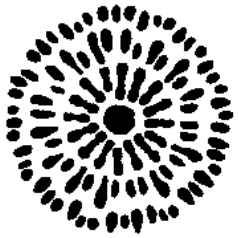
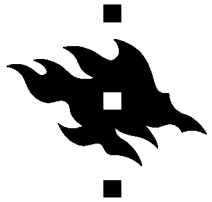
Introduction to Bioinformatics

Esa Pitkänen
esa.pitkanen@cs.helsinki.fi
Autumn 2008, I period
www.cs.helsinki.fi/mbi/courses/08-09/itb



MBI MASTER'S DEGREE
PROGRAMME IN BIOINFORMATICS



Introduction to Bioinformatics

Lecture 1:
Administrative issues
MBI Programme, Bioinformatics courses
What is bioinformatics?
Molecular biology primer

How to enrol for the course?

- Use the registration system of the Computer Science department: https://ilmo.cs.helsinki.fi
  - You need your user account at the IT department (“cc account”)
- If you cannot register yet, don’t worry: attend the lectures and exercises; just register when you are able to do so

Teachers

- Esa Pitkänen, Department of Computer Science, University of Helsinki
- Elja Arjas, Department of Mathematics and Statistics, University of Helsinki
- Sami Kaski, Department of Information and Computer Science, Helsinki University of Technology
- Lauri Eronen, Department of Computer Science, University of Helsinki (exercises)

Lectures and exercises

- Lectures: Tuesday and Friday 14.15-16.00 Exactum C221
- Exercises: Tuesday 16.15-18.00 Exactum C221
  - First exercise session on Tue 9 September

Status & Prerequisites

- Advanced level course at the Department of Computer Science, U. Helsinki
- 4 credits
- Prerequisites:
  - Basic mathematics skills (probability calculus, basic statistics)
  - Familiarity with computers
  - Basic programming skills recommended
  - No biology background required

Course contents

- What is bioinformatics?
- Molecular biology primer
- Biological words
- Sequence assembly
- Sequence alignment
- Fast sequence alignment using FASTA and BLAST
- Genome rearrangements
- Motif finding (tentative)
- Phylogenetic trees
- Gene expression analysis

How to pass the course?

- Recommended method:
  - Attend the lectures (not obligatory though)
  - Do the exercises
  - Take the course exam
- Or:
  - Take a separate exam

How to pass the course?

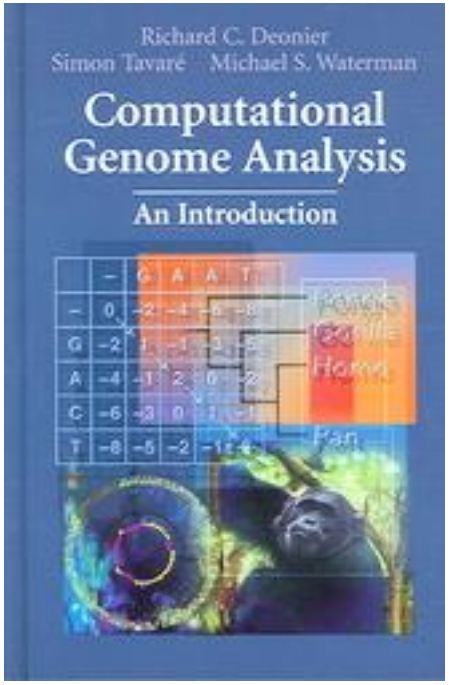
- Exercises give you max. 12 points
  - 0% completed assignments gives you 0 points, 80% gives 12 points, the rest by linear interpolation
  - “A completed assignment” means that
    - You are willing to present your solution in the exercise session and
    - You return notes by e-mail to Lauri Eronen (see course web page for contact info) describing the main phases you took to solve the assignment
  - Return notes at latest on Tuesdays 16.15
- Course exam gives you max. 48 points

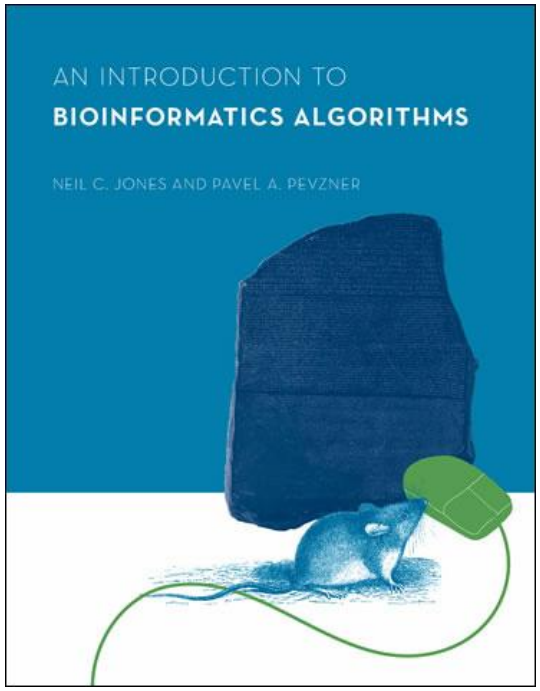
How to pass the course?

- Grading: on the scale 0-5
  - To get the lowest passing grade 1, you need to get at least 30 points out of 60 maximum
- Course exam: Wed 15 October 16.00-19.00 Exactum A111
- See course web page for separate exams
- Note: if you take the first separate exam, the best of the following options will be considered:
  - Exam gives you 48 points, exercises 12 points
  - Exam gives you 60 points
- In second and subsequent separate exams, only the 60 point option is in use

Literature

- Deonier, Tavaré, Waterman: Computational Genome Analysis, an Introduction. Springer, 2005
- Jones, Pevzner: An Introduction to Bioinformatics Algorithms. MIT Press, 2004
- Slides for some lectures will be available on the course web page

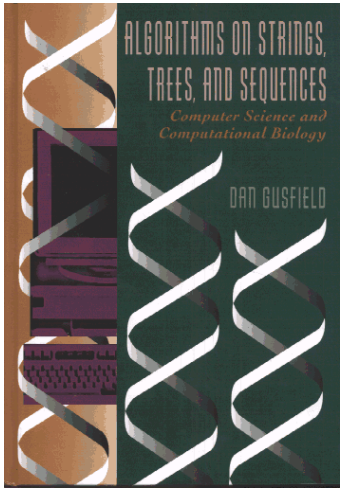


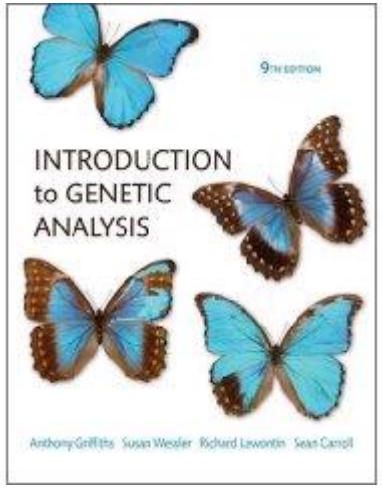


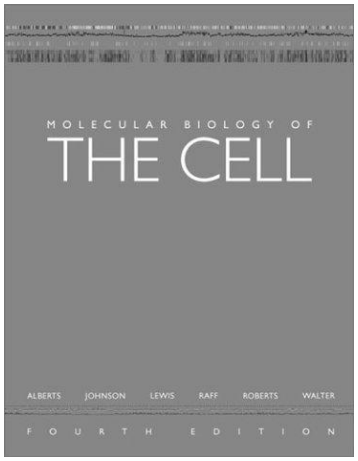
Additional literature

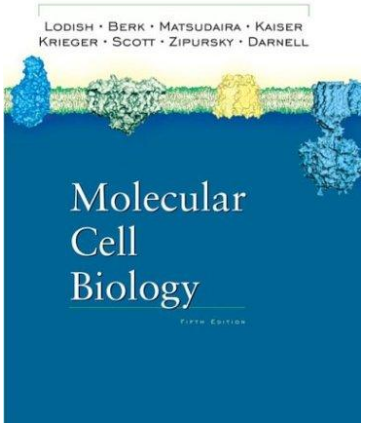
- Gusfield: Algorithms on strings, trees and sequences
- Griffiths et al: Introduction to genetic analysis
- Alberts et al.: Molecular biology of the cell
- Lodish et al.: Molecular cell biology

- Check the course web site









Questions about administrative & practical stuff?

Master's Degree Programme in Bioinformatics (MBI)

- Two-year MSc programme
- Admission for 2009-2010 in January 2009
  - You need to have your Bachelor's degree ready by August 2009

MBI programme organizers

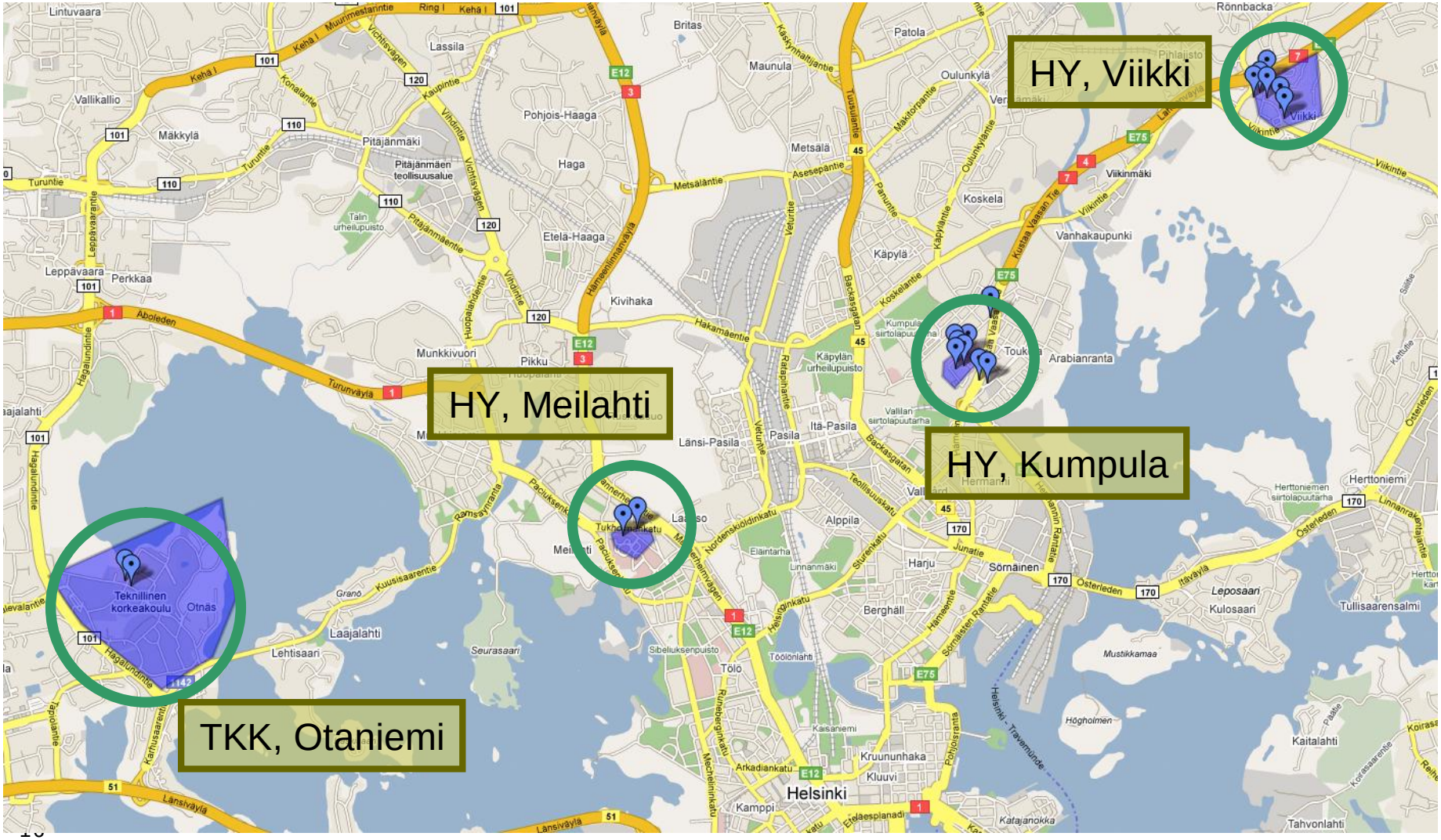
Department of Computer Science,
Department of Mathematics and Statistics
Faculty of Science, Kumpula Campus, HY

Laboratory of Computer and Information Science, Laboratory of CS and Engineering,TKK

Faculty of Medicine, Meilahti Campus, HY

Faculty of Biosciences
Faculty of Agriculture and Forestry
Viikki Campus, HY

Four MBI campuses



MBI highlights

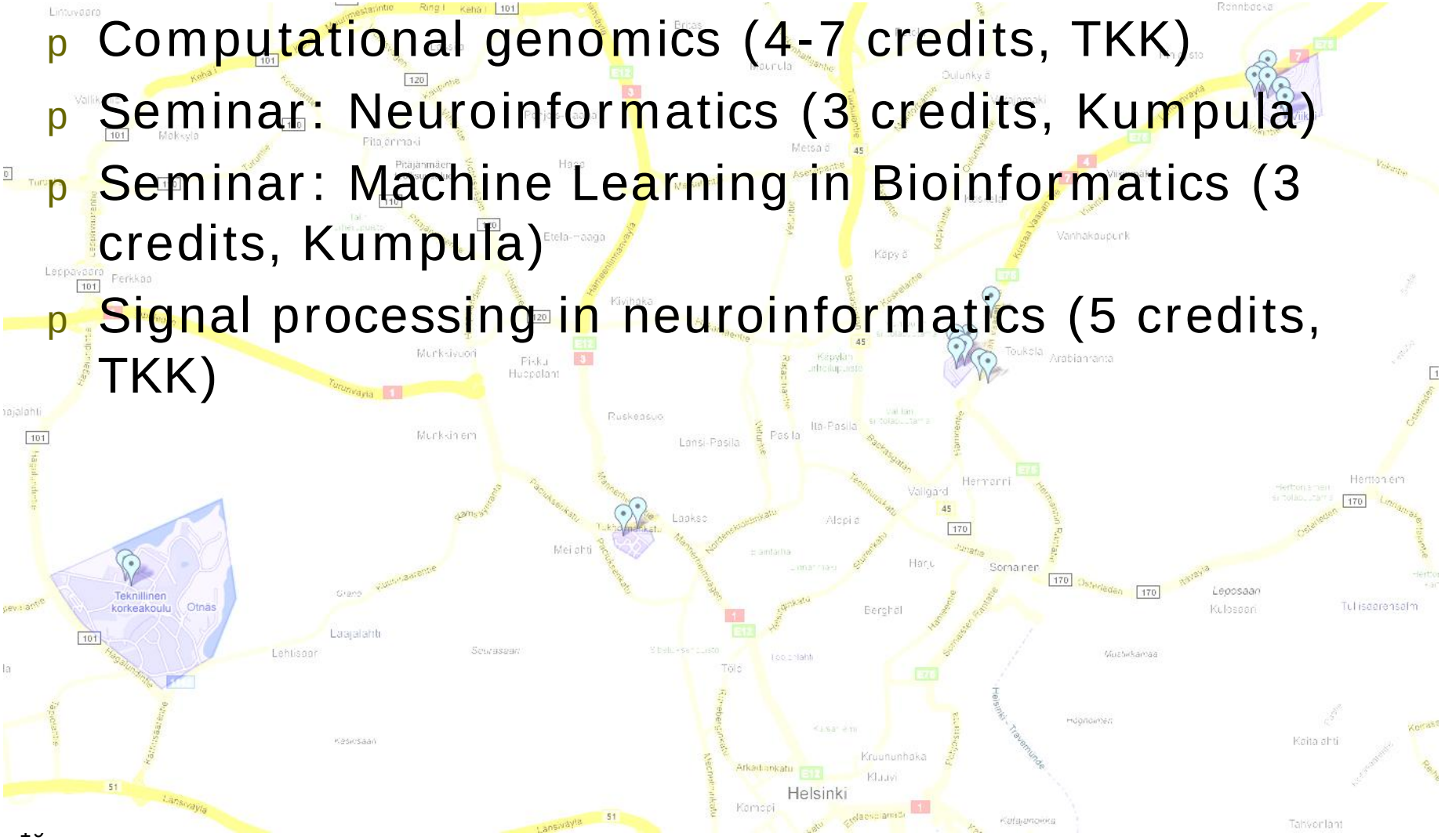
- You can take courses from both HY and TKK
- Two biology courses tailored specifically for MBI
- Bioinformatics is a new exciting field, with a high demand for experts in job market
- Go to www.cs.helsinki.fi/mbi/careers to find out what a bioinformatician could do for living

Admission

- Admission requirements
  - Bachelor's degree in a suitable field (e.g., computer science, mathematics, statistics, biology or medicine)
  - At least 60 ECTS credits in total in computer science, mathematics and statistics
  - Proficiency in English (standardized language test: TOEFL, IELTS)
- Admission period opens in late Autumn 2009 and closes in 2 February 2009
- Details on admission will be posted in www.cs.helsinki.fi/mbi during this autumn

Bioinformatics courses in Helsinki region: 1st period

- Computational genomics (4-7 credits, TKK)
- Seminar: Neuroinformatics (3 credits, Kumpula)
- Seminar: Machine Learning in Bioinformatics (3 credits, Kumpula)
- Signal processing in neuroinformatics (5 credits, TKK)

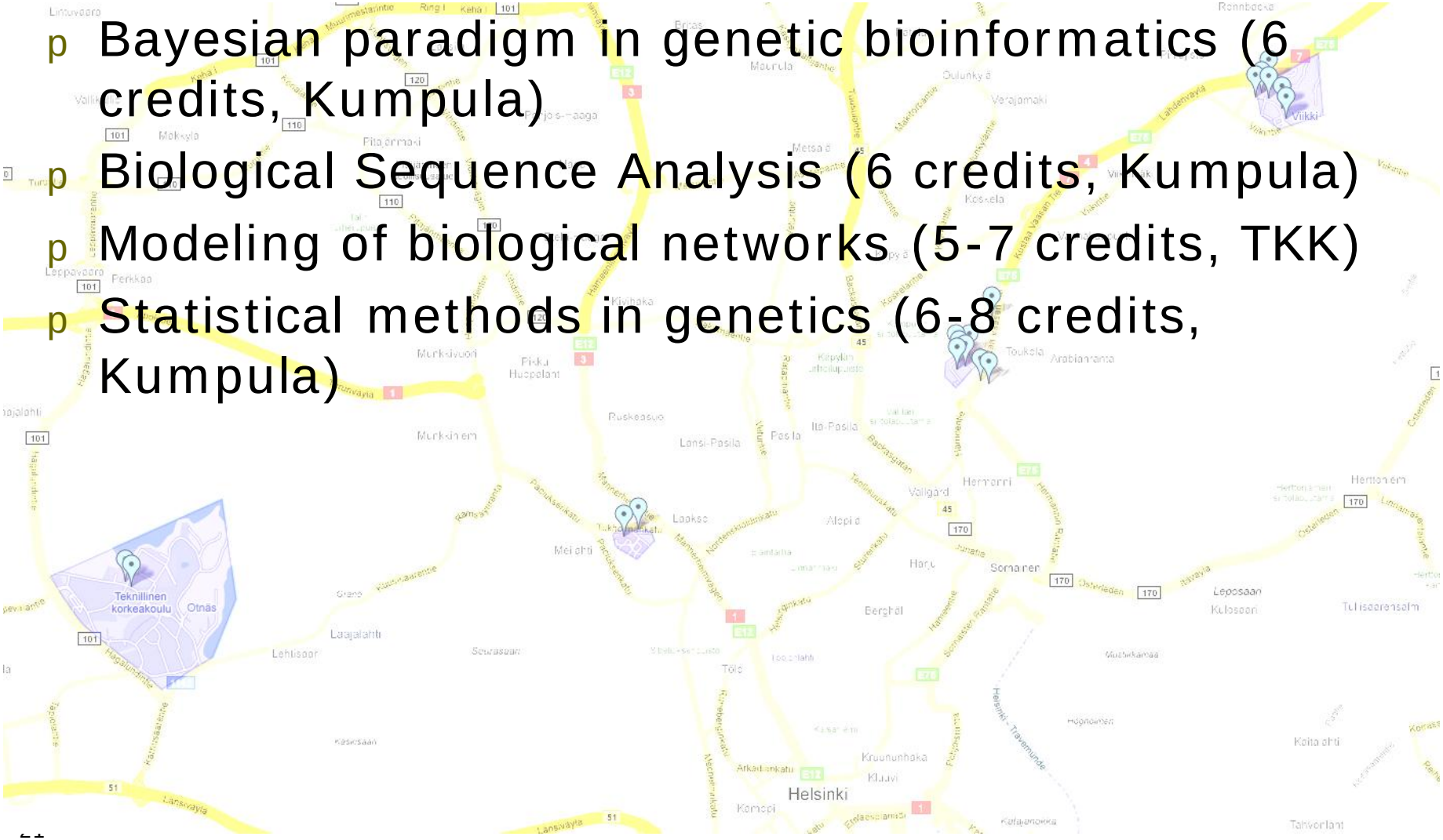


A good biology course for computer scientists and mathematicians?

- Biology for methodological scientists (8 credits, Meilahti)
  - Course organized by the Faculties of Bioscience and Medicine for the MBI programme
  - Introduction to basic concepts of microarrays, medical genetics and developmental biology
  - Study group + book exam in I period (2 cr)
  - Three lectured modules, 2 cr each
  - Each module has an individual registration so you can participate even if you missed the first module
  - www.cs.helsinki.fi/mbi/courses/08-09/bfms/

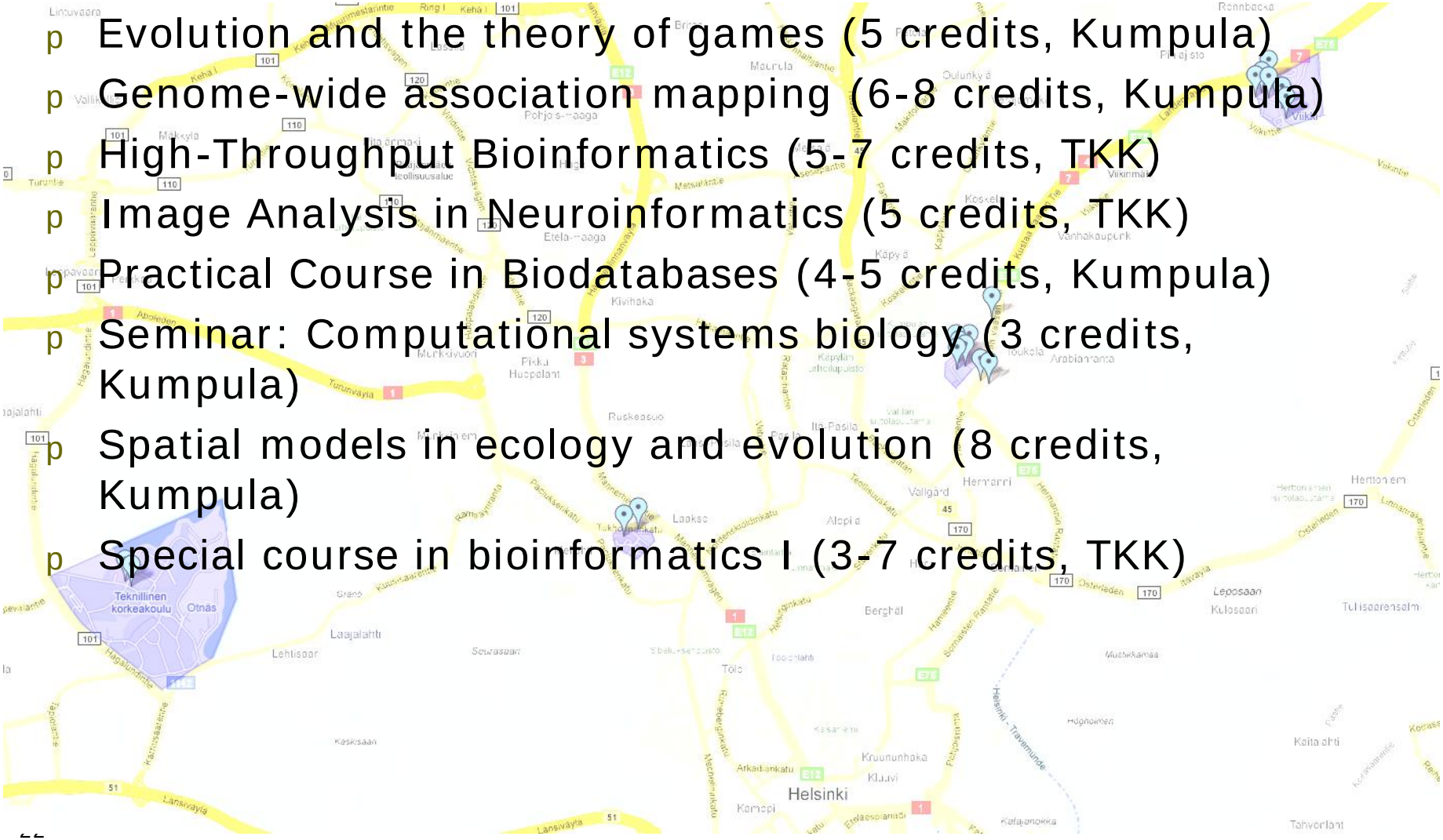
Bioinformatics courses in Helsinki region: 2nd period

- Bayesian paradigm in genetic bioinformatics (6 credits, Kumpula)
- Biological Sequence Analysis (6 credits, Kumpula)
- Modeling of biological networks (5-7 credits, TKK)
- Statistical methods in genetics (6-8 credits, Kumpula)



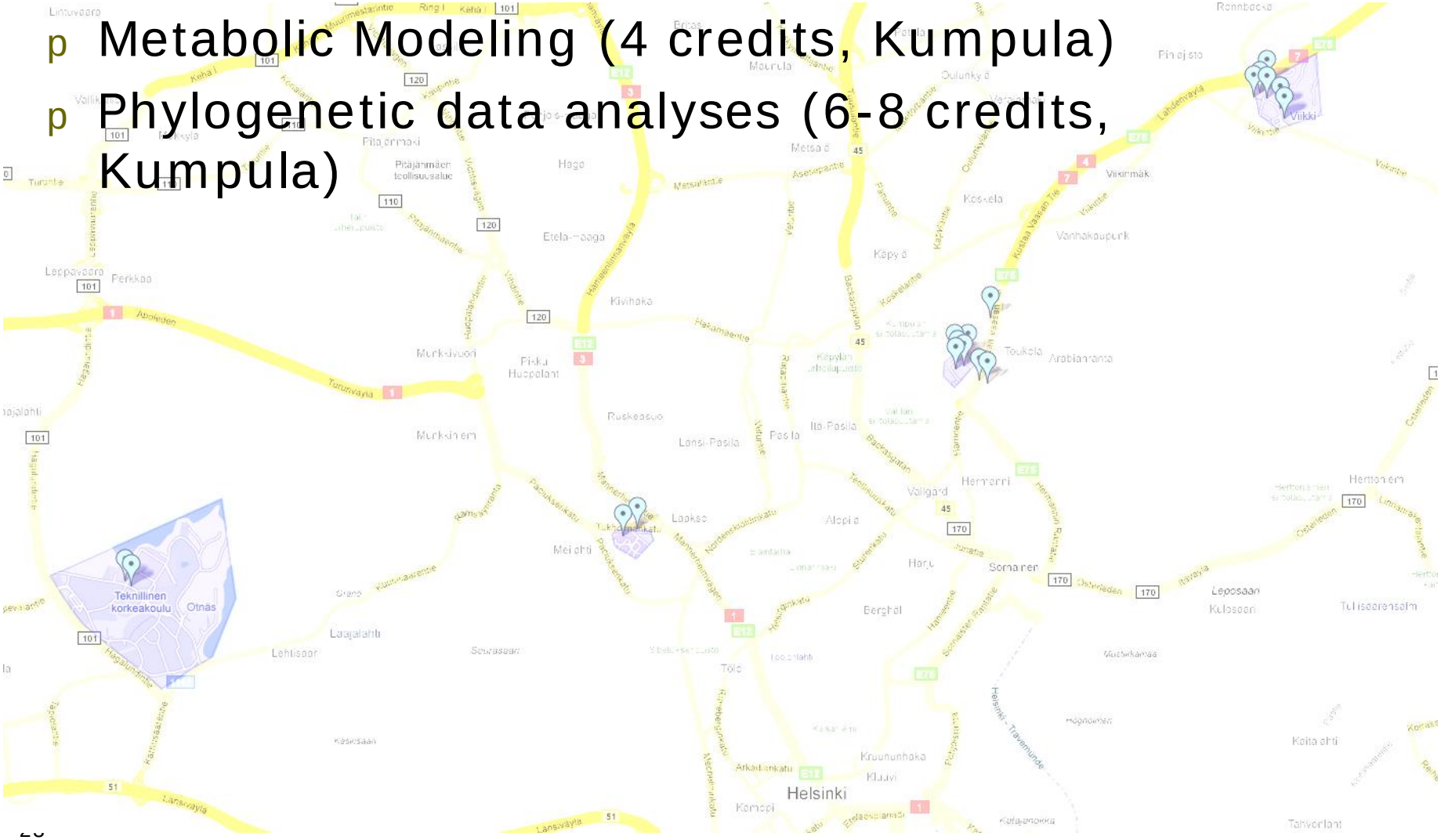
Bioinformatics courses in Helsinki region: 3rd period

- Evolution and the theory of games (5 credits, Kumpula)
- Genome-wide association mapping (6-8 credits, Kumpula)
- High-Throughput Bioinformatics (5-7 credits, TKK)
- Image Analysis in Neuroinformatics (5 credits, TKK)
- Practical Course in Biodatabases (4-5 credits, Kumpula)
- Seminar: Computational systems biology (3 credits, Kumpula)
- Spatial models in ecology and evolution (8 credits, Kumpula)
- Special course in bioinformatics I (3-7 credits, TKK)

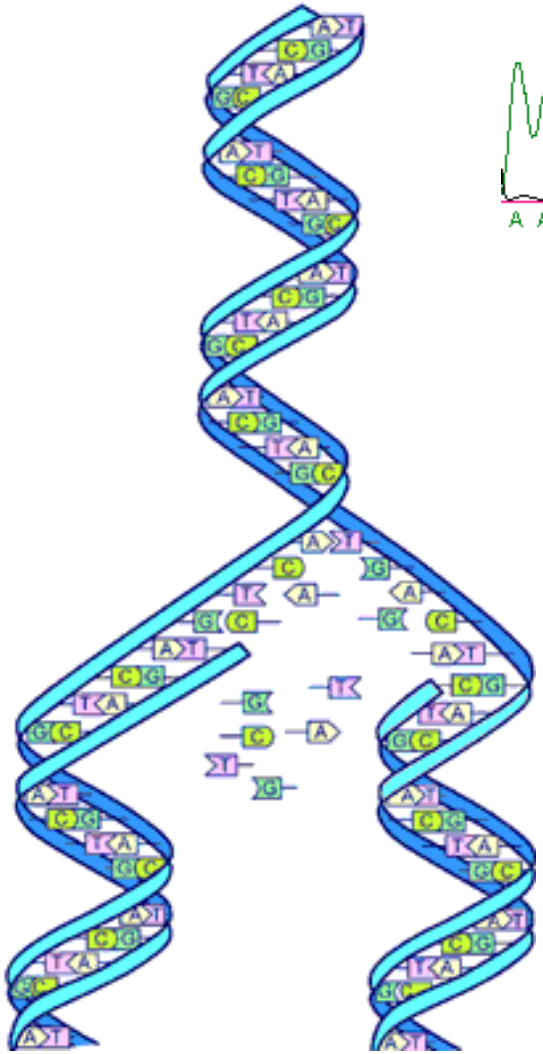


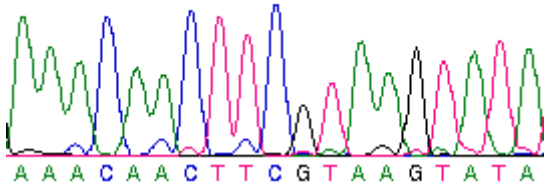
Bioinformatics courses in Helsinki region: 4th period

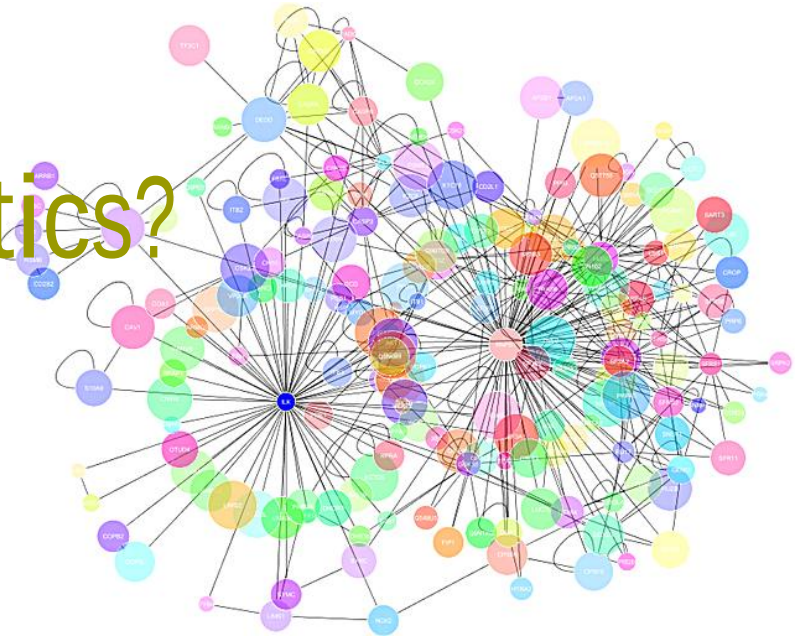
- Metabolic Modeling (4 credits, Kumpula)
- Phylogenetic data analyses (6-8 credits, Kumpula)

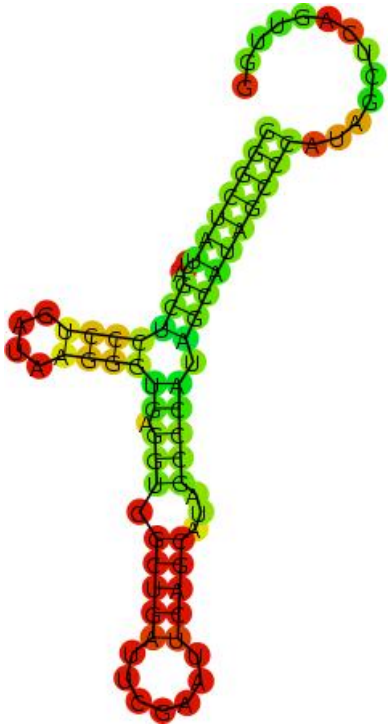


1. What is bioinformatics?









What is bioinformatics?

- Bioinformatics, *n.* The science of information and information flow in biological systems, esp. of the use of computational methods in genetics and genomics. (Oxford English Dictionary)

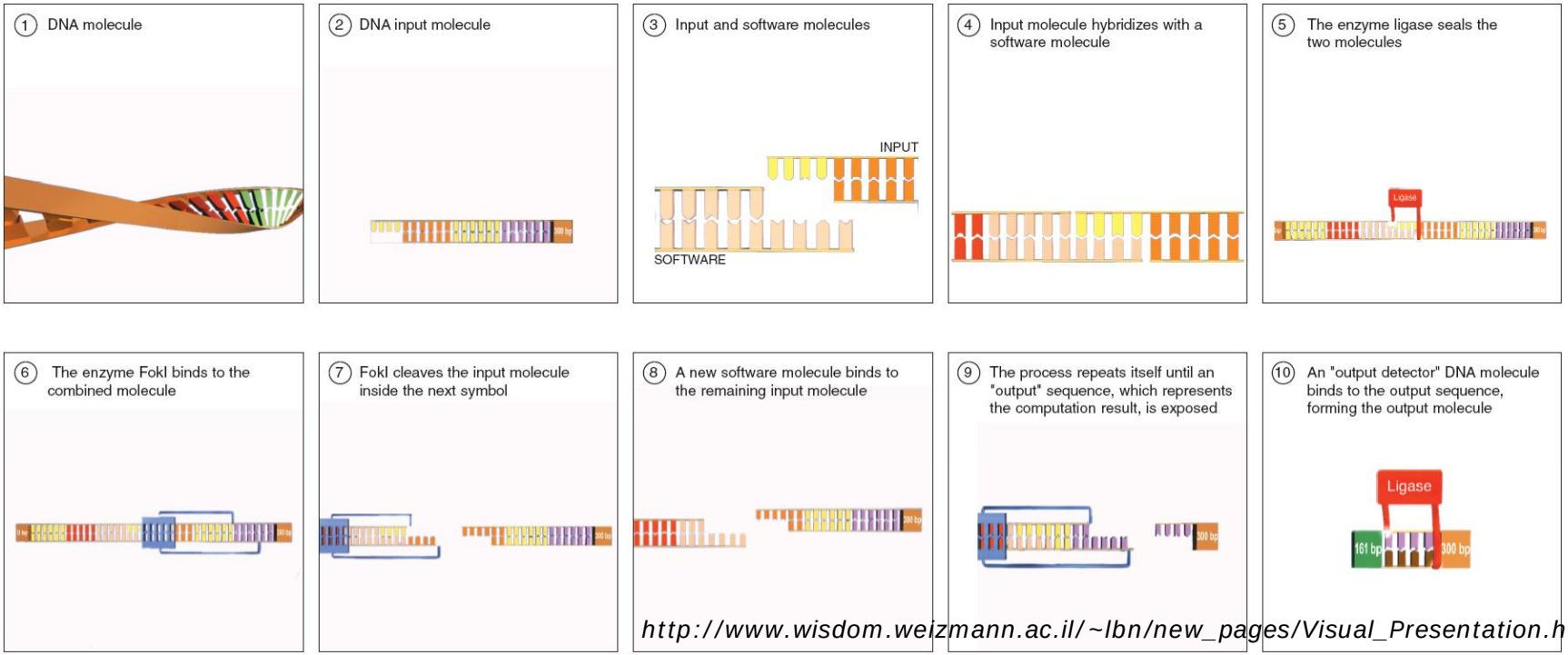
- "The mathematical, statistical and computing methods that aim to solve biological problems using DNA and amino acid sequences and related information." -- Fredj Tekaia

What is bioinformatics?

- "I do not think all biological computing is bioinformatics, e.g. mathematical modelling is not bioinformatics, even when connected with biology-related problems. In my opinion, bioinformatics has to do with management and the subsequent use of biological information, particular genetic information."
  -- Richard Durbin

What is *not* bioinformatics?

- Biologically-inspired computation, e.g., genetic algorithms and neural networks
- However, application of neural networks to solve some biological problem, could be called bioinformatics
- What about DNA computing?



http://www.wisdom.weizmann.ac.il/~lbn/new_pages/Visual_Presentation.html

Computational biology

- Application of computing to biology (broad definition)
- Often used interchangeably with bioinformatics
- Or: *Biology* that is done with computational means

Biometry & biophysics

- Biometry: the statistical analysis of biological data
  - Sometimes also the field of identification of individuals using biological traits (a more recent definition)
- Biophysics: "an interdisciplinary field which applies techniques from the physical sciences to understanding biological structure and function" -- British Biophysical Society

Mathematical biology

- Mathematical biology "tackles biological problems, but the methods it uses to tackle them need not be numerical and need not be implemented in software or hardware."

  -- Damian Counsell

Alan Turing

THE CHEMICAL BASIS OF MORPHOGENESIS

By A. M. TURING, F.R.S. *University of Manchester*

(*Received* 9 *November* 1951—*Revised* 15 *March* 1952)

It is suggested that a system of chemical substances, called morphogens, reacting together and diffusing through a tissue, is adequate to account for the main phenomena of morphogenesis. Such a system, although it may originally be quite homogeneous, may later develop a pattern or structure due to an instability of the homogeneous equilibrium, which is triggered off by random disturbances. Such reaction-diffusion systems are considered in some detail in the case of an isolated ring of cells, a mathematically convenient, though biologically unusual system. The investigation is chiefly concerned with the onset of instability. It is found that there are six essentially different forms which this may take. In the most interesting form stationary waves appear on the ring. It is suggested that this might account, for instance, for the tentacle patterns on *Hydra* and for whorled leaves. A system of reactions and diffusion on a sphere is also considered. Such a system appears to account for gastrulation. Another reaction system in two dimensions gives rise to patterns reminiscent of dappling. It is also suggested that stationary waves in two dimensions could account for the phenomena of phyllotaxis.

The purpose of this paper is to discuss a possible mechanism by which the genes of a zygote may determine the anatomical structure of the resulting organism. The theory does not make any new hypotheses; it merely suggests that certain well-known physical laws are sufficient to account for many of the facts. The full understanding of the paper requires a good knowledge of mathematics, some biology, and some elementary chemistry. Since readers cannot be expected to be experts in all of these subjects, a number of elementary facts are explained, which can be found in text-books, but whose omission would make the paper difficult reading.

1. A MODEL OF THE EMBRYO. MORPHOGENS

In this section a mathematical model of the growing embryo will be described. This model will be a simplification and an idealization, and consequently a falsification. It is to be hoped that the features retained for discussion are those of greatest importance in the present state of knowledge.

The model takes two slightly different forms. In one of them the cell theory is recognized but the cells are idealized into geometrical points. In the other the matter of the organism is imagined as continuously distributed. The cells are not, however, completely ignored, for various physical and physico-chemical characteristics of the matter as a whole are assumed to have values appropriate to the cellular matter.

With either of the models one proceeds as with a physical theory and defines an entity called 'the state of the system'. One then describes how that state is to be determined from the state at a moment very shortly before. With either model the description of the state consists of two parts, the mechanical and the chemical. The mechanical part of the state describes the positions, masses, velocities and elastic properties of the cells, and the forces between them. In the continuous form of the theory essentially the same information is given in the form of the stress, velocity, density and elasticity of the matter. The chemical part of the state is given (in the cell form of theory) as the chemical composition of each separate cell; the diffusibility of each substance between each two adjacent cells must also

VOL. 237. B. 641. (Price 8s.) 5 [Published 14 August 1952

Turing on biological complexity

- "It must be admitted that the biological examples which it has been possible to give in the present paper are very limited.

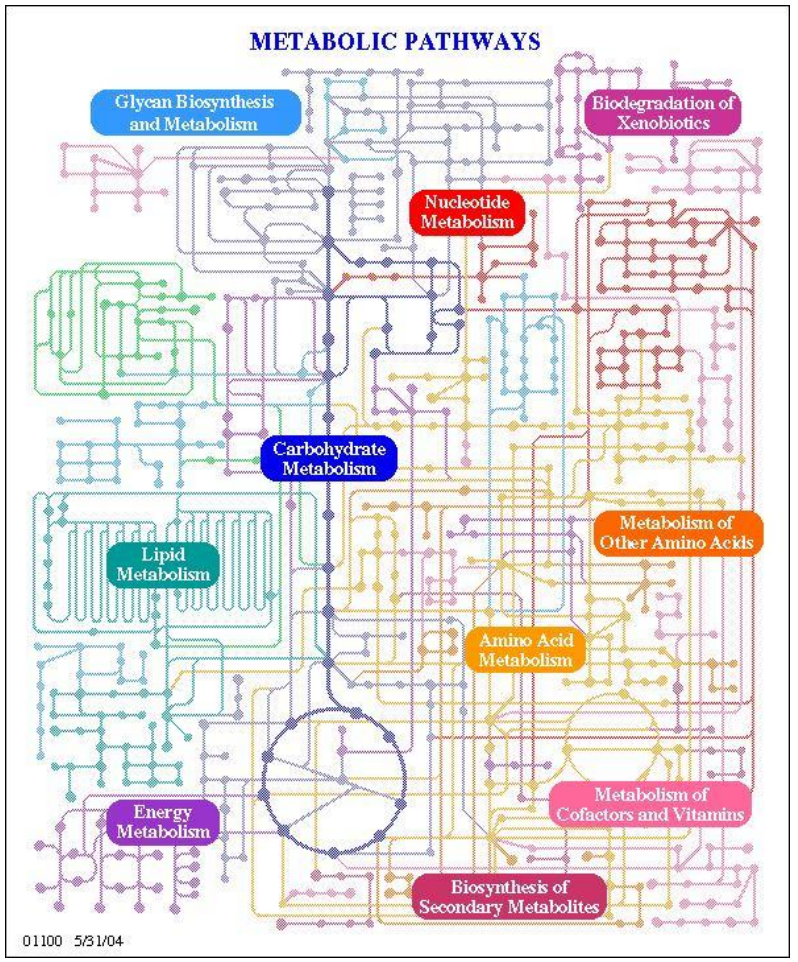
  This can be ascribed quite simply to the fact that biological phenomena are usually very complicated. Taking this in combination with the relatively elementary mathematics used in this paper one could hardly expect to find that many observed biological phenomena would be covered.

  It is thought, however, that the imaginary biological systems which have been treated, and the principles which have been discussed, should be of some help in interpreting real biological forms."

  – Alan Turing, The Chemical Basis of Morphogenesis, 1952

Related concepts

- Systems biology
  - "Biology of networks"
  - Integrating different levels of information to understand how biological systems work
- Computational systems biology



Overview of metabolic pathways in KEGG database, www.genome.jp/kegg/

Why is bioinformatics important?

- New measurement techniques produce huge quantities of biological data
  - Advanced data analysis methods are needed to make sense of the data
  - Typical data sources produce noisy data with a lot of missing values
- Paradigm shift in biology to utilise bioinformatics in research

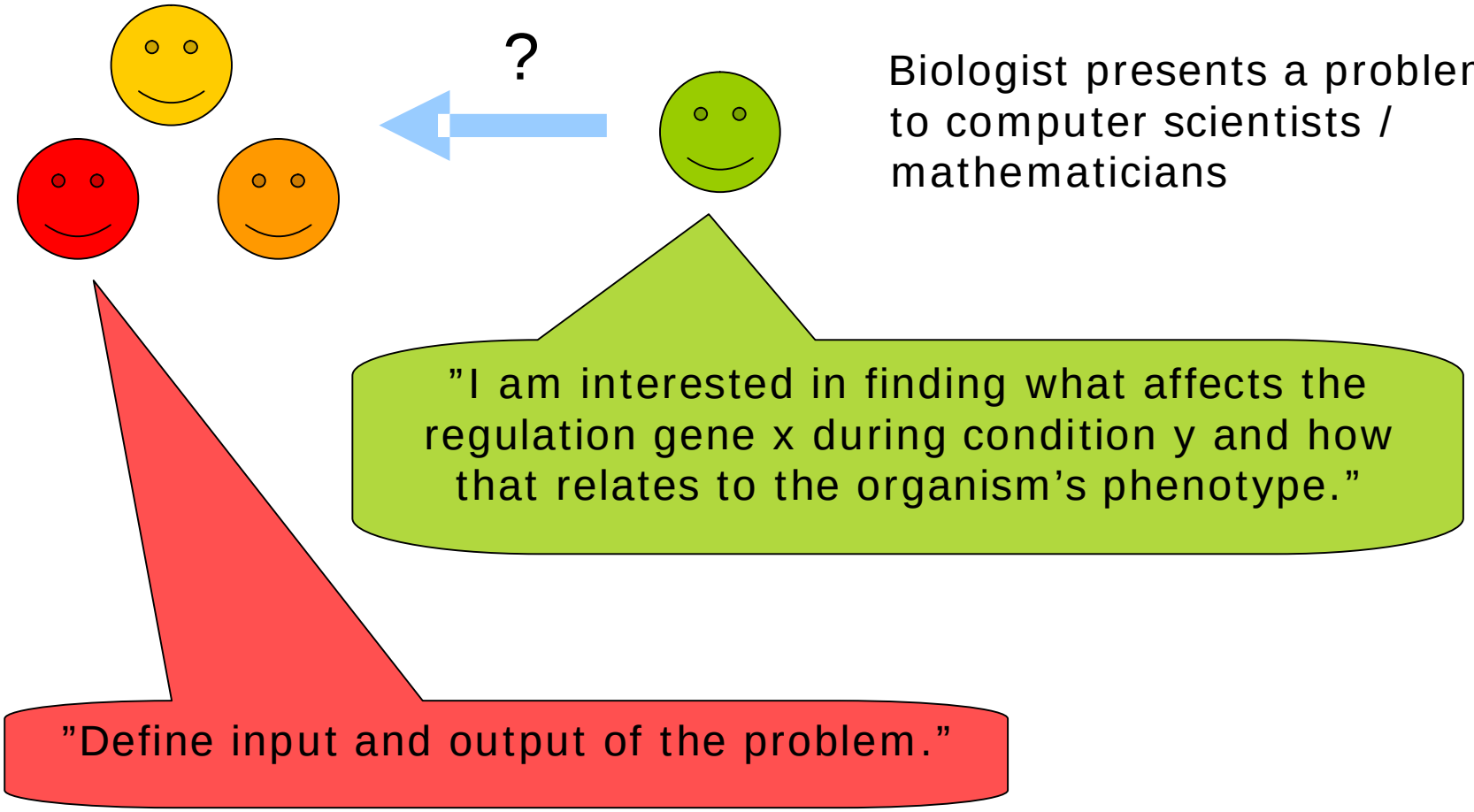
Bioinformatician's skill set

- Statistics, data analysis methods
  - Lots of data
  - High noise levels, missing values
  - #attributes >> #data points
- Programming languages
  - Scripting languages: Python, Perl, Ruby, ...
  - Extensive use of text file formats: need parsers
  - Integration of both data and tools
- Data structures, databases

Bioinformatician's skill set

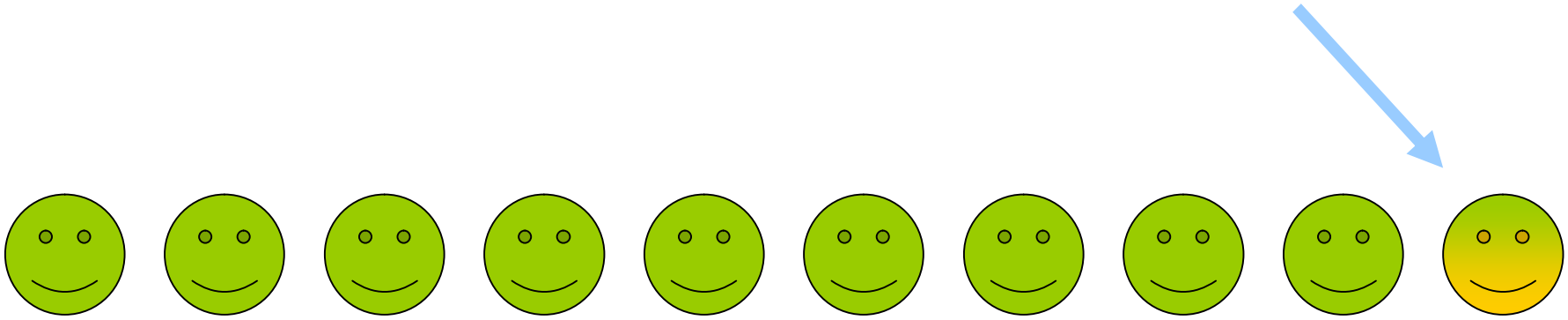
- Modelling
  - Discrete vs continuous domains
  - -> Systems biology
- Scientific computation packages
  - R, Matlab/Octave, ...
- Communication skills!

Communication skills: case 1

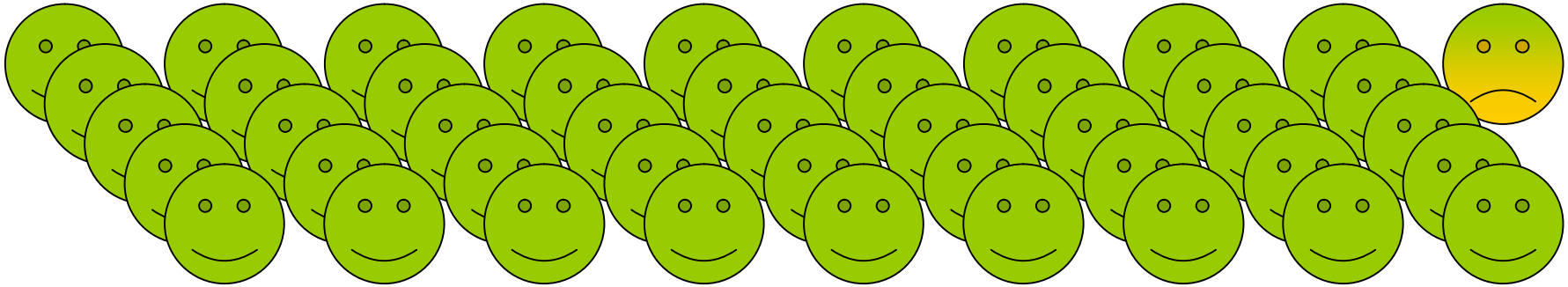


Communication skills: case 2

Bioinformatician is a part of a group that consists mostly of biologists.

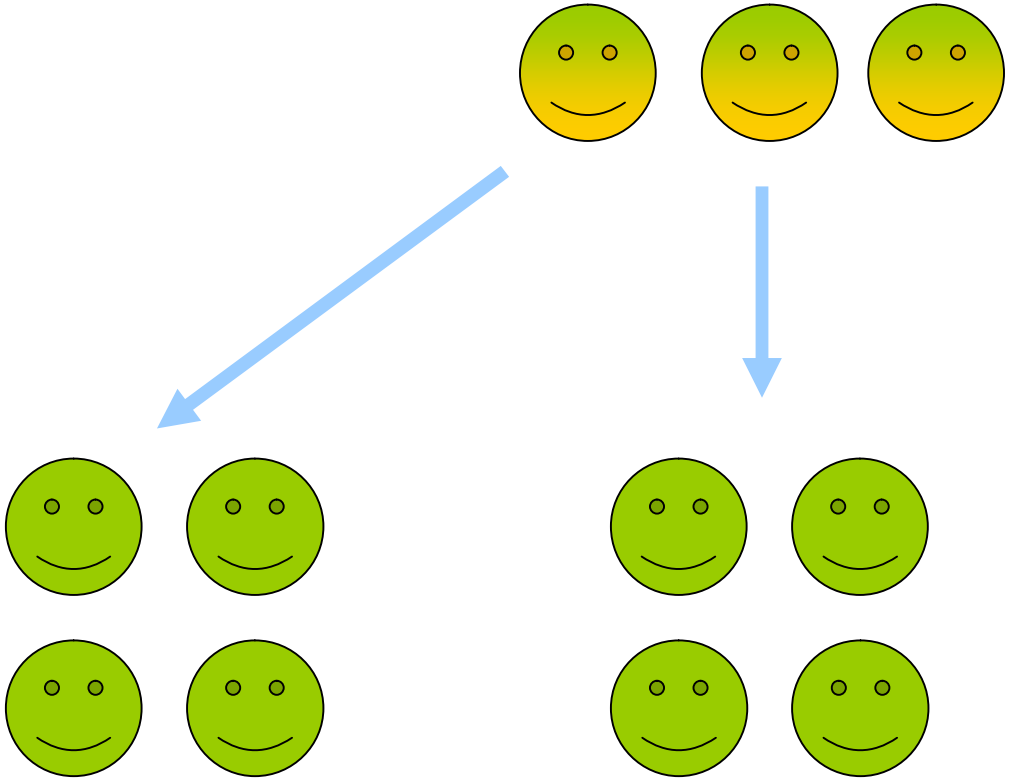


Communication skills: case 2



...biologist/bioinformatician ratio is important!

Communication skills: case 3



A group of bioinformaticians offers their services to more than one group

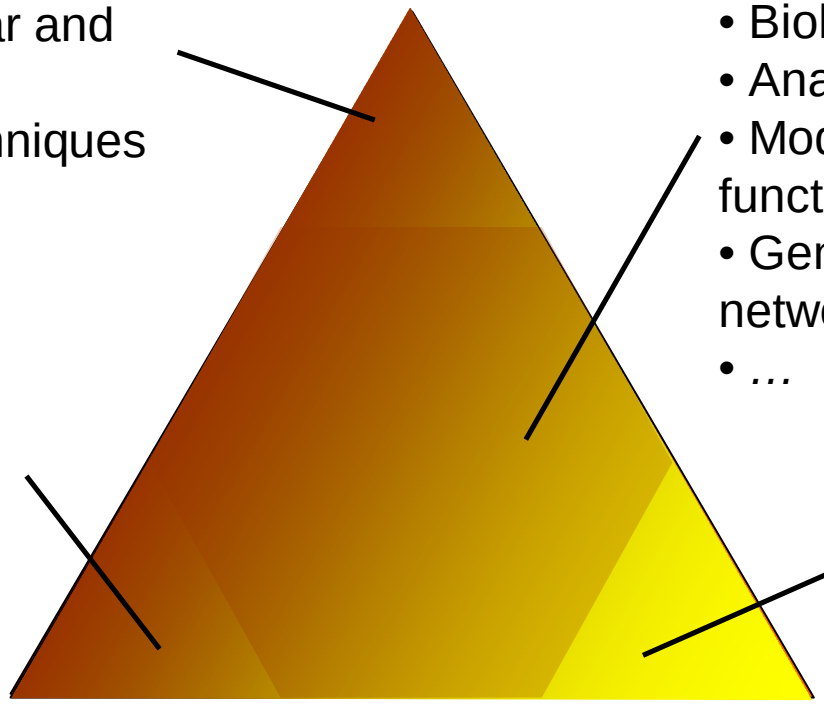
Bioinformatician's skill set

- How much biology you should know?

Bioinformatician's skill set

Biology & Medicine
- Basics in molecular and cell biology
- Measurement techniques

Computer Science
- Programming
- Databases
- Algorithmics



Bioinformatics
- Biological sequence analysis
- Biological databases
- Analysis of gene expression
- Modeling protein structure and function
- Gene, protein and metabolic networks
- *…*

Mathematics and statistics
- Calculus
- Probability calculus
- Linear algebra

Where would you be in this triangle?

Prof. Juho Rousu, 2006

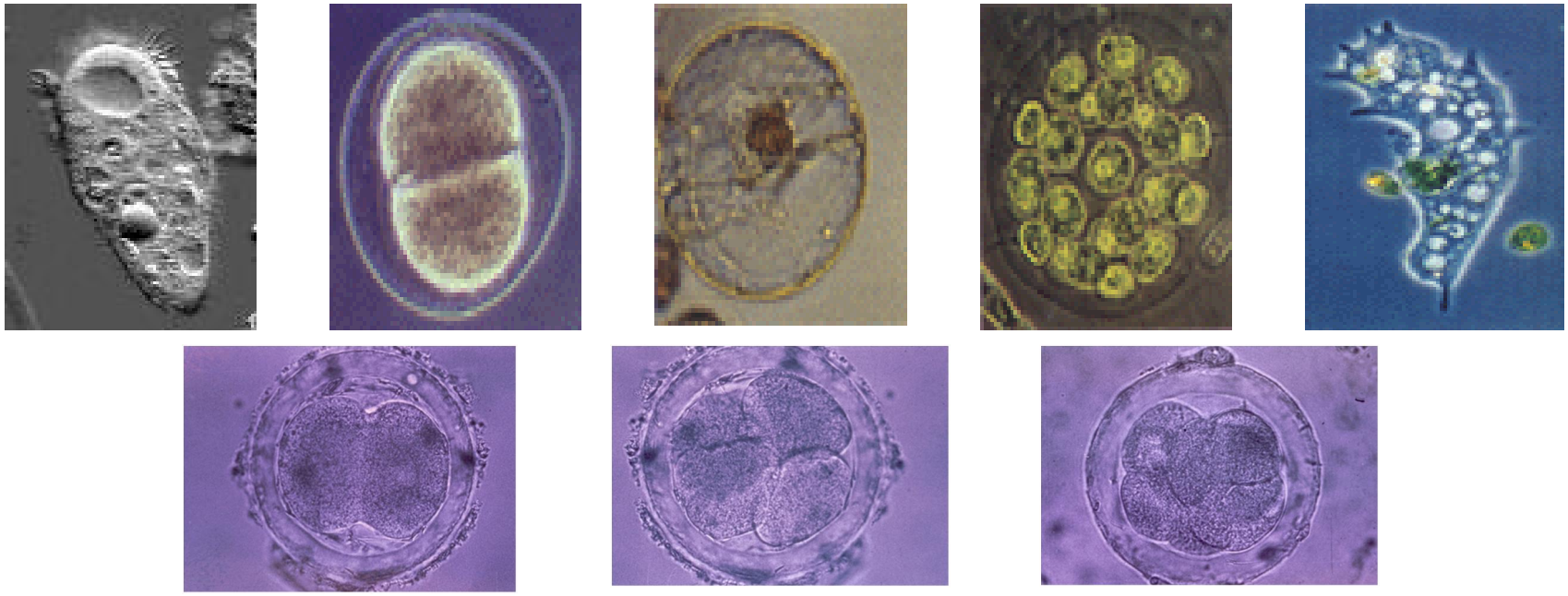
A problem involving bioinformatics?

- ”I found a fruit fly that is immune to all diseases!”

- ”It was one of these”

Pertti Jarla, http://www.hs.fi/fingerpori/

Molecular biology primer

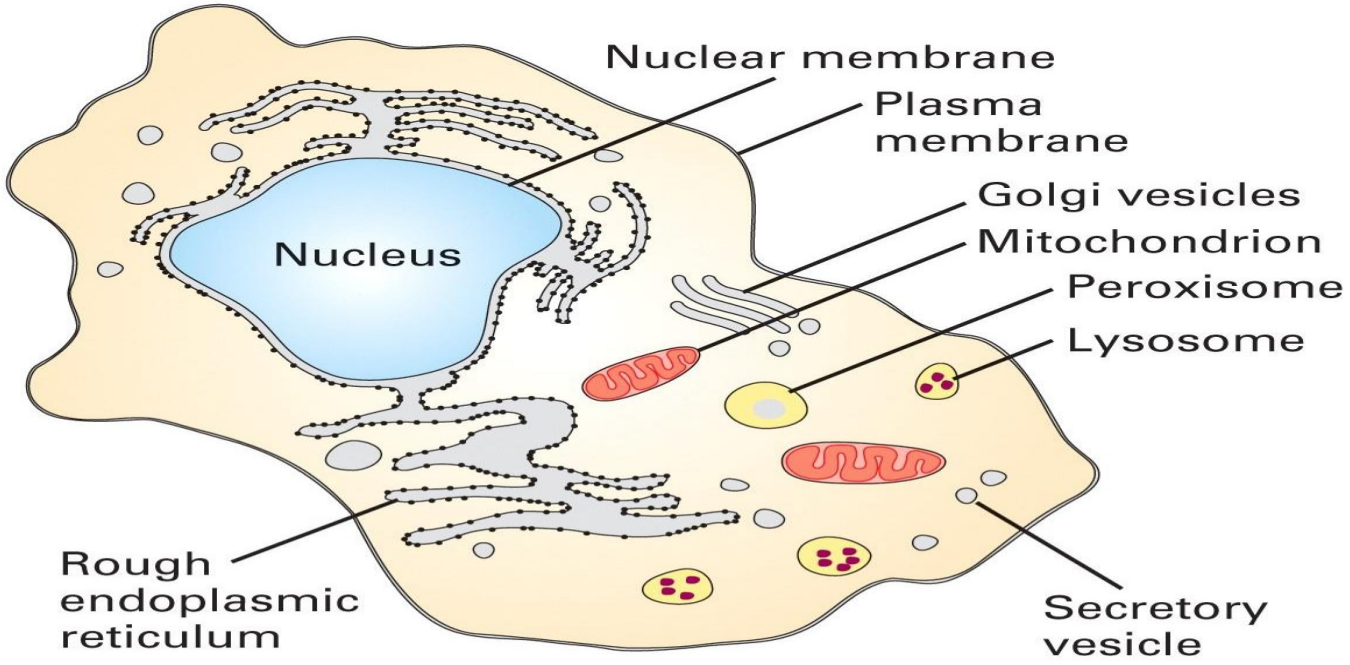


Molecular Biology Primer by Angela Brooks, Raymond Brown, Calvin Chen, Mike Daly, Hoa Dinh, Erinn Hama, Robert Hinman, Julio Ng, Michael Sneddon, Hoa Troung, Jerry Wang, Che Fung Yung
Edited for Introduction to Bioinformatics (Autumn 2007, Summer 2008, Autumn 2008) by Esa Pitkänen

Molecular biology primer

- Part 1: What is life made of?
- Part 2: Where does the variation in genomes come from?

Life begins with Cell

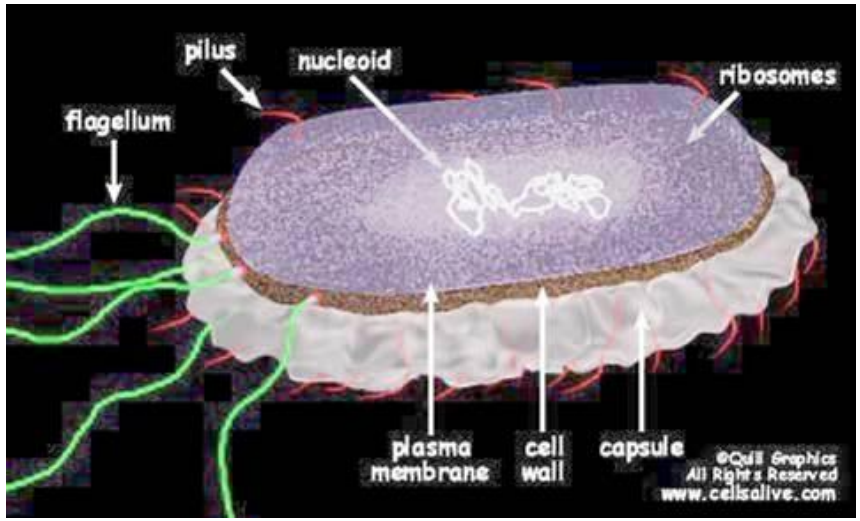


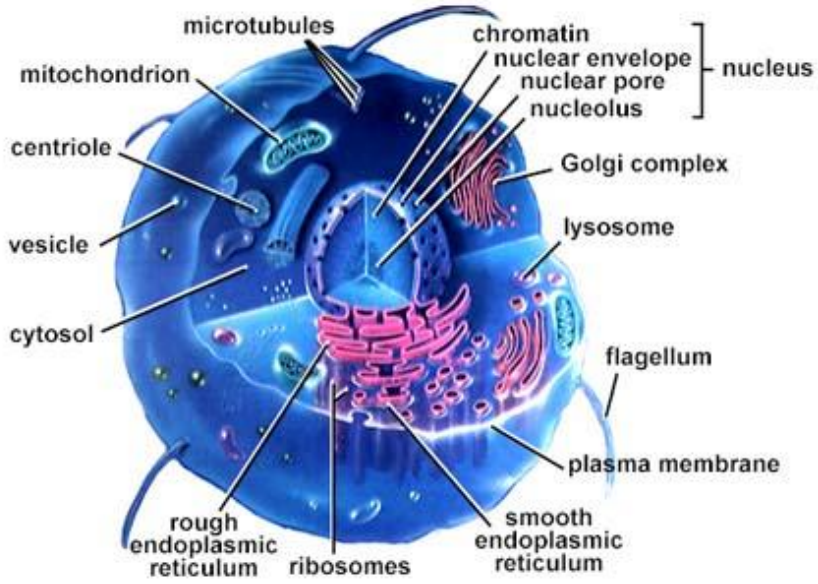
- A cell is a smallest structural unit of an organism that is capable of independent functioning
- All cells have some common features

Cells

- Fundamental working units of every living system.
- Every organism is composed of one of two radically different types of cells:
  - prokaryotic cells or
  - eukaryotic cells.
- Prokaryotes and Eukaryotes are descended from the same primitive cell.
  - All prokaryotic and eukaryotic cells are the result of a total of 3.5 billion years of evolution.

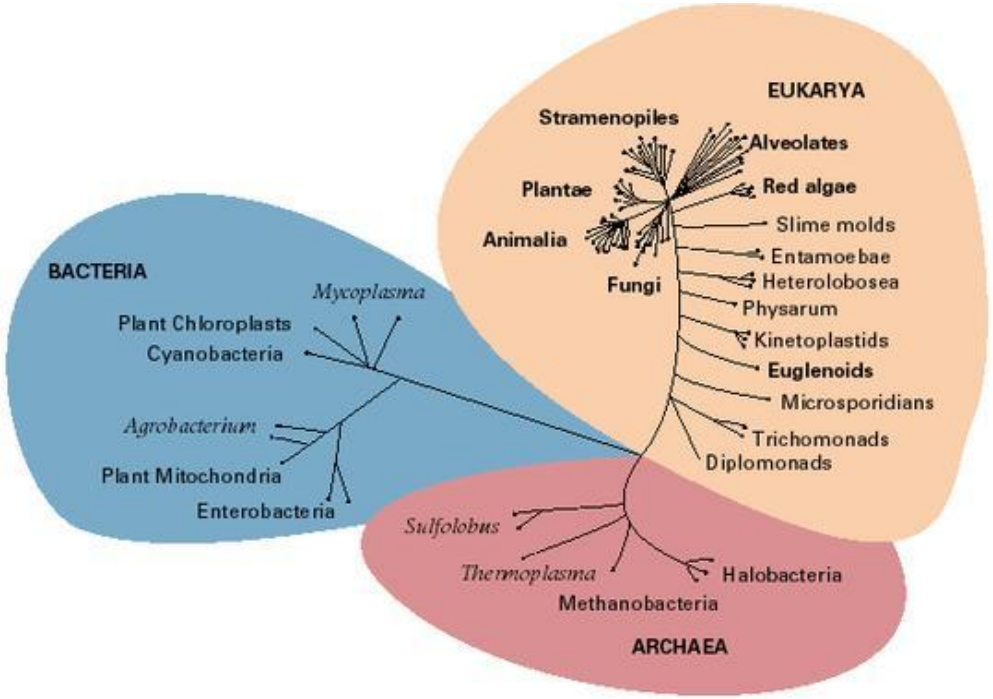
Two types of cells: Prokaryotes and Eukaryotes





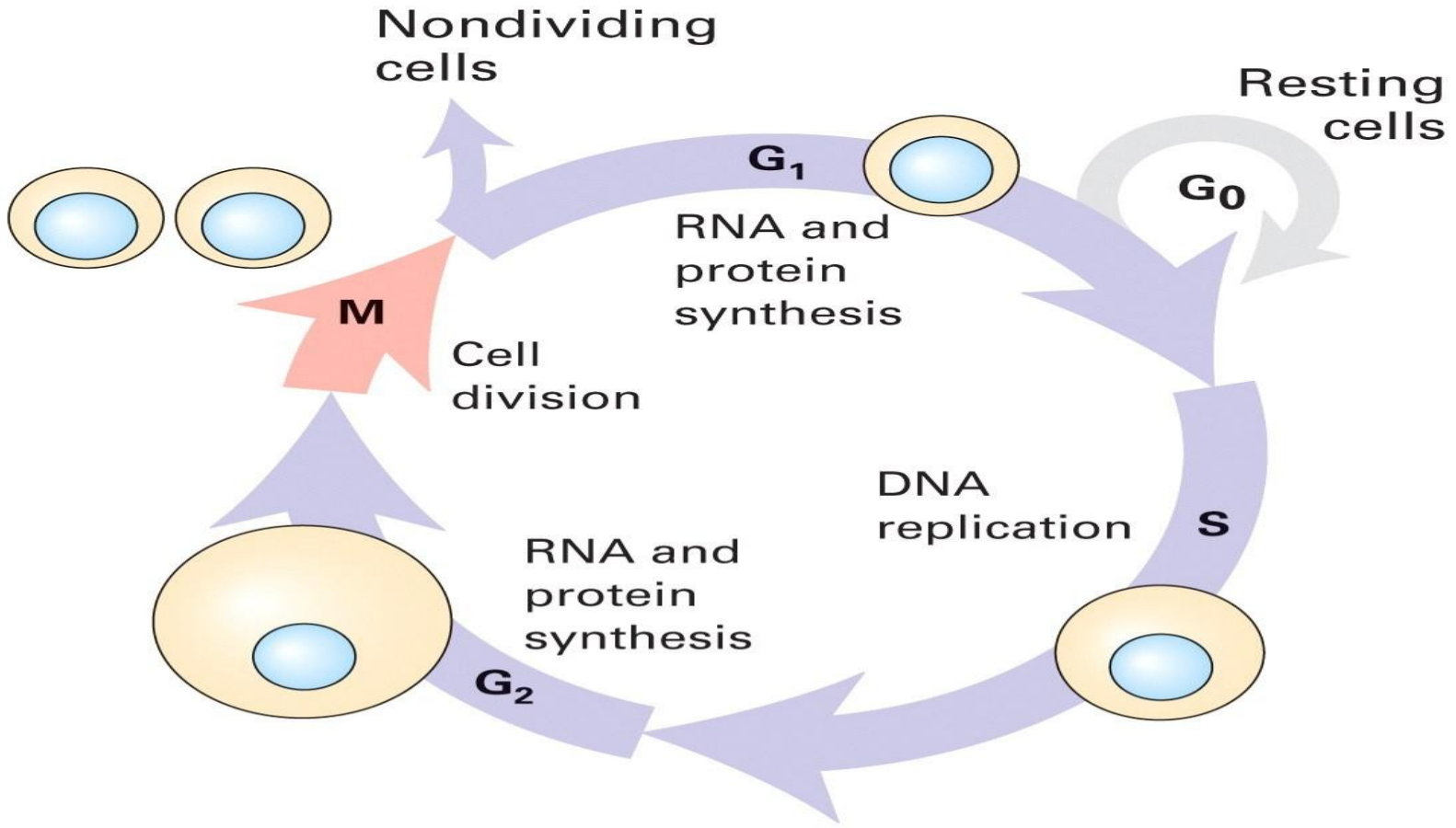
Prokaryotes and Eukaryotes

- According to the most recent evidence, there are three main branches to the tree of life
- Prokaryotes include Archaea (“ancient ones”) and bacteria
- Eukaryotes are kingdom Eukarya and includes plants, animals, fungi and certain algae



► Lecture: Phylogenetic trees

All Cells have common Cycles



- Born, eat, replicate, and die

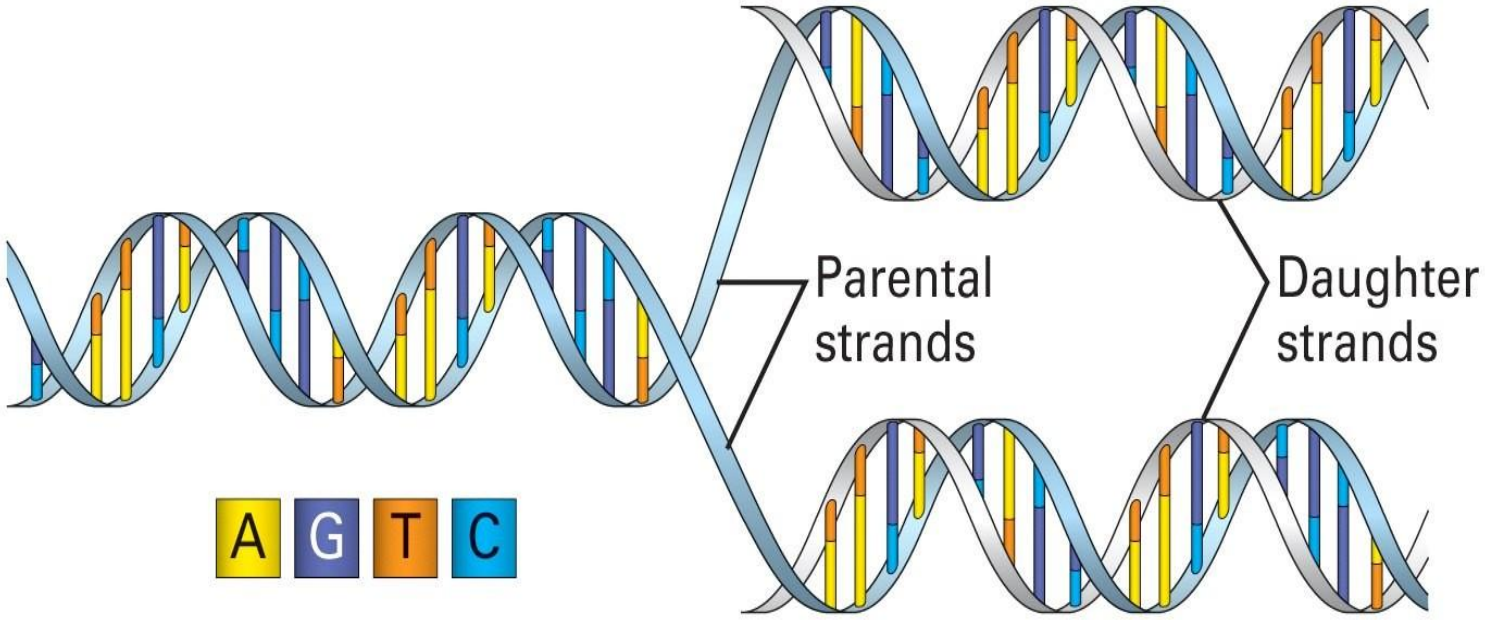
Common features of organisms

- Chemical energy is stored in ATP
- Genetic information is encoded by DNA
- Information is transcribed into RNA
- There is a common triplet genetic code
- Translation into proteins involves ribosomes
- Shared metabolic pathways
- Similar proteins among diverse groups of organisms

All Life depends on 3 critical molecules

- DNAs (Deoxyribonucleic acid)
  - Hold information on how cell works
- RNAs (Ribonucleic acid)
  - Act to transfer short pieces of information to different parts of cell
  - Provide templates to synthesize into protein
- Proteins
  - Form enzymes that send signals to other cells and regulate gene activity
  - Form body's major components (e.g. hair, skin, etc.)
  - "Workhorses" of the cell

DNA: The Code of Life

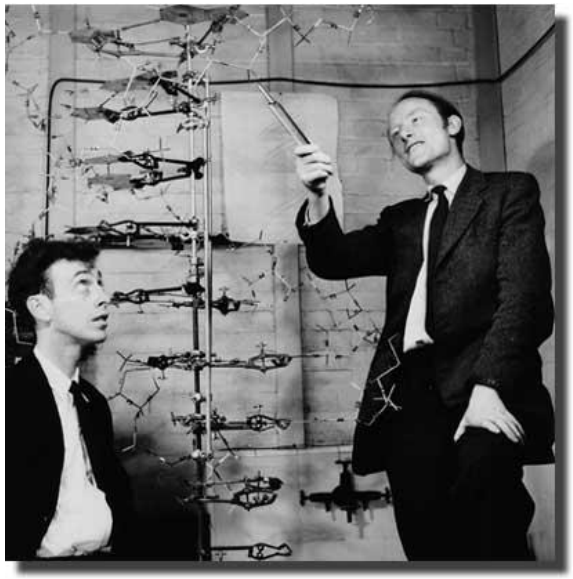


- The structure and the four genomic letters code for all living organisms
- Adenine, Guanine, Thymine, and Cytosine which pair A-T and C-G on complimentary strands.

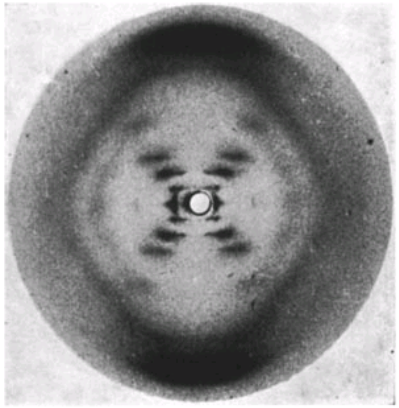
► Lecture: Genome sequencing and assembly

Discovery of the structure of DNA

- 1952-1953 James D. Watson and Francis H. C. Crick deduced the double helical structure of DNA from X-ray diffraction images by Rosalind Franklin and data on amounts of nucleotides in DNA



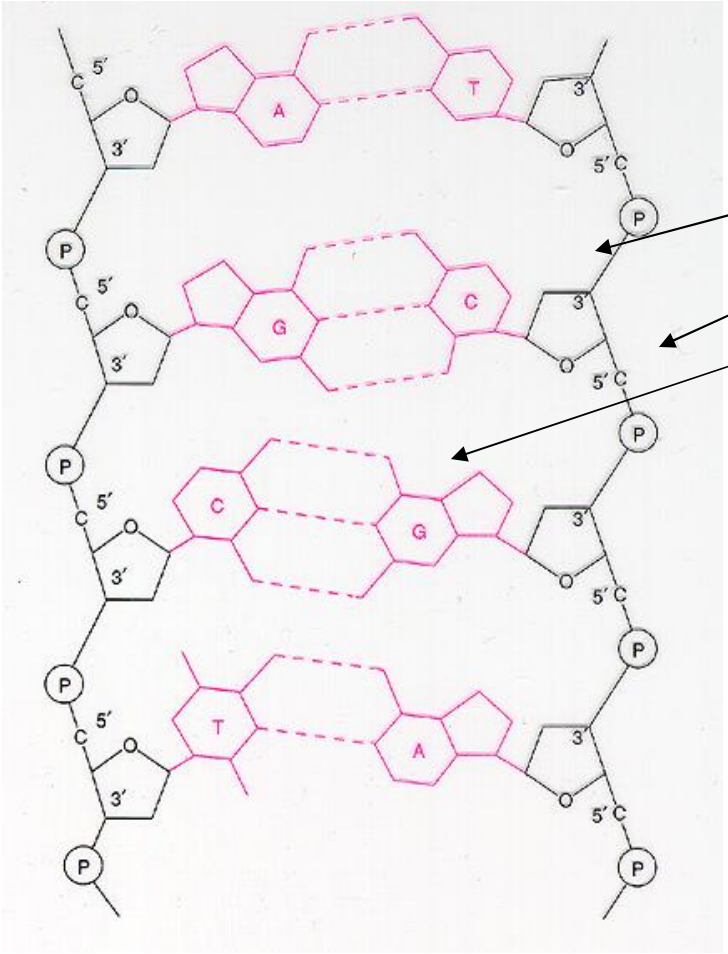
James Watson and Francis Crick



”Photo 51”

Rosalind Franklin

DNA, continued

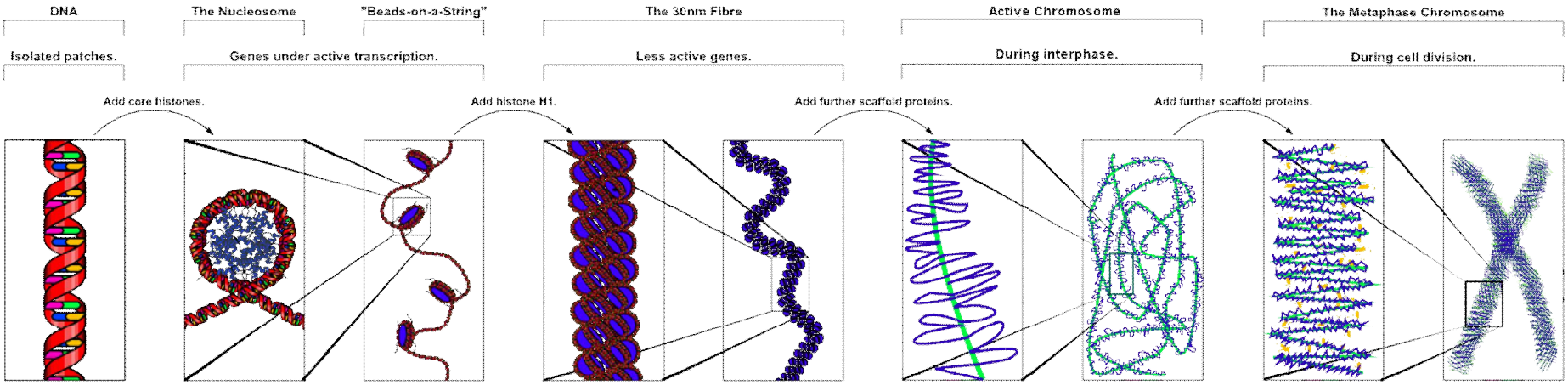


- DNA has a double helix structure which is composed of
  - sugar molecule
  - phosphate group
  - and a base (A,C,G,T)
- By convention, we read DNA strings in direction of transcription: from 5’ end to 3’ end

  5’ ATTTAGGCC 3’

  3’ TAAATCCGG 5’

DNA is contained in chromosomes

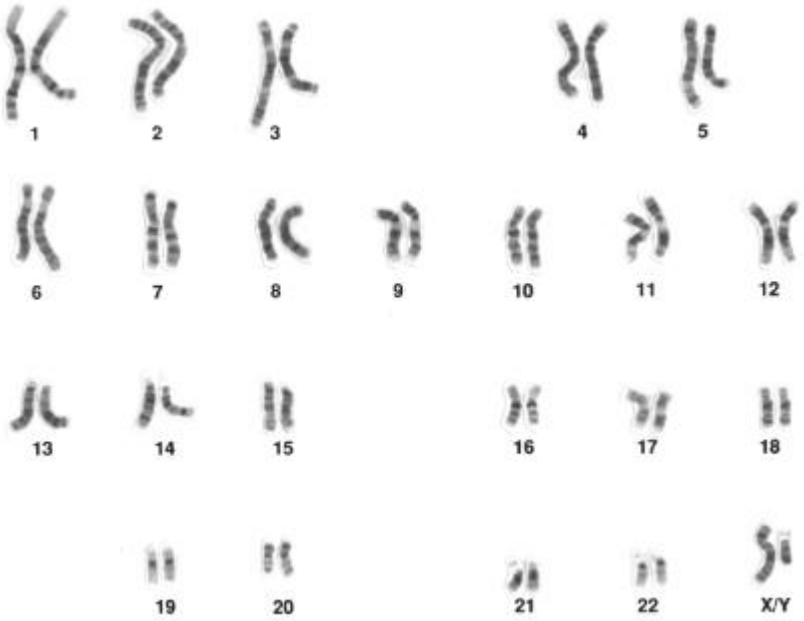


- In eukaryotes, DNA is packed into *chromatids*
  - In metaphase, the "X" structure consists of two identical chromatids
- In prokaryotes, DNA is usually contained in a single, circular chromosome

http://en.wikipedia.org/wiki/Image:Chromatin_Structures.png

Human chromosomes

- Somatic cells in humans have 2 pairs of 22 chromosomes + XX (female) or XY (male) = total of 46 chromosomes
- Germline cells have 22 chromosomes + either X or Y = total of 23 chromosomes



Karyogram of human male using Giemsa staining (http://en.wikipedia.org/wiki/Karyotype)

Length of DNA and number of chromosomes

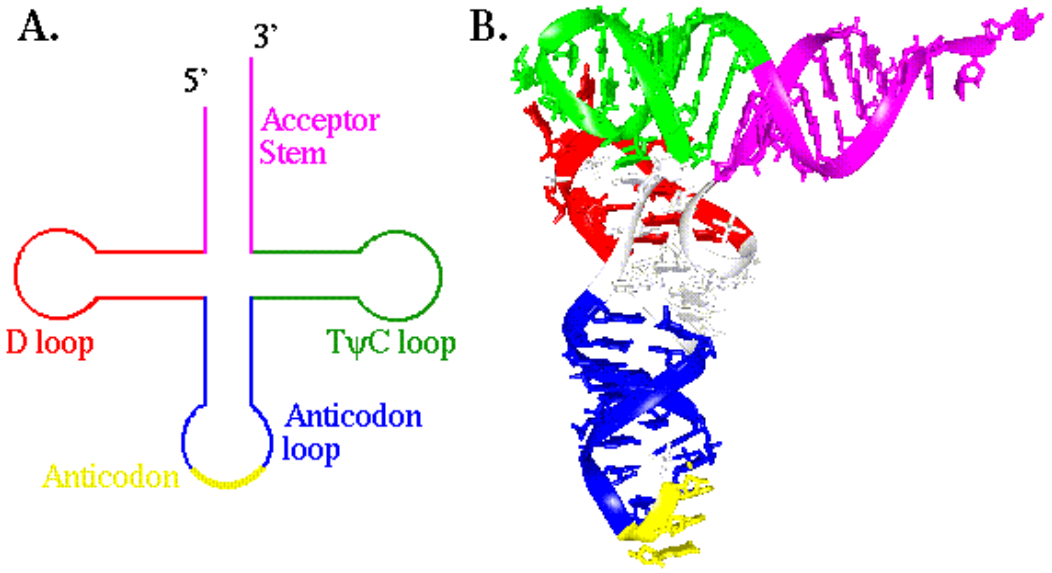
Organism	#base pairs	#chromosomes (germline)
Prokayotic		
Escherichia coli (bacterium)	$4x10^6$	1
Eukaryotic		
Saccharomyces cerevisia (yeast)	$1.35x10^7$	17
Drosophila melanogaster (insect)	$1.65x10^8$	4
Homo sapiens (human)	$2.9x10^9$	23
Zea mays (corn / maize)	$5.0x10^9$	10

Hepatitis delta virus, complete genome

```
   1 atgagccaag ttccgaacaa ggattcgcgg ggaggataga tcagcgcccg agaggggtga
  61 gtcggtaaag agcattggaa cgtcggagat acaactccca agaaggaaaa aagagaaagc
 121 aagaagcgga tgaatttccc cataacgcca gtgaaactct aggaagggga aagagggaag
 181 gtggaagaga aggaggcggg cctcccgatc cgaggggccc ggcggccaag tttggaggac
 241 actccggccc gaagggttga gagtacccca gagggaggaa gccacacgga gtagaacaga
 301 gaaatcacct ccagaggacc ccttcagcga acagagagcg catcgcgaga gggagtagac
 361 catagcgata ggaggggatg ctaggagttg ggggagaccg aagcgaggag gaaagcaaag
 421 agagcagcgg ggctagcagg tgggtgttcc gccccccgag aggggacgag tgaggcttat
 481 cccggggaac tcgacttatc gtccccacat agcagactcc cggacccccc ttcaaagtga
 541 ccgaggggggg tgactttgaa cattggggac cagtggagcc atgggatgct cctcccgatt
 601 ccgcccaagc tccttccccc caagggtcgc ccaggaatgg cgggacccca ctctgcaggg
 661 tccgcgttcc atcctttctt acctgatggc cggcatggtc ccagcctcct cgctggcgcc
 721 ggctgggcaa cattccgagg ggaccgtccc ctcggtaatg gcgaatggga cccacaaatc
 781 tctctagctt cccagagaga agcgagagaa aagtggctct cccttagcca tccgagtgga
 841 cgtgcgtcct ccttcggatg cccaggtcgg accgcgagga ggtggagatg ccatgccgac
 901 ccgaagagga aagaaggacg cgagacgcaa acctgcgagt ggaaacccgc tttattcact
 961 ggggtcgaca actctgggga gaggagggag ggtcggctgg gaagagtata tcctatggga
1021 atccctggct tccccttatg tccagtccct ccccggtccg agtaaagggg gactccggga
1081 ctccttgcat gctggggacg aagccgcccc cgggcgctcc cctcgttcca ccttcgaggg
1141 ggttcacacc cccaacctgc gggccggcta ttcttctttc ccttctctcg tcttcctcgg
1201 tcaacctcct aagttcctct tcctcctcct tgctgaggtt ctttcccccc gccgatagct
1261 gctttctctt gttctcgagg gccttccttc gtcggtgatc ctgcctctcc ttgtcggtga
1321 atcctccccc ggaaggcctc ttcctaggtc cggagtctac ttccatctgg tccgttcggg
1381 ccctcttcgc cgggggagcc ccctctccat ccttatcttt ctttccgaga attcctttga
1441 tgtttcccag ccagggatgt tcatcctcaa gtttcttgat tttcttctta accttccgga
1501 ggtctctctc gagttcctct aacttctttc ttccgctcac ccactgctcg agaacctctt
1561 ctctcccccc gcggtttttc cttccttcgg gccggctcat cttcgactag aggcgacggt
1621 cctcagtact cttactcttt tctgtaaaga ggagactgct ggccctgtcg cccaagttcg
1681 ag
```

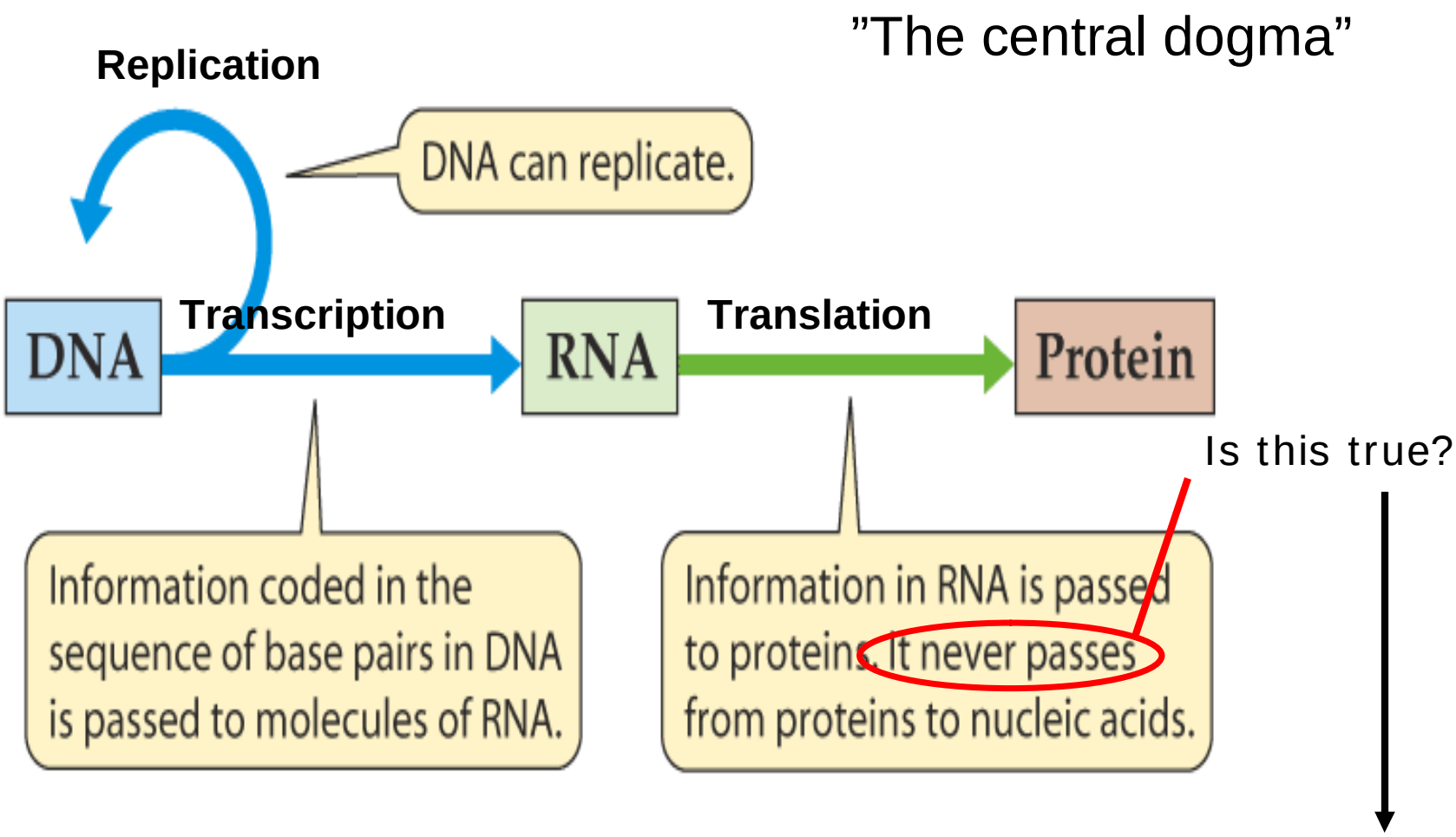
RNA

- RNA is similar to DNA chemically. It is usually only a single strand. T(hyamine) is replaced by U(racil)
- Several types of RNA exist for different functions in the cell.



tRNA linear and 3D view: http://www.cgl.ucsf.edu/home/glasfeld/tutorial/trna/trna.gif

DNA, RNA, and the Flow of Information



Denis Noble: The principles of Systems Biology illustrated using the virtual heart
http://velblod.videolectures.net/2007/pascal/eccs07_dresden/noble_denis/eccs07_noble_psb_01.ppt

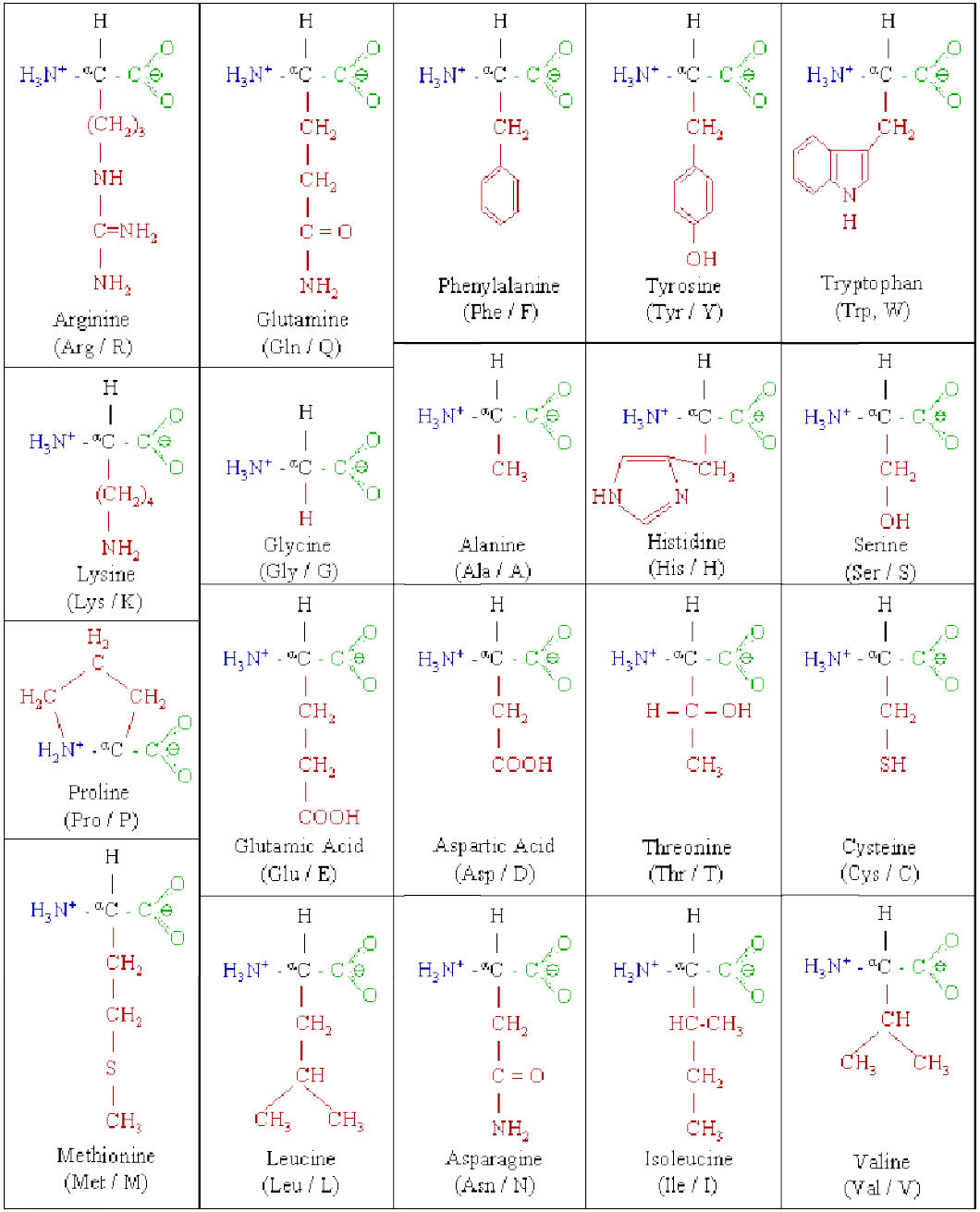
Proteins

- Proteins are polypeptides (strings of amino acid residues)
- Represented using strings of letters from an alphabet of 20: AEGLV…WKKLAG
- Typical length 50…1000 residues

Urease enzyme from Helicobacter pylori

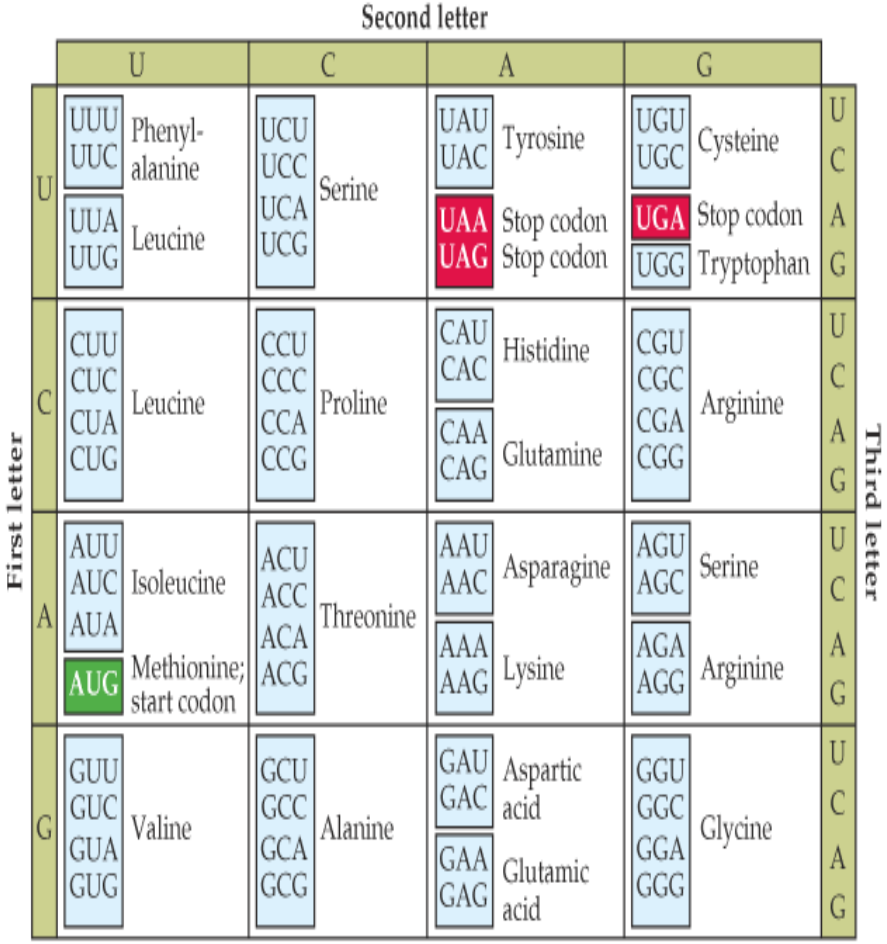
Amino acids

http://upload.wikimedia.org/wikipedia/commons/c/c5/Amino_acids_2.png



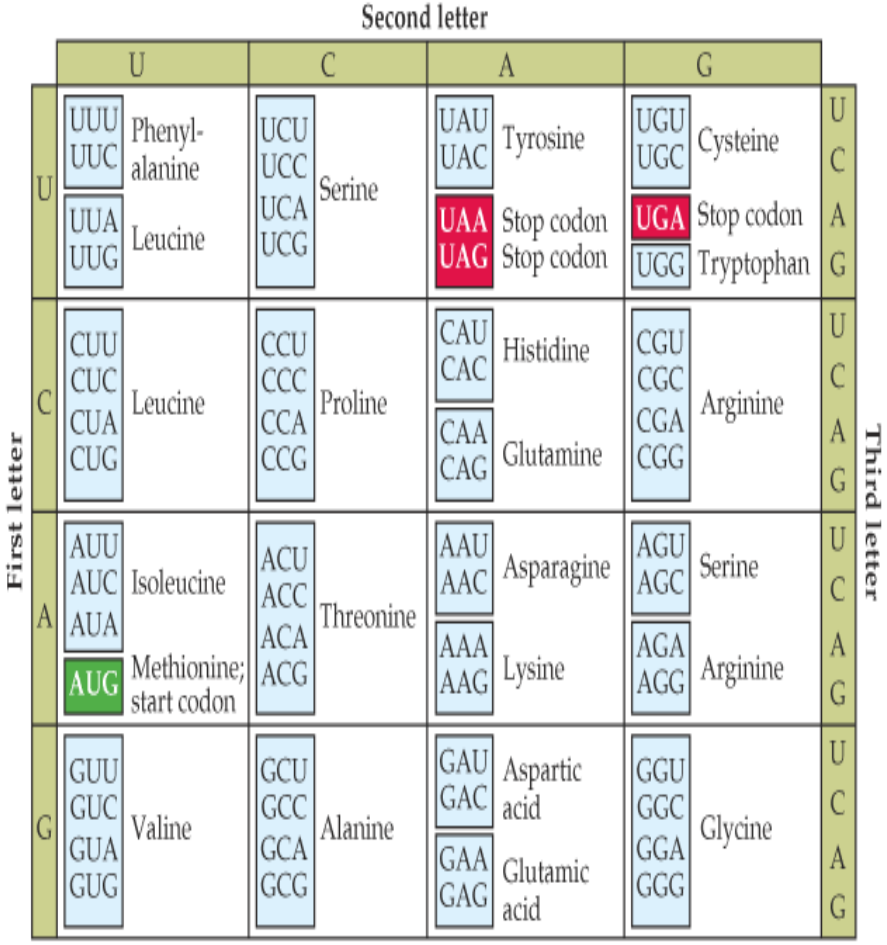
How DNA/RNA codes for protein?

- DNA alphabet contains four letters but must specify protein, or polypeptide sequence of 20 letters.
- Dinucleotides are not enough: $4^2 = 16$ possible dinucleotides
- Trinucleotides (triplets) allow $4^3 = 64$ possible trinucleotides
- Triplets are also called *codons*



How DNA/RNA codes for protein?

- Three of the possible triplets specify ”stop translation”
- Translation usually starts at triplet AUG (this codes for methionine)
- Most amino acids may be specified by more than triplet
- How to find a gene? Look for start and stop codons (not that easy though)



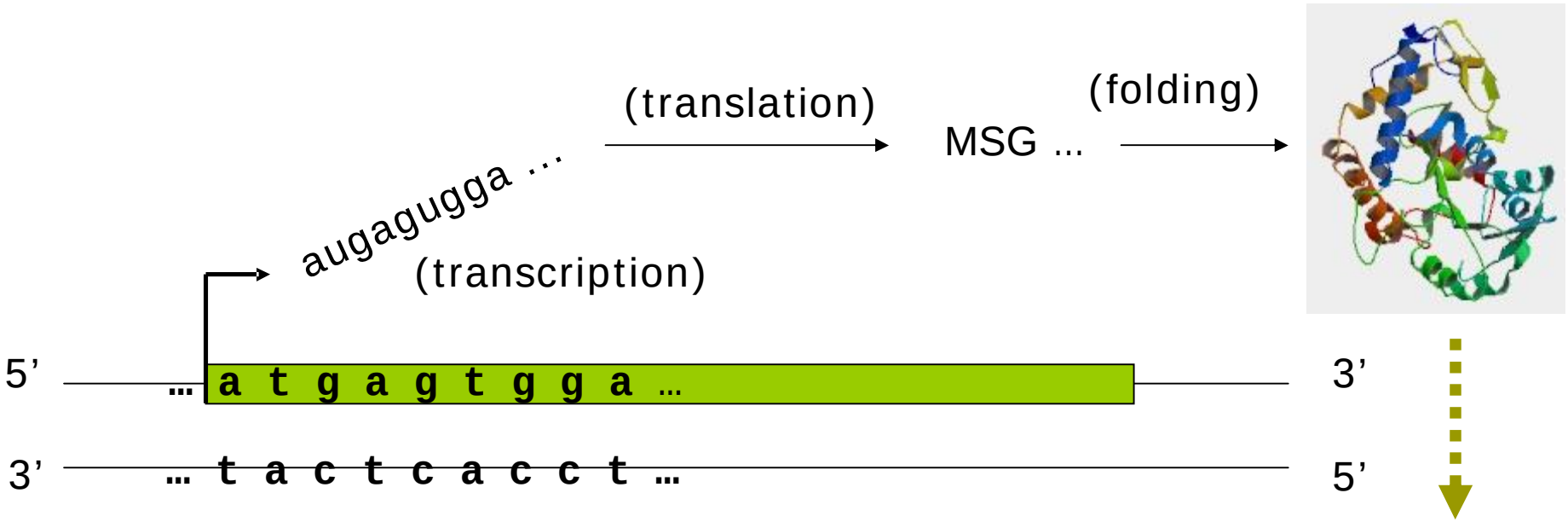
Proteins: Workhorses of the Cell

- 20 different amino acids
  - different chemical properties cause the protein chains to fold up into specific three-dimensional structures that define their particular functions in the cell.
- Proteins do all essential work for the cell
  - build cellular structures
  - digest nutrients
  - execute metabolic functions
  - mediate information flow within a cell and among cellular communities.
- Proteins work together with other proteins or nucleic acids as "molecular machines"
  - structures that fit together and function in highly specific, lock-and-key ways.

► Lecture 8: Proteomics

Genes

- "A gene is a union of genomic sequences encoding a coherent set of potentially overlapping functional products" --Gerstein et al.
- A DNA segment whose information is expressed either as an RNA molecule or protein



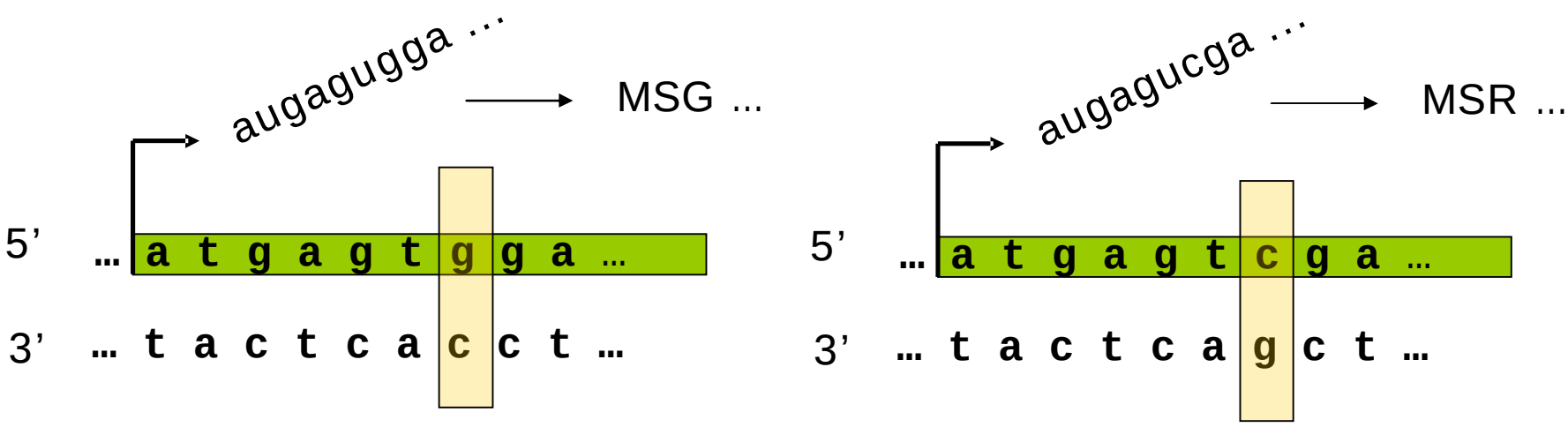
http://fold.it

FoldIt: Protein folding game

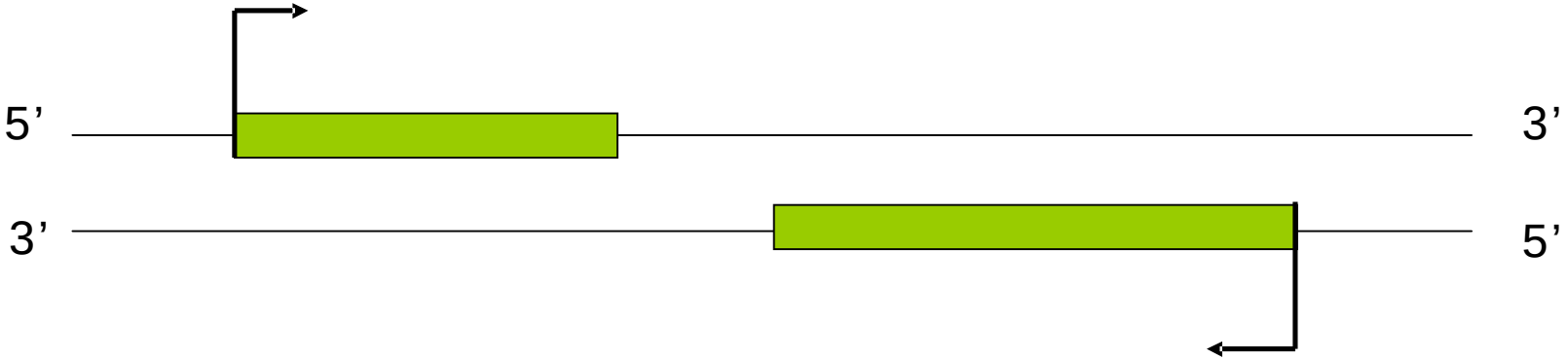
http://fold.it

Genes & alleles

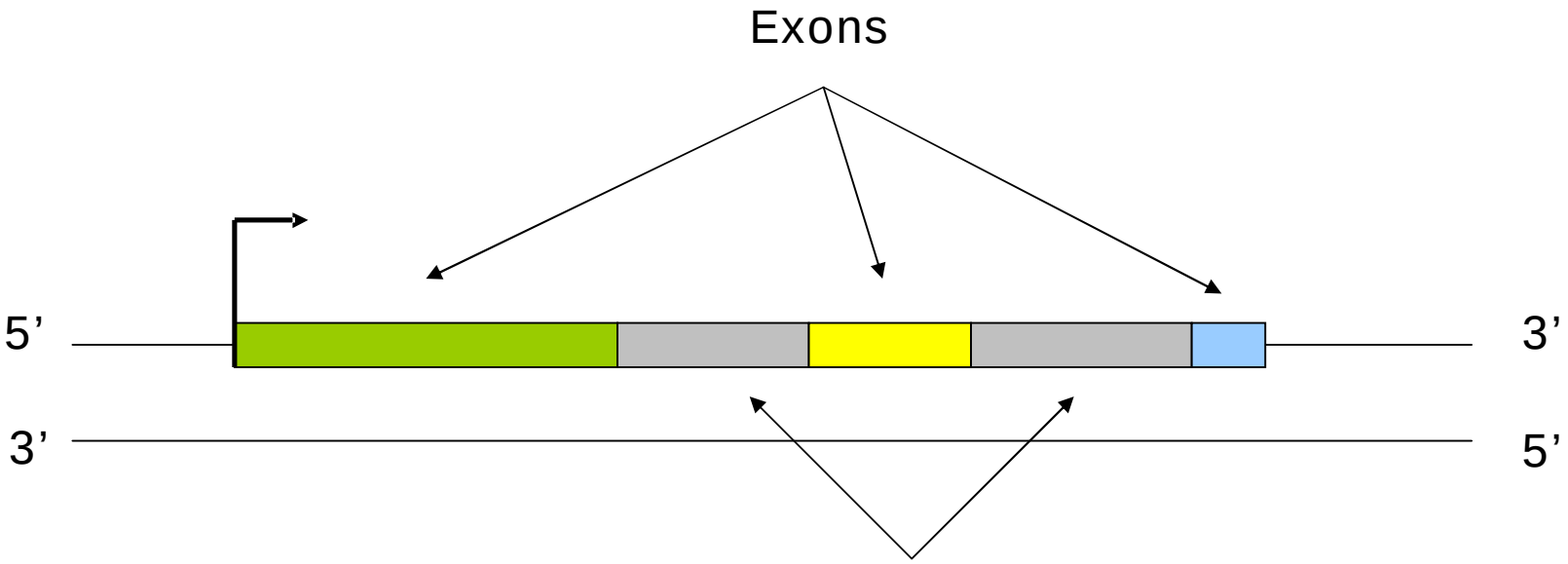
- A gene can have different variants
- The variants of the same gene are called *alleles*



Genes can be found on both strands



Exons and introns & splicing



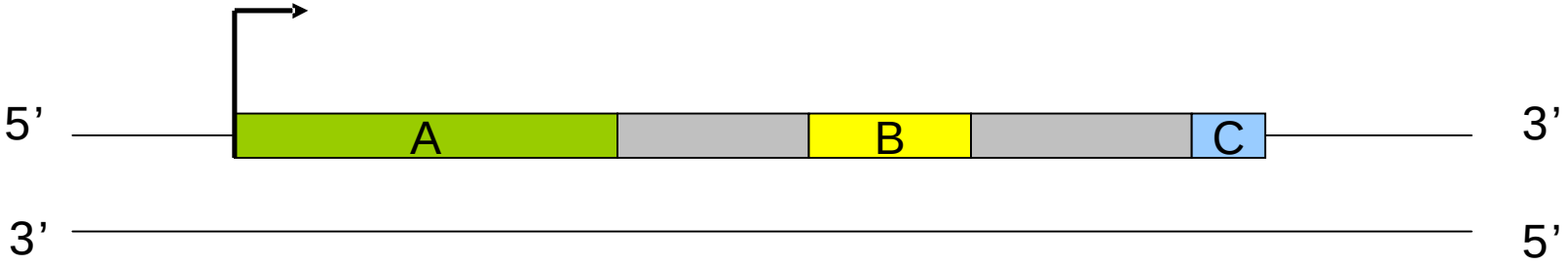
Introns are removed from RNA after transcription

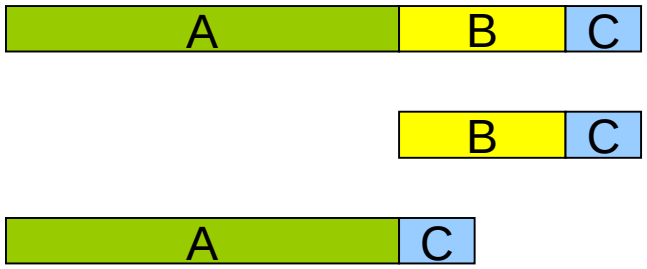
Exons are joined:

This process is called *splicing*

Alternative splicing

Different *splice variants* may be generated

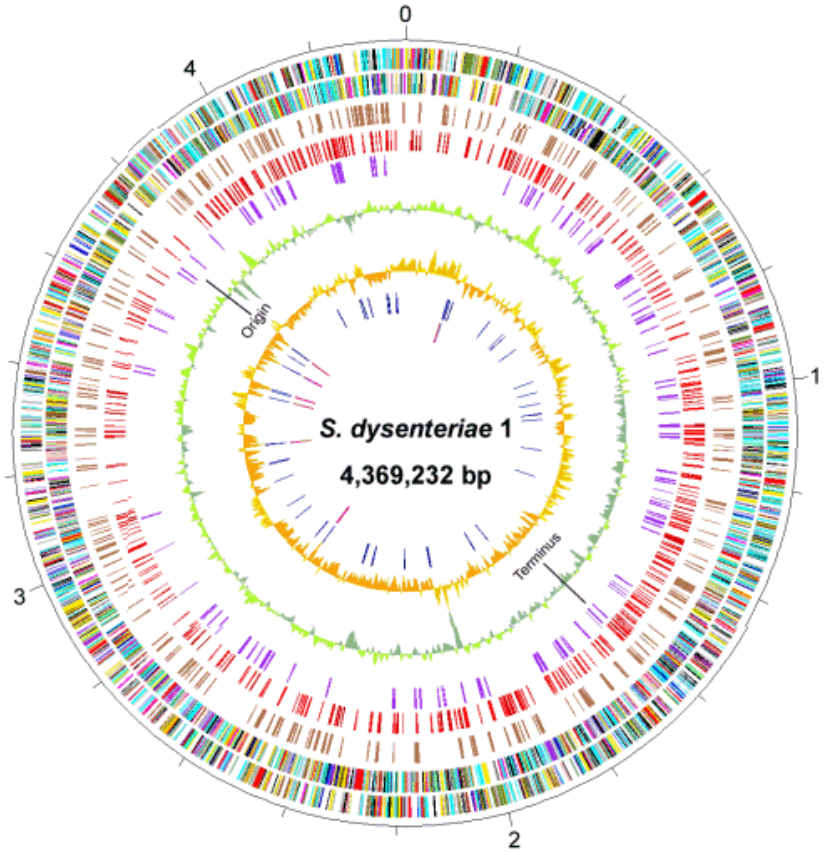




...

Where does the variation in genomes come from?

- Prokaryotes are typically haploid: they have a single (circular) chromosome
- DNA is usually inherited vertically (parent to daughter)
- Inheritance is clonal
  - Descendants are faithful copies of an ancestral DNA
  - Variation is introduced via mutations, transposable elements, and horizontal transfer of DNA



Chromosome map of *S. dysenteriae*, the nine rings describe different properties of the genome http://www.mgc.ac.cn/ShiBASE/circular_Sd197.htm

Causes of variation

- Mistakes in DNA replication
- Environmental agents (radiation, chemical agents)
- Transposable elements (transposons)
  - A part of DNA is moved or copied to another location in genome
- Horizontal transfer of DNA
  - Organism obtains genetic material from another organism that is not its parent
  - Utilized in genetic engineering

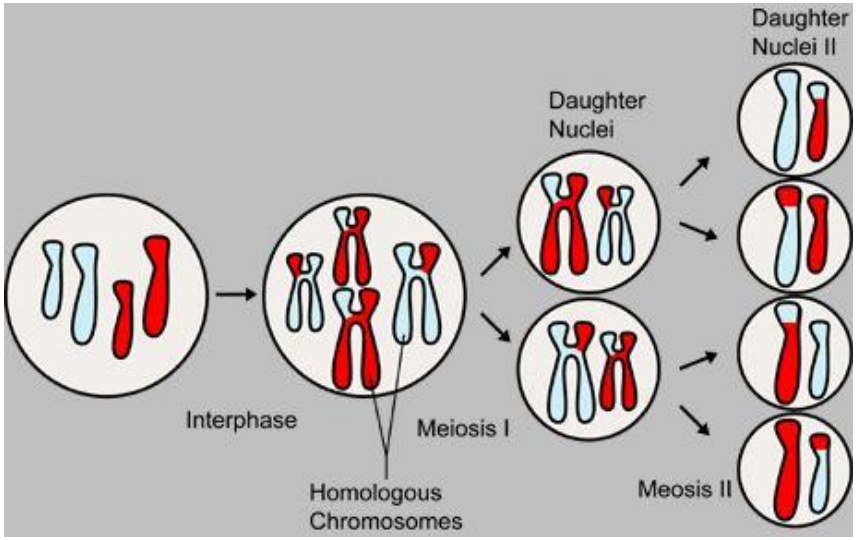
Biological string manipulation

- Point mutation: substitution of a base
  - ...ACGGCT... => ...ACGCCT...
- Deletion: removal of one or more contiguous bases (substring)
  - ...TTGATCA... => ...TTTCA...
- Insertion: insertion of a substring
  - ...GGCTAG... => ...GGTCAACTAG...

► Lecture: Sequence alignment
Lecture: Genome rearrangements

Meiosis

- Sexual organisms are usually diploid
  - Germline cells (gametes) contain N chromosomes
  - Somatic (body) cells have 2N chromosomes
- Meiosis: reduction of chromosome number from 2N to N during reproductive cycle
  - One chromosome doubling is followed by two cell divisions



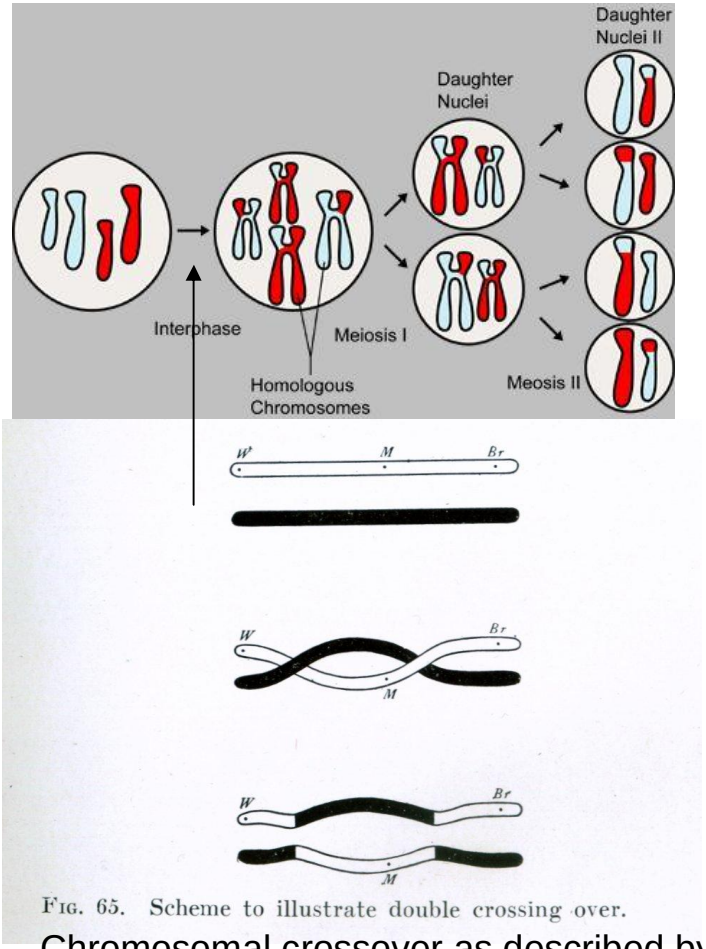
Major events in meiosis

http://en.wikipedia.org/wiki/Meiosis

http://www.ncbi.nlm.nih.gov/About/Primer

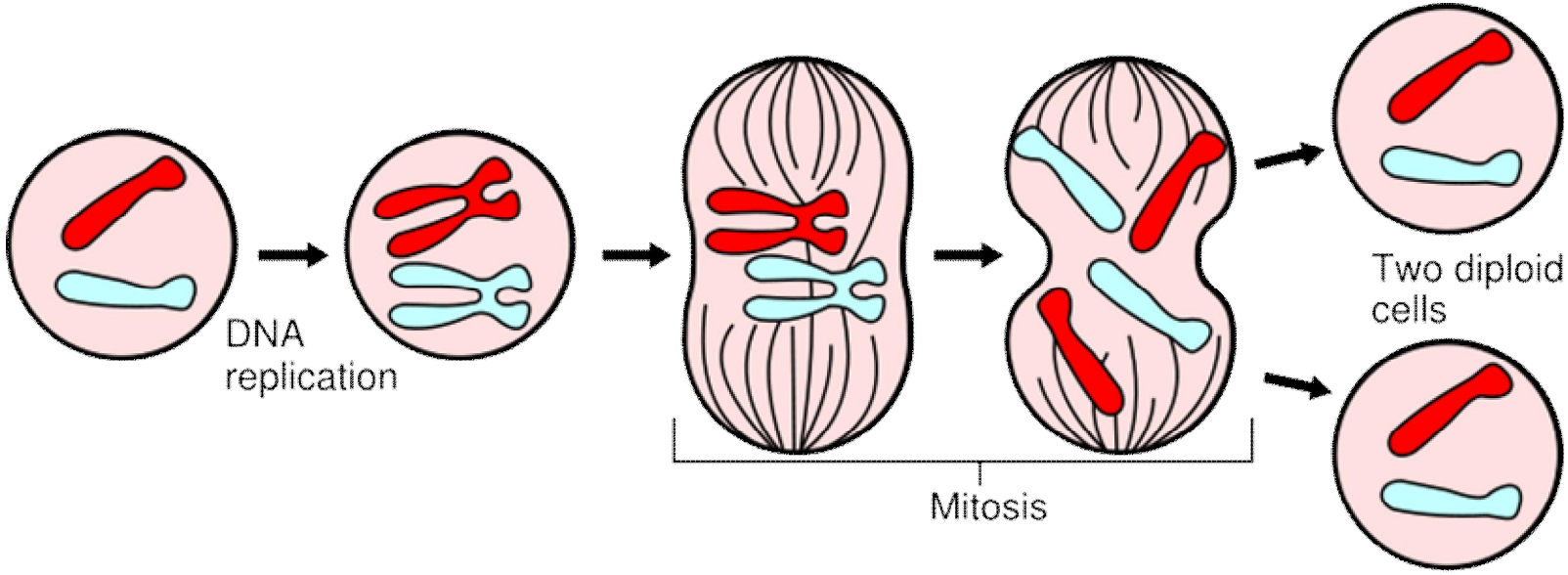
Recombination and variation

- Recap: Allele is a viable DNA coding occupying a given locus (position in the genome)
- In recombination, alleles from parents become suffled in offspring individuals via chromosomal crossover over
- Allele combinations in offspring are usually different from combinations found in parents
- Recombination errors lead into additional variations



Chromosomal crossover as described by T. H. Morgan in 1916

Mitosis



- Mitosis: growth and development of the organism
  - One chromosome doubling is followed by one cell division

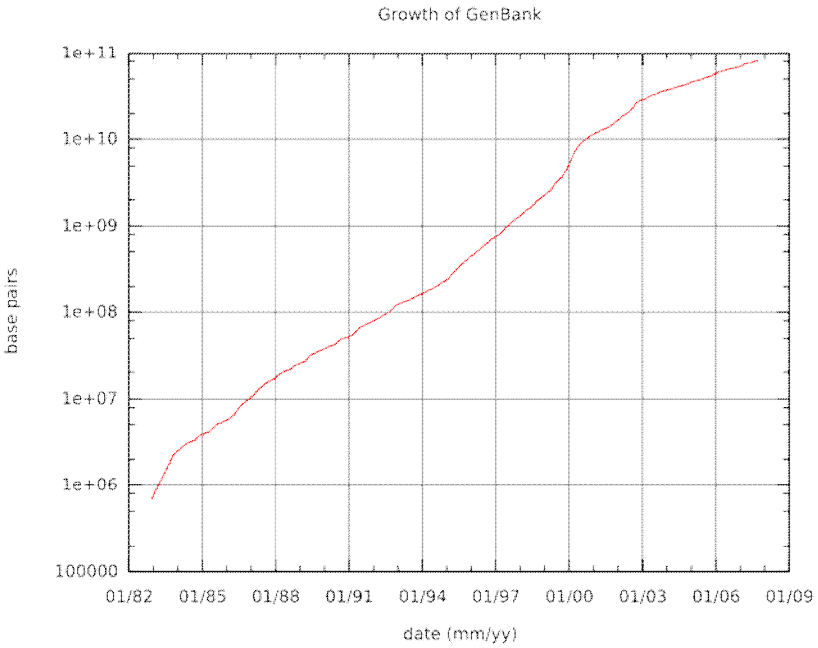
http://en.wikipedia.org/wiki/Image:Major_events_in_mitosis.svg

Recombination frequency and linked genes

- Genetic marker: some DNA sequence of interest (e.g., gene or a part of a gene)
- Recombination is more likely to separate two distant markers than two close ones
- Linked markers: ”tend” to be inherited together
- Marker distances measured in centimorgans: 1 centimorgan corresponds to 1% chance that two markers are separated in recombination

Biological databases

- Exponential growth of biological data
  - New measurement techniques
  - Before we are able to use the data, we need to store it efficiently -> biological databases
  - Published data is submitted to databases
- General vs specialised databases
- This topic is discussed extensively in *Practical course in biodatabases* (III period)

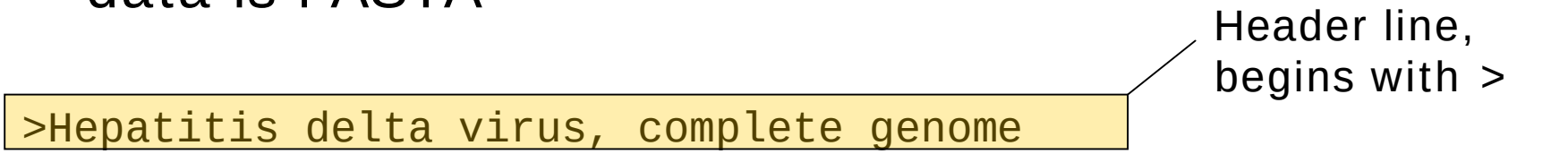


10 most important biodatabases… according to ”Bioinformatics for dummies”

GenBank/DDJB/EMBL	www.ncbi.nlm.nih.gov	Nucleotide sequences
Ensembl	www.ensembl.org	Human/mouse genome
PubMed	www.ncbi.nlm.nih.gov	Literature references
NR	www.ncbi.nlm.nih.gov	Protein sequences
UniProt	www.expasy.org	Protein sequences
InterPro	www.ebi.ac.uk	Protein domains
OMIM	www.ncbi.nlm.nih.gov	Genetic diseases
Enzymes	www.expasy.org	Enzymes
PDB	www.rcsb.org/pdb/	Protein structures
KEGG	www.genome.ad.jp	Metabolic pathways

FASTA format

- A simple format for DNA and protein sequence data is FASTA




```
>Hepatitis delta virus, complete genome
atgagccaagttccgaacaaggattcgcggggaggatagatcagcgcccgagaggggtga
gtcggtaaagagcattggaacgtcggagatacaactcccaagaaggaaaaaagagaaagc
aagaagcggatgaatttccccataacgccagtgaaactctaggaaggggaaagagggaag
gtggaagagaaggaggcgggcctcccgatccgaggggcccggcggccaagtttggaggac
actccggcccgaagggttgagagtaccccagagggaggaagccacacggagtagaacaga
gaaatcacctccagaggacccccttcagcgaacagagagcgcatcgcgagagggagtagac
catagcgataggaggggatgctaggagttgggggagaccgaagcgaggaggaaagcaaag
agagcagcggggctagcaggtgggtgttccgcccccgagaggggacgagtgaggcttat
cccggggaactcgacttatcgtccccacatagcagactcccggacccccctttcaaagtga
…
```

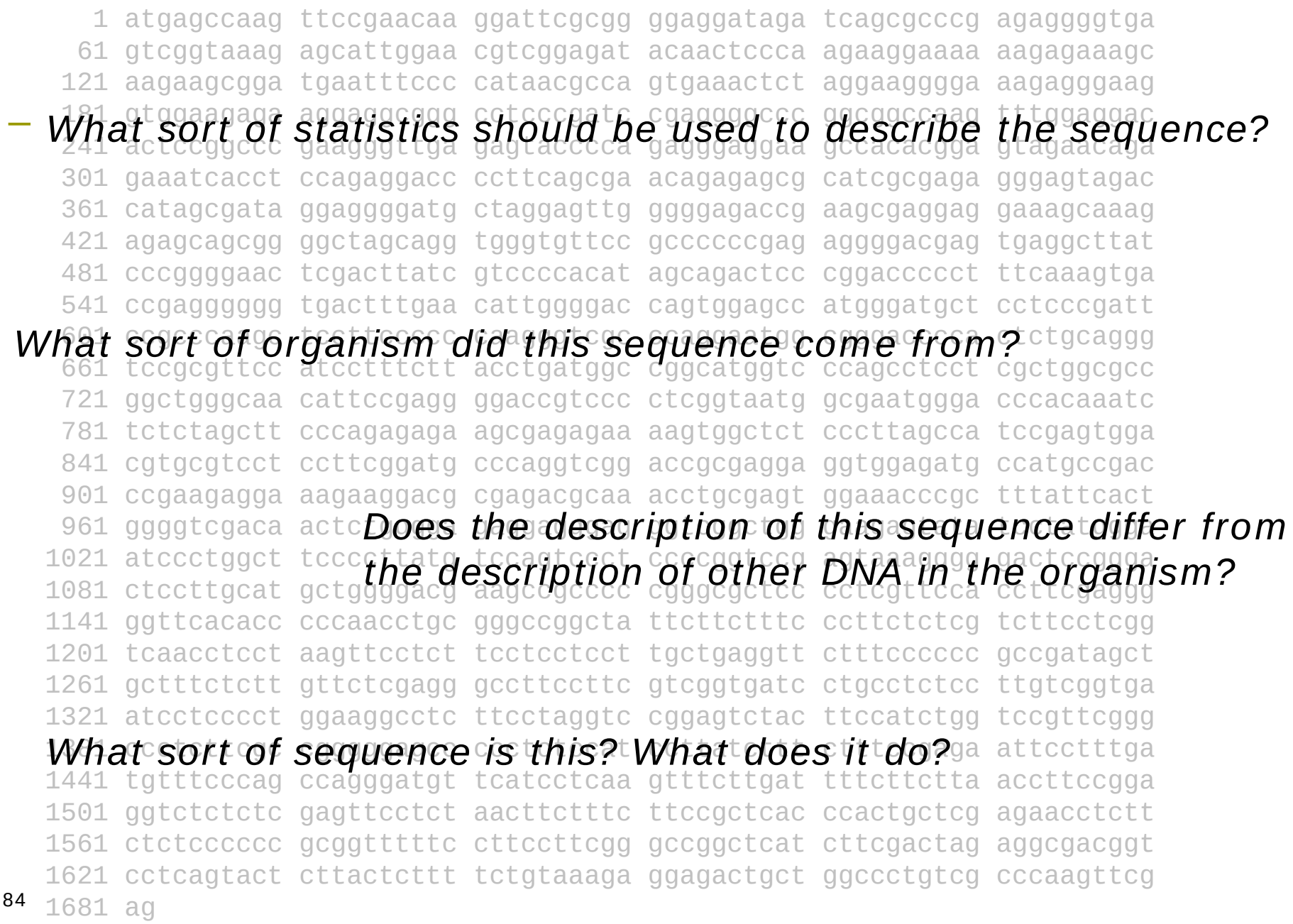
Introduction to Bioinformatics

Biological words

Recap

- DNA codes information with alphabet of 4 letters: A, C, G, T
- In proteins, the alphabet size is 20
- DNA -> RNA -> Protein (genetic code)
  - Three DNA bases (triplet, codon) code for one amino acid
  - Redundancy, start and stop codons

Given a DNA sequence, we might ask a number of questions



- *What sort of statistics should be used to describe the sequence?*

What sort of organism did this sequence come from?

Does the description of this sequence differ from the description of other DNA in the organism?

What sort of sequence is this? What does it do?

Biological words

- We can try to answer questions like these by considering the *words* in a sequence
- A *k*-word (or a *k-tuple*) is a string of length *k* drawn from some alphabet
- A DNA k-word is a string of length k that consists of letters A, C, G, T
  - 1-words: individual nucleotides (bases)
  - 2-words: dinucleotides (AA, AC, AG, AT, CA, …)
  - 3-words: codons (AAA, AAC, …)
  - 4-words and beyond

1-words: base composition

- Typically DNA exists as *duplex* molecule (two complementary strands)

5′-GGATCGAAGCTAAGGGCT-3′
3′-CCTAGCTTCGATTCCCGA-5′

Top strand: 7 G, 3 C, 5 A, 3 T
Bottom strand: 3 G, 7 C, 3 A, 5 T
Duplex molecule: 10 G, 10 C, 8 A, 8 T
Base frequencies: 10/36 10/36 8/36 8/36

These are something we can determine experimentally.

fr(G + C) = 20/36, fr(A + T) = 1 – fr(G + C) = 16/36

G+C content

- fr(G + C), or *G+C content* is a simple statistics for describing genomes
- Notice that one value is enough characterise fr(A), fr(C), fr(G) and fr(T) for duplex DNA
- Is G+C content (= base composition) able to tell the difference between genomes of different organisms?
  - Simple computational experiment, if we have the genome sequences under study (-> exercises)

G+C content and genome sizes (in megabasepairs, Mb) for various organisms

Mycoplasma genitalium	31.6%	0.585
Escherichia coli K-12	50.7%	4.693
Pseudomonas aeruginosa PAO1	66.4%	6.264
Pyrococcus abyssi	44.6%	1.765
Thermoplasma volcanium	39.9%	1.585
Caenorhabditis elegans	36%	97
Arabidopsis thaliana	35%	125
Homo sapiens	41%	3080

Base frequencies in duplex molecules

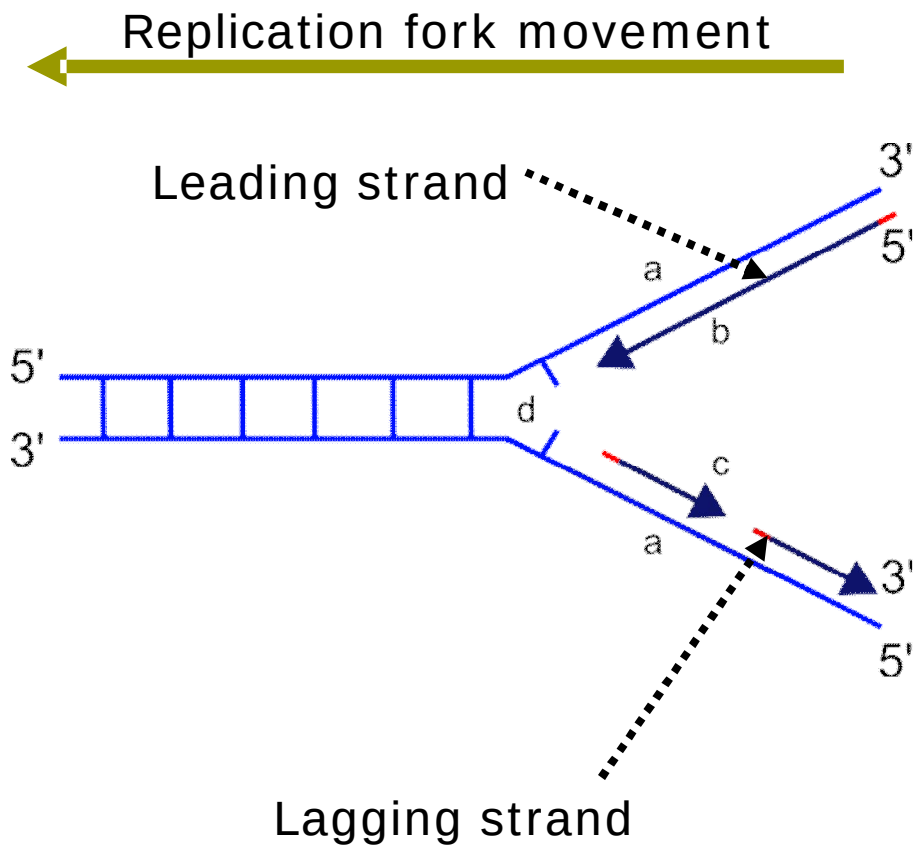
- Consider a DNA sequence generated randomly, with probability of each letter being independent of position in sequence
- You could expect to find a uniform distribution of bases in genomes…

```
5′-...GGATCGAAGCTAAGGGCT...-3′
3′-...CCTAGCTTCGATTCCCGA...-5′
```

- This is not, however, the case in genomes, especially in prokaryotes
  - This phenomena is called *GC skew*

DNA replication fork

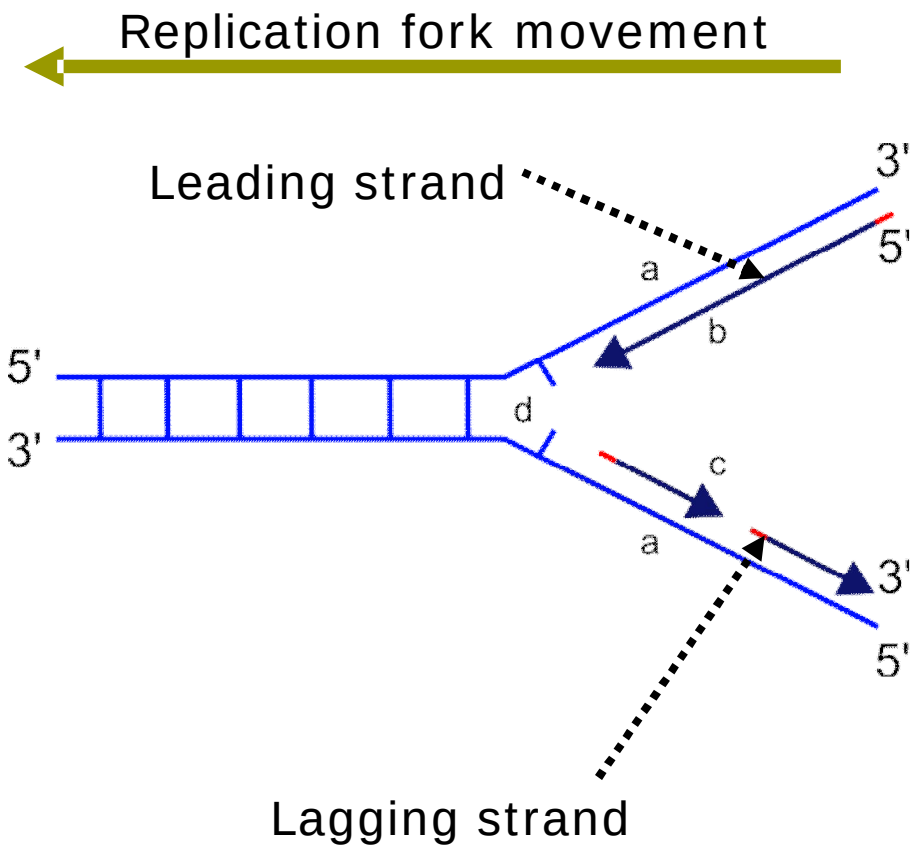
- When DNA is replicated, the molecule takes the *replication fork* form
- New complementary DNA is synthesised at both strands of the ”fork”
- New strand in 5’-3’ direction corresponding to replication fork movement is called *leading strand* and the other *lagging strand*



Replication fork

DNA replication fork

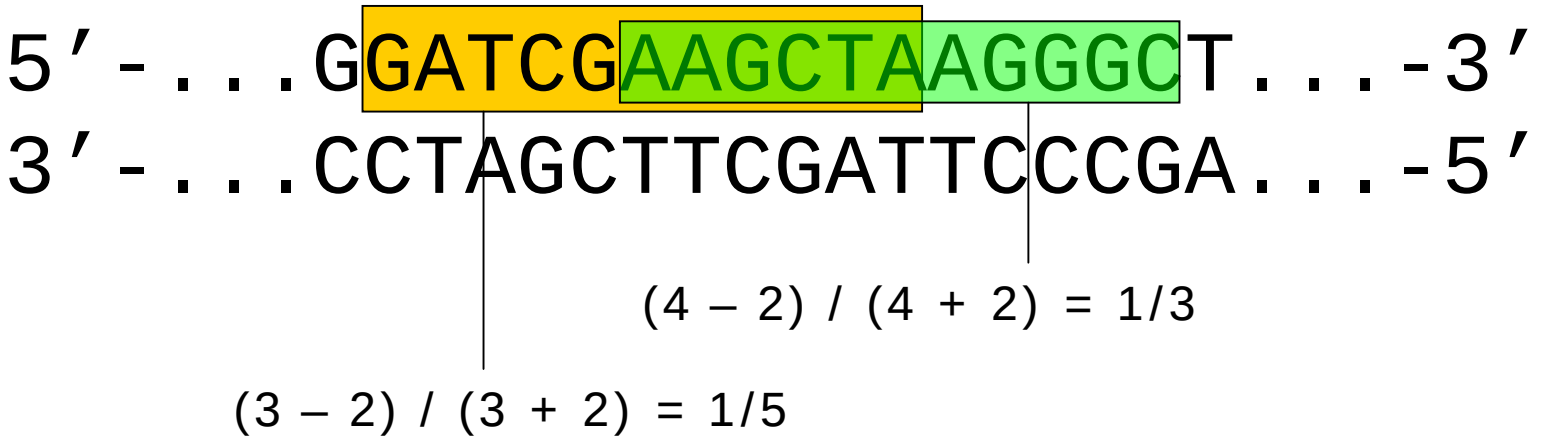
- This process has specific starting points in genome (*origins of replication)*
- Observation: Leading strands have an excess of G over C
- This can be described by *GC skew* statistics



Replication fork

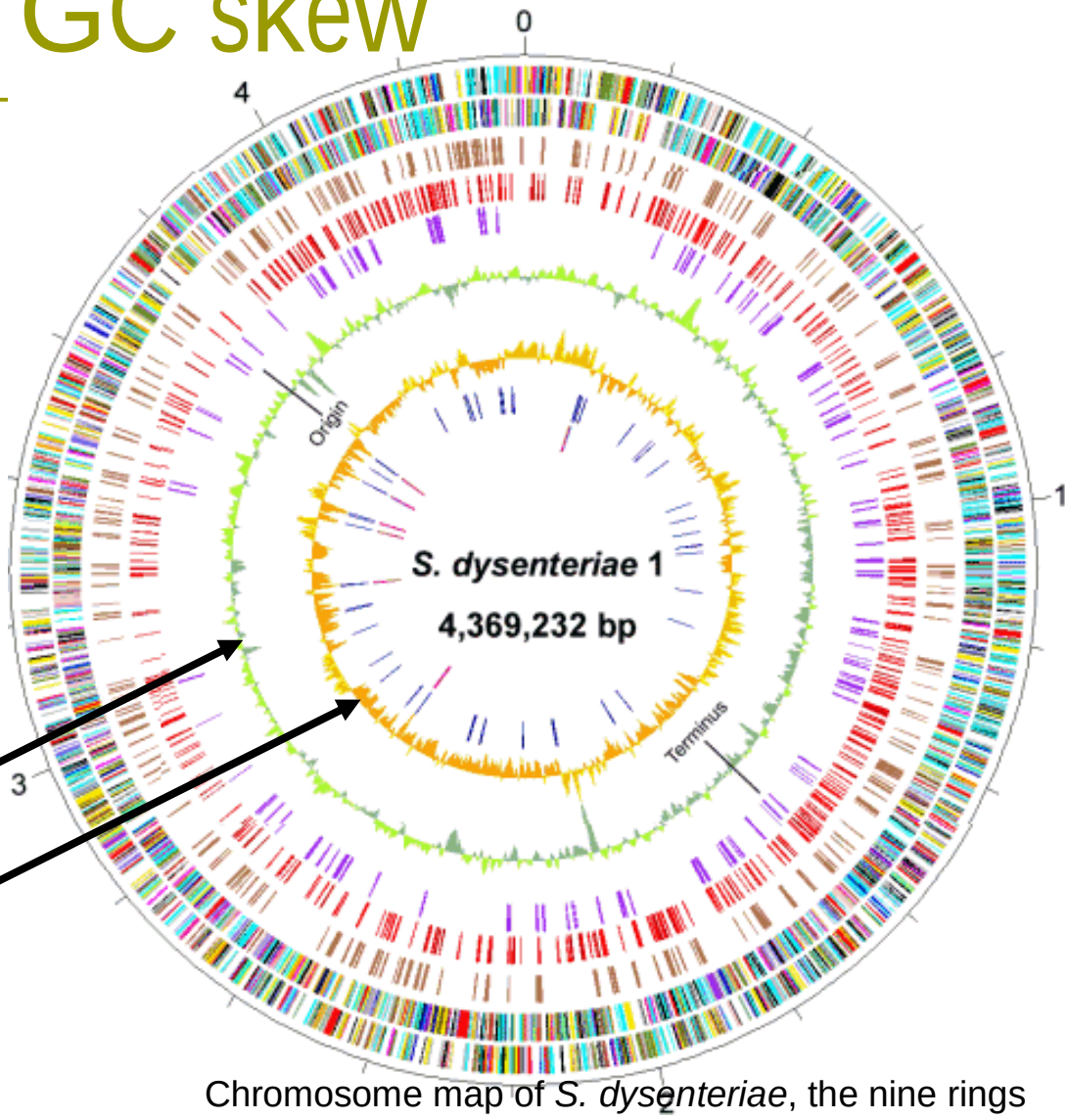
GC skew

- GC skew is defined as (#G - #C) / (#G + #C)
- It is calculated at successive positions in intervals (windows) of specific width



G-C content & GC skew

- G-C content & GC skew statistics can be displayed with a *circular genome map*



G+C content

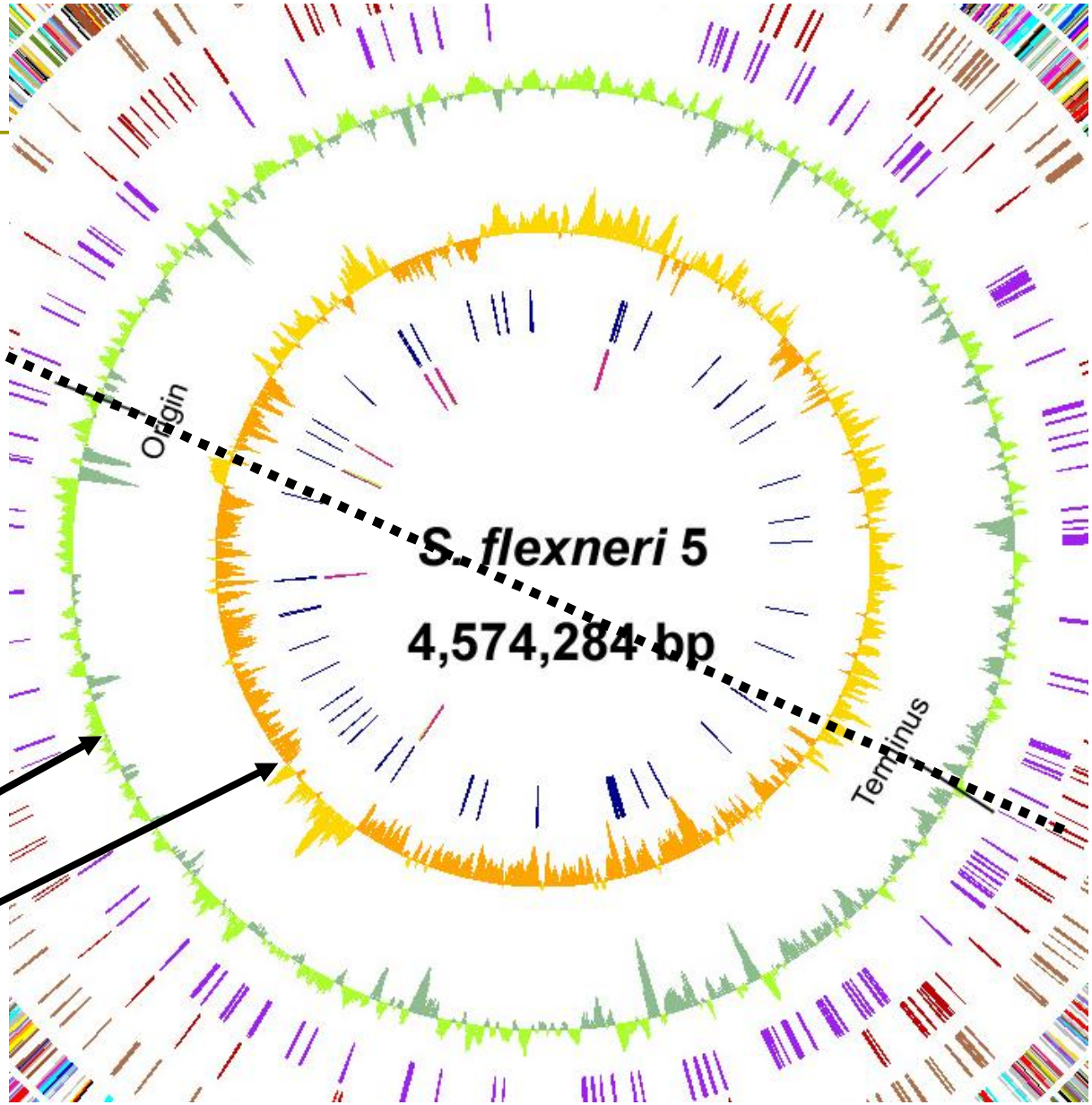
GC skew

(10kb window size)

Chromosome map of *S. dysenteriae*, the nine rings describe different properties of the genome
http://www.mgc.ac.cn/ShiBASE/circular_Sd197.htm

GC skew

- GC skew often changes sign at origins and termini of replication



G+C content

GC skew

(10kb window size)

Nie et al., BMC Genomics, 2006

2-words: dinucleotides

- Let's consider a sequence $L_1,L_2,\ldots,L_n$ where each letter L_i is drawn from the DNA alphabet {A, C, G, T}
- We have 16 possible dinucleotides $l_i l_{i+1}$: AA, AC, AG, …, TG, TT.

i.i.d. model for nucleotides

- Assume that bases
  - occur independently of each other
  - bases at each position are identically distributed
- Probability of the base A, C, G, T occuring is p_A, p_C, p_G, p_T, respectively
  - For example, we could use $p_A = p_C = p_G = p_T = 0.25$ or estimate the values from known genome data
- Probability of $l_i l_{i+1}$ is then $P_{li} P_{li+1}$
  - For example, $P(TG) = p_T p_G$

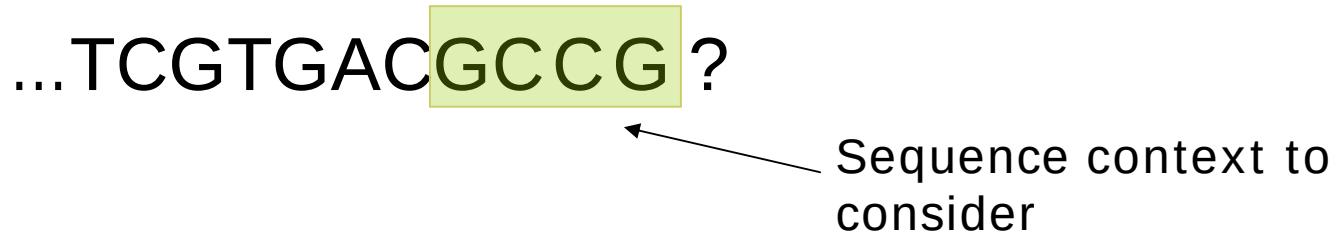
2-words: is what we see surprising?

- We can test whether a sequence is ”unexpected”, for example, with a χ^2 test
- Test statistic for a particular dinucleotide r_1r_2 is $\chi^2 = (O - E)^2 / E$ where
  - O is the observed number of dinucleotide r_1r_2
  - E is the expected number of dinucleotide r_1r_2
  - $E = (n - 1)p_{r1}p_{r2}$ under i.i.d. model
- Basic idea: high values of χ^2 indicate deviation from the model
  - Actual procedure is more detailed -> basic statistics courses

Refining the i.i.d. model

- i.i.d. model describes some organisms well (see Deonier's book) but fails to characterise many others

- We can refine the model by having the DNA letter at some position depend on letters at preceding positions

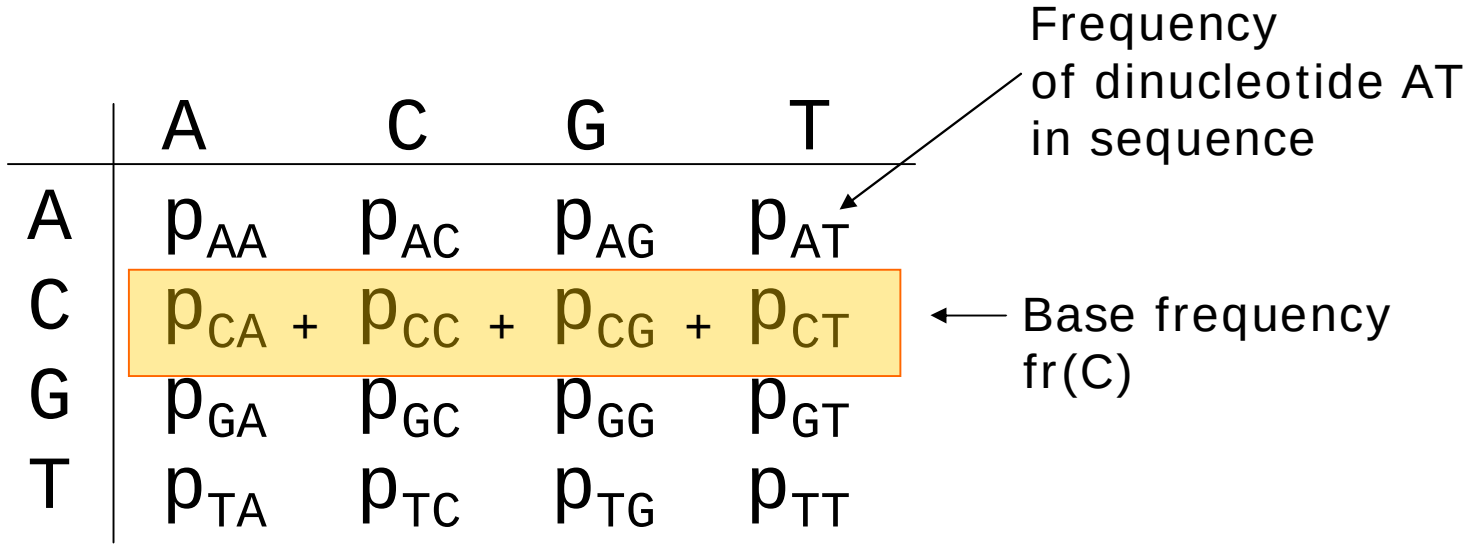


First-order Markov chains

- Lets assume that in sequence X the letter at position t, X_t, depends only on the previous letter X_{t-1} (*first-order markov chain*)
- Probability of letter j occuring at position t given X_{t-1} = i: $p_{ij} = P(X_t = j \mid X_{t-1} = i)$
- We consider *homogeneous* markov chains: probability p_{ij} is independent of position t

Estimating p_{ij}

- We can estimate probabilities p_{ij} (”the probability that j follows i”) from observed dinucleotide frequencies



	A	C	G	T
A	p_{AA}	p_{AC}	p_{AG}	p_{AT}
C	p_{CA} +	p_{CC} +	p_{CG} +	p_{CT}
G	p_{GA}	p_{GC}	p_{GG}	p_{GT}
T	p_{TA}	p_{TC}	p_{TG}	p_{TT}

…the values p_{AA}, p_{AC}, …, p_{TG}, p_{TT} sum to 1

Estimating p_{ij}

Dinucleotide frequency

- $p_{ij} = P(X_t = j \mid X_{t-1} = i) = \dfrac{P(X_t = j, X_{t-1} = i)}{P(X_{t-1} = i)}$

Probability of transition i -> j

Base frequency of nucleotide i, fr(i)

0.052 / 0.345 ≈ 0.151

	A	C	G	T
A	0.146	0.052	0.058	0.089
C	0.063	0.029	0.010	0.056
G	0.050	0.030	0.028	0.051
T	0.086	0.047	0.063	0.140

$P(X_t = j, X_{t-1} = i)$

	A	C	G	T
A	0.423	0.151	0.168	0.258
C	0.399	0.184	0.063	0.354
G	0.314	0.189	0.176	0.321
T	0.258	0.138	0.187	0.415

$P(X_t = j \mid X_{t-1} = i)$

Simulating a DNA sequence

- From a transition matrix, it is easy to generate a DNA sequence of length n:
  - First, choose the starting base randomly according to the base frequency distribution
  - Then, choose next base according to the distribution $P(x_t \mid x_{t-1})$ until n bases have been chosen

T T C T T C A A

Look for R code in Deonier's book

	A	C	G	T
A	0.423	0.151	0.168	0.258
C	0.399	0.184	0.063	0.354
G	0.314	0.189	0.176	0.321
T	0.258	0.138	0.187	0.415

$P(X_t = j \mid X_{t-1} = i)$

Simulating a DNA sequence

- Now we can quickly generate sequences of arbitrary length...

```
ttcttcaaaataaggatagtgattcttattggcttaagggataacaatttagatctttttcatgaatcatgtatgtcaacgttaaaagttgaactgcaataagttc
ttacacacgattgtttatctgcgtgcgaagcatttcactacatttgccgatgcagccaaaagtatttaacatttggtaaacaaattgacttaaatcgcgcacttaga
gtttgacgtttcatagttgatgcgtgtctaacaattacttttagtttttaaatgcgtttgtctacaatcattaatcagctctggaaaaacattaatgcatttaaac
cacaatggataattagttacttattttaaaattcacaaagtaattattcgaatagtgccctaagagagtactggggttaatggcaaagaaaattactgtagtgaaga
ttaagcctgttattatcacctgggtactctggtgaatgcacataagcaaatgctacttcagtgtcaaagcaaaaaaatttactgataggactaaaaacccttttattt
ttagaatttgtaaaaatgtgacctcttgcttataacatcatatttattgggtcgttctaggacactgtgattgccttctaactcttatttagcaaaaaattgtcata
gctttgaggtcagacaaacaagtgaatggaagacagaaaaagctcagcctagaattagcatgttttgagtggggaattacttggttaactaaagtgttcatgactgt
tcagcatatgattgttggtgagcactacaaagatagaagagttaaactaggtagtggtgatttcgctaacacagttttcatacaagttctattttctcaatggtttt
ggataagaaaacagcaaacaaatttagtattatttcctagtaaaaagcaaacatcaaggagaaattggaagctgcttgttcagtttgcattaaattaaaaatttat
ttgaagtattcgagcaatgttgacagtctgcgttcttcaaataagcagcaaatcccctcaaaattgggcaaaaacctaccctggcttcttttaaaaaaccaagaaa
agtcctatataagcaacaaatttcaaacctttgttaaaaattctgctgctgaataaataggcattacagcaatgcaattaggtgcaaaaaaggccatcctctttct
ttttttgtacaattgttcaagcaactttgaatttgcagattttaacccactgtctatatgggacttcgaattaaattgactggtctgcatcacaaatttcaactgcc
caatgtaatcatattctagagtattaaaaatacaaaaagtacaattagttatgcccattggcctggcaatttatttactccactttccacgttttggggatatttta
acttgaatagttcacaatcaaaacataggaaggatctactgctaaaagcaaaagcgtattggaatgataaaaaactttgatgtttaaaaaactacaaccttaatgaa
ttaaagttgaaaaaatattcaaaaaaagaaattcagttcttggcgagtaatatttttgatgtttgagatcagggttacaaaataagtgcatgagattaactcttcaa
atataaactgatttaagtgtatttgctaataacattttcgaaaaggaatattatggtaagaattcataaaaatgtttaatactgatacaactttcttttatatcctc
catttggccagaatactgttgcacacaactaattggaaaaaaaatagaacgggtcaatctcagtgggaggagaagaaaaaagttggtgcaggaaatagtttctacta
acctggtataaaaacatcaagtaacattcaaattgcaaatgaaaactaaccgatctaagcattgattgatttttctcatgcctttcgcctagttttaataaacgcgc
cccaactctcatcttcggttcaaatgatctattgtatttatgcactaacgtgcttttatgttagcatttttcaccctgaagttccgagtcattggcgtcactcacaa
atgacattacaatttttctatgttttgttctgttgagtcaaagtgcatgcctacaattctttcttatatagaactagacaaaatagaaaaaggcacttttggagtct
gaatgtcccttagtttcaaaaaggaaattgttgaatttttgtggttagttaaattttgaacaaactagtatagtggtgacaaacgatcaccttgagtcggtgacta
taaaagaaaaaggagattaaaaatacctgcggtgccacatttttgttacgggcatttaaggtttgcatgtgttgagcaattgaaacctacaactcaataagtcatg
ttaagtcacttctttgaaaaaaaaaaagacccttaagcaagctc
```

Simulating a DNA sequence

Dinucleotide frequencies

	Simulated	Observed
aa	0.145	0.146
ac	0.050	0.052
ag	0.055	0.058
at	0.092	0.089
ca	0.065	0.063
cc	0.028	0.029
cg	0.011	0.010
ct	0.058	0.056
ga	0.048	0.050
gc	0.032	0.030
gg	0.029	0.028
gt	0.050	0.051
ta	0.084	0.086
tc	0.052	0.047
tg	0.064	0.063
tt	0.138	0.0140

n = 10000

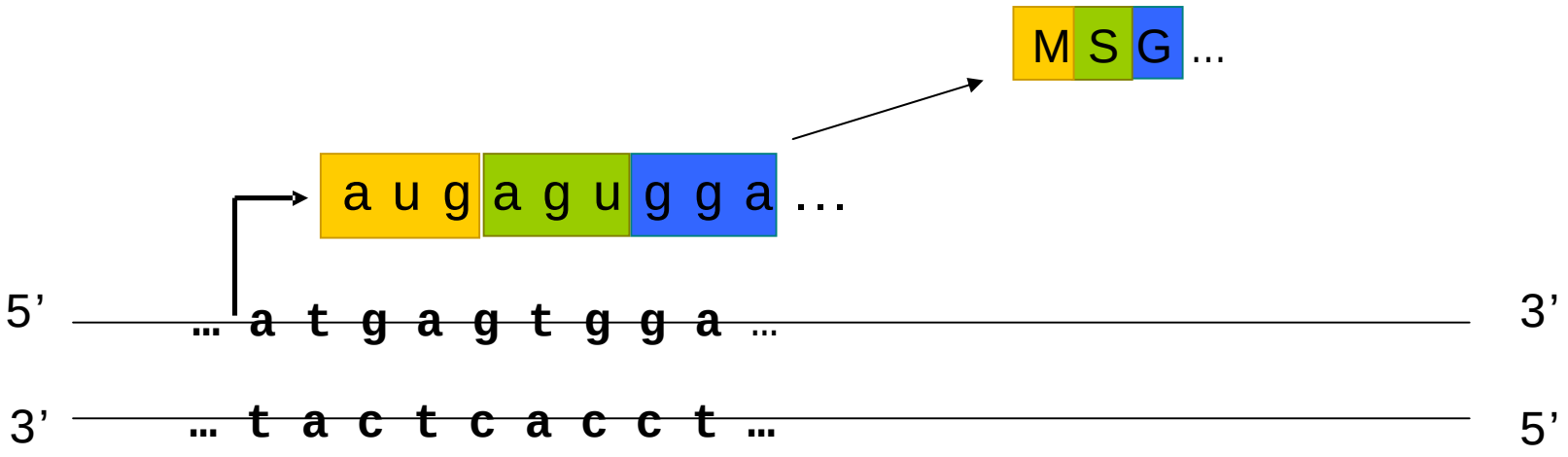
Simulating a DNA sequence

- The model is able to generate correct proportions of 1- and 2-words in genomes...
- ...but fails with k=3 and beyond.

```
ttcttcaaaataaggatagtgattcttattggcttaagggataacaatttagatctttttcatgaatcatgtatgtcaacgttaaaagttgaactgcaataagttc
ttacacacgattgtttatctgcgtgcgaagcatttcactacatttgccgatgcagccaaaagtatttaacatttggtaaacaaattgacttaaatcgcgcacttaga
gtttgacgtttcatagttgatgcgtgtctaacaattacttttagtttttaaatgcgtttgtctacaatcattaatcagctctggaaaaacattaatgcatttaaac
cacaatggataattagttacttattttaaaattcacaaagtaattattcgaatagtgccctaagagagtactggggttaatggcaaagaaaattactgtagtgaaga
ttaagcctgttattatcacctgggtactctggtgaatgcacataagcaaatgctacttcagtgtcaaagcaaaaaaatttactgataggactaaaaacccttttattt
ttagaatttgtaaaaatgtgacctcttgcttataacatcatatttattgggtcgttctaggacactgtgattgccttctaactcttatttagcaaaaaattgtcata
gctttgaggtcagacaaacaagtgaatggaagacagaaaaagctcagcctagaattagcatgttttgagtggggaattacttggttaactaaagtgttcatgactgt
tcagcatatgattgttggtgagcactacaaagatagaagagttaaactaggtagtggtgatttcgctaacacagttttcatacaagttctattttctcaatggtttt
ggataagaaaacagcaaacaaatttagtattatttcctagtaaaaagcaaacatcaaggagaaattggaagctgcttgttcagtttgcattaaattaaaaatttat
ttgaagtattcgagcaatgttgacagtctgcgttcttcaaataagcagcaaatcccctcaaaattgggcaaaaacctaccctggcttctttttaaaaaaccaagaaa
agtcctatataagcaacaaatttcaaacctttgttaaaaattctgctgctgaataaataggcattacagcaatgcaattaggtgcaaaaaaggccatcctctttct
ttttttgtacaattgttcaagcaactttgaatttgcagattttaacccactgtctatatgggacttcgaattaaattgactggtctgcatcacaaatttcaactgcc
caatgtaatcatattctagagtattaaaaatacaaaaagtacaattagttatgcccattggcctggcaatttatttactccactttccacgttttggggatatttta
acttgaatagttcacaatcaaaacataggaaggatctactgctaaaagcaaaagcgtattggaatgataaaaaactttgatgtttaaaaaactacaaccttaatgaa
ttaaagttgaaaaaatattcaaaaaaagaaattcagttcttggcgagtaatatttttgatgtttgagatcagggttacaaaataagtgcatgagattaactcttcaa
atataaactgatttaagtgtatttgctaataacattttcgaaaaggaatattatggtaagaattcataaaaatgtttaatactgatacaactttcttttatatcctc
catttggccagaatactgttgcacacaactaattggaaaaaaaatagaacgggtcaatctcagtgggaggagaagaaaaaagttggtgcaggaaatagtttctacta
acctggtataaaaacatcaagtaacattcaaattgcaaatgaaaactaaccgatctaagcattgattgatttttctcatgcctttcgcctagttttaataaacgcgc
cccaactctcatcttcggttcaaatgatctattgtatttatgcactaacgtgcttttatgttagcatttttcaccctgaagttccgagtcattggcgtcactcacaa
atgacattacaatttttctatgttttgttctgttgagtcaaagtgcatgcctacaattctttcttatatagaactagacaaaatagaaaaaggcacttttggagtct
gaatgtcccttagtttcaaaaaggaaattgttgaatttttgtggttagttaaatttgaacaaactagtatagtggtgacaaacgatcaccttgagtcggtgacta
taaaagaaaaaggagattaaaaatacctgcggtgccacatttttgttacgggcatttaaggtttgcatgtgttgagcaattgaaacctacaactcaataagtcatg
ttaagtcacttctttgaaaaaaaaaaagacccttaagcaagctc
```

3-words: codons

- We can extend the previous method to 3-words
- k=3 is an important case in study of DNA sequences because of genetic code



3-word probabilities

- Let's again assume a sequence L of independent bases
- Probability of 3-word $r_1r_2r_3$ at position $i,i+1,i+2$ in sequence L is

$$P(L_i = r_1, L_{i+1} = r_2, L_{i+2} = r_3) =$$
$$P(L_i = r_1)P(L_{i+1} = r_2)P(L_{i+2} = r_3)$$

3-words in Escherichia coli genome

Word	Count	Observed	Expected	Word	Count	Observed	Expected
AAA	108924	**0.02348**	**0.01492**	CAA	76614	0.01651	0.01541
AAC	82582	0.01780	0.01541	CAC	66751	0.01439	0.01591
AAG	63369	0.01366	0.01537	CAG	104799	0.02259	0.01588
AAT	82995	0.01789	0.01490	CAT	76985	0.01659	0.01539
ACA	58637	0.01264	0.01541	CCA	86436	0.01863	0.01591
ACC	74897	0.01614	0.01591	CCC	47775	0.01030	0.01643
ACG	73263	0.01579	0.01588	CCG	87036	0.01876	0.01640
ACT	49865	0.01075	0.01539	CCT	50426	0.01087	0.01589
AGA	56621	0.01220	0.01537	CGA	70938	0.01529	0.01588
AGC	80860	0.01743	0.01588	CGC	115695	**0.02494**	**0.01640**
AGG	50624	0.01091	0.01584	CGG	86877	0.01872	0.01636
AGT	49772	0.01073	0.01536	CGT	73160	0.01577	0.01586
ATA	63697	0.01373	0.01490	CTA	26764	**0.00577**	**0.01539**
ATC	86486	0.01864	0.01539	CTC	42733	0.00921	0.01589
ATG	76238	0.01643	0.01536	CTG	102909	0.02218	0.01586
ATT	83398	0.01797	0.01489	CTT	63655	0.01372	0.01537

3-words in Escherichia coli genome

Word	Count	Observed	Expected	Word	Count	Observed	Expected
GAA	83494	0.01800	0.01537	TAA	68838	0.01484	0.01490
GAC	54737	0.01180	0.01588	TAC	52592	0.01134	0.01539
GAG	42465	0.00915	0.01584	*TAG*	27243	**0.00587**	**0.01536**
GAT	86551	0.01865	0.01536	TAT	63288	0.01364	0.01489
GCA	96028	0.02070	0.01588	TCA	84048	0.01812	0.01539
GCC	92973	0.02004	0.01640	TCC	56028	0.01208	0.01589
GCG	114632	**0.02471**	**0.01636**	TCG	71739	0.01546	0.01586
GCT	80298	0.01731	0.01586	TCT	55472	0.01196	0.01537
GGA	56197	0.01211	0.01584	TGA	83491	0.01800	0.01536
GGC	92144	0.01986	0.01636	TGC	95232	0.02053	0.01586
GGG	47495	0.01024	0.01632	TGG	85141	0.01835	0.01582
GGT	74301	0.01601	0.01582	TGT	58375	0.01258	0.01534
GTA	52672	0.01135	0.01536	TTA	68828	0.01483	0.01489
GTC	54221	0.01169	0.01586	TTC	83848	0.01807	0.01537
GTG	66117	0.01425	0.01582	TTG	76975	0.01659	0.01534
GTT	82598	0.01780	0.01534	TTT	109831	0.02367	0.01487

2nd order Markov Chains

- Markov chains readily generalise to higher orders
- In 2nd order markov chain, position t depends on positions t-1 and t-2
- Transition matrix:

	A	C	G	T
AA				
AC				
AG				
AT				
CA				
…				

- Probabilistic models for DNA and amino acid sequences will be discussed in Biological sequence analysis course (II period)

Codon Adaptation Index (CAI)

- Observation: cells prefer certain codons over others in highly expressed genes
  - Gene expression: DNA is transcribed into RNA (and possibly translated into protein)

Amino acid	Codon	Predicted	Gene class I	Gene class II
Phe	TTT	0.493	0.551	0.291
	TTC	0.507	0.449	0.709
Ala	GCT	0.246	0.145	0.275
	GCC	0.254	0.276	0.164
	GCA	0.246	0.196	0.240
	GCG	0.254	0.382	0.323
Asn	AAT	0.493	0.409	0.172
	AAC	0.507	0.591	0.828

Moderately expressed

Highly expressed

Codon frequencies for some genes in E. coli

Codon Adaptation Index (CAI)

- CAI is a statistic used to compare the distribution of codons observed with the preferred codons for highly expressed genes

Codon Adaptation Index (CAI)

- Consider an amino acid sequence $X = x_1x_2...x_n$
- Let p_k be the probability that codon k is used in highly expressed genes
- Let q_k be the highest probability that a codon coding for the same amino acid as codon k has
  - For example, if codon k is ”GCC”, the corresponding amino acid is Alanine (see genetic code table; also GCT, GCA, GCG code for Alanine)
  - Assume that $p_{GCC} = 0.164$, $p_{GCT} = 0.275$, $p_{GCA} = 0.240$, $p_{GCG} = 0.323$
  - Now $q_{GCC} = q_{GCT} = q_{GCA} = q_{GCG} = 0.323$

Codon Adaptation Index (CAI)

- CAI is defined as

$$CAI = \left(\prod_{k=1}^{n} p_k / q_k\right)^{1/n}$$

- CAI can be given also in *log-odds* form:

$$\log(CAI) = (1/n) \sum_{k=1}^{n} \log(p_k / q_k)$$

CAI: example with an E. coli gene

q_k
p_k

M	A	L	T	K	A	E	M	S	E	Y	L	…
ATG	GCG	CTT	ACA	AAA	GCT	GAA	ATG	TCA	GAA	TAT	CTG	
1.00	**0.47**	0.02	0.45	**0.80**	**0.47**	**0.79**	**1.00**	**0.43**	**0.79**	0.19	0.02	
	0.06	0.02	**0.47**	0.20	0.06	0.21		0.32	0.21	**0.81**	0.02	
	0.28	0.04	0.04		0.28			0.03			0.04	
	0.20	0.03	0.05		0.20			0.01			0.03	
		0.01						0.04			0.01	
		0.89						0.18			**0.89**	
ATG	**GCT**	TTA	ACT	**AAA**	**GCT**	**GAA**	**ATG**	**TCT**	**GAA**	TAT	TTA	
	GCC	TTG	**ACC**	AAG	GCC	GAG		TCC	GAG	**TAC**	TTG	
	GCA	CTT	ACA		GCA			TCA			CTT	
	GCG	CTC	ACG		GCG			TCG			CTC	
		CTA						AGT			CTA	
		CTG						AGC			**CTG**	
1.00	0.20	0.04	0.04	0.80	0.47	0.79	1.00	0.03	0.79	0.19	0.89	…
1.00	0.47	0.89	0.47	0.80	0.47	0.79	1.00	0.43	0.79	0.81	0.89	

$1/n$

CAI: properties

- CAI = 1.0 : each codon was the most frequently used codon in highly expressed genes
- Log-odds used to avoid numerical problems
  - What happens if you multiply many values <1.0 together?
- In a sample of E.coli genes, CAI ranged from 0.2 to 0.85
- CAI correlates with mRNA levels: can be used to predict high expression levels

Biological words: summary

- Simple 1-, 2- and 3-word models can describe interesting properties of DNA sequences
  - GC skew can identify DNA replication origins
  - It can also reveal *genome rearrangement* events and *lateral transfer* of DNA
  - GC content can be used to locate genes: human genes are comparably GC-rich
  - CAI predicts high gene expression levels

Biological words: summary

- k=3 models can help to identify correct *reading frames*
  - Reading frame starts from a start codon and stops in a stop codon
  - Consider what happens to translation when a single extra base is introduced in a reading frame
- Also word models for k > 3 have their uses

Next lecture

- Genome sequencing & assembly – where do we get sequence data?

Note on programming languages

- Working with probability distributions is straightforward with R, for example
  - Deonier's book contains many computational examples
  - You can use R in CS Linux systems
- Python works too!

Example Python code for generating DNA sequences with first-order Markov chains.

```
#!/usr/bin/env python

import sys, random

n = int(sys.argv[1])

tm = {'a' : {'a' : 0.423, 'c' : 0.151, 'g' : 0.168, 't' : 0.258},
      'c' : {'a' : 0.399, 'c' : 0.184, 'g' : 0.063, 't' : 0.354},
      'g' : {'a' : 0.314, 'c' : 0.189, 'g' : 0.176, 't' : 0.321},
      't' : {'a' : 0.258, 'c' : 0.138, 'g' : 0.187, 't' : 0.415}}

pi = {'a' : 0.345, 'c' : 0.158, 'g' : 0.159, 't' : 0.337}

def choose(dist):
    r = random.random()
    sum = 0.0
    keys = dist.keys()
    for k in keys:
        sum += dist[k]
        if sum > r:
            return k
    return keys[-1]

c = choose(pi)
for i in range(n - 1):
    sys.stdout.write(c)
    c = choose(tm[c])
sys.stdout.write(c)
sys.stdout.write("\n")
```

Initialisation: use packages 'sys' and 'random', read sequence length from input.

Transition matrix tm and initial distribution pi.

Function choose(), returns a key (here 'a', 'c', 'g' or 't') of the dictionary 'dist' chosen randomly according to probabilities in dictionary values.

Choose the first letter, then choose next letter according to $P(x_t \mid x_{t-1})$.

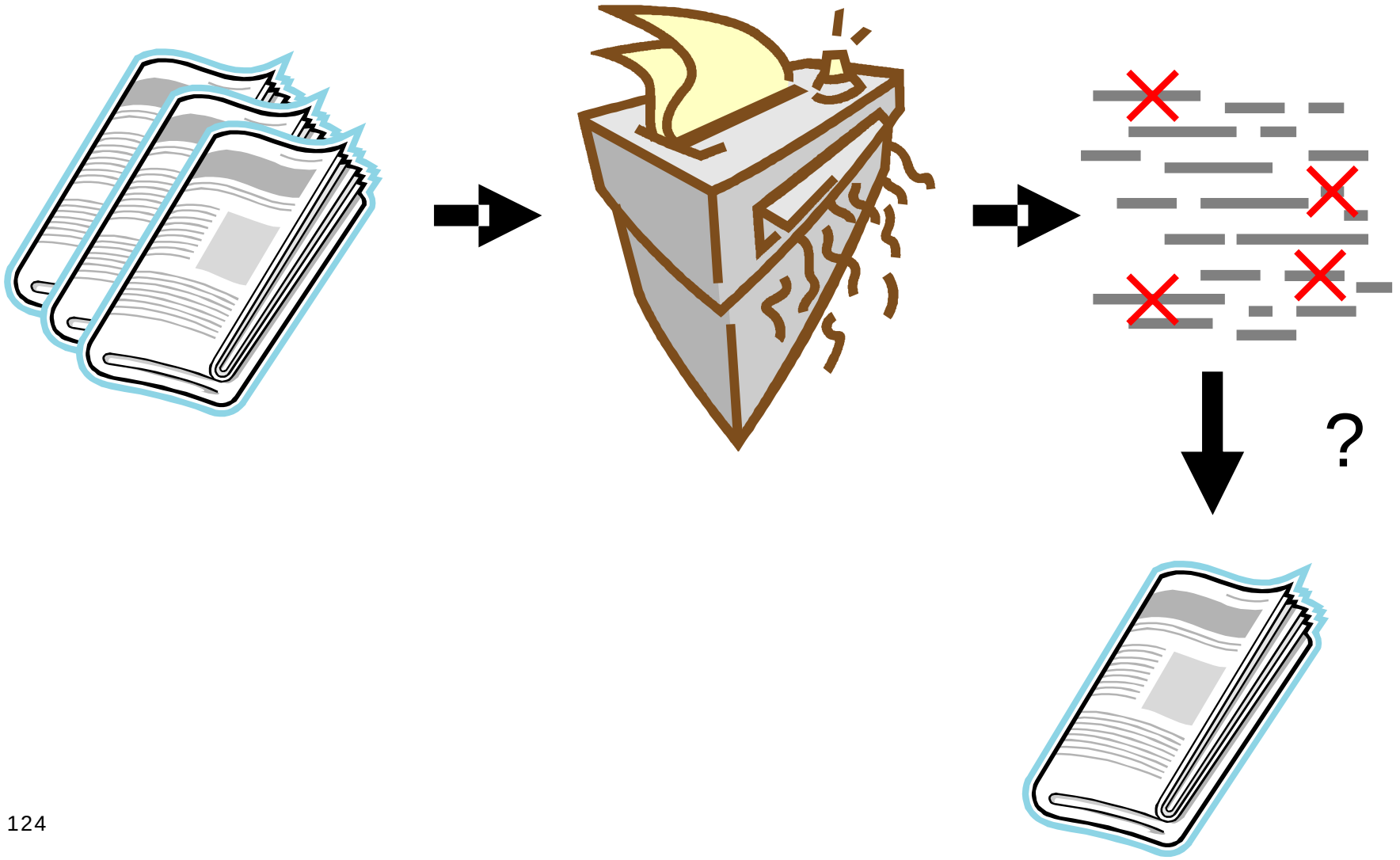
Introduction to Bioinformatics

Genome sequencing & assembly

Genome sequencing & assembly

- DNA sequencing
  - How do we obtain DNA sequence information from organisms?
- Genome assembly
  - What is needed to put together DNA sequence information from sequencing?
- First statement of sequence assembly problem (according to G. Myers):
  - Peltola, Söderlund, Tarhio, Ukkonen: Algorithms for some string matching problems arising in molecular genetics. Proc. 9th IFIP World Computer Congress, 1983

Recovery of shredded newspaper



DNA sequencing

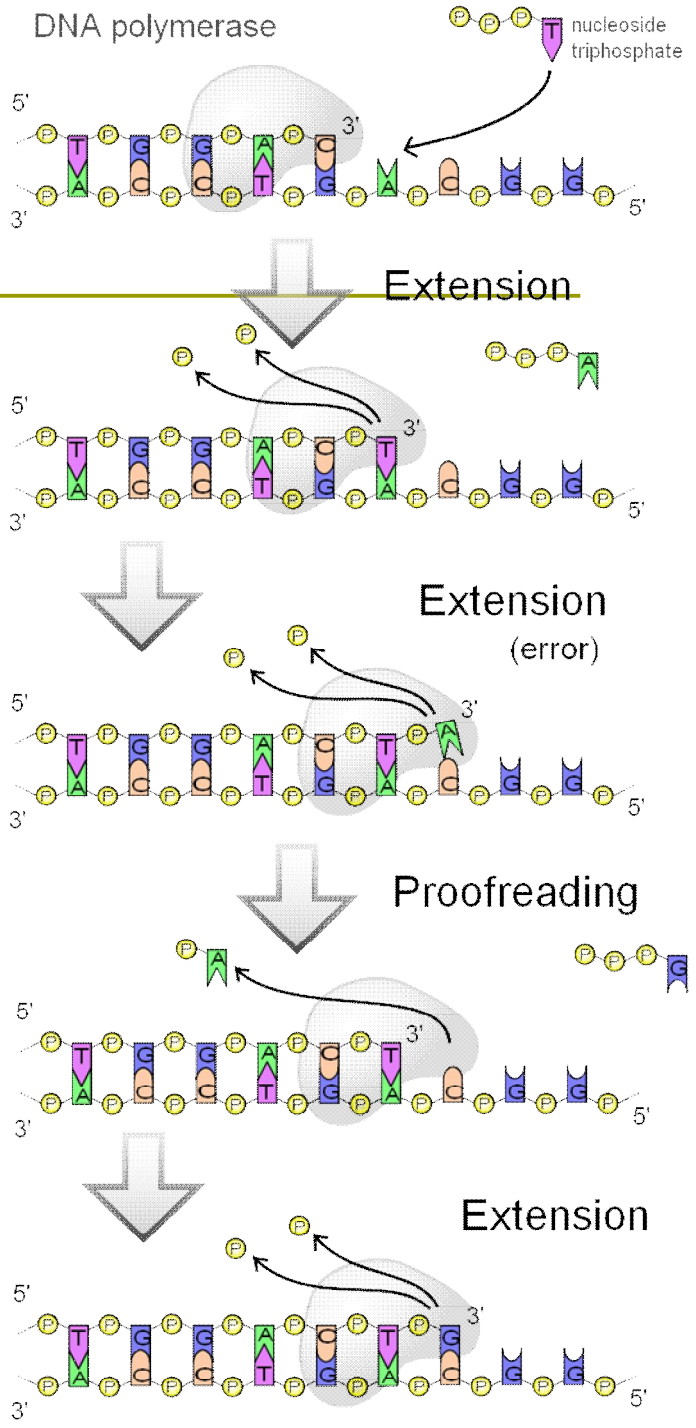
- DNA sequencing: resolving a nucleotide sequence (whole-genome or less)
- Many different methods developed
  - Maxam-Gilbert method (1977)
  - Sanger method (1977)
  - High-throughput methods

Sanger sequencing: sequencing by synthesis

- A sequencing technique developed by Fred Sanger
- Also called *dideoxy sequencing*

DNA polymerase

- A *DNA polymerase* is an enzyme that catalyzes DNA synthesis
- DNA polymerase needs a *primer*
  - Synthesis proceeds always in 5'->3' direction



http://en.wikipedia.org/wiki/DNA_polymerase

Dideoxy sequencing

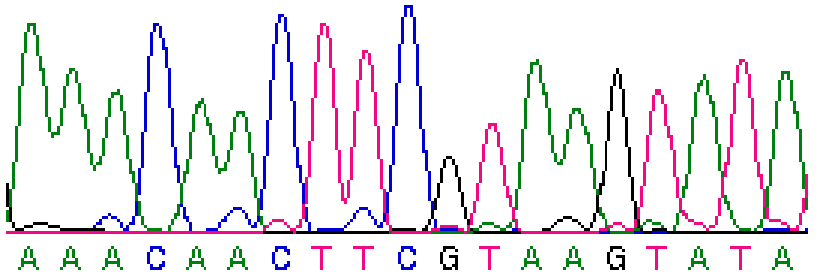
- In Sanger sequencing, chain-terminating dideoxynucleoside triphosphates (ddXTPs) are employed
  - ddATP, ddCTP, ddGTP, ddTTP lack the 3'-OH tail of dXTPs
- A mixture of dXTPs with small amount of ddXTPs is given to DNA polymerase with DNA template and primer
- ddXTPs are given fluorescent labels

Dideoxy sequencing

- When DNA polymerase encounters a ddXTP, the synthesis cannot proceed
- The process yields copied sequences of different lengths
- Each sequence is terminated by a labeled ddXTP

Determining the sequence

- Sequences are sorted according to length by capillary electrophoresis
- Fluorescent signals corresponding to labels are registered
- *Base calling*: identifying which base corresponds to each position in a read
  - Non-trivial problem!



Output sequences from base calling are called reads

Reads are short!

- Modern Sanger sequencers can produce quality reads up to ~750 bases[1]
  - Instruments provide you with a quality file for bases in reads, in addition to actual sequence data
- Compare the read length against the size of the human genome (2.9×10^9 bases)
- Reads have to be assembled!

[1] *Nature Methods* - 5, 16 - 18 (2008)

Problems with sequencing

- Sanger sequencing error rate per base varies from 1% to 3%[1]
- Repeats in DNA
  - For example, ~300 base *Alu* sequence repeated is over million times in human genome
  - Repeats occur in different scales
- What happens if repeat length is longer than read length?
  - We will get back to this problem later

[1] Jones, Pevzner (2004)

Shortest superstring problem

- Find the shortest string that ”explains” the reads
- *Given a set of strings (reads), find a shortest string that contains all of them*

Example: Shortest superstring

Set of strings: {000, 001, 010, 011, 100, 101, 110, 111}

Concetenation of strings: 000001010011100101110111

```
                    010
                  110
                011
              000
Shortest superstring: 0001110100
                  001
                    111
                      101
                        100
```

Shortest superstrings: issues

- NP-complete problem: unlike to have an efficient (exact) algorithm
- Reads may be from either strand of DNA
- Is the shortest string necessarily the correct assembly?
- What about errors in reads?
- Low *coverage* -> gaps in assembly
  - Coverage: average number of times each base occurs in the set of reads (e.g., 5x coverage)

Sequence assembly and combination locks

- What is common with sequence assembly and opening keypad locks?

Whole-genome shotgun sequence

- *Whole-genome shotgun sequence assembly* starts with a large sample of genomic DNA
  1. Sample is randomly partitioned into *inserts* of length > 500 bases
  2. Inserts are multiplied by cloning them into *a vector* which is used to infect bacteria
  3. DNA is collected from bacteria and sequenced
  4. Reads are assembled

Assembly of reads with Overlap-Layout-Consensus algorithm

- Overlap
  - Finding potentially overlapping reads
- Layout
  - Finding the order of reads along DNA
- Consensus (Multiple alignment)
  - Deriving the DNA sequence from the layout

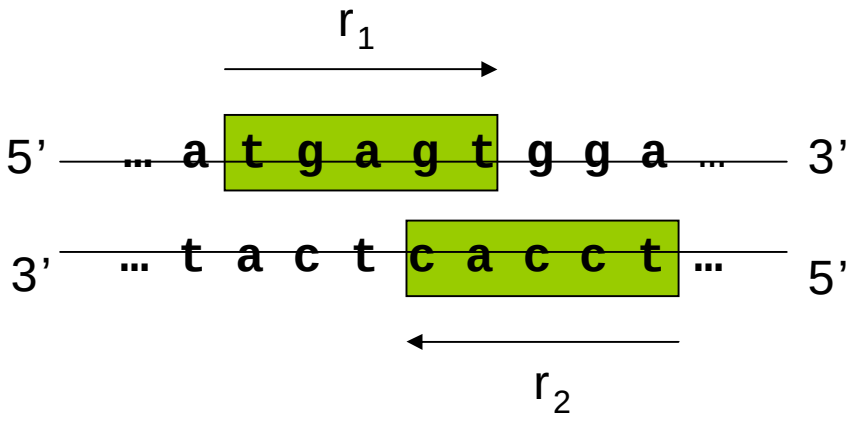
- Next, the method is described at a very abstract level, skipping a lot of details

Kececioglu, J.D. and E.W. Myers. 1995. Combinatorial algorithms for DNA sequence assembly. *Algorithmica* 13: 7-51.

Finding overlaps

- First, pairwise overlap alignment of reads is resolved
- Reads can be from either DNA strand: The *reverse complement* r* of each read r has to be considered

acggagtcc
agtccgcgctt



r_1: tgagt, r_1^*: actca
r_2: tccac, r_2^*: gtgga

Example sequence to assemble

5’ – CAGCGCGCTGCGTGACGAGTCTGACAAAGACGGTATGCGCATCG
TGATTGAAGTGAAACGCGATGCGGTCGGTCGGTGAAGTTGTGCT - 3’

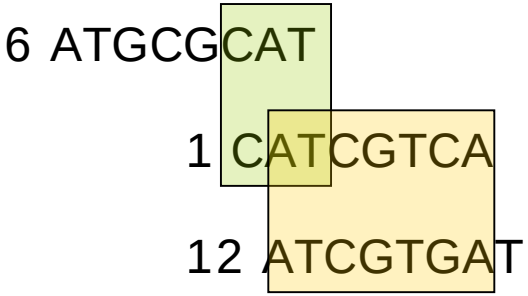
- 20 reads:

#	Read	Read*	#	Read	Read*
1	CATCGTCA	TCACGATG	11	GGTCGGTG	CACCGACC
2	CGGTGAAG	CTTCACCG	12	ATCGTGAT	ATCACGAT
3	TATGCGCA	TGCGCATA	13	GCGCTGCG	CGCAGCGC
4	GACGAGTC	GACTCGTC	14	GCATCGTG	CACGATGC
5	CTGACAAA	TTTGTCAG	15	**AGCGCGCT**	**AGCGCGCT**
6	ATGCGCAT	ATGCGCAT	16	GAAGTTGT	ACAACTTC
7	**ATGCGGTC**	GACCGCAT	17	AGTGAAAC	GTTTCACT
8	CTGCGTGA	TCACGCAG	18	ACGCGATG	CATCGCGT
9	GCGTGACG	CGTCACGC	19	GCGCATCG	CGATGCGC
10	GTCGGTGA	TCACCGAC	20	AAGTGAAA	TTTCACTT

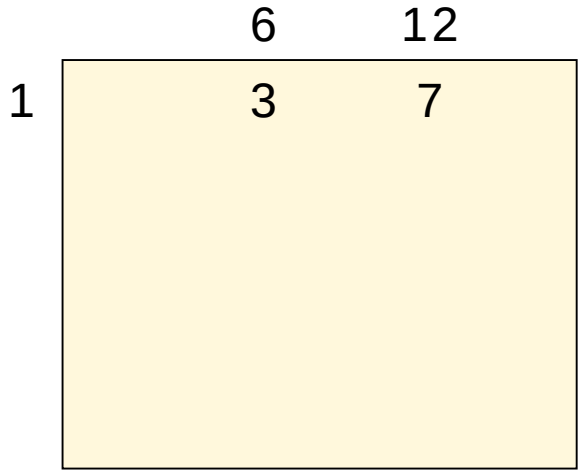
Finding overlaps

- Overlap between two reads can be found with a *dynamic programming* algorithm
  - Errors can be taken into account
- Dynamic programming will be discussed more on next lecture
- Overlap scores stored into the overlap matrix
  - Entries (i, j) below the diagonal denote overlap of read r_i and r_j*

Overlap(1, 6) = 3

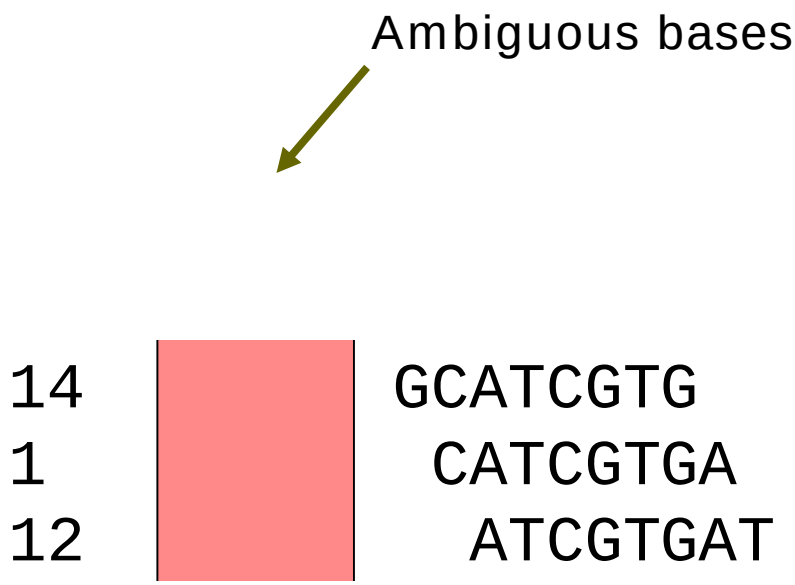


Overlap(1, 12) = 7



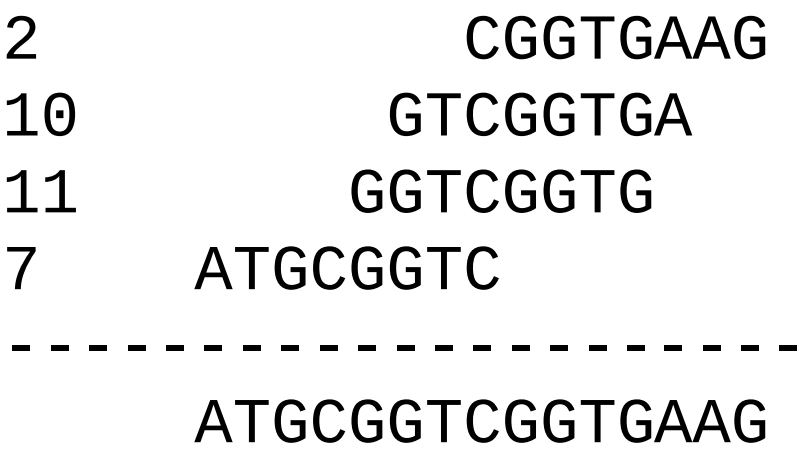
Finding layout & consensus

- Method extends the assembly *greedily* by choosing the best overlaps
- Both orientations are considered
- Sequence is extended as far as possible

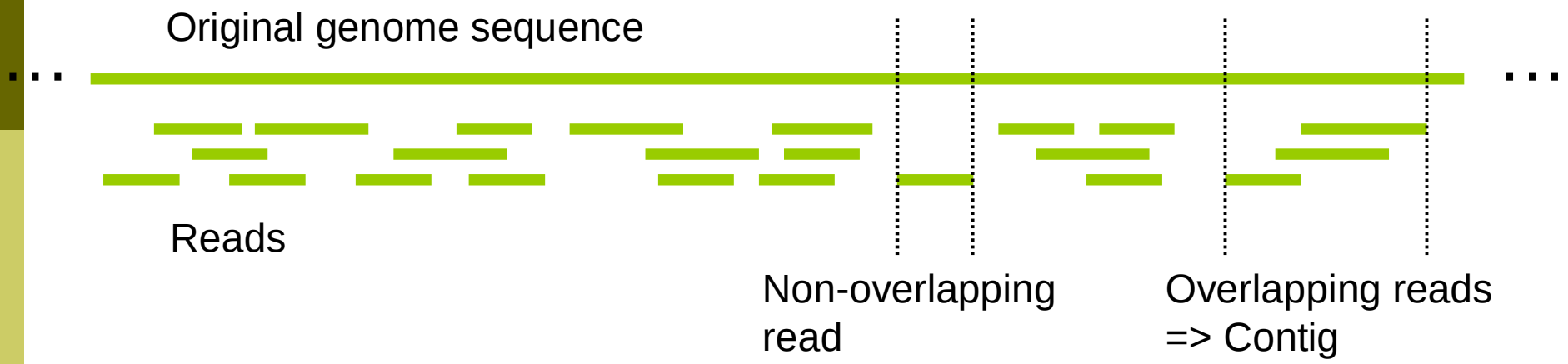


Finding layout & consensus

- We move on to next best overlaps and extend the sequence from there
- The method stops when there are no more overlaps to consider
- A number of contigs is produced
- Contig stands for contiguous sequence, resulting from merging reads



Whole-genome shotgun sequencing: summary



- Ordering of the reads is initially unknown
- Overlaps resolved by aligning the reads
- In a $3x10^9$ bp genome with 500 bp reads and 5x coverage, there are ~10^7 reads and ~$10^7(10^7-1)/2$ = ~$5x10^{13}$ pairwise sequence comparisons

Repeats in DNA and genome assembly

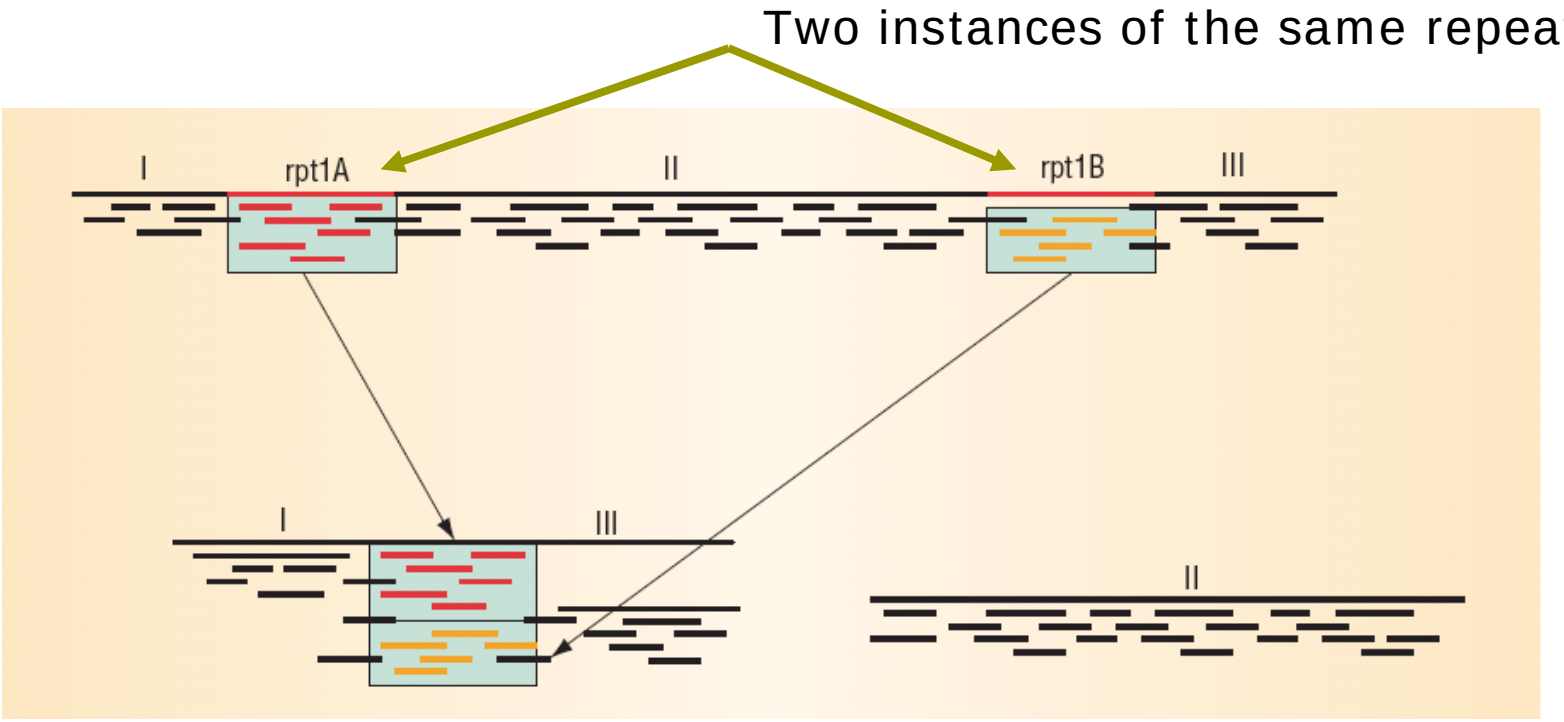


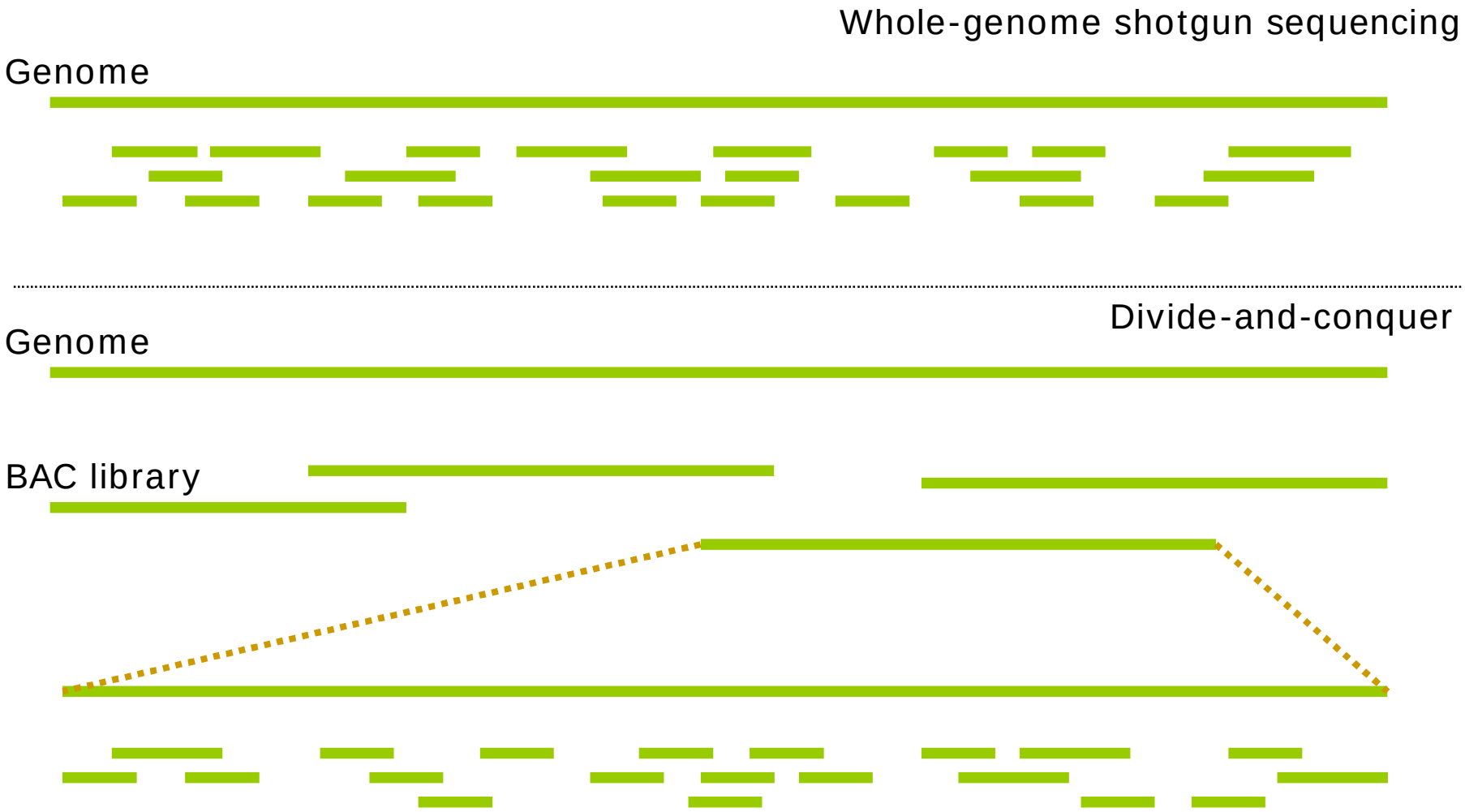
Figure 2. Repeat sequence. The top represents the correct layout of three DNA sequences. The bottom shows a repeat collapsed in a misassembly.

Pop, Salzberg, Shumway (2002)

Repeats in DNA cause problems in sequence assembly

- Recap: if repeat length exceeds read length, we might not get the correct assembly
- This is a problem especially in eukaryotes
  - ~3.1% of genome consists of repeats in Drosophila, ~45% in human
- Possible solutions
  1. Increase read length – feasible?
  2. Divide genome into smaller parts, with known order, and sequence parts individually

”Divide and conquer” sequencing approaches: BAC-by-BAC



BAC-by-BAC sequencing

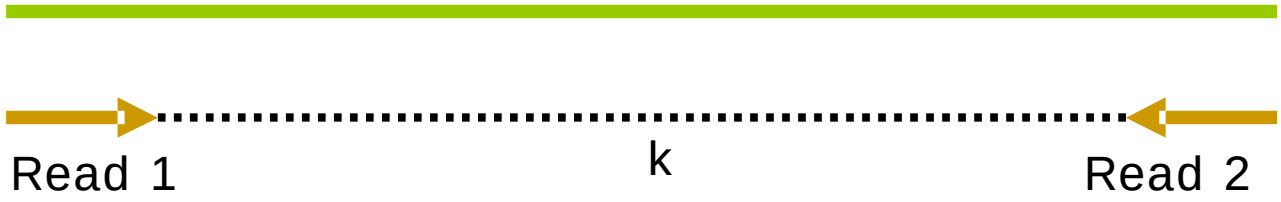
- Each BAC (Bacterial Artificial Chromosome) is about 150 kbp
- Covering the human genome requires ~30000 BACs
- BACs shotgun-sequenced separately
  - Number of repeats in each BAC is significantly smaller than in the whole genome...
  - ...needs much more manual work compared to whole-genome shotgun sequencing

Hybrid method

- Divide-and-conquer and whole-genome shotgun approaches can be combined
  - Obtain high coverage from whole-genome shotgun sequencing for short contigs
  - Generate of a set of BAC contigs with low coverage
  - Use BAC contigs to ”bin” short contigs to correct places
- This approach was used to sequence the brown Norway rat genome in 2004

Paired end sequencing

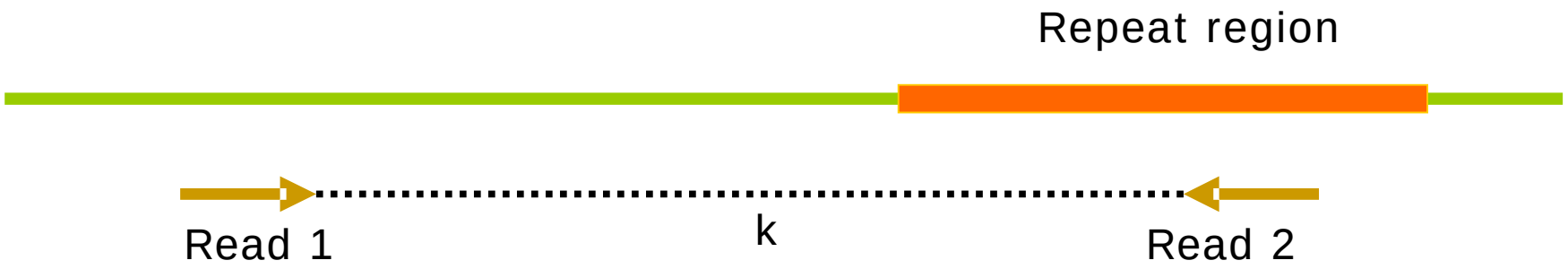
- *Paired end* (or *mate-pair*) sequencing is technique where
  - both ends of an insert are sequenced
  - For each insert, we get two reads
  - We know the distance between reads, and that they are in opposite orientation



  - Typically read length < insert length

Paired end sequencing

- The key idea of paired end sequencing:
  - Both reads from an insert are unlikely to be in repeat regions
  - If we know where the first read is, we know also second's location



- This technique helps to WGSS higher organisms

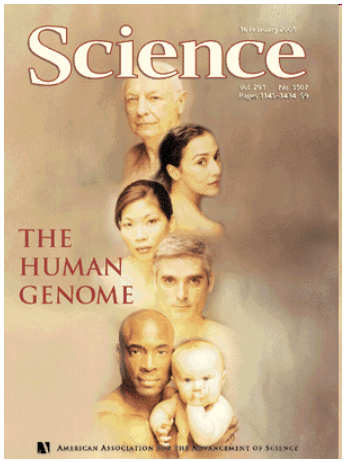
First whole-genome shotgun sequencing project: Drosophila melanogaster

- Fruit fly is a common *model organism* in biological studies
- Whole-genome assembly reported in Eugene Myers, *et al.*, A Whole-Genome Assembly of *Drosophila*, *Science* 24, 2000
- Genome size 120 Mbp

http://en.wikipedia.org/wiki/Drosophila_melanogaster

Sequencing of the Human Genome

- The (draft) human genome was published in 2001
- Two efforts:
  - Human Genome Project (public consortium)
  - Celera (private company)
- HGP: BAC-by-BAC approach
- Celera: whole-genome shotgun sequencing



HGP: Nature 15 February 2001
Vol 409 Number 6822

Celera: Science 16 February 2001
Vol 291, Issue 5507

Genome assembly software

- phrap (Phil's revised assembly program)
- AMOS (A Modular, Open-Source whole-genome assembler)
- CAP3 / PCAP
- TIGR assembler

Next generation sequencing techniques

- Sanger sequencing is the prominent first-generation sequencing method
- Many new sequencing methods are emerging

- See Lars Paulin's slides (course web page) for details

Next-gen sequencing: 454

- Genome Sequencer FLX (454 Life Science / Roche)
  - >100 Mb / 7.5 h run
  - Read length 250-300 bp
  - >99.5% accuracy / base in a single run
  - >99.99% accuracy / base in consensus

Next-gen sequencing: Illumina Solexa

- Illumina / Solexa Genome Analyzer
  - Read length 35 - 50 bp
  - 1-2 Gb / 3-6 day run
  - > 98.5% accuracy / base in a single run
  - 99.99% accuracy / consensus with 3x coverage

Next-gen sequencing: SOLiD

- SOLiD
  - Read length 25-30 bp
  - 1-2 Gb / 5-10 day run
  - >99.94% accuracy / base
  - >99.999% accuracy / consensus with 15x coverage

Next-gen sequencing: Helicos

- Helicos: Single Molecule Sequencer
  - No amplification of sequences needed
  - Read length up to 55 bp
    - Accuracy does not decrease when read length is increased
    - Instead, throughput goes down
  - 25-90 Mb / h
  - >2 Gb / day

Next-gen sequencing: Pacific Biosciences

- Pacific Biosciences
  - Single-Molecule Real-Time (SMRT) DNA sequencing technology
  - Read length "thousands of nucleotides"
    - Should overcome most problems with repeats
  - Throughput estimate: 100 Gb / hour
  - First instruments in 2010?

Exercise groups

- 1st group: Tuesdays 16.15-18.00 C221
- 2nd group: Wednesday 14.15-16.00 B120

- You can choose freely which group you want to attend to
- Send exercise notes before the 1st group starts (Tue 16.15), even if you go to the 2nd group

Introduction to Bioinformatics

Lecture 3:

Sequence alignment

Sequence alignment

- *The biological problem*
- Global alignment
- Local alignment
- Multiple alignment

Background: comparative genomics

- Basic question in biology: *what properties are shared among organisms?*
- Genome sequencing allows comparison of organisms at DNA and protein levels
- Comparisons can be used to
  - Find evolutionary relationships between organisms
  - Identify functionally conserved sequences
  - Identify corresponding genes in human and model organisms: develop models for human diseases

Homologs

- Two genes (sequences in general) g_B and g_C evolved from the same ancestor gene g_A are called *homologs*
- Homologs usually exhibit conserved functions
- Close evolutionary relationship => expect a high number of homologs

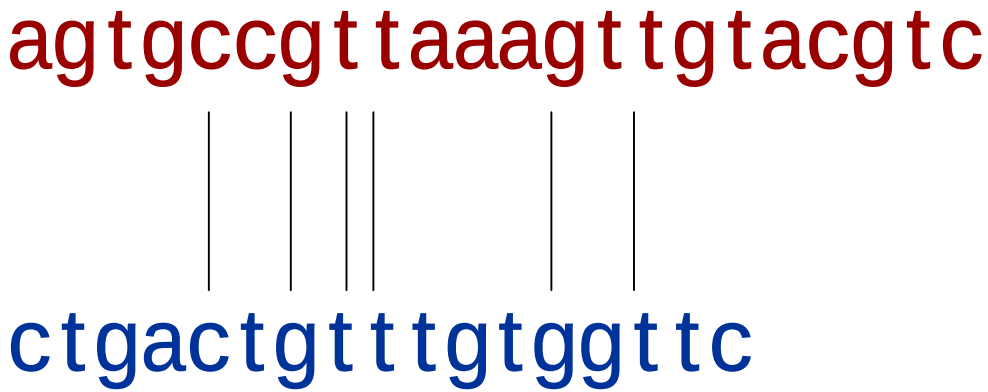
g_A = agtgtccgttaagtgcgttc

g_B = agtgccgttaaagttgtacgtc

g_C = ctgactgtttgtggttc

Sequence similarity

- We expect homologs to be ”similar” to each other
- Intuitively, similarity of two sequences refers to the degree of match between corresponding positions in sequence



- What about sequences that differ in length?

Similarity vs homology

- Sequence similarity is not sequence homology
  - If the two sequences g_B and g_C have accumulated enough mutations, the similarity between them is likely to be low

#mutations		#mutations	
0	agtgtccgttaagtgcgttc	64	acagtccgttcgggctattg
1	agtgtccgttatagtgcgttc	128	cagagcactaccgc
2	agtgtccgcttatagtgcgttc	256	cacgagtaagatatagct
4	agtgtccgcttaagggcgttc	512	taatcgtgata
8	agtgtccgcttcaaggggcgt	1024	acccttatctacttcctggagtt
16	gggccgttcatgggggt	2048	agcgacctgcccaa
32	gcagggcgtcactgagggct	4096	caaac

Homology is more difficult to detect over greater evolutionary distances.

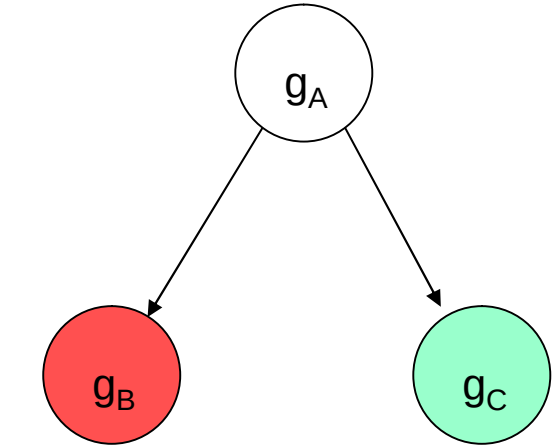
Similarity vs homology (2)

- Sequence similarity can occur by chance
  - *Similarity does not imply homology*

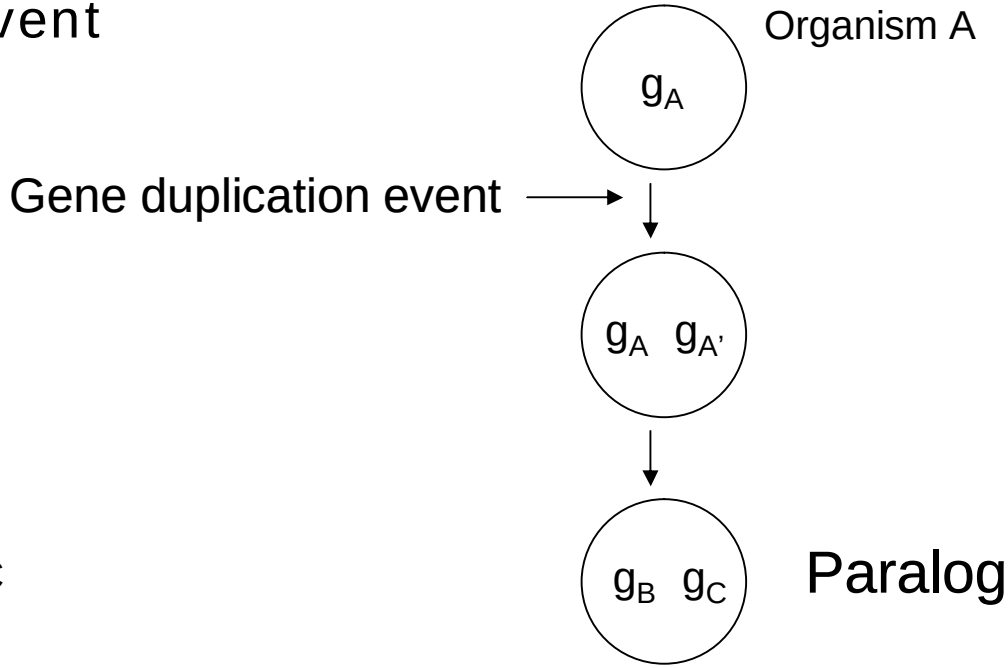
- Consider comparing two short sequences against each other

Orthologs and paralogs

- We distinguish between two types of homology
  - Orthologs: homologs from two different species, separated by a *speciation* event
  - Paralogs: homologs within a species, separated by a *gene duplication* event



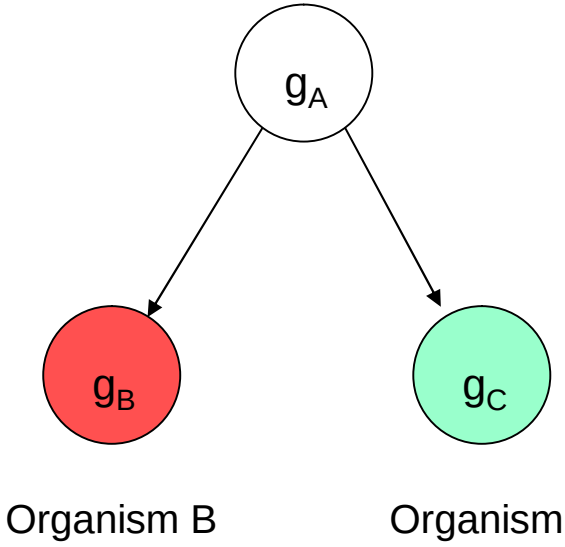
Orthologs

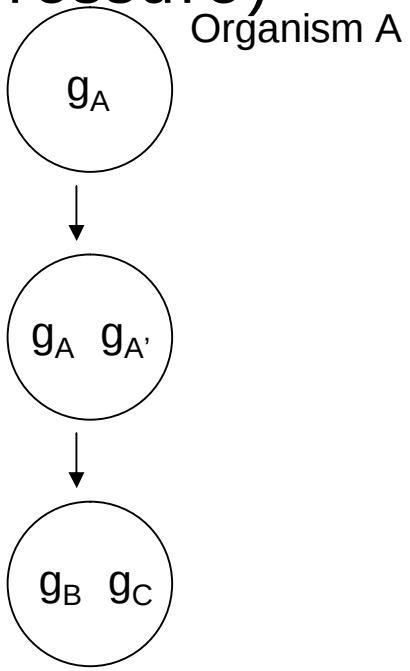


Paralogs

Orthologs and paralogs (2)

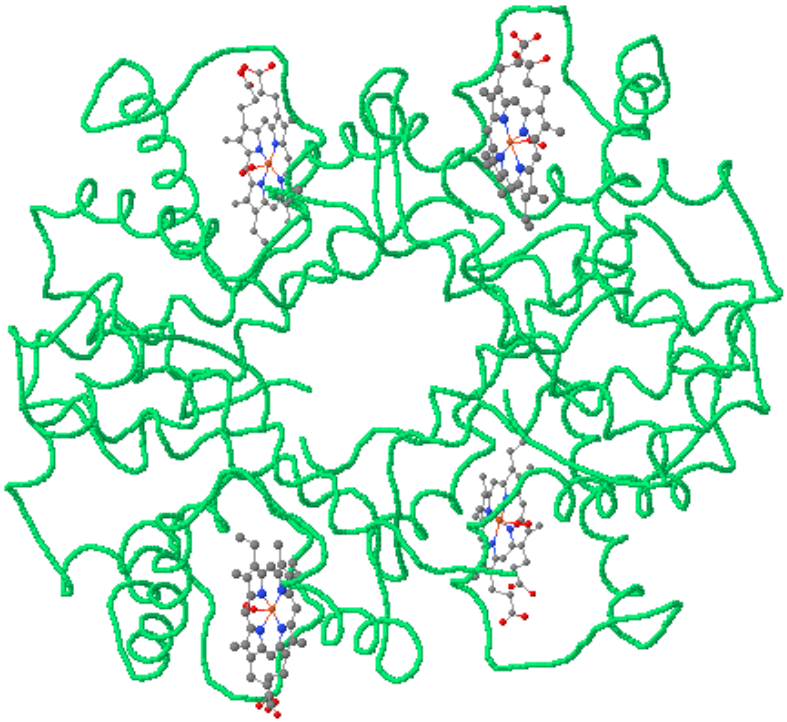
- Orthologs typically retain the original function
- In paralogs, one copy is free to mutate and acquire new function (no selective pressure)





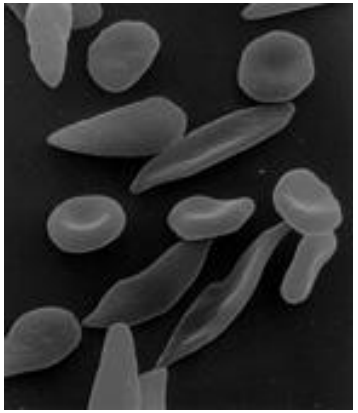
Paralogy example: hemoglobin

- Hemoglobin is a protein complex which transports oxygen
- In humans, hemoglobin consists of four protein subunits and four non-protein heme groups



Hemoglobin A, www.rcsb.org/pdb/explore.do?structureId=1GZX

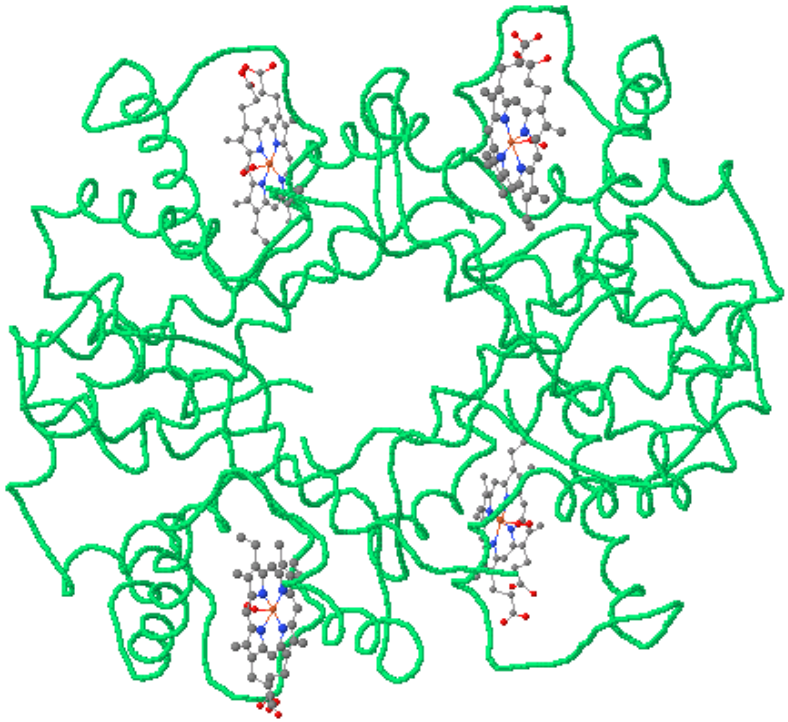
Sickle cell diseases are caused by mutations in hemoglobin genes



http://en.wikipedia.org/wiki/Image:Sicklecells.jpg

Paralogy example: hemoglobin

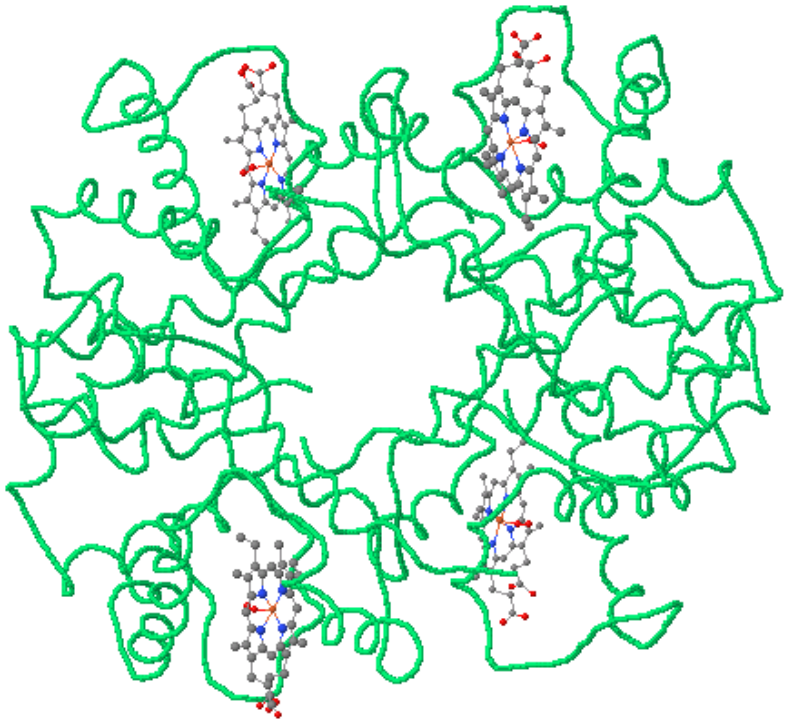
- In adults, three types are normally present
  - Hemoglobin A: 2 alpha and 2 beta subunits
  - Hemoglobin A2: 2 alpha and 2 delta subunits
  - Hemoglobin F: 2 alpha and 2 gamma subunits
- Each type of subunit (alpha, beta, gamma, delta) is encoded by a separate gene



Hemoglobin A, www.rcsb.org/pdb/explore.do?structureId=1GZX

Paralogy example: hemoglobin

- The subunit genes are paralogs of each other, i.e., they have a common ancestor gene
- Exercise: hemoglobin human paralogs in NCBI sequence databases
  http://www.ncbi.nlm.nih.gov/sites/entrez?db=Nucleotide
  - Find human hemoglobin alpha, beta, gamma and delta
  - Compare sequences



Hemoglobin A, www.rcsb.org/pdb/explore.do?structureId=1GZX

Orthology example: insulin

- The genes coding for insulin in human (*Homo sapiens*) and mouse (*Mus musculus*) are orthologs:
  - They have a common ancestor gene in the ancestor species of human and mouse
  - Exercise: find insulin orthologs from human and mouse in NCBI sequence databases

Sequence alignment

- Alignment specifies which positions in two sequences match

```
acgtctag    acgtctag    acgtctag
||            |||||     || |||||
actctag-    -actctag    ac-tctag
```

2 matches	5 matches	7 matches
5 mismatches	2 mismatches	0 mismatches
1 not aligned	1 not aligned	1 not aligned

Sequence alignment

- Maximum alignment length is the total length of the two sequences

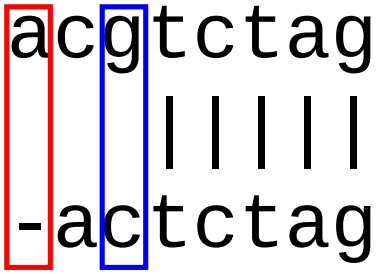
```
acgtctag-------        -------acgtctag

--------actctag        actctag--------
```

0 matches
0 mismatches
15 not aligned

0 matches
0 mismatches
15 not aligned

Mutations: Insertions, deletions and substitutions

Indel: insertion or deletion of a base with respect to the ancestor sequence



Mismatch: substitution (point mutation) of a single base

- Insertions and/or deletions are called *indels*
  - *We can't tell whether the ancestor sequence had a base or not at indel position!*

Problems

- What sorts of alignments should be considered?
- How to score alignments?
- How to find optimal or good scoring alignments?
- How to evaluate the statistical significance of scores?

In this course, we discuss each of these problems briefly.

Sequence Alignment (chapter 6)

- The biological problem
- *Global alignment*
- Local alignment
- Multiple alignment

Global alignment

- Problem: find optimal scoring alignment between two sequences (Needleman & Wunsch 1970)
- Every position in both sequences is included in the alignment
- We give score for each position in alignment
  - Identity (match) +1
  - Substitution (mismatch) -μ
  - Indel -δ
- Total score: sum of position scores

Scoring: Toy example

- Consider two sequences with characters drawn from the English language alphabet: WHAT, WHY

WHAT

| |

WH-Y

S(WHAT/WH-Y) = 1 + 1 – δ – μ

WHAT

-WHY

S(WHAT/-WHY) = – δ – μ – μ – μ

Dynamic programming

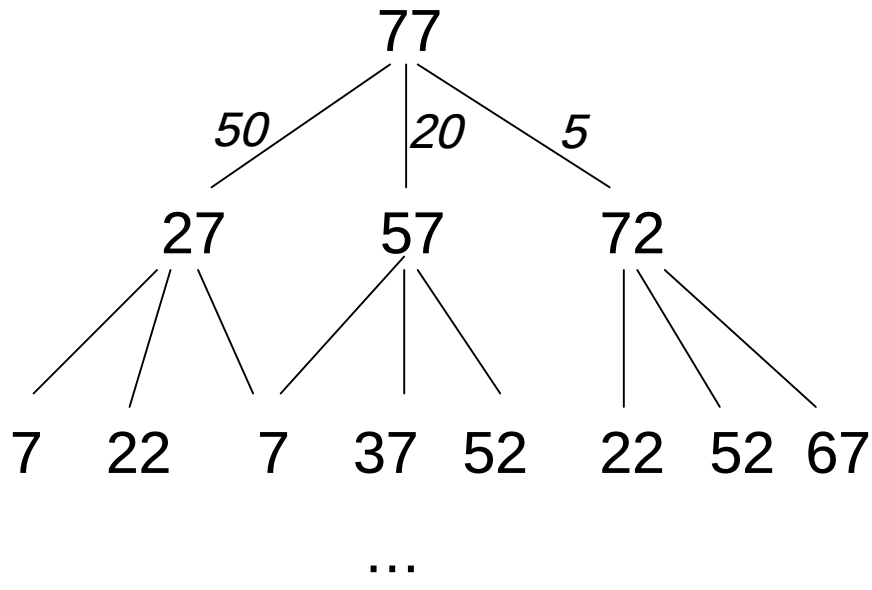
- How to find the optimal alignment?
- We use previous solutions for optimal alignments of smaller subsequences
- This general approach is known as dynamic programming

Introduction to dynamic programming: the money change problem

- Suppose you buy a pen for 4.23€ and pay for with a 5€ note
- You get 77 cents in change – what coins is the cashier going to give you if he or she tries to minimise the number of coins?
- The usual algorithm: start with largest coin (denominator), proceed to smaller coins until no change is left:
  - 50, 20, 5 and 2 cents
- This greedy algorithm is *incorrect*, in the sense that it does not always give you the correct answer

The money change problem

- How else to compute the change?
- We could consider all possible ways to reduce the amount of change
- Suppose we have 77 cents change, and the following coins: 50, 20, 5 cents
- We can compute the change with *recursion*
  - *C(n) = min { C(n – 50) + 1, C(n – 20) + 1, C(n – 5) + 1 }*
- Figure shows the recursion tree for the example

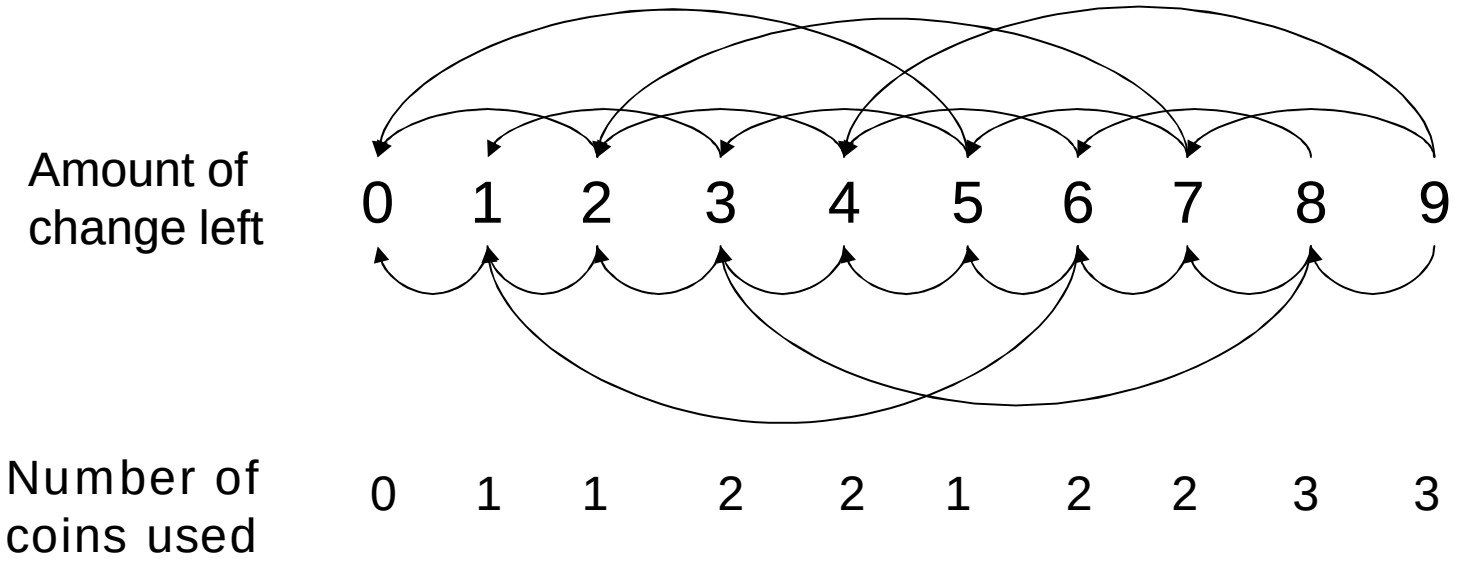


- Many values are computed more than once!
- This leads to a correct but very inefficient algorithm

The money change problem

- We can speed the computation up by solving the change problem for all i ≤ n
  - Example: solve the problem for 9 cents with available coins being 1, 2 and 5 cents
- Solve the problem in steps, first for 1 cent, then 2 cents, and so on
- In each step, utilise the solutions from the previous steps

The money change problem



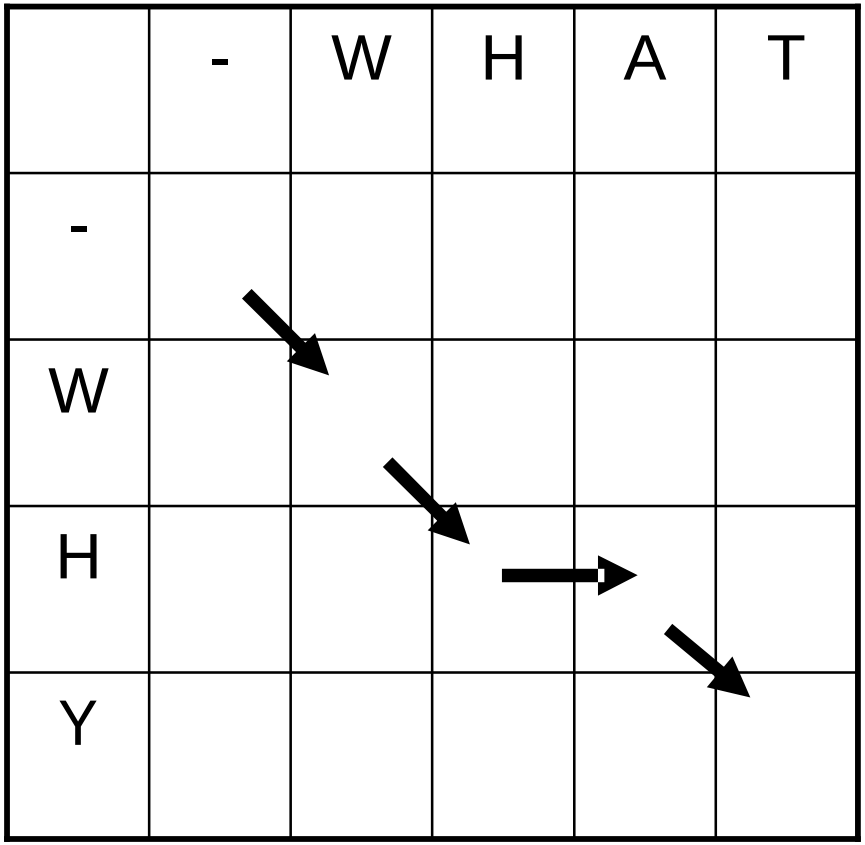
- Algorithm runs in time proportional to Md, where M is the amount of change and d is the number of coin types
- The same technique of storing solutions of subproblems can be utilised in aligning sequences

Representing alignments and scores

Alignments can be represented in the following tabular form.

Each alignment corresponds to a path through the table.

```
WHAT
|  |
WH-Y
```

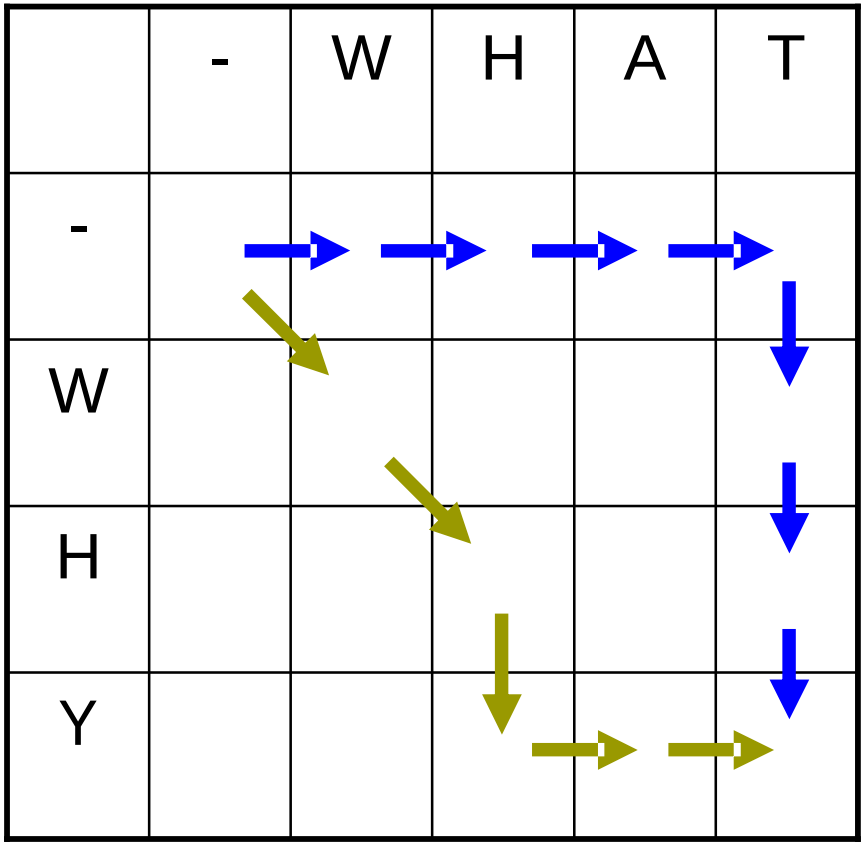


Representing alignments and scores

```
WH-AT
||
WHY--
```

```
WHAT---
```

```
----WHY
```



Representing alignments and scores

WHAT

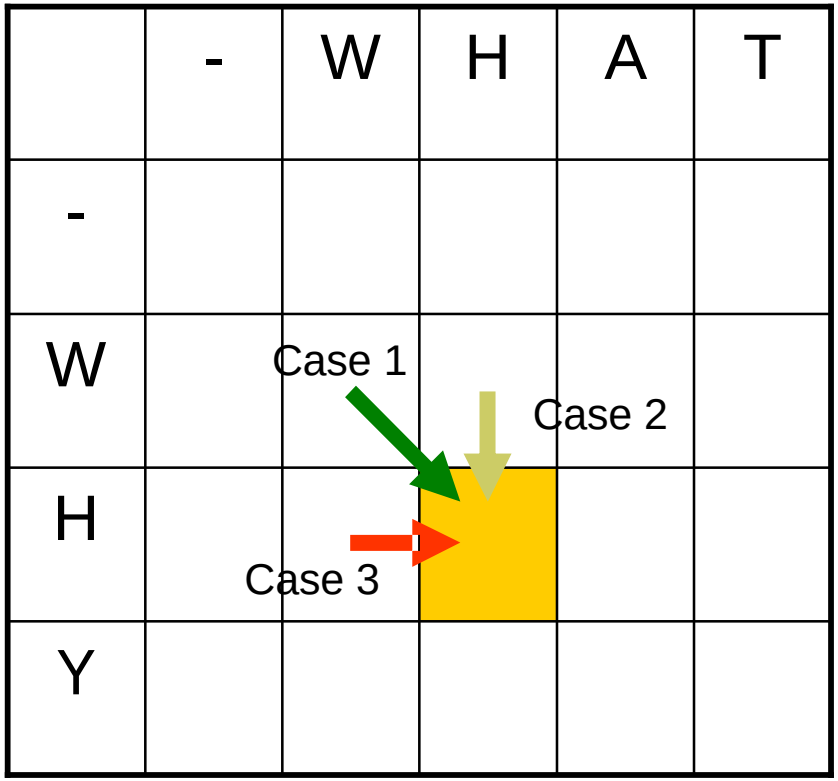
||

WH-Y

Global alignment score $S_{3,4} = 2-\delta-\mu$

	-	W	H	A	T
-	0				
W		1			
H			2	2-δ	
Y					2-δ-μ

Filling the alignment matrix



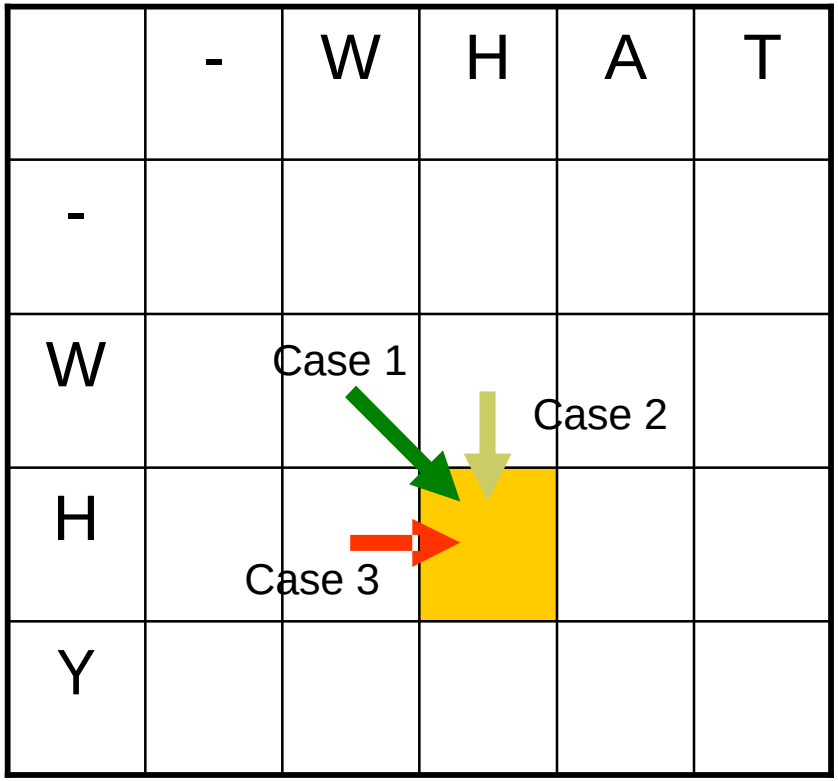
Consider the alignment process at shaded square.

Case 1. Align H against H (match)

Case 2. Align H in WHY against – (indel) in WHAT

Case 3. Align H in WHAT against – (indel) in WHY

Filling the alignment matrix (2)



Scoring the alternatives.

Case 1. $S_{2,2} = S_{1,1} + s(2, 2)$

Case 2. $S_{2,2} = S_{1,2} - \delta$

Case 3. $S_{2,2} = S_{2,1} - \delta$

$s(i, j) = 1$ for matching positions,

$s(i, j) = -\mu$ for substitutions.

Choose the case (path) that yields the maximum score.

Keep track of path choices.

Global alignment: formal development

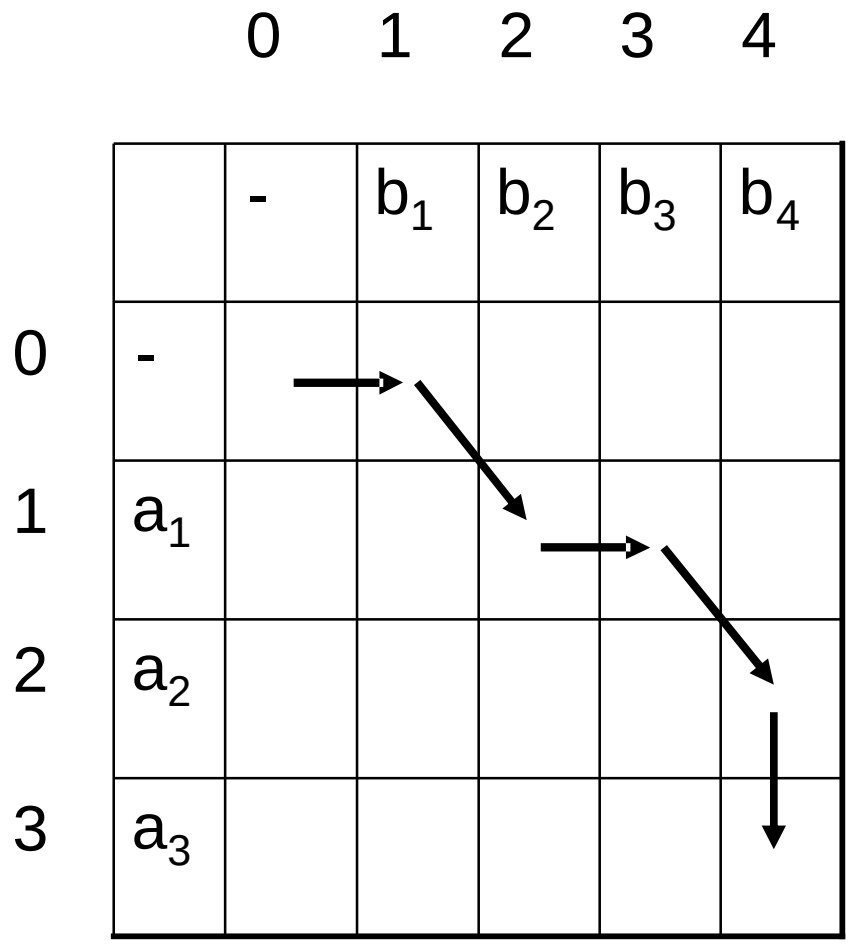
$A = a_1a_2a_3\ldots a_n$,

$B = b_1b_2b_3\ldots b_m$

b_1	b_2	b_3	b_4	-
-	a_1	-	a_2	a_3

- Any alignment can be written as a unique path through the matrix
- Score for aligning A and B up to positions i and j:

$S_{i,j} = S(a_1a_2a_3\ldots a_i, b_1b_2b_3\ldots b_j)$



Scoring partial alignments

- Alignment of A = $a_1a_2a_3...a_i$ with B = $b_1b_2b_3...b_j$ can be end in three possible ways
  - Case 1: $(a_1a_2...a_{i-1})$ a_i
    $(b_1b_2...b_{j-1})$ b_j
  - Case 2: $(a_1a_2...a_{i-1})$ a_i
    $(b_1b_2...b_j)$ -
  - Case 3: $(a_1a_2...a_i)$ –
    $(b_1b_2...b_{j-1})$ b_j

Scoring alignments

- Scores for each case:
  - Case 1: $(a_1 a_2 \ldots a_{i-1})\ a_i$
    $(b_1 b_2 \ldots b_{j-1})\ b_j$
  - Case 2: $(a_1 a_2 \ldots a_{i-1})\ a_i$
    $(b_1 b_2 \ldots b_j)\ -$
  - Case 3: $(a_1 a_2 \ldots a_i)\ -$
    $(b_1 b_2 \ldots b_{j-1})\ b_j$

$$s(a_i, b_j) = \begin{cases} +1 \text{ if } a_i = b_j \\ -\mu \text{ otherwise} \end{cases}$$

$$s(a_i, -) = s(-, b_j) = -\delta$$

Scoring alignments (2)

- First row and first column correspond to initial alignment against indels:

  $S(i, 0) = -i\,\delta$

  $S(0, j) = -j\,\delta$

- Optimal global alignment score $S(A, B) = S_{n,m}$

		0	1	2	3	4
		-	b_1	b_2	b_3	b_4
0	-	0	-δ	-2δ	-3δ	-4δ
1	a_1	-δ				
2	a_2	-2δ				
3	a_3	-3δ				

Algorithm for global alignment

Input sequences A, B, n = |A|, m = |B|

Set $S_{i,0} := -\delta i$ for all i

Set $S_{0,j} := -\delta j$ for all j

for i := 1 to n

  for j := 1 to m

    $S_{i,j} := \max\{S_{i-1,j} - \delta, S_{i-1,j-1} + s(a_i,b_j), S_{i,j-1} - \delta\}$

  end

end

Algorithm takes O(nm) time

Global alignment: example

$\mu = 1$

$\delta = 2$

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2					
T	-4					
C	-6					
G	-8					
T	-10					?

Global alignment: example

μ = 1

δ = 2

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2	-1	-3			
T	-4					
C	-6					
G	-8					
T	-10					?

Global alignment: example (2)

μ = 1

δ = 2

ATCGT-

| | ||

-TGGTG

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2	-1	-3	-5	-7	-9
T	-4	-1	-2	-4	-4	-6
C	-6	-3	-2	-3	-5	-5
G	-8	-5	-2	-1	-3	-4
T	-10	-7	-4	-3	0	-2

Sequence Alignment (chapter 6)

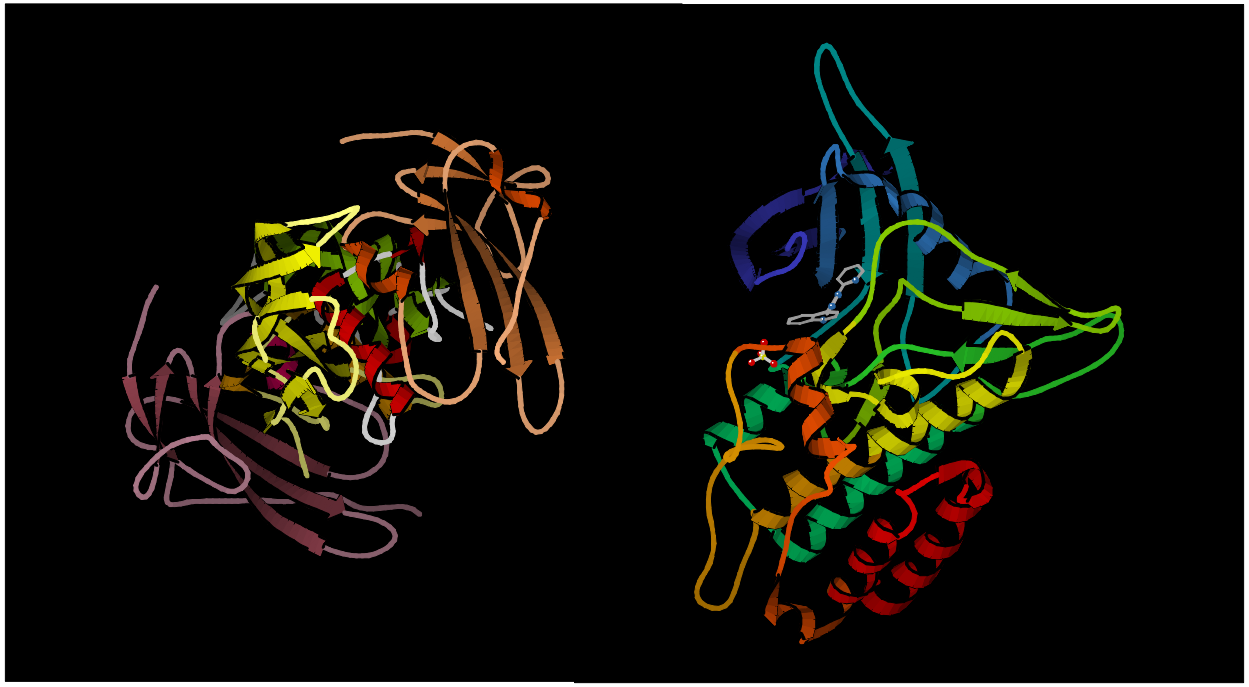
- The biological problem
- Global alignment
- *Local alignment*
- Multiple alignment

Local alignment: rationale

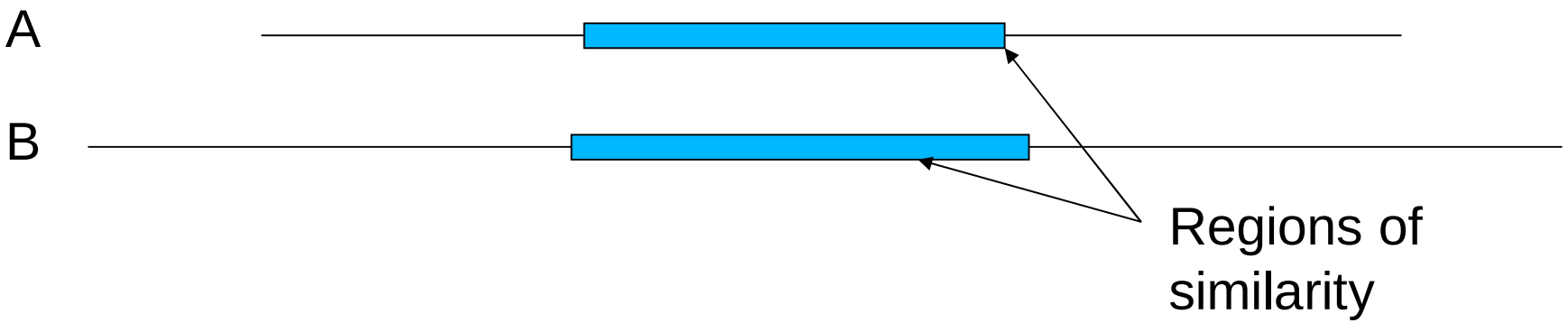
- Otherwise dissimilar proteins may have local regions of similarity
  -> Proteins may share a function

Human bone morphogenic protein receptor type II precursor (left) has a 300 aa region that resembles 291 aa region in TGF-β receptor (right).

The shared function here is protein kinase.



Local alignment: rationale



- Global alignment would be inadequate
- Problem: find the highest scoring *local* alignment between two sequences
- Previous algorithm with minor modifications solves this problem (Smith & Waterman 1981)

From global to local alignment

- Modifications to the global alignment algorithm
  - Look for the highest-scoring path in the alignment matrix (not necessarily through the matrix), or in other words:
  - Allow preceding and trailing indels without penalty

Scoring local alignments

$A = a_1a_2a_3\ldots a_n$, $B = b_1b_2b_3\ldots b_m$

Let I and J be intervals (substrings) of A and B, respectively:

$$I \subset A \quad J \subset B$$

Best local alignment score:

$$M(A, B) = \max\{S(I, J) : I \subset A, J \subset B\}$$

where S(I, J) is the alignment score for substrings I and J.

Allowing preceding and trailing indels

- First row and column initialised to zero:

  $M_{i,0} = M_{0,j} = 0$

```
b1  b2  b3
-   -   a1
```

		0	1	2	3	4
		-	b_1	b_2	b_3	b_4
0	-	0	0	0	0	0
1	a_1	0				
2	a_2	0				
3	a_3	0				

Recursion for local alignment

- $M_{i,j} = \max \{$
  - $M_{i-1,j-1} + s(a_i, b_i)$,
  - $M_{i-1,j} - \delta$,
  - $M_{i,j-1} - \delta$,
  - 0
  - $\}$

Allow alignment to start anywhere in sequences

	-	T	G	G	T	G
-	0	0	0	0	0	0
A	0	0	0	0	0	0
T	0	1	0	0	1	0
C	0	0	0	0	0	0
G	0	0	1	1	0	1
T	0	1	0	0	2	0

Finding best local alignment

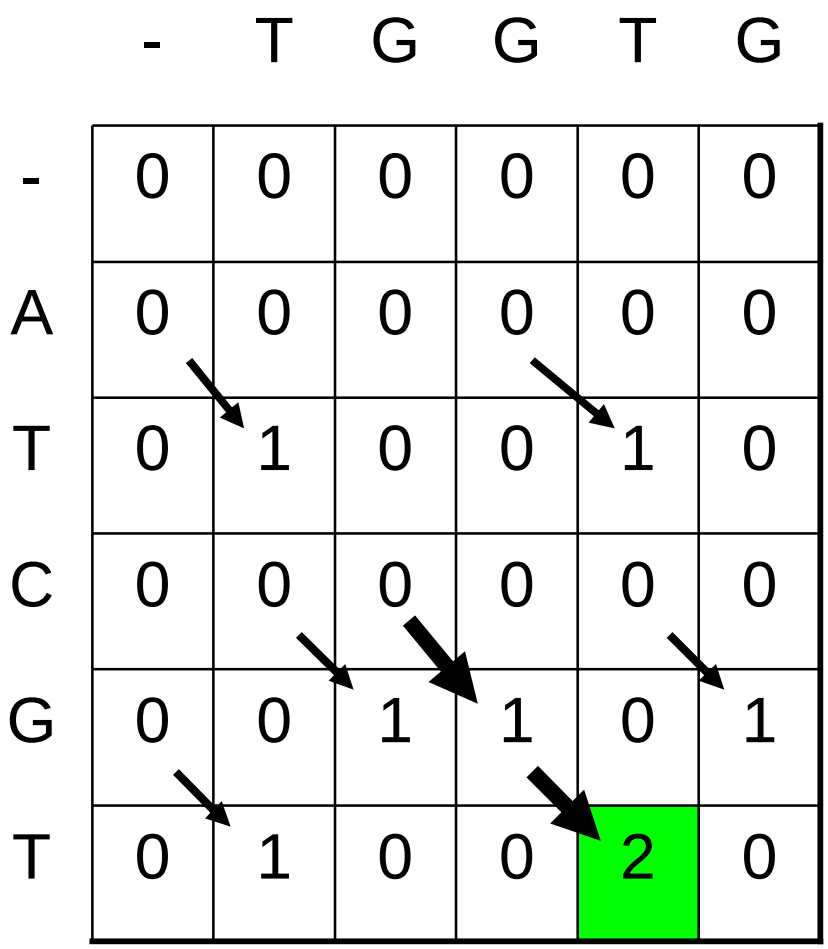
- Optimal score is the highest value in the matrix

$$M(A,B) = \max\{S(I,J) : I \subset A, J \subset B\}$$

$$= \max_{i,j} M_{i,j}$$

- Best local alignment can be found by backtracking from the highest value in M

- What is the best local alignment in this example?



Local alignment: example

$$M_{i,j} = \max \{ M_{i-1,j-1} + s(a_i, b_i),\ M_{i-1,j} - \delta,\ M_{i,j-1} - \delta,\ 0 \}$$

Scoring (for example)
Match: +2
Mismatch: -1
Indel: -2

		0	1	2	3	4	5	6	7	8	9	10
		-	G	G	C	T	C	A	A	T	C	A
0	-	0	0	0	0	0	0	0	0	0	0	0
1	A	0	0									
2	C	0										
3	C	0										
4	T	0										
5	A	0										
6	A	0										
7	G	0										
8	G	0										

Local alignment: example

$$M_{i,j} = \max \{ M_{i-1,j-1} + s(a_i, b_j),\ M_{i-1,j} - \delta,\ M_{i,j-1} - \delta,\ 0 \}$$

Scoring (for example)
Match: +2
Mismatch: -1
Indel: -2

		0	1	2	3	4	5	6	7	8	9	10
		-	G	G	C	T	C	A	A	T	C	A
0	-	0	0	0	0	0	0	0	0	0	0	0
1	A	0	0	0	0	0	0	2				
2	C	0										
3	C	0										
4	T	0										
5	A	0										
6	A	0										
7	G	0										
8	G	0										

Local alignment: example

Optimal local alignment:

C T – A A
C T C A A

		0	1	2	3	4	5	6	7	8	9	10
		-	G	G	C	T	C	A	A	T	C	A
0	-	0	0	0	0	0	0	0	0	0	0	0
1	A	0	0	0	0	0	0	2	2	0	0	2
2	C	0	0	0	2	0	2	0	1	1	2	0
3	C	0	0	0	2	1	2	1	0	0	3	1
4	T	0	0	0	0	4	2	1	0	2	1	2
5	A	0	0	0	0	2	3	4	3	1	1	3
6	A	0	0	0	0	0	1	5	6	4	2	3
7	G	0	2	2	0	0	0	3	4	5	3	1
8	G	0	2	4	2	0	0	1	2	3	4	2

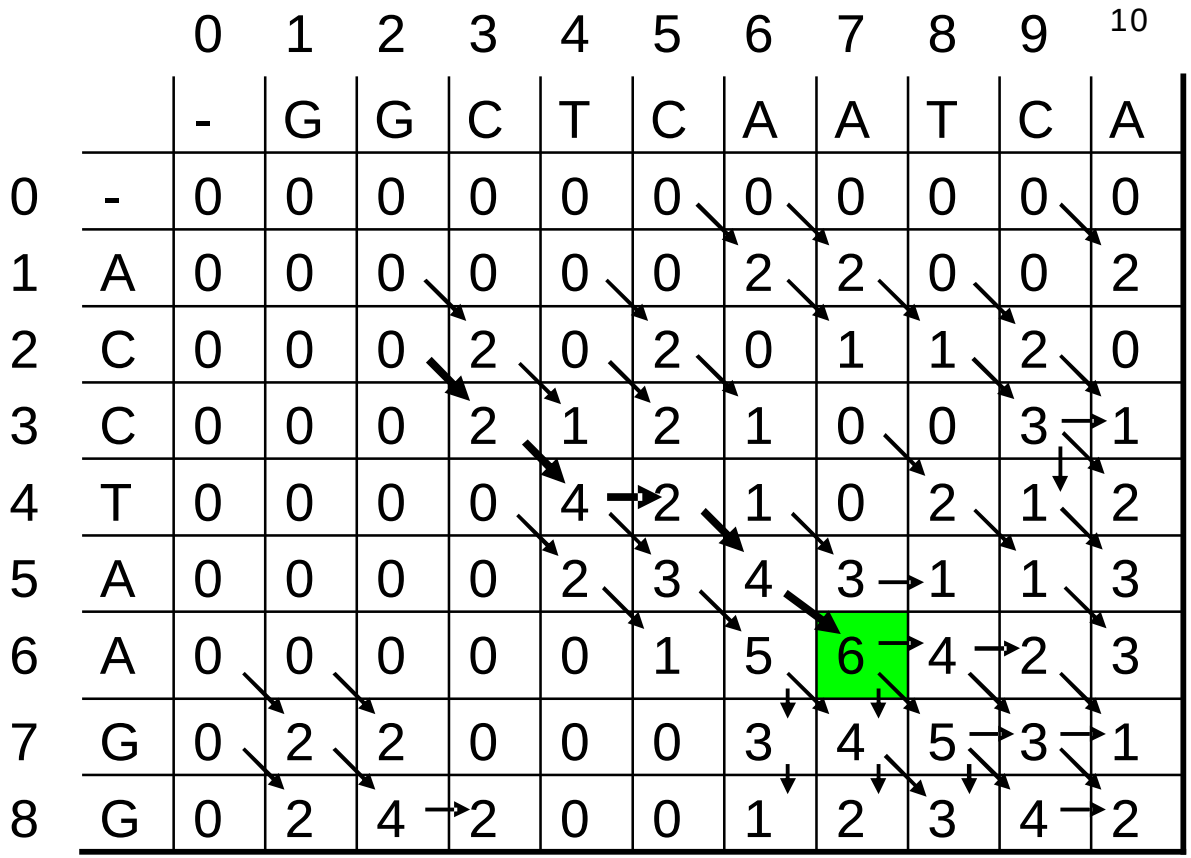
Scoring (for example)
Match: +2
Mismatch: -1
Indel: -2

Multiple optimal alignments
Non-optimal, good-scoring alignments

How can you find

1. Optimal alignments if more than one exist?

2. Non-optimal, good-scoring alignments?



Overlap alignment

- Overlap matrix used by Overlap-Layout-Consensus algorithm can be computed with dynamic programming
- Initialization: $O_{i,0} = O_{0,j} = 0$ for all i, j
- Recursion:

$$O_{i,j} = \max \{ O_{i-1,j-1} + s(a_i, b_i), O_{i-1,j} - \delta, O_{i,j-1} - \delta, \}$$

Best overlap: maximum value from rightmost column and bottom row

Non-uniform mismatch penalties

- We used uniform penalty for mismatches:

  s('A', 'C') = s('A', 'G') = … = s('G', 'T') = μ

- Transition mutations (A->G, G->A, C->T, T->C) are approximately twice as frequent than transversions (A->T, T->A, A->C, G->T)
  - use non-uniform mismatch penalties collected into a *substitution matrix*

	A	C	G	T
A	1	-1	-0.5	-1
C	-1	1	-1	-0.5
G	-0.5	-1	1	-1
T	-1	-0.5	-1	1

Gaps in alignment

- Gap is a succession of indels in alignment

```
C T – - - A A
C T C G C A A
```

- Previous model scored a length k gap as $w(k) = -k\delta$
- Replication processes may produce longer stretches of insertions or deletions
  - In coding regions, insertions or deletions of codons may preserve functionality

Gap open and extension penalties (2)

- We can design a score that allows the penalty opening gap to be larger than extending the gap:

$$w(k) = -\alpha - \beta(k - 1)$$

- Gap open cost α, Gap extension cost β
- Alignment algorithms can be extended to use w(k) (not discussed on this course)

Amino acid sequences

- We have discussed mainly DNA sequences
- Amino acid sequences can be aligned as well
- However, the design of the substitution matrix is more involved because of the larger alphabet
- More on the topic in the course Biological sequence analysis

Demonstration of the EBI web site

- European Bioinformatics Institute (EBI) offers many biological databases and bioinformatics tools at http://www.ebi.ac.uk/
  - Sequence alignment: Tools -> Sequence Analysis -> Align

Sequence Alignment (chapter 6)

- The biological problem
- Global alignment
- Local alignment
- *Multiple alignment*

Multiple alignment

- Consider a set of n sequences on the right
  - Orthologous sequences from different organisms
  - Paralogs from multiple duplications
- How can we study relationships between these sequences?

aggcgagctgcgagtgcta
cgttagattgacgctgac
ttccggctgcgac
gacacggcgaacgga
agtgtgcccgacgagcgaggac
gcgggctgtgagcgcta
aagcggcctgtgtgccccta
atgctgctgccagtgta
agtcgagccccgagtgc
agtccgagtcc
actcggtgc

Optimal alignment of three sequences

- Alignment of A = $a_1a_2...a_i$ and B = $b_1b_2...b_j$ can end either in (-, b_j), (a_i, b_j) or (a_i, -)
- $2^2 - 1 = 3$ alternatives
- Alignment of A, B and C = $c_1c_2...c_k$ can end in $2^3 - 1$ ways: (a_i, -, -), (-, b_j, -), (-, -, c_k), (-, b_j, c_k), (a_i, -, c_k), (a_i, b_j, -) or (a_i, b_j, c_k)
- Solve the recursion using three-dimensional dynamic programming matrix: $O(n^3)$ time and space
- Generalizes to n sequences but impractical with even a moderate number of sequences

Multiple alignment in practice

- In practice, real-world multiple alignment problems are usually solved with heuristics
- Progressive multiple alignment
  - Choose two sequences and align them
  - Choose third sequence w.r.t. two previous sequences and align the third against them
  - Repeat until all sequences have been aligned
  - Different options how to choose sequences and score alignments
  - Note the similarity to Overlap-Layout-Consensus

Multiple alignment in practice

- Profile-based progressive multiple alignment: CLUSTALW
  - Construct a distance matrix of all pairs of sequences using dynamic programming
  - Progressively align pairs in order of decreasing similarity
  - CLUSTALW uses various heuristics to contribute to accuracy

Additional material

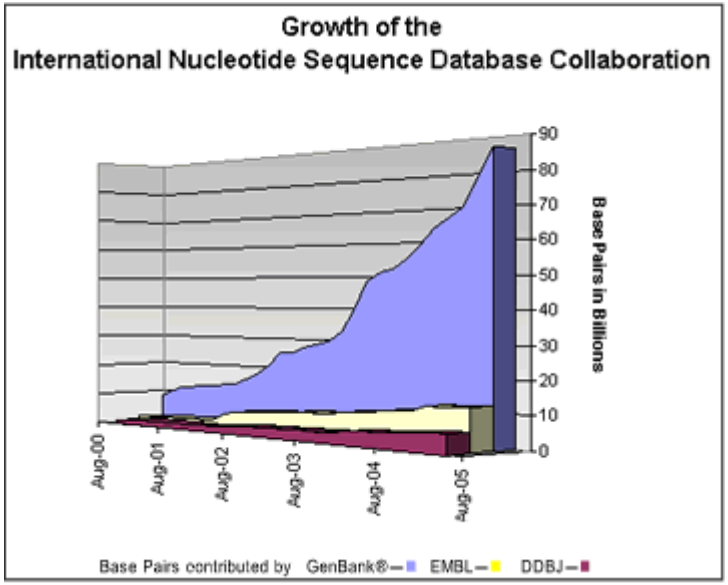
- R. Durbin, S. Eddy, A. Krogh, G. Mitchison: Biological sequence analysis
- N. C. Jones, P. A. Pevzner: An introduction to bioinformatics algorithms
- Course Biological sequence analysis in period II, 2008

Rapid alignment methods: FASTA and BLAST

- *The biological problem*
- *Search strategies*
- FASTA
- BLAST

The biological problem

- Global and local alignment algoritms are slow in practice
- Consider the scenario of aligning a *query sequence* against a large database of sequences
  - New sequence with unknown function



- NCBI GenBank size in January 2007 was 65 369 091 950 bases (61 132 599 sequences)
- Feb 2008: 85 759 586 764 bases (82 853 685 sequences)

Problem with large amount of sequences

- Exponential growth in both number and total length of sequences
- Possible solution: Compare against model organisms only
- With large amount of sequences, chances are that matches occur by random
  - Need for statistical analysis

Rapid alignment methods: FASTA and BLAST

- The biological problem
- Search strategies
- *FASTA*
- BLAST

FASTA

- FASTA is a multistep algorithm for sequence alignment (Wilbur and Lipman, 1983)
- The sequence file format used by the FASTA software is widely used by other sequence analysis software
- Main idea:
  - Choose regions of the two sequences (query and database) that look promising (have some degree of similarity)
  - Compute local alignment using dynamic programming in these regions

FASTA outline

- FASTA algorithm has five steps:
  - *1. Identify common k-words between I and J*
  - 2. Score diagonals with k-word matches, identify 10 best diagonals
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - 5. Perform dynamic programming to find final alignments

Search strategies

- How to speed up the computation?
  - Find ways to limit the number of pairwise comparisons
- Compare the sequences at word level to find out common words
  - Word means here a k-tuple (or a k-word), a substring of length k

Analyzing the word content

- Example query string I: TGATGATGAAGACATCAG
- For k = 8, the set of k-words (substring of length k) of I is

TGATGATG

GATGATGA

ATGATGAA

TGATGAAG

...

GACATCAG

Analyzing the word content

- There are n-k+1 k-words in a string of length n
- If at least one word of I is not found from another string J, we know that I differs from J
- Need to consider statistical significance: I and J might share words by chance only
- Let n=|I| and m=|J|

Word lists and comparison by content

- The k-words of I can be arranged into a table of word occurences $L_w(I)$
- Consider the k-words when k=2 and I=GCATCGGC:

GC, CA, AT, TC, CG, GG, GC

AT: 3

CA: 2

CG: 5

GC: 1, 7 ← Start indecies of k-word GC in I

GG: 6

TC: 4

Building $L_w(I)$ takes O(n) time

Common k-words

- Number of common k-words in I and J can be computed using $L_w(I)$ and $L_w(J)$
- For each word w in I, there are $|L_w(J)|$ occurences in J
- Therefore I and J have $\sum_w |L_w(I)||L_w(J)|$ common words
- This can be computed in $O(n + m + 4^k)$ time
  - $O(n + m)$ time to build the lists
  - $O(4^k)$ time to calculate the sum (in DNA strings)

Common k-words

- I = GCATCGGC
- J = CCATCGCCATCG

L_w(I)	L_w(J)	Common words
AT: 3	AT: 3, 9	2
CA: 2	CA: 2, 8	2
	CC: 1, 7	0
CG: 5	CG: 5, 11	2
GC: 1, 7	GC: 6	2
GG: 6		0
TC: 4	TC: 4, 10	2
		10 in total

Properties of the common word list

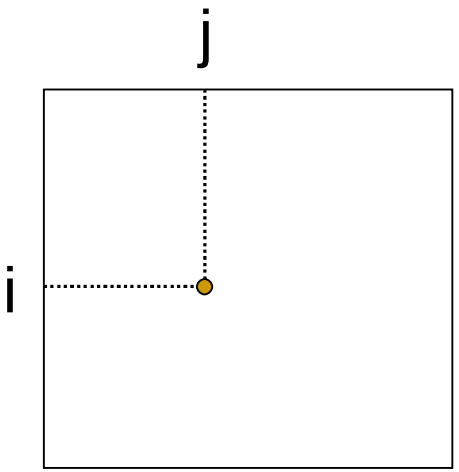
- Exact matches can be found using binary search (e.g., where TCGT occurs in I?)
  - $O(\log 4^k)$ time
- For large k, the table size is too large to compute the common word count in the previous fashion
- Instead, an approach based on merge sort can be utilised (details skipped)
- The common k-word technique can be combined with the local alignment algorithm to yield a rapid alignment approach

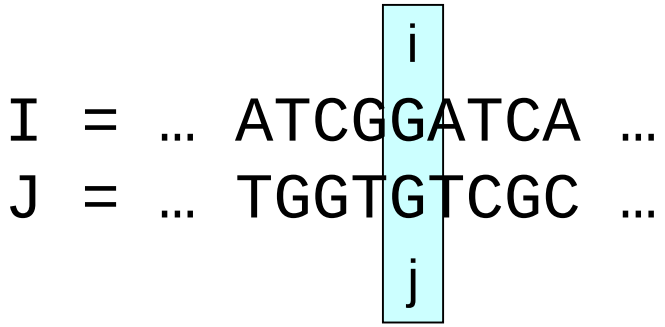
FASTA outline

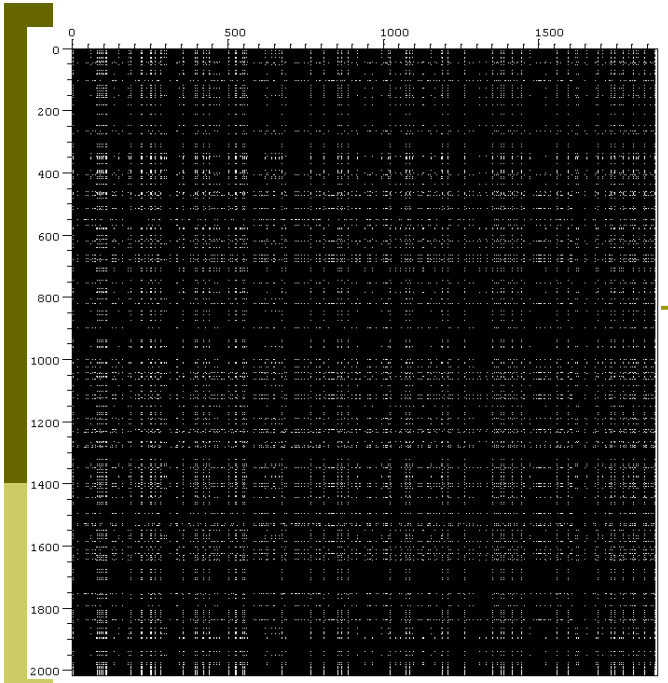
- FASTA algorithm has five steps:
  - 1. Identify common k-words between I and J
  - *2. Score diagonals with k-word matches, identify 10 best diagonals*
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - 5. Perform dynamic programming to find final alignments

Dot matrix comparisons

- Word matches in two sequences I and J can be represented as a *dot matrix*
- Dot matrix element (i, j) has ”a dot”, if the word starting at position i in I is identical to the word starting at position j in J
- The dot matrix can be plotted for various k

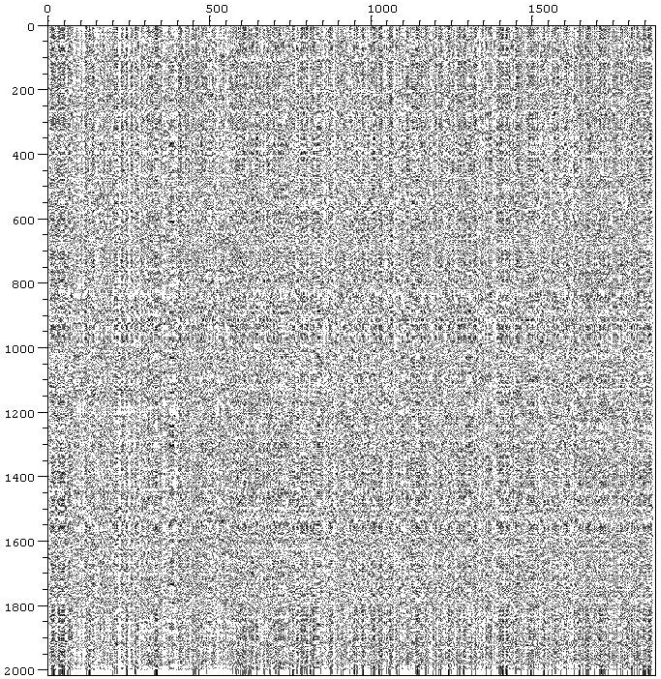




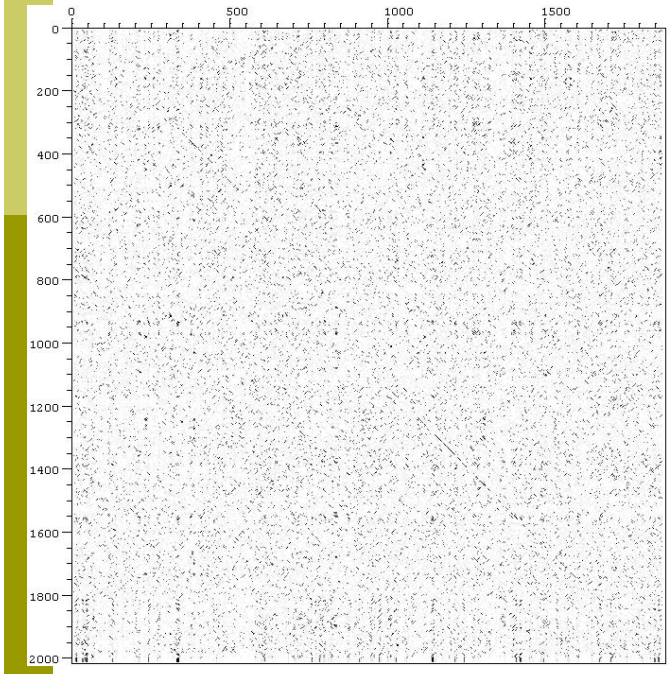


k=1

k=4

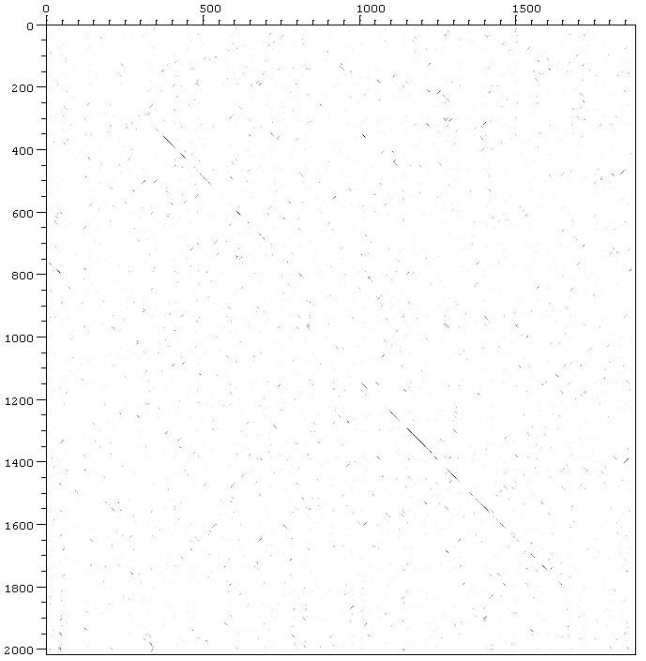


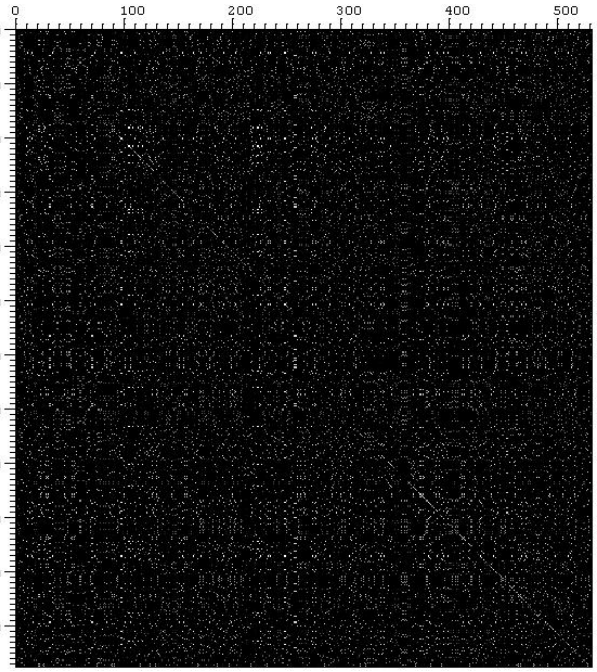
Dot matrix (k=1,4,8,16)
for two **DNA** sequences
X85973.1 (1875 bp)
Y11931.1 (2013 bp)



k=8

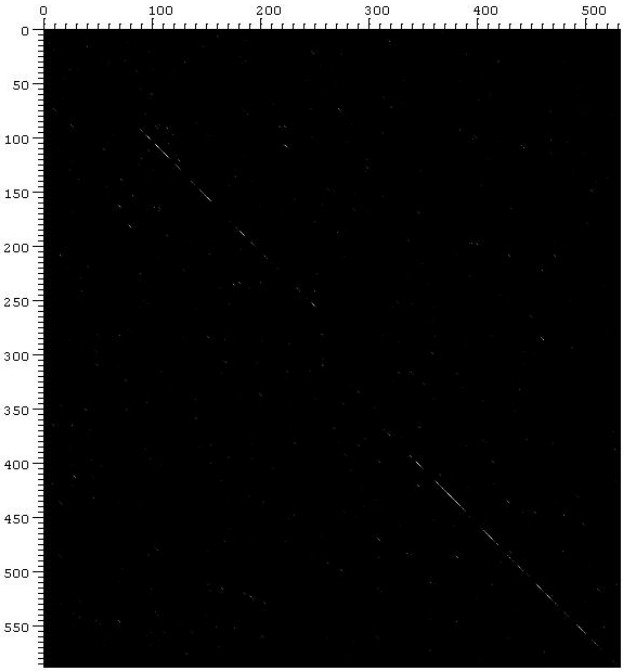
k=16



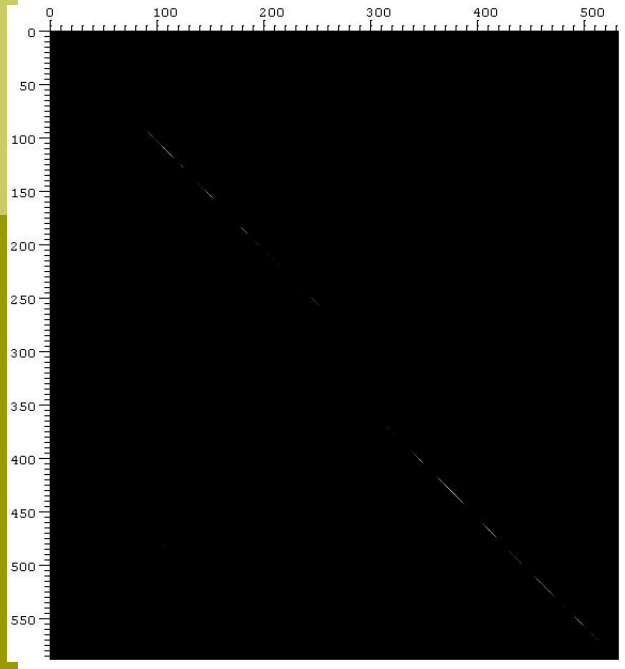


k=1

k=4

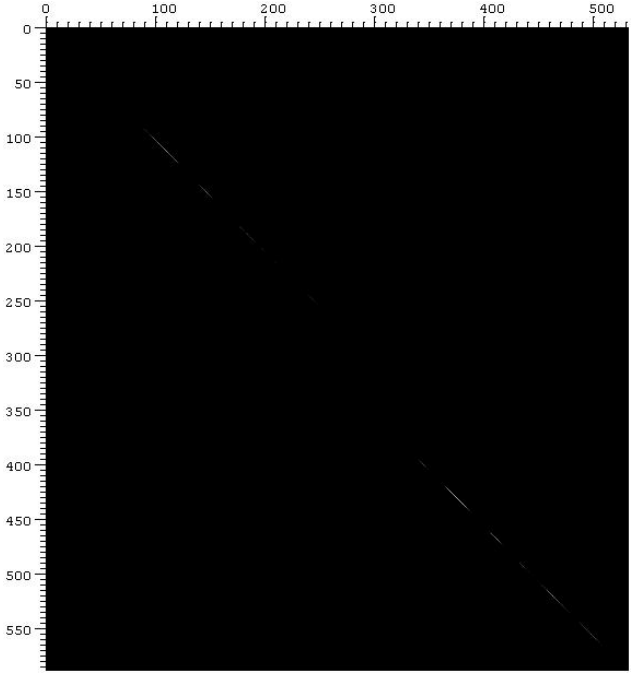


Dot matrix (k=1,4,8,16) for two **protein** sequences
CAB51201.1 (531 aa)
CAA72681.1 (588 aa)



k=8

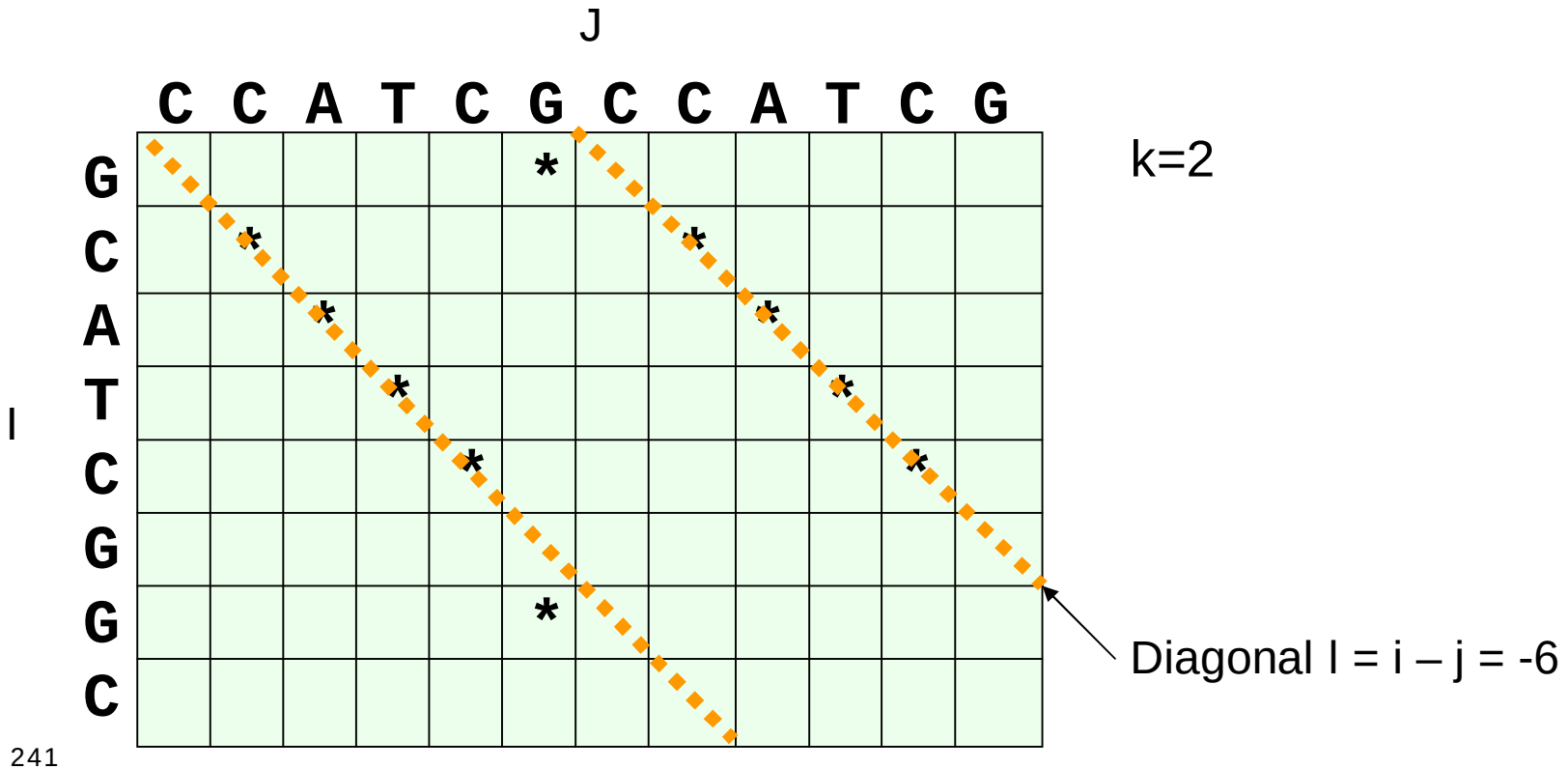
k=16



Shading indicates now the match score according to a score matrix (Blosum62 here)

Computing diagonal sums

- We would like to find high scoring diagonals of the dot matrix
- Lets index diagonals by the offset, l = i - j



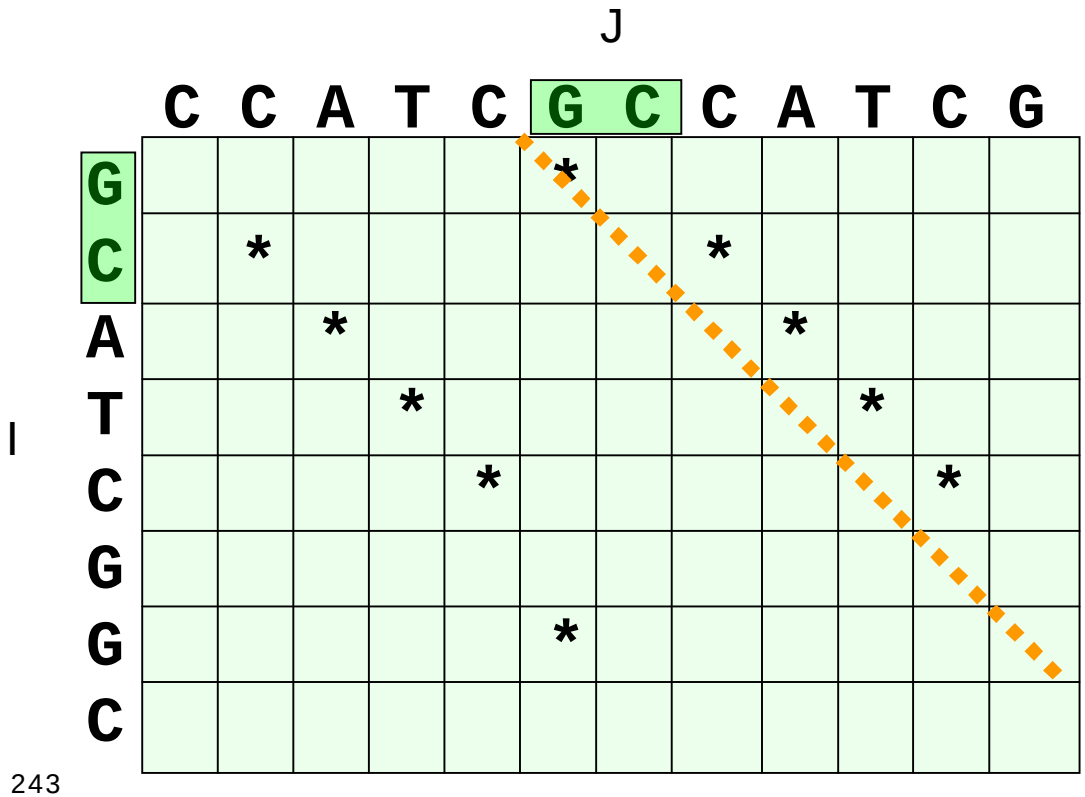
Computing diagonal sums

- As an example, lets compute diagonal sums for I = GCATCGGC, J = CCATCGCCATCG, k = 2
- 1. Construct k-word list $L_w(J)$
- 2. Diagonal sums S_l are computed into a table, indexed with the offset and initialised to zero

l	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
S_l	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Computing diagonal sums

- 3. Go through k-words of I, look for matches in $L_w(J)$ and update diagonal sums

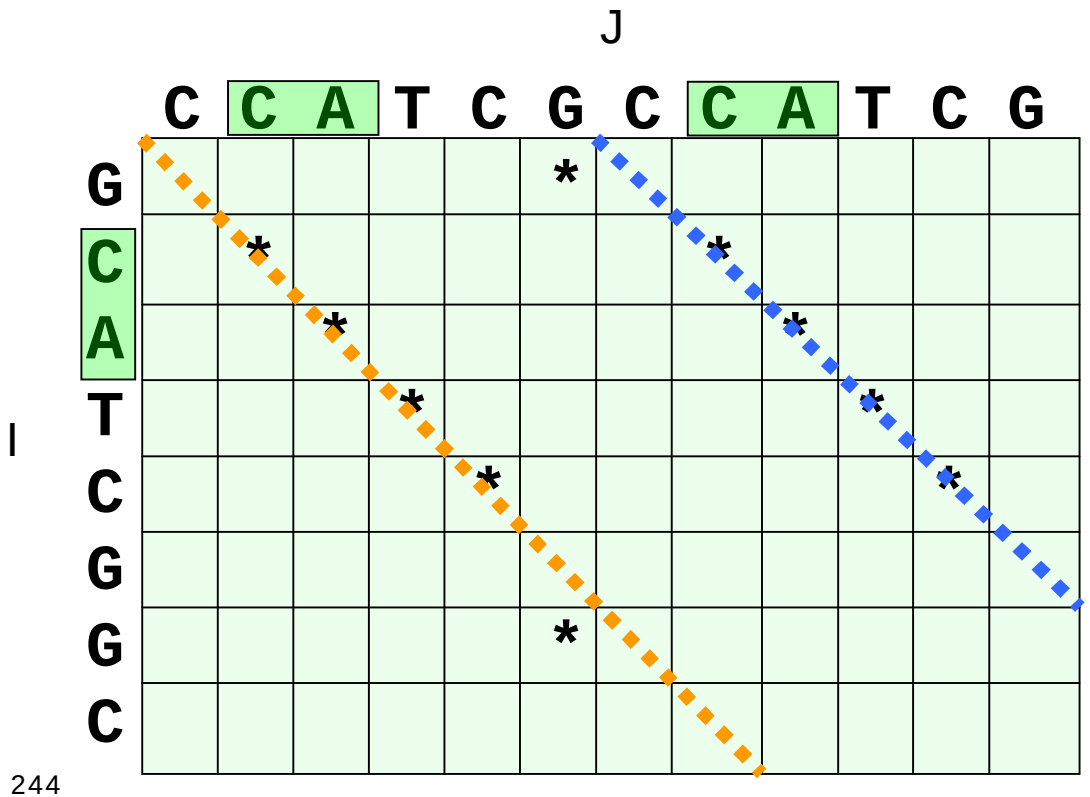


For the first 2-word in I, GC, $L_{GC}(J) = \{6\}$.

We can then update the sum of diagonal $I = i - j = 1 - 6 = -5$ to $S_{-5} := S_{-5} + 1 = 0 + 1 = 1$

Computing diagonal sums

- 3. Go through k-words of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-word in I is CA, for which $L_{CA}(J) = \{2, 8\}$.

Two diagonal sums are updated:

$I = i - j = 2 - 2 = 0$

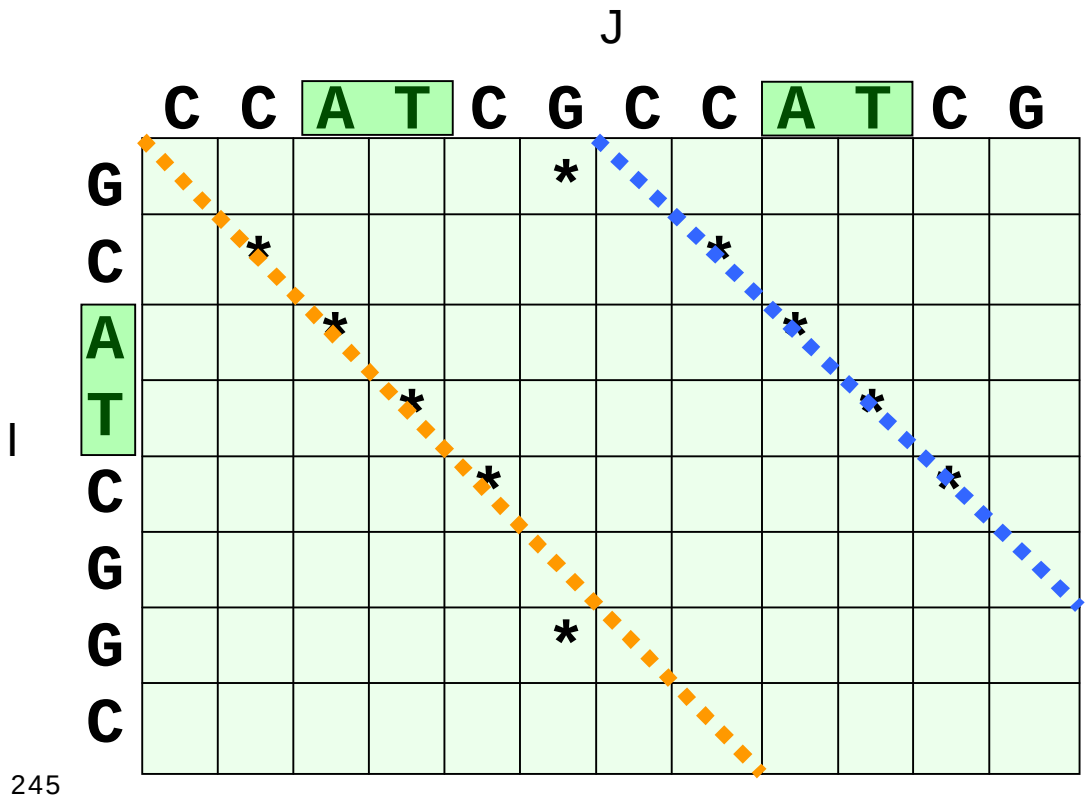
$S_0 := S_0 + 1 = 0 + 1 = 1$

$I = i - j = 2 - 8 = -6$

$S_{-6} := S_{-6} + 1 = 0 + 1 = 1$

Computing diagonal sums

- 3. Go through k-words of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-word in I is AT, for which $L_{AT}(J) = \{3, 9\}$.

Two diagonal sums are updated:

$I = i - j = 3 - 3 = 0$

$S_0 := S_0 + 1 = 1 + 1 = 2$

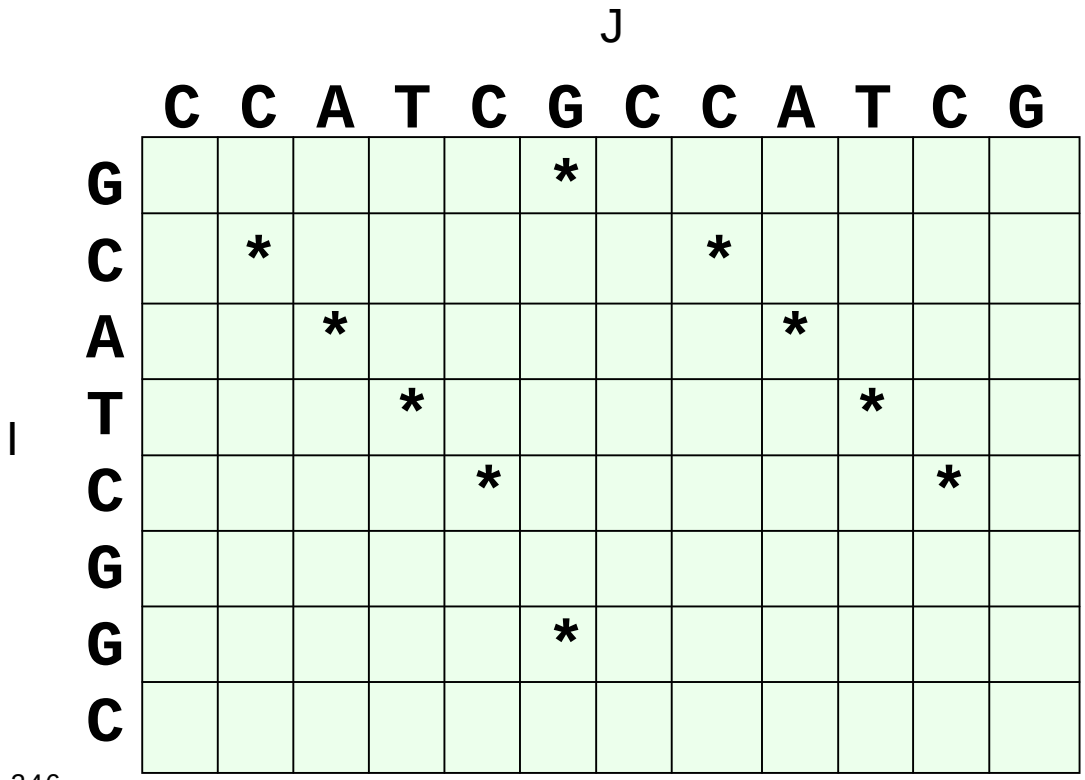
$I = i - j = 3 - 9 = -6$

$S_{-6} := S_{-6} + 1 = 1 + 1 = 2$

Computing diagonal sums

After going through the k-words of I, the result is:

l	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
S_l	0	0	0	0	4	1	0	0	0	0	4	1	0	0	0	0	0



Algorithm for computing diagonal sum of scores

$S_l := 0$ for all $1 - m \leq l \leq n - 1$

Compute $L_w(J)$ for all words w

for $i := 1$ to $n - k - 1$ do

  $w := I_i I_{i+1} \ldots I_{i+k-1}$

  for $j \in L_w(J)$ do

    $l := i - j$

    $S_l := S_l + 1$ ← Match score is here 1

  end

end

FASTA outline

- FASTA algorithm has five steps:
  - 1. Identify common k-words between I and J
  - 2. Score diagonals with k-word matches, identify 10 best diagonals
  - *3. Rescore initial regions with a substitution score matrix*
  - *4. Join initial regions using gaps, penalise for gaps*
  - 5. Perform dynamic programming to find final alignments

Rescoring initial regions

- Each high-scoring diagonal chosen in the previous step is rescored according to a score matrix
- This is done to find subregions with identities shorter than k
- Non-matching ends of the diagonal are trimmed

```
I: C C A T C G C C A T C G
J: C C A A C G C A A T C A
```

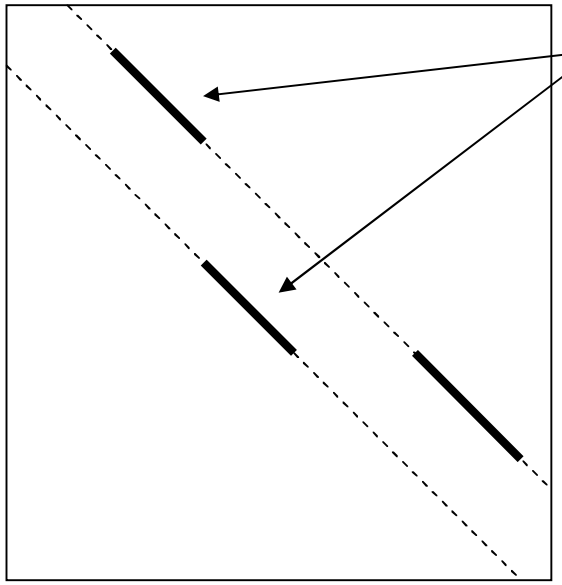
75% identity, no 4-word identities

```
I': C C A T C G C C A T C G
J': A C A T C A A A T A A A
```

33% identity, one 4-word identity

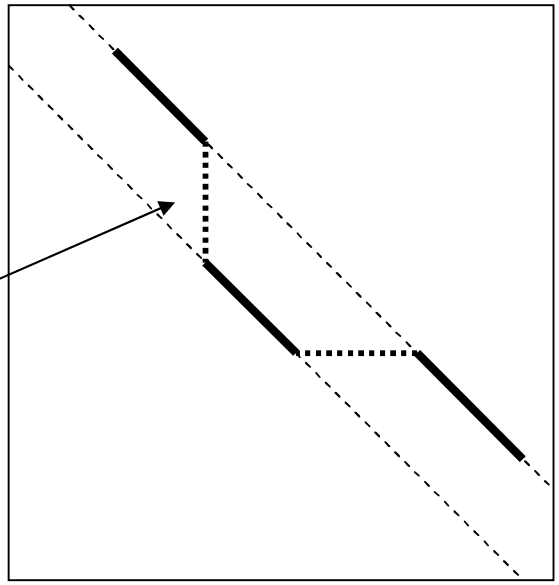
Joining diagonals

- Two offset diagonals can be joined with a gap, if the resulting alignment has a higher score
- Separate gap open and extension are used
- Find the best-scoring combination of diagonals



High-scoring diagonals

Two diagonals joined by a gap

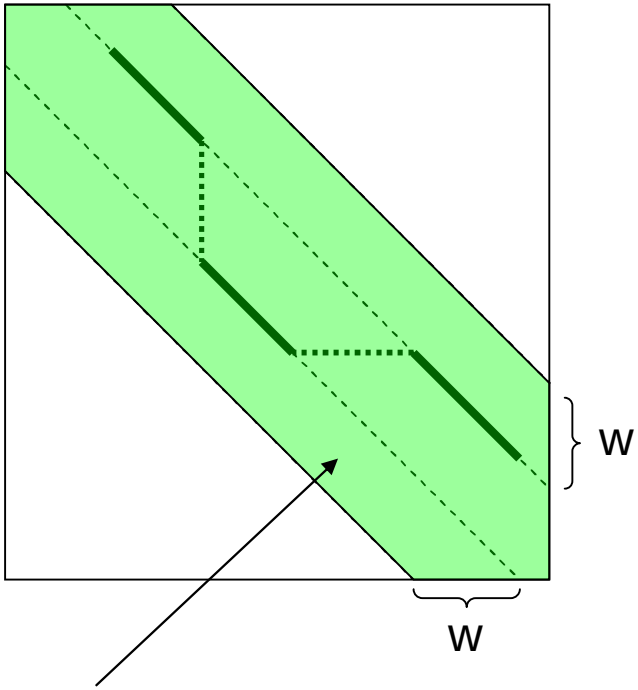


FASTA outline

- FASTA algorithm has five steps:
  - 1. Identify common k-words between I and J
  - 2. Score diagonals with k-word matches, identify 10 best diagonals
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - *5. Perform dynamic programming to find final alignments*

Local alignment in the highest-scoring region

- Last step of FASTA: perform local alignment using dynamic programming around the highest-scoring
- Region to be aligned covers –w and +w offset diagonal to the highest-scoring diagonals
- With long sequences, this region is typically very small compared to the whole n x m matrix



Dynamic programming matrix
M filled only for the green region

Properties of FASTA

- Fast compared to local alignment using dynamic programming only
  - Only a narrow region of the full matrix is aligned
- Increasing parameter k decreases the number of hits:
  - Increases specificity
  - Decreases sensitivity
  - Decreases running time
- FASTA can be very specific when identifying long regions of low similarity
  - Specific method does not find many incorrect results
  - Sensitive method finds many of the correct results

Properties of FASTA

- FASTA looks for initial exact matches to query sequence
  - Two proteins can have very different amino acid sequences and still be biologically similar
  - This may lead into a lack of sensitivity with diverged sequences

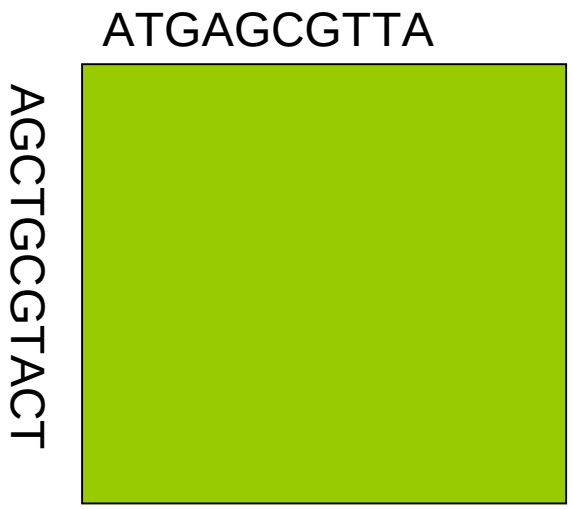
Demonstration of FASTA at EBI

- http://www.ebi.ac.uk/fasta/
- Note that parameter ktup in the software corresponds to parameter k in lectures

Note on sequences and alignment matrices in exercises

- Example solutions to alignment problems will have sequences arranged like this:
  - Perform global alignment of the sequences
    - s = AGCTGCGTACT
    - t = ATGAGCGTTA

So if you want to be able to compare your solution easily against the example, use this convention.



Rapid alignment methods: FASTA and BLAST

- The biological problem
- Search strategies
- FASTA
- *BLAST*

BLAST: Basic Local Alignment Search Tool

- BLAST (Altschul et al., 1990) and its variants are some of the most common sequence search tools in use
- Roughly, the basic BLAST has three parts:
  - 1. Find *segment pairs* between the query sequence and a database sequence above score threshold (”seed hits”)
  - 2. Extend seed hits into *locally maximal segment pairs*
  - 3. Calculate p-values and a rank ordering of the local alignments
- Gapped BLAST introduced in 1997 allows for gaps in alignments

Finding seed hits

- First, we generate a set of *neighborhood sequences* for given k, *match score matrix and threshold* T
- Neighborhood sequences of a k-word w include all strings of length k that, when aligned against w, have the alignment score at least T
- For instance, let I = GCATCGGC, J = CCATCGCCATCG and k = 5, match score be 1, mismatch score be 0 and T = 4

Finding seed hits

- I = GCATCGGC, J = CCATCGCCATCG, k = 5, match score 1, mismatch score 0, T = 4
- This allows for one mismatch in each k-word
- The neighborhood of the first k-word of I, GCATC, is GCATC and the 15 sequences

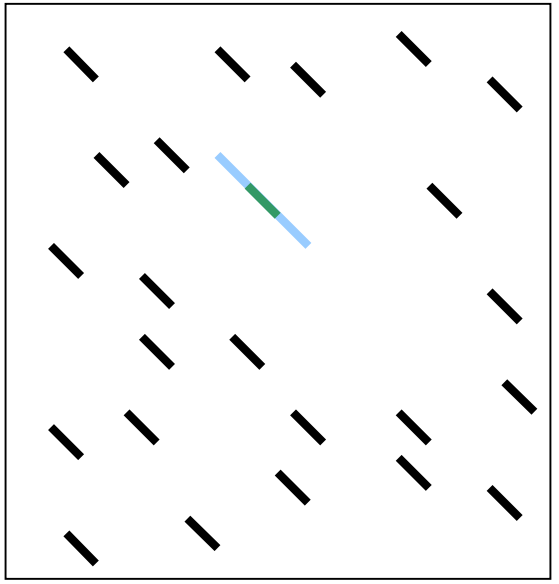
$$\begin{Bmatrix} A \\ CCATC, G \\ T \end{Bmatrix} \begin{Bmatrix} A \\ GATC, GC \\ T \end{Bmatrix} \begin{Bmatrix} C \\ GTC, GCA \\ T \end{Bmatrix} \begin{Bmatrix} A \\ CC, GCAT \\ G \end{Bmatrix} \begin{Bmatrix} A \\ G \\ T \end{Bmatrix}$$

Finding seed hits

- I = GCATCGGC has 4 k-words and thus 4x16 = 64 5-word patterns to locate in J
  - Occurences of patterns in J are called seed hits
- Patterns can be found using exact search in time proportional to the sum of pattern lengths + length of J + number of matches (Aho-Corasick algorithm)
  - Methods for pattern matching are developed on course 58093 String processing algorithms
- Compare this approach to FASTA

Extending seed hits: original BLAST

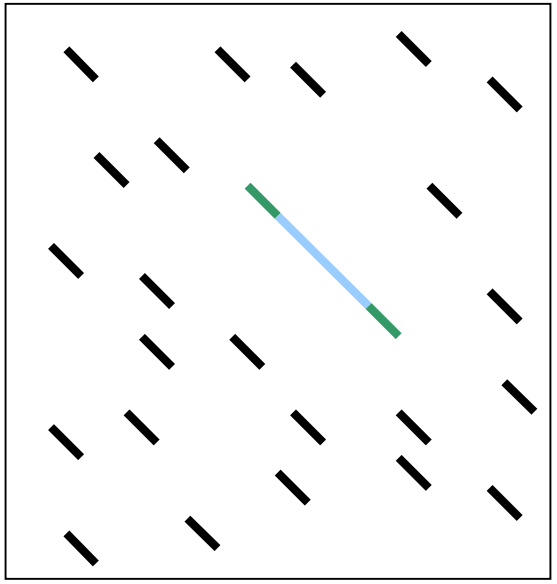
- Initial seed hits are extended into locally maximal segment pairs or High-scoring Segment Pairs (HSP)
- Extensions do not add gaps to the alignment
- Sequence is extended until the alignment score drops below the maximum attained score minus a threshold parameter value
- All statistically significant HSPs reported



Altschul, S.F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J., *J. Mol. Biol.*, 215, 403-410, 1990

Extending seed hits: gapped BLAST

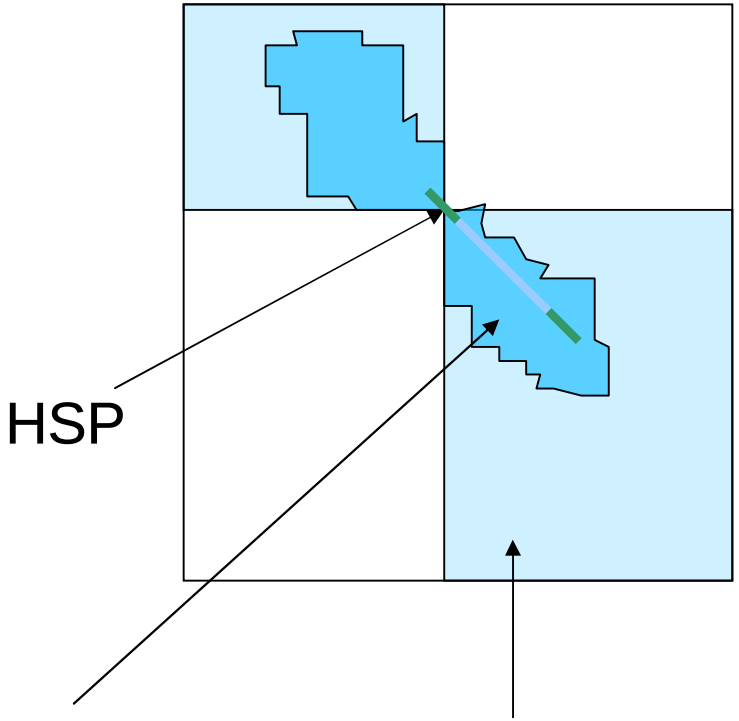
- In a later version of BLAST, two seed hits have to be found on the same diagonal
  - Hits have to be non-overlapping
  - If the hits are closer than A (additional parameter), then they are joined into a HSP
- Threshold value T is lowered to achieve comparable sensitivity
- If the resulting HSP achieves a score at least S_g, a *gapped extension* is triggered



Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, and Lipman DJ, *Nucleic Acids Res*. 1;25(17), 3389-402, 1997

Gapped extensions of HSPs

- Local alignment is performed starting from the HSP
- Dynamic programming matrix filled in ”forward” and ”backward” directions (see figure)
- Skip cells where value would be X_g below the best alignment score found so far



Estimating the significance of results

- In general, we have a score S(D, X) = s for a sequence X found in database D
- BLAST rank-orders the sequences found by p-values
- The p-value for this hit is P(S(D, Y) ≥ s) where Y is a random sequence
  - Measures the amount of ”surprise” of finding sequence X
- A smaller p-value indicates more significant hit
  - A p-value of 0.1 means that one-tenth of random sequences would have as large score as our result

Estimating the significance of results

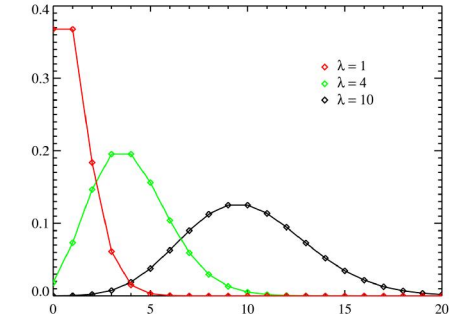
- In BLAST, p-values are computed roughly as follows
- There are nm places to begin an optimal alignment in the n x m alignment matrix
- Optimal alignment is preceded by a mismatch and has t matching (identical) letters
  - (Assume match score 1 and mismatch/indel score $-\infty$)
- Let p = P(two random letters are equal)
- The probability of having a mismatch and then t matches is $(1-p)p^t$

Estimating the significance of results

- We model this event by a Poisson distribution (why?) with mean $\lambda = nm(1-p)p^t$
- P(there is local alignment t or longer)

  $\approx 1 - P(\text{no such event})$

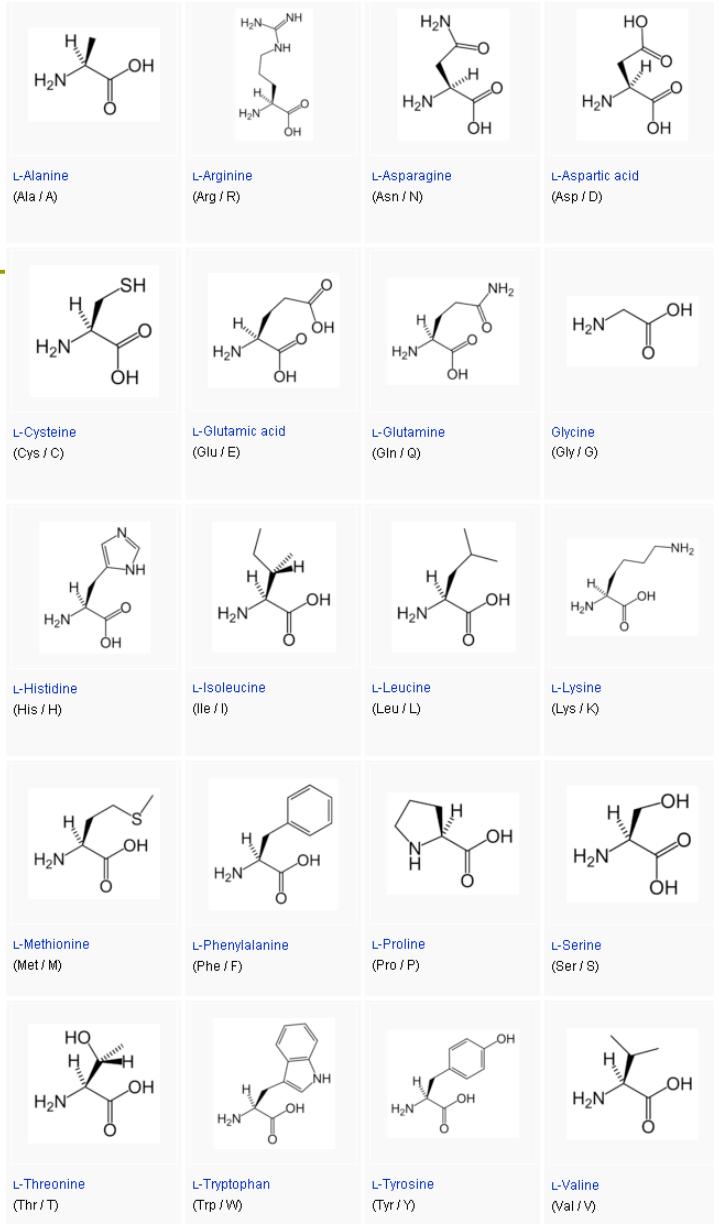
  $= 1 - e^{-\lambda} = 1 - \exp(-nm(1-p)p^t)$



- An equation of the same form is used in Blast:
- E-value = $P(S(D, Y) \geq s) \approx 1 - \exp(-nm\gamma\xi^t)$ where $\gamma > 0$ and $0 < \xi < 1$
- Parameters γ and ξ are estimated from data

Scoring amino acid alignments

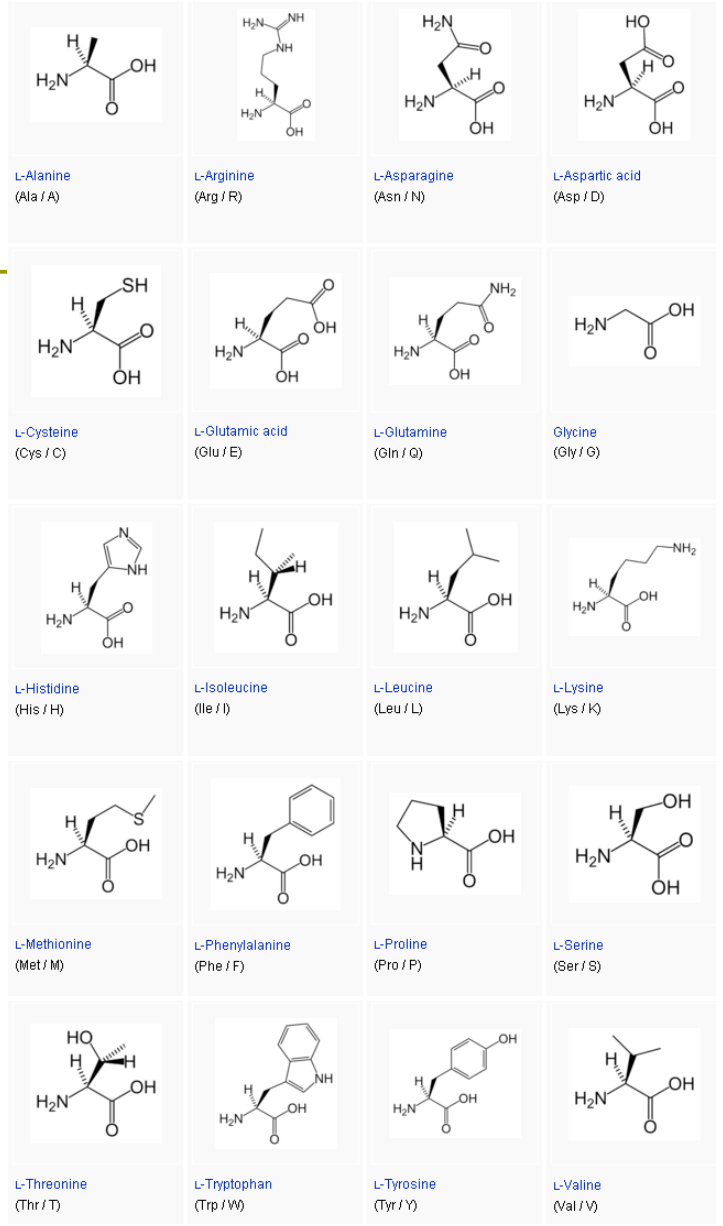
- We need a way to compute the score S(D, X) for aligning the sequence X against database D
- Scoring DNA alignments was discussed previously
- Constructing a scoring model for amino acids is more challenging
  - 20 different amino acids vs. 4 bases
- Figure shows the molecular structures of the 20 amino acids



http://en.wikipedia.org/wiki/List_of_standard_amino_acids

Scoring amino acid alignments

- Substitutions between chemically similar amino acids are more frequent than between dissimilar amino acids
- We can check our scoring model against this



http://en.wikipedia.org/wiki/List_of_standard_amino_acids

Score matrices

- Scores s = S(D, X) are obtained from score matrices
- Let A = $A_1a_2...a_n$ and B = $b_1b_2...b_n$ be sequences of equal length (no gaps allowed to simplify things)
- To obtain a score for alignment of A and B, where a_i is aligned against b_i, we take the ratio of two probabilities
  - The probability of having A and B where the characters match (match model M)
  - The probability that A and B were chosen randomly (random model R)

Score matrices: random model

- Under the random model, the probability of having X and Y is

$$P(A, B|R) = \prod_i q_{ai} \prod_i q_{bi}$$

  where q_{xi} is the probability of occurence of amino acid type x_i

- Position where an amino acid occurs does not affect its type

Score matrices: match model

- Let p_{ab} be the probability of having amino acids of type a and b aligned against each other given they have evolved from the same ancestor c
- The probability is

$$P(A, B|M) = \prod_i p_{a_i b_i}$$

Score matrices: log-odds ratio score

- We obtain the score S by taking the ratio of these two probabilities

$$\frac{P(A,B|M)}{P(A,B|R)} = \frac{\prod_i p_{a_i b_i}}{\prod_i q_{a_i} \prod_i q_{b_i}} = \prod_i \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}}$$

and taking a logarithm of the ratio

$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^{n} \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^{n} s(a_i, b_i)$$

Score matrices: log-odds ratio score

$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^{n} \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^{n} s(a_i, b_i)$$

- The score S is obtained by summing over character pair-specific scores:

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

- The probabilities q_a and p_{ab} are extracted from data

Calculating score matrices for amino acids

- Probabilities q_a are in principle easy to obtain:
  - Count relative frequencies of every amino acid in a sequence database

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

Calculating score matrices for amino acids

- To calculate p_{ab} we can use a known pool of aligned sequences
- BLOCKS is a database of highly conserved regions for proteins
- It lists multiply aligned, ungapped and conserved protein segments
- Example from BLOCKS shows genes related to human gene associated with DNA-repair defect xeroderma pigmentosum

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

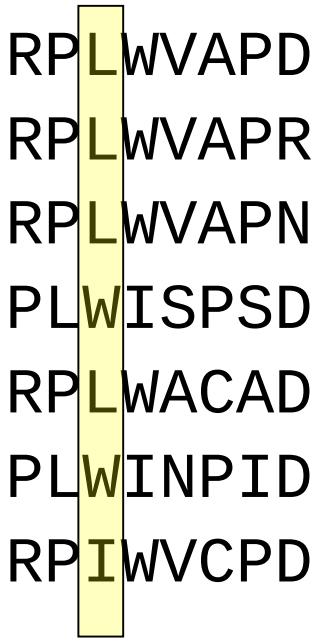
Block PR00851A
ID XRODRMPGMNTB; BLOCK
AC PR00851A; distance from previous block=(52,131)
DE Xeroderma pigmentosum group B protein signature
BL adapted; width=21; seqs=8; 99.5%=985; strength=1287

```
XPB_HUMAN|P19447 ( 74)    RPLWVAPDGHIFLEAFSPVYK 54
XPB_MOUSE|P49135 ( 74)    RPLWVAPDGHIFLEAFSPVYK 54
P91579 ( 80)              RPLYLAPDGHIFLESFSPVYK 67
XPB_DROME|Q02870 ( 84)    RPLWVAPNGHVFLESFSPVYK 79
RA25_YEAST|Q00578 ( 131) PLWISPSDGRIILESFSPLAE 100
Q38861 ( 52)              RPLWACADGRIFLETFSPLYK 71
O13768 ( 90)              PLWINPIDGRIILEAFSPLAE 100
O00835 ( 79)              RPIWVCPDGHIFLETFSAIYK 86
```

http://blocks.fhcrc.org

BLOSUM matrix

- BLOSUM is a score matrix for amino acid sequences derived from BLOCKS data
- First, count pairwise matches $f_{x,y}$ for every *amino acid type pair (x, y)*
- For example, for column 3 and amino acids L and W, we find 8 pairwise matches: $f_{L,W} = f_{W,L} = 8$



Creating a BLOSUM matrix

- Probability p_{ab} is obtained by dividing f_{ab} with the total number of pairs (note difference with course book):

$$p_{ab} = f_{ab} / \sum_{x=1}^{20} \sum_{y=1}^{x} f_{xy}$$

- We get probabilities q_a by

$$q_a = \sum_{b=1}^{20} p_{ab}$$

RPLWVAPD
RPLWVAPR
RPLWVAPN
PLWISPSD
RPLWACAD
PLWINPID
RPIWVCPD

Creating a BLOSUM matrix

- The probabilities p_{ab} and q_a can now be plugged into

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

  to get a 20 x 20 matrix of scores s(a, b).
- Next slide presents the BLOSUM62 matrix
  - Values scaled by factor of 2 and rounded to integers
  - Additional step required to take into account expected evolutionary distance
  - Described in Deonier's book in more detail

BLOSUM62

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Using BLOSUM62 matrix

```
MQLEANADTSV
  |   | |
LQEQAEAQGEM
```

$$s = \sum_{i=1}^{11} s(a_i, b_i)$$

= 2 + 5 – 3 – 4 + 4 + 0 + 4 + 0 – 2 + 0 + 1

= 7

Demonstration of BLAST at NCBI

- http://www.ncbi.nlm.nih.gov/BLAST/

Introduction to Bioinformatics

Lecture 4:

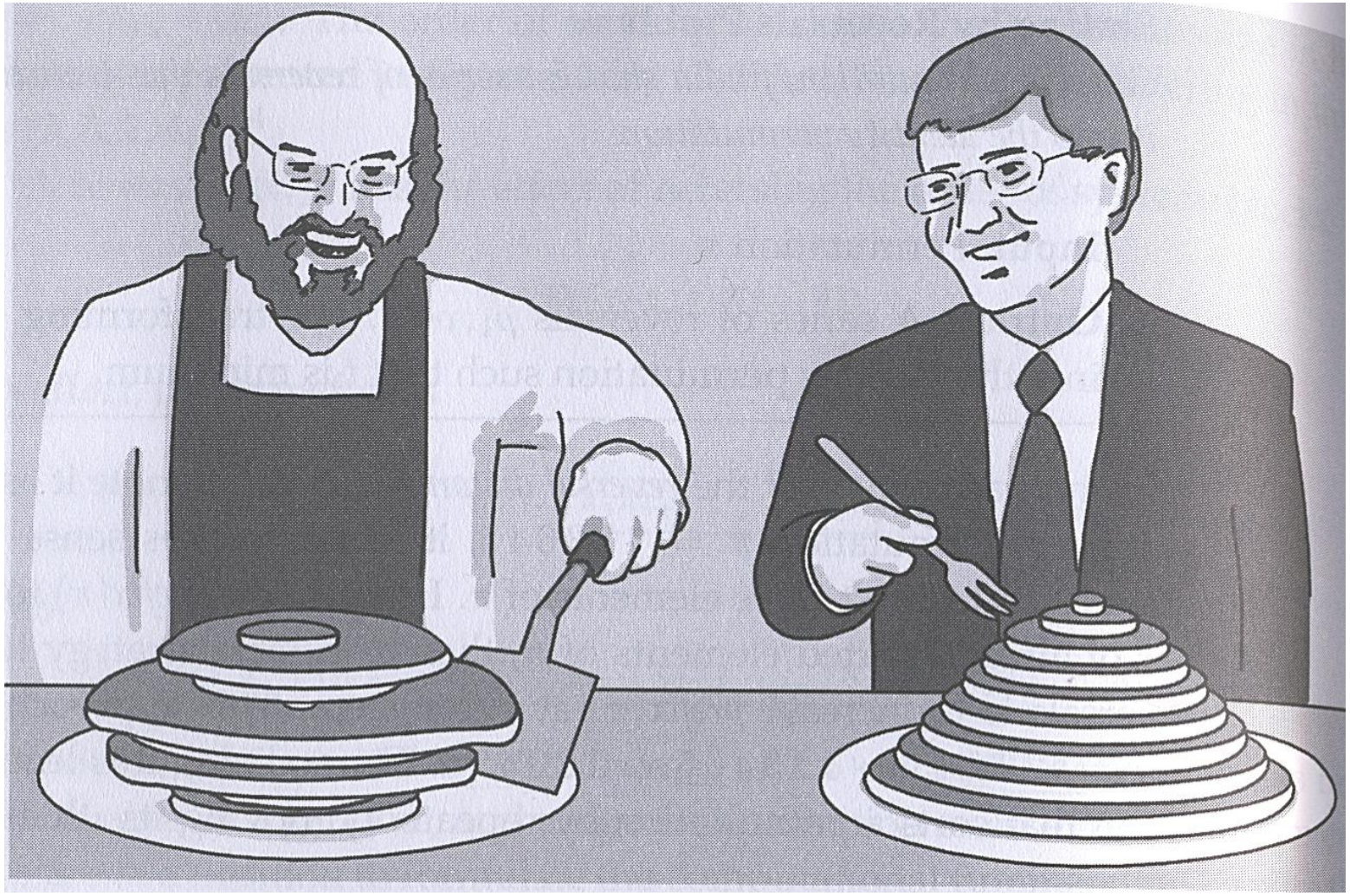
Genome rearrangements

Why study genome rearrangements?

- Provide insight into evolution of species
- Fun algorithmic problem!

- Structure of this lecture:
  - The biological phenomenon
  - How to computationally model it?
  - How to compute interesting things?
  - Studying the phenomenon using existing tools (continued in exercises)

Genome rearrangements as an algorithmic problem



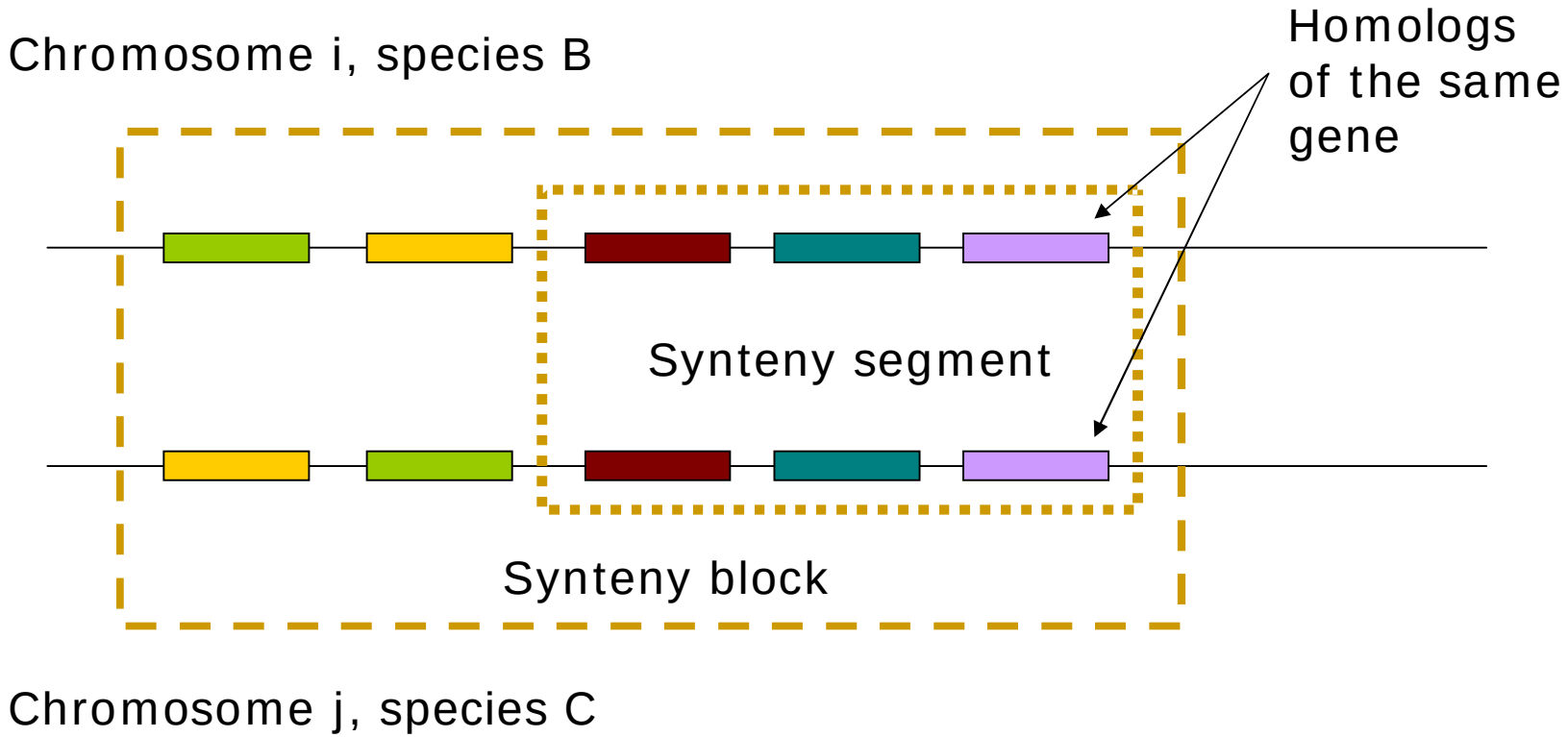
Background

- Genome sequencing enables us to compare genomes of two or more different species
  - -> Comparative genomics
- Basic observation:
  - Closely related species (such as human and mouse) can be almost identical in terms of genome contents...
  - ...but the order of genomic segments can be very different between species

Synteny blocks and segments

- Synteny – derived from Greek 'on the same ribbon' – means genomic segments located on the same chromosome
  - Genes, markers (any sequence)
- Synteny block (or syntenic block)
  - A set of genes or markers that co-occur together in two species
- Synteny segment (or syntenic segment)
  - Syntenic block where the *order* of genes or markers is preserved

Synteny blocks and segments

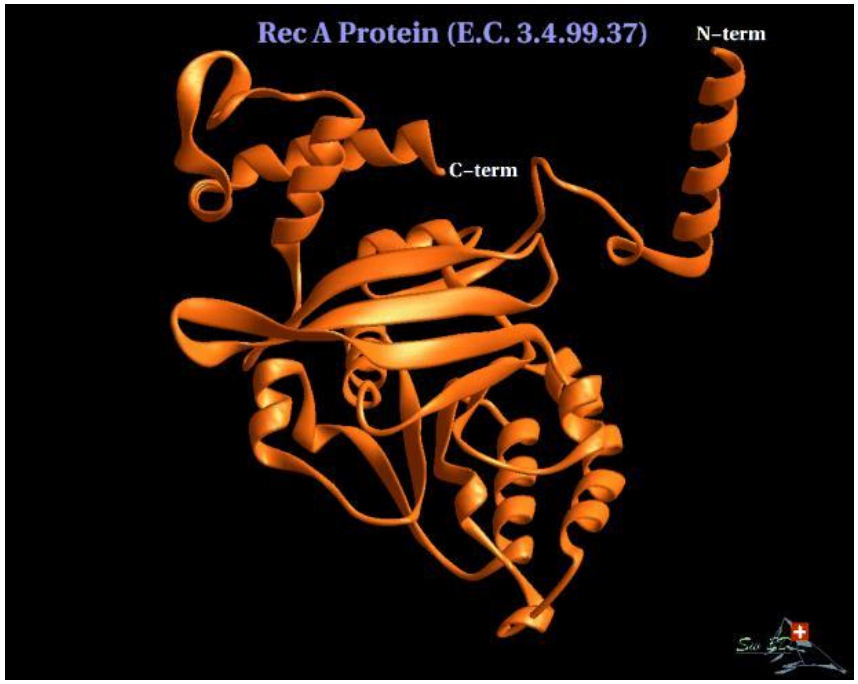


Observations from sequencing

1. Large chromosome *inversions* and *translocations* (we'll get to these shortly) are common
   - ...Even between closely related species
2. Chromosome inversions are usually symmetric around the *origin of DNA replication*
3. Inversions are less common *within species*...

What causes rearrangements?

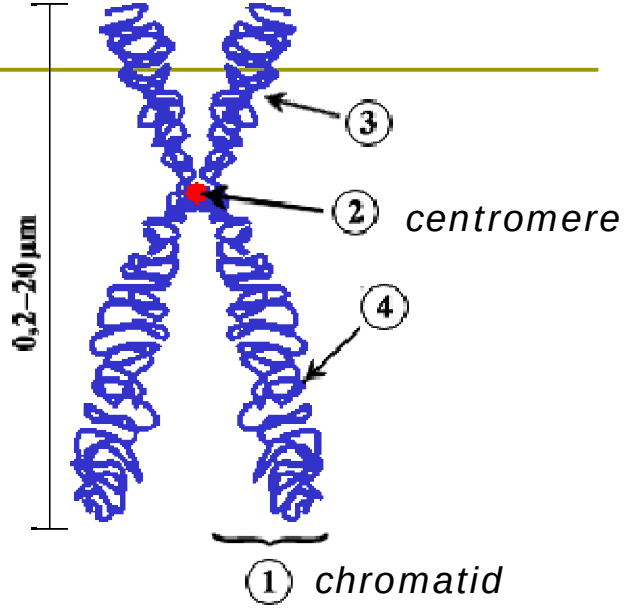
- RecA, Recombinase A, is a protein used to repair chromosomal damage
- It uses a duplicate copy of the damaged sequence as template
- Template is usually a homologous sequence on a *sister chromosome*



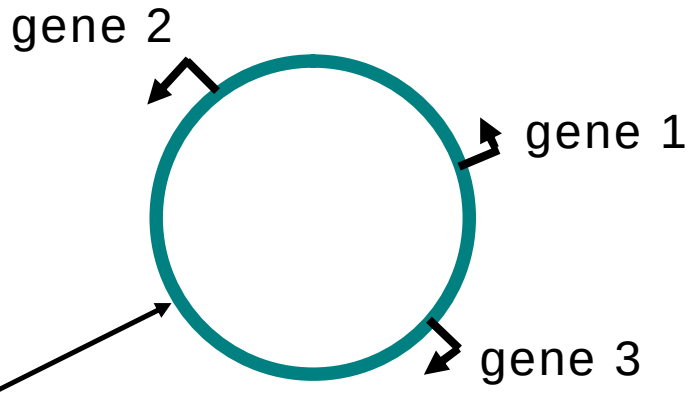
Diarmaid Hughes: Evaluating genome dynamics: the constraints on rearrangements within bacterial genomes, *Genome Biology* 2000, 1

Chromosomes: recap

- Linear chromosomes
  - Eukaryotes (mostly)



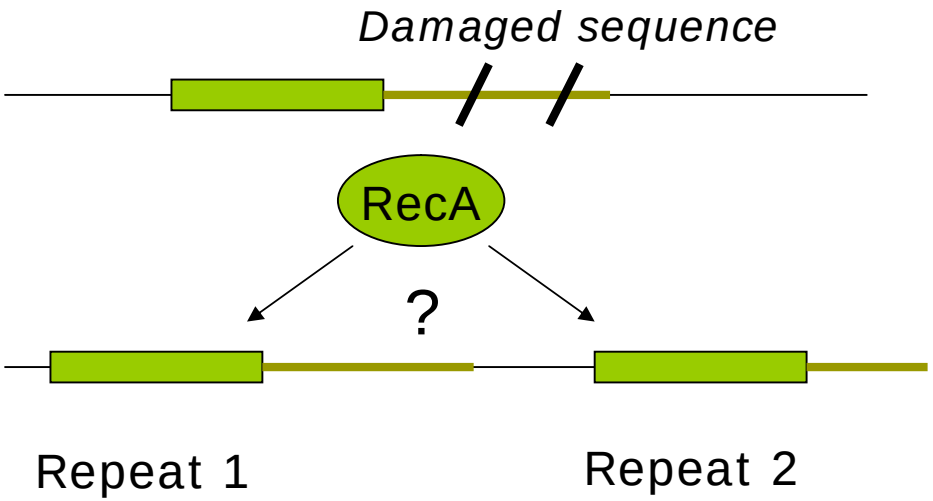
- Circular chromosomes
  - Prokaryotes (mostly)
  - Mitochondria

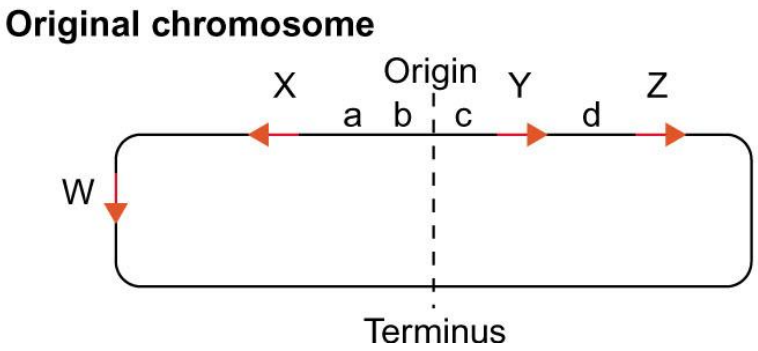


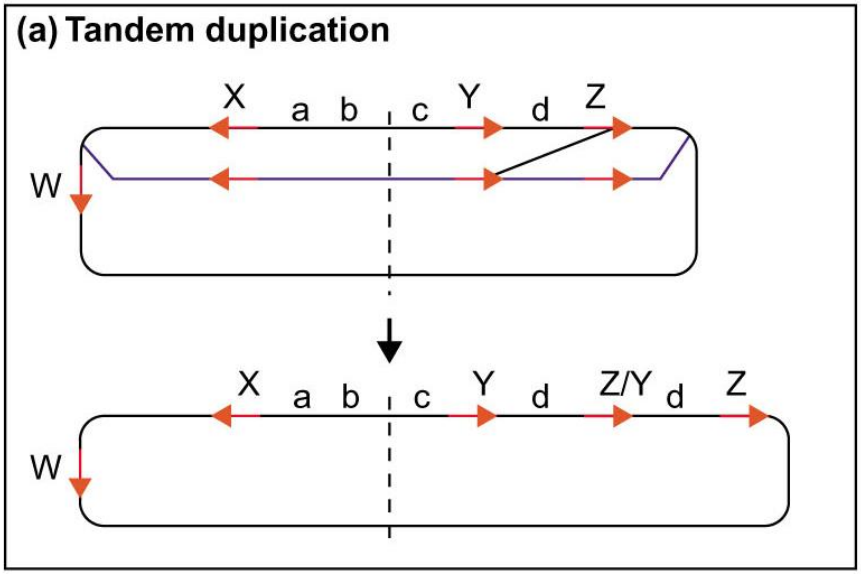
Also double-stranded: genes can be found on both strands (*orientations*)

What effects does RecA have on genome?

- Repeated sequences cause RecA to fail to choose correct recombination start position
- This leads to
  - Tandem duplications
  - Translocations
  - Inversions







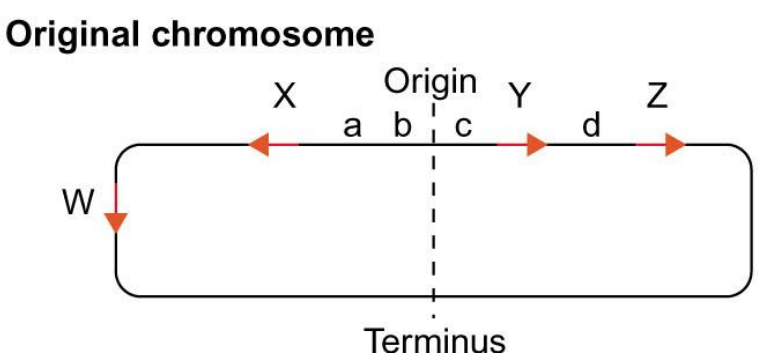
X, Y, Z and W are repeats of the same sequence.

a, b, c and d are sequences on genome bounded by repeats.

In a tandem duplication example, RecA recombines a sequence that starts from Y instead of Z after Z.

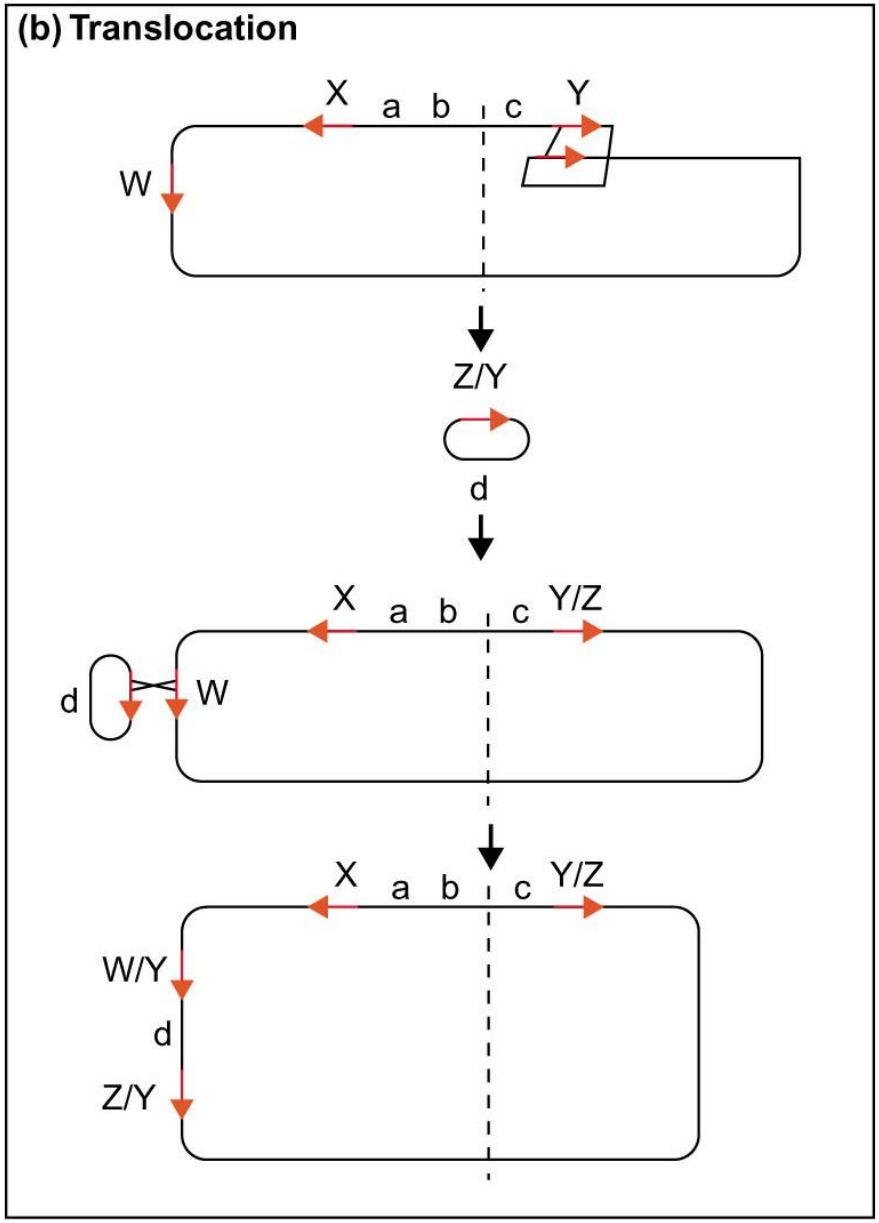
This leads to duplication of segment Y-Z.

Diarmaid Hughes: Evaluating genome dynamics: the constraints on rearrangements within bacterial genomes, *Genome Biology* 2000, 1

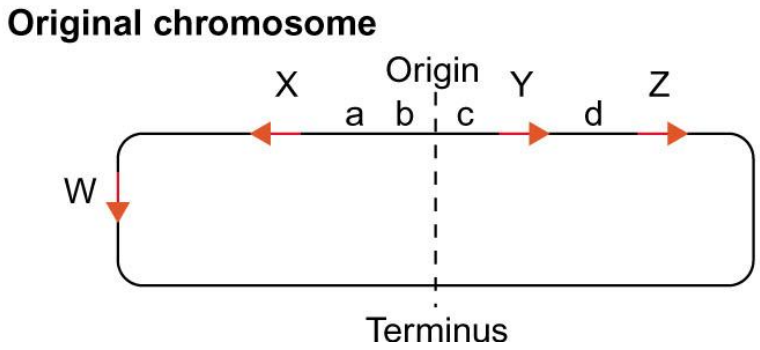


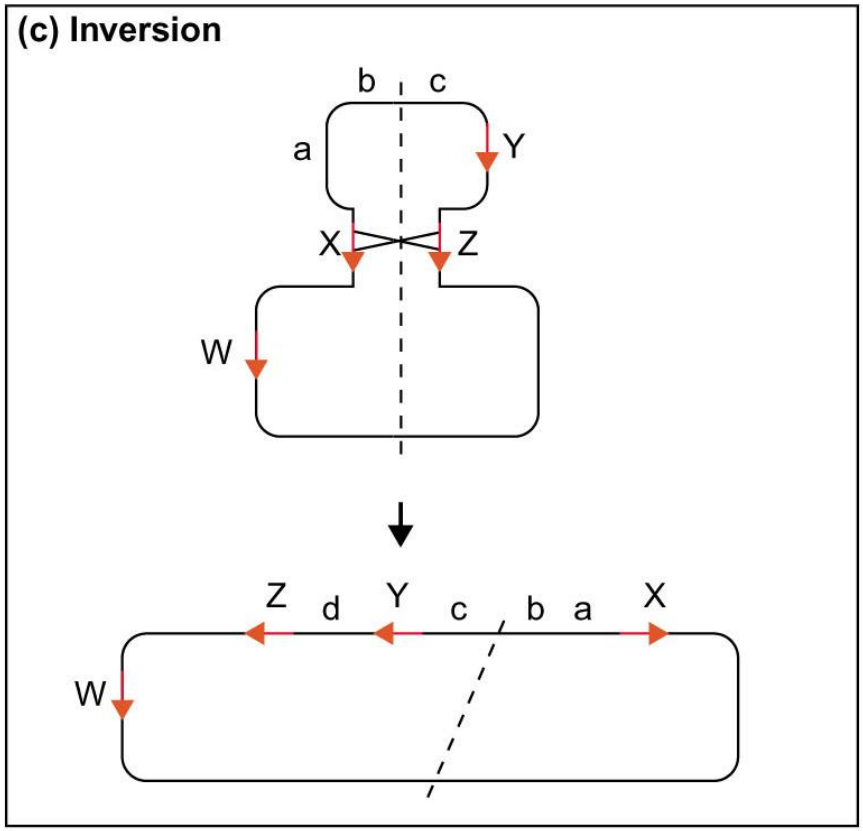
Recombination of two repeat sequences in the *same* chromosome can lead to a fragment translocation

Here sequence d is translocated



Diarmaid Hughes: Evaluating genome dynamics: the constraints on rearrangements within bacterial genomes, *Genome Biology* 2000, 1





Inversion happens when two sequences of *opposite* orientations are recombined.

Diarmaid Hughes: Evaluating genome dynamics: the constraints on rearrangements within bacterial genomes, *Genome Biology* 2000, 1

Example: human vs mouse genome

- Human and mouse genomes share thousands of homologous genes, but they are
  - Arranged in different order
  - Located in different chromosomes
- Examples
  - Human chromosome 6 contains elements from six different mouse chromosomes
  - Analysis of X chromosome indicates that rearrangements have happened primarily *within* chromosome

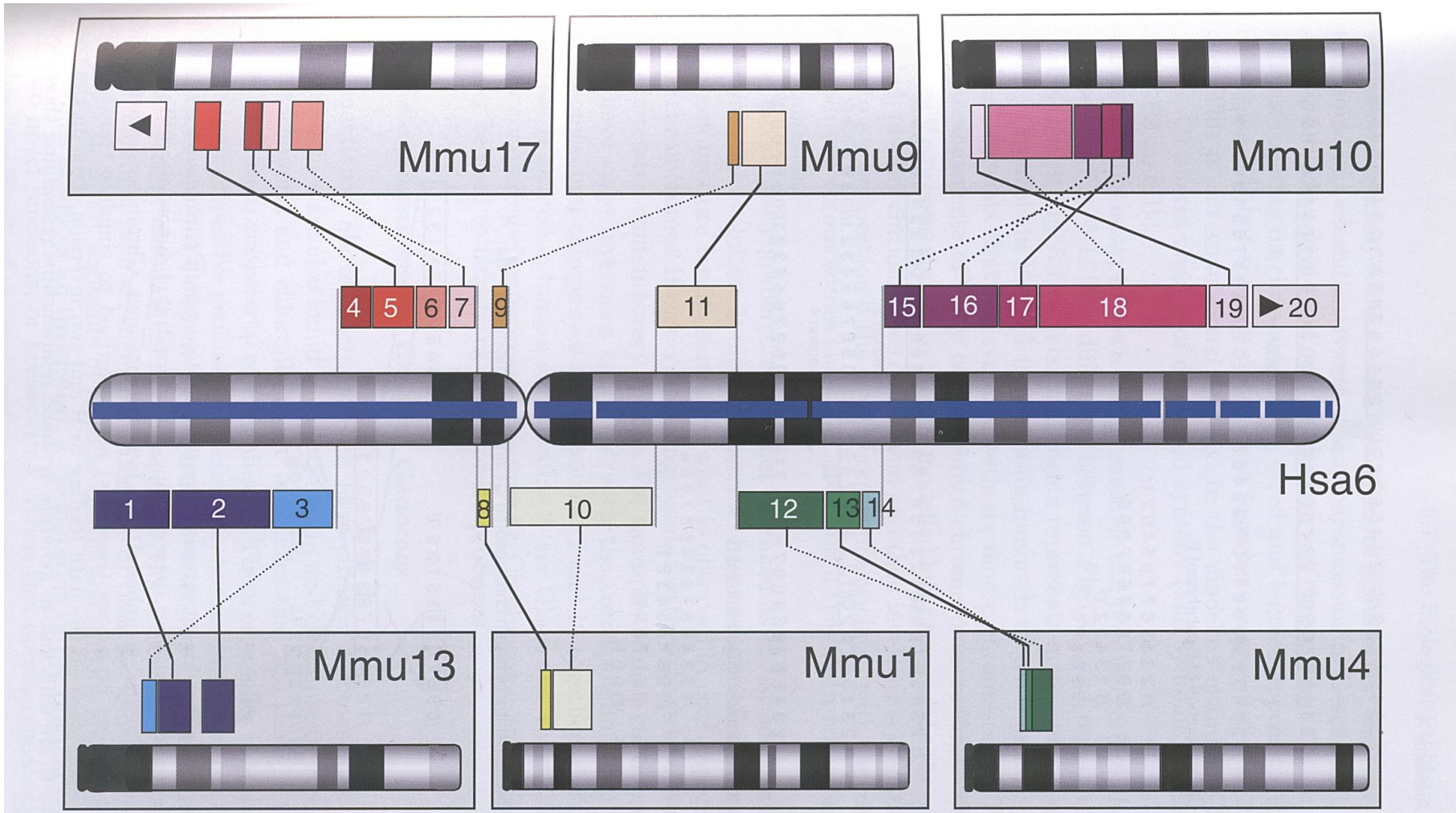


Fig. 5.1. Syntenic blocks conserved between human chromosome Hsa6 and mouse chromosomes. Broken lines indicate regions that appear in inverted orders in the two organisms. Reprinted, with permission, from Gregory SG et al. (2002) *Nature* 418:743–750. Copyright 2002 Nature Publishing Group.

Jones & Pevzner, 2004

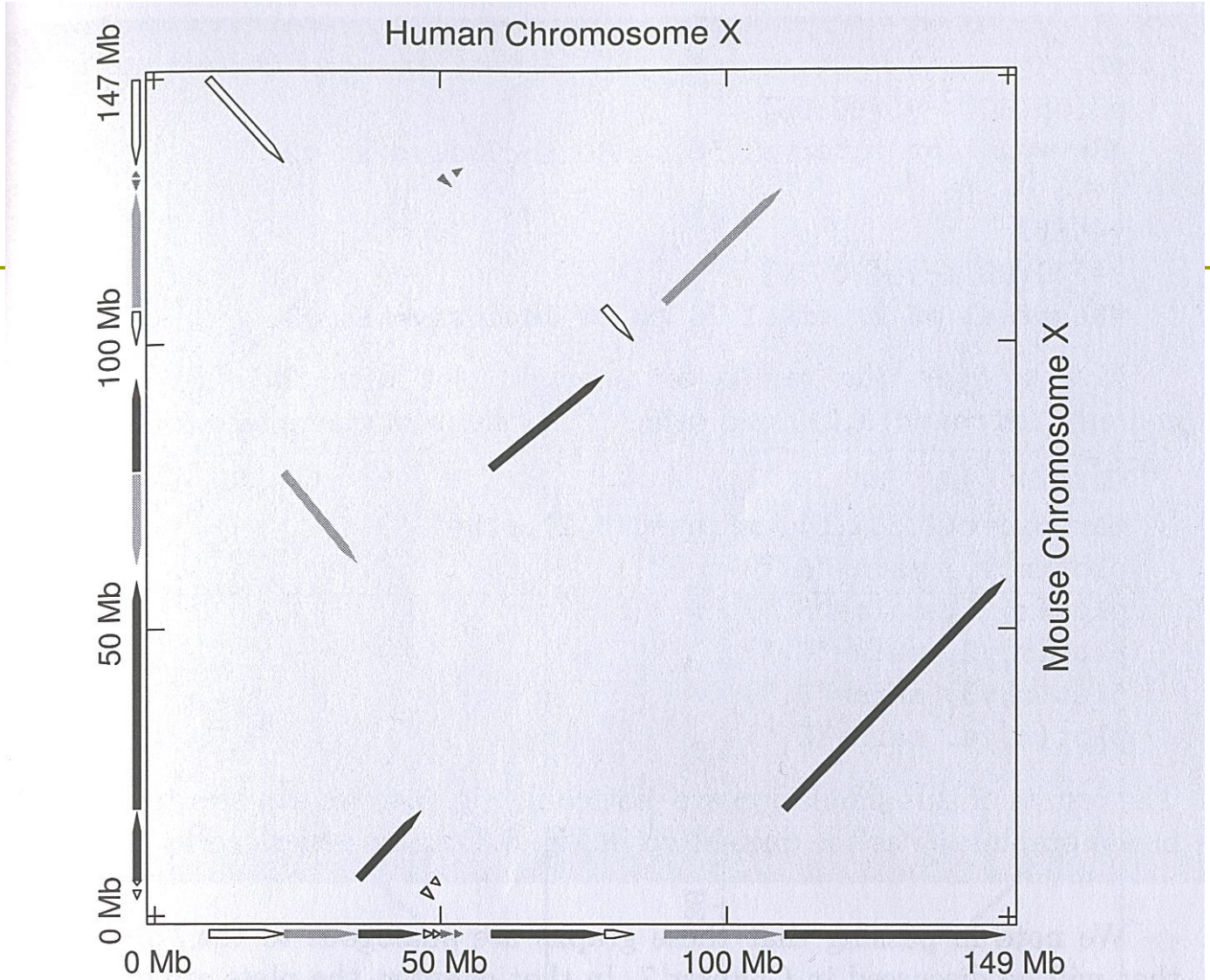


Fig. 5.3. Synteny blocks shared by human and mouse X chromosomes. The arrowhead for each block indicates the direction of increasing coordinate values for the human X chromosome. Reprinted, with permission, from Pevzner P and Tesler G (2003) *Genome Research* 13:37–45. Copyright 2003 Cold Spring Harbor Laboratory Press.

Representing genome rearrangments

- When comparing two genomes, we can find homologous sequences in both using BLAST, for example
- This gives us a map between sequences in both genomes

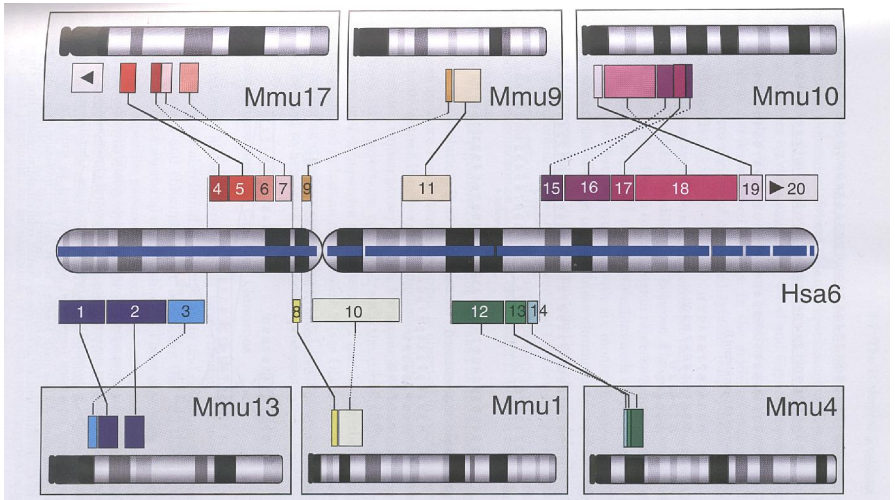
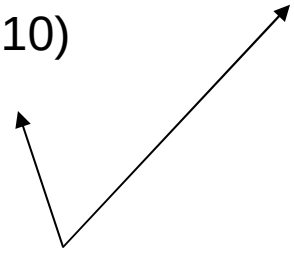


Fig. 5.1. Syntenic blocks conserved between human chromosome Hsa6 and mouse chromosomes. Broken lines indicate regions that appear in inverted orders in the two organisms. Reprinted, with permission, from Gregory SG et al. (2002) *Nature* 418:743–750. Copyright 2002 Nature Publishing Group.

Representing genome rearrangments

- We assign numbers 1,…,n to the found homologous sequences
- By convention, we number the sequences in the first genome by their order of appearance in chromosomes
- If the homolog of *i* is in reverse orientation, it receives number *–i* (*signed data*)
- For example, consider human vs mouse gene numbering on the right

Human		**Mouse**	
1	(gnat2)	12	(inpp1)
2	(nras)	13	(cd28)
3	(ngfb)	14	(fn1)
4	(gba)	15	(pax3)
5	(pklr)	-9	(il10)
6	(at3)	-8	(pdc)
7	(lamc1)	-7	(lamc1)
8	(pdc)	-6	(at3)
9	(il10)		



List order corresponds to *physical order* on chromosomes!

Permutations

- The basic data structure in the study of genome rearrangements is *permutation*
- A permutation of a sequence of *n* numbers is a reordering of the sequence
- For example, 4 1 3 2 5 is a permutation of 1 2 3 4 5

Genome rearrangement problem

- Given two genomes (set of markers), how many
  - duplications,
  - inversions and
  - translocations

  do we need to do to transform the first genome to the second?

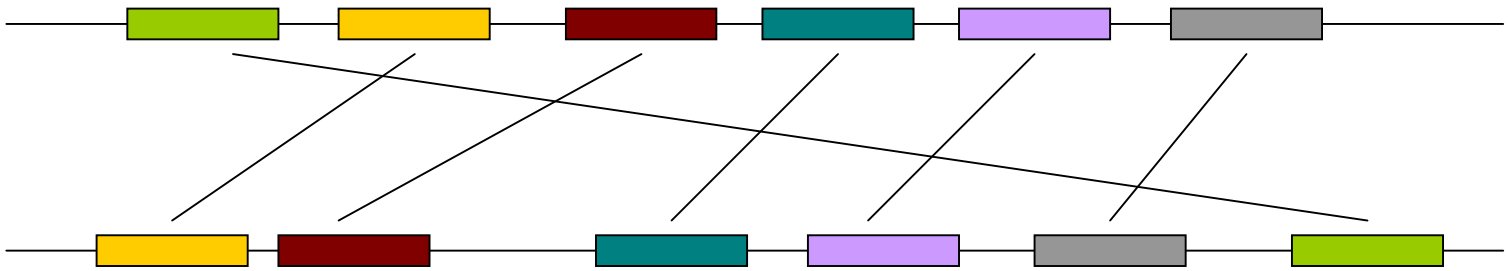
  Minimum number of operations?

  What operations? Which order?

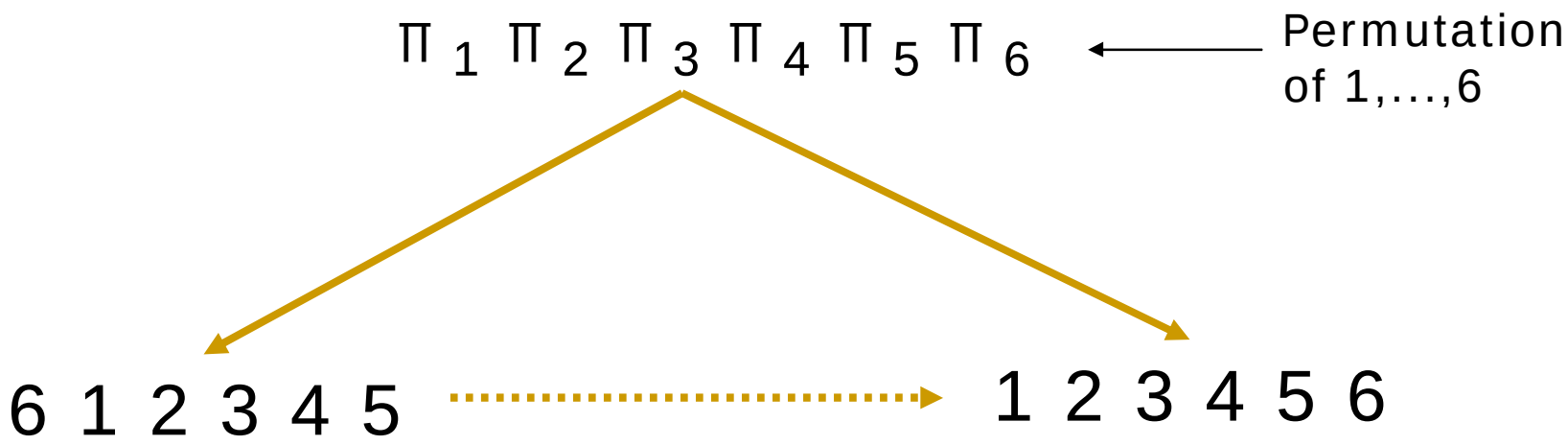
Genome rearrangement problem

#duplications?
#inversions?
#translocations?

6 1 2 3 4 5 → 1 2 3 4 5 6



Genome rearrangement problem

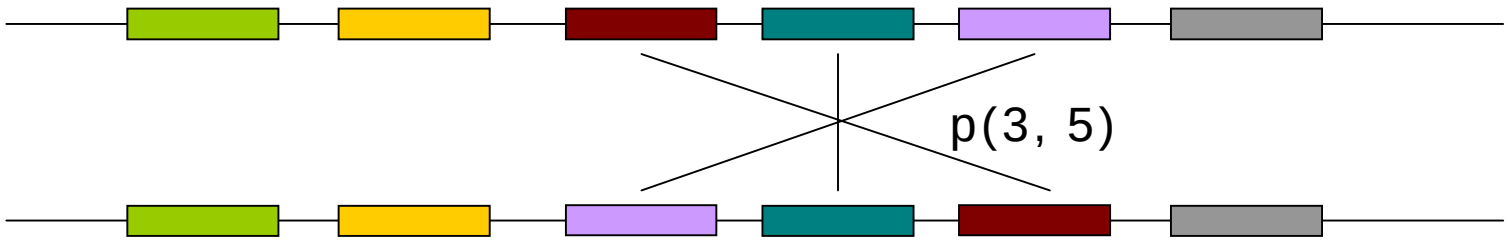


Keep in mind, that the two genomes have been evolved from a common ancestor genome!

Genome rearrangements using reversals (=inversions) only

- Lets consider a simpler problem where we just study reversals with *unsigned data*
- A reversal p(i, j) reverses the order of the segment $\Pi_i \Pi_{i+1} \ldots \Pi_{j-1} \Pi_j$ (indexing starts from 1)
- For example, given permutation

  6 1 2 3 4 5 and reversal p(3, 5) we get permutation 6 1 4 3 2 5



...note that we do not care about exact *positions* on the genome

Reversal distance problem

- Find the shortest series of reversals that, given a permutation Π, transforms it to the *identity* permutation (1, 2, …, n)
- This quantity is denoted by $d(\Pi)$

- Reversal distance for a pair of chromosomes:
  - Find synteny blocks in both
  - Number blocks in the first chromosome to identity
  - Set Π to correspond matching of second chromosome's blocks against the first
  - Find reversal distance

Reversal distance problem: discussion

- If we can find the minimal series of reversals for some pair of genomes
  - Is that what happened during evolution?
  - If not, is it the correct number of reversals?

- In any case, reversal distance gives us a measure of evolutionary distance between the two genomes and species

Solving the problem by sorting

- Our first approach to solve the reversal distance problem:
  - Examine each position i of the permutation
  - At each position, if $\Pi_i \neq i$, do a reversal such that $\Pi_i = i$
- This is a *greedy* approach: we try to choose the best option at each step

Simple reversal sort: example

6 1 2 3 4 5 -> 1 6 2 3 4 5 -> 1 2 6 3 4 5 -> 1 2 3 4 6 5

-> 1 2 3 4 5 6

Reversal series: p(1,2), p(2,3), p(3,4), p(5,6)

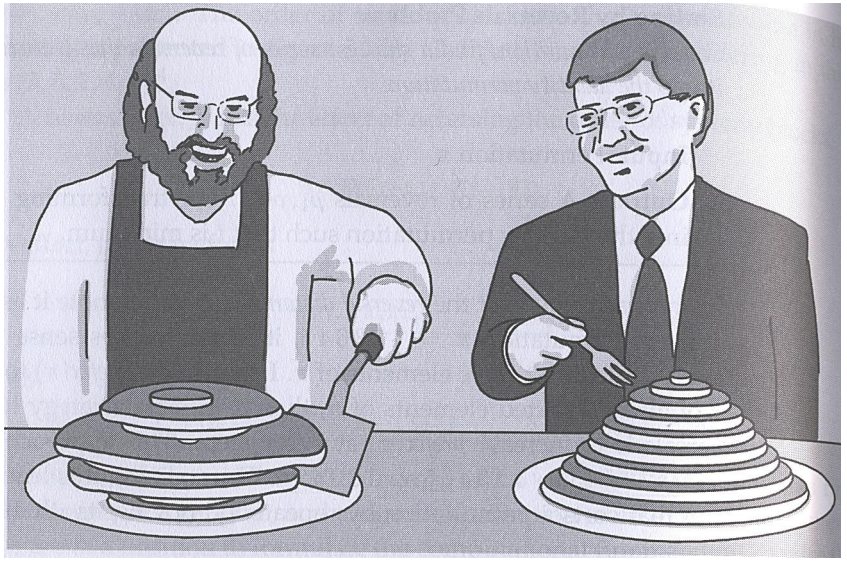
Is d(6 1 2 3 4 5) then 4?

6 1 2 3 4 5 -> 5 4 3 2 1 6 -> 1 2 3 4 5 6

D(6 1 2 3 4 5) = 2

Pancake flipping problem

- No pancake made by the chef is of the same size
- Pancakes need to be rearranged before delivery
- Flipping operation: take some from the top and flip them over
- This corresponds to always reversing the sequence *prefix*



1 2 3 6 4 5 -> 6 3 2 1 4 5 ->

5 4 1 2 3 6 -> 3 2 1 4 5 6 ->

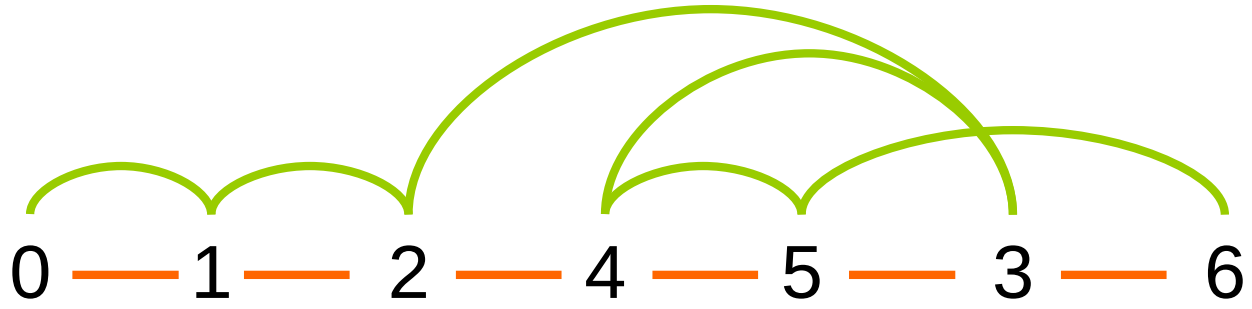
1 2 3 4 5 6

How good is simple reversal sort?

- Not so good actually
- It has to do at most n-1 reversals with permutation of length n
- The algorithm can return a distance that is as large as (n – 1)/2 times the correct result d(Π)
  - For example, if n = 1001, result can be as bad as 500 x d(Π)

Estimating reversal distance by cycle decomposition

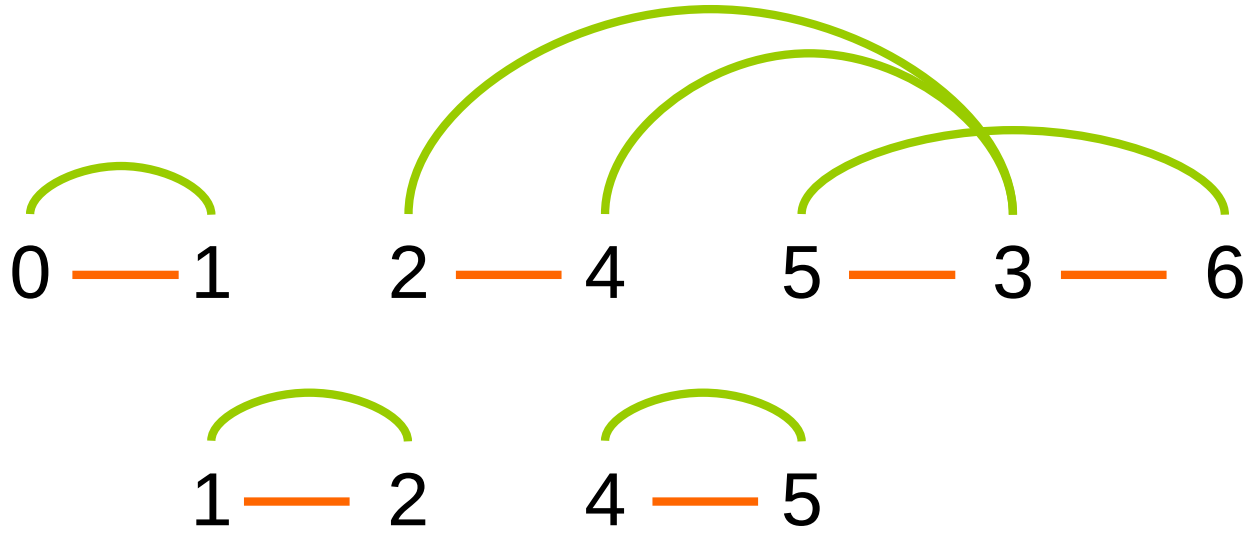
- We can estimate d(Π) by *cycle decomposition*
- Lets represent permutation Π = 1 2 4 5 3 with the following graph



where edges correspond to adjacencies (identity, permutation F)

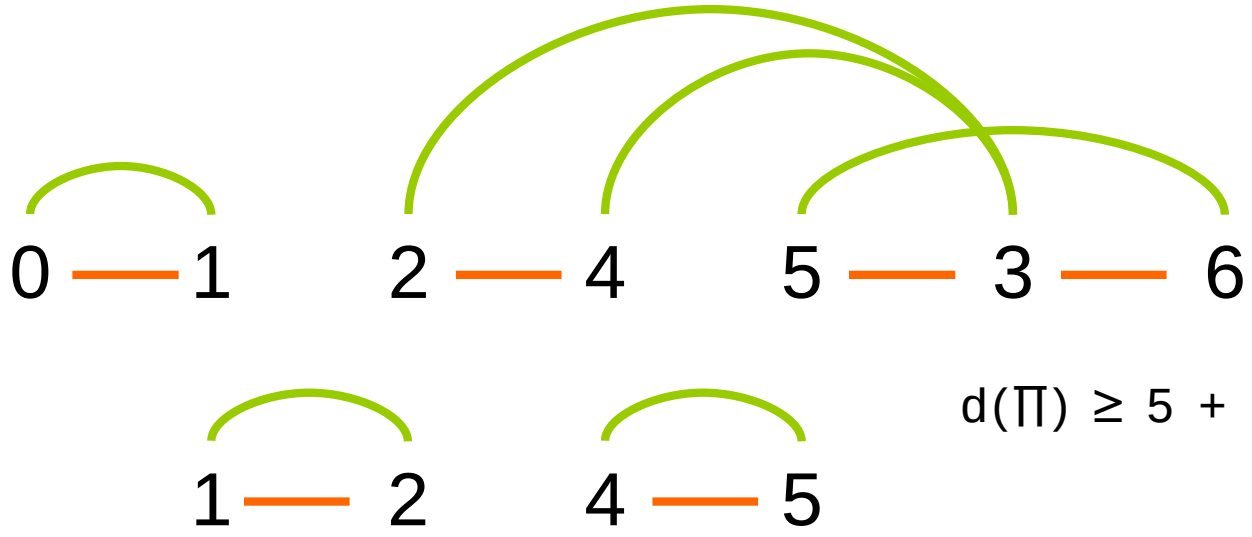
Estimating reversal distance by cycle decomposition

- Cycle decomposition: a set of cycles that
  - have edges with alternating colors
  - do not share edges with other cycles (=cycles are edge disjoint)



Cycle decompositions

- Let c(Π) the maximum number of alternating, edge-disjoint cycles in the graph representation of Π
- The following formula allows estimation of d(Π)
  - d(Π) ≥ n + 1 – c(Π), where n is the permutation length



d(Π) ≥ 5 + 1 – 4 = 2

Claim in Deonier: equality holds for ”most of the usual and interesting biological systems.

Cycle decompositions

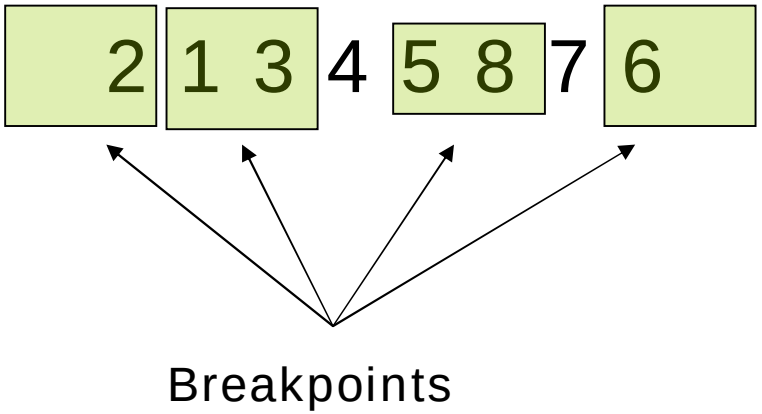
- Cycle decomposition is NP-complete
  - We cannot solve the general problem exactly for large instances
- However, with signed data the problem becomes easy
  - Before going into signed data, lets discuss another algorithm for the general case

Computing reversals with breakpoints

- Lets investigate a better way to compute reversal distance
- First, some concepts related to permutation $\Pi_1\Pi_{2,,,}\Pi_{n-1}\Pi_n$
  - Breakpoint: two elements Π_i and Π_{i+1} are a *breakpoint*, if they are not consecutive numbers
  - Adjacency: if Π_i and Π_{i+1} are consecutive, they are called *adjacency*

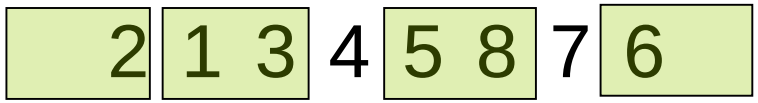
Breakpoints and adjacencies

This permutation contains
four breakpoints *begin*-2, 13, 58, 6-*end* and
five adjacencies 21, 34, 45, 87, 76



Breakpoints

- Each breakpoint in permutation needs to be removed to get to the identity permutation (=our target)
  - Identity permutation does not contain any breakpoints



$b(\Pi) = 4$

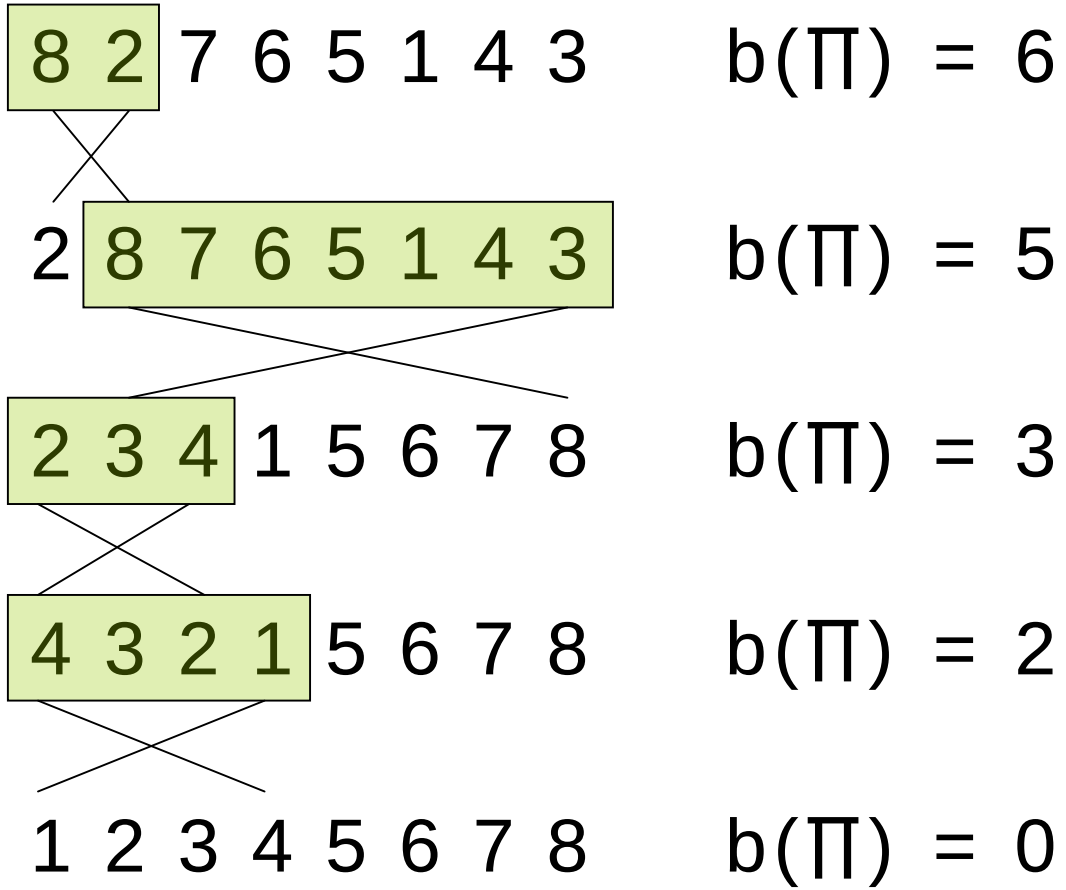
- First and last positions special cases
- Note that each reversal can remove *at most* two breakpoints
- Denote the number of breakpoints by $b(\Pi)$

Breakpoint reversal sort

- Idea: try to remove as many breakpoints as possible (max 2) in every step

1. While b(Π) > 0
2. Choose reversal p that removes most breakpoints
3. Perform reversal p to Π
4. Output Π
5. return

Breakpoint removal: example



Breakpoint removal

- The previous algorithm needs refinement to be correct
- Consider the following permutation:

1 5 6 7 2 3 4 8

- There is no reversal that decreases the number of breakpoints!
- See Jones & Pevzner for detailed description on this

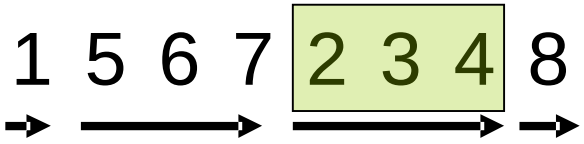
Strip: maximal segment without breakpoints

Breakpoint removal

Increasing strip

Decreasing strip

- Reversal can only decrease breakpoint count if permutation contains *decreasing strips*



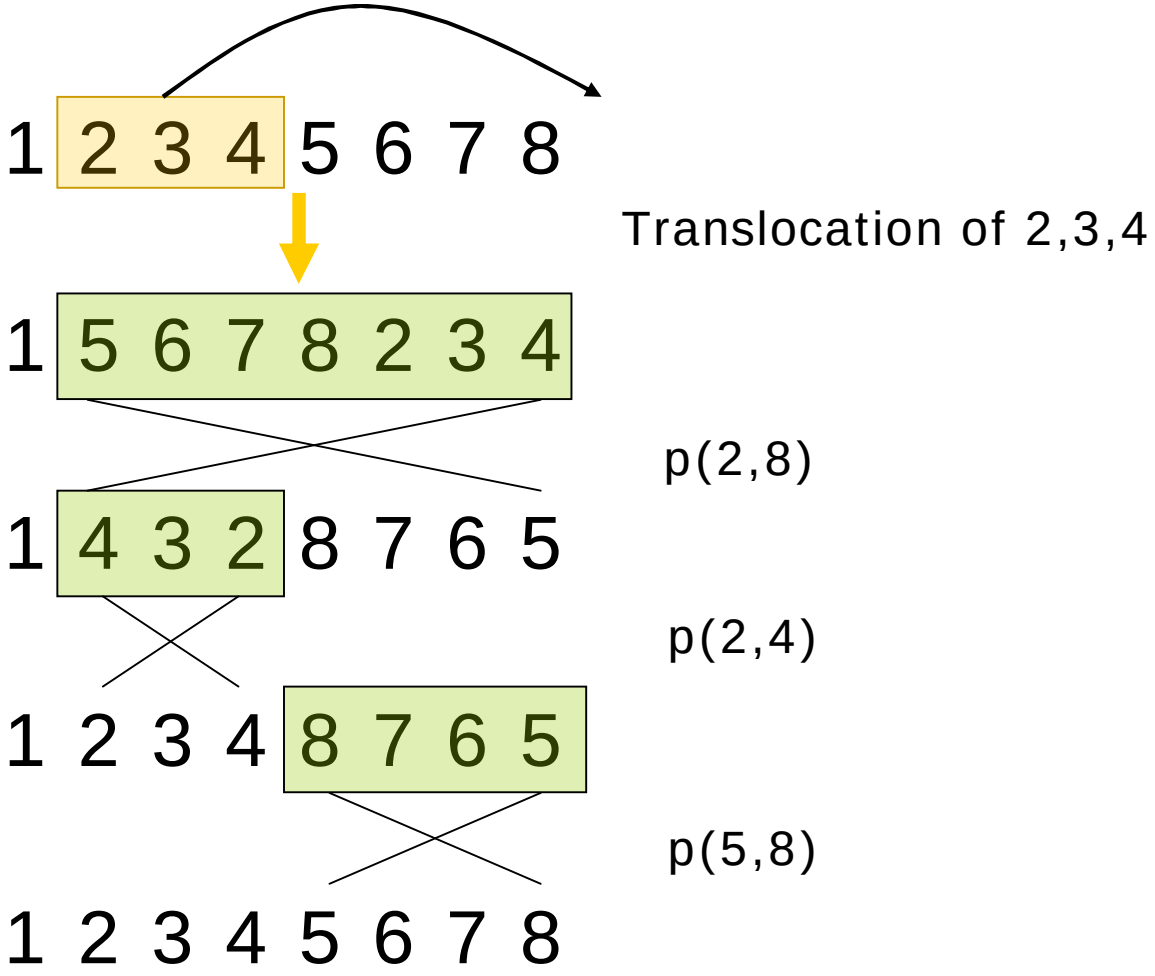
Improved breakpoint reversal sort

1. While b(Π) > 0
2. If Π has a decreasing strip
3. Do reversal p that removes most BPs
4. Else
5. Reverse an increasing strip
6. Output Π
7. return

Is Improved BP removal enough?

- The algorithm works pretty well:
  - It produces a result that is at most four times worse than the optimal result
  - ...is this good?
- We considered only reversals
- What about translocations & duplications?

Translocations via reversals



Genome rearrangements with reversals

- With *unsigned* data, the problem of finding minimum reversal distances is *NP-complete*
  - *Why is this so if sorting is easy?*
- An algorithm has been developed that achieves 1.375-approximation
- However, reversal distance in *signed data* can be computed quickly!
  - It takes linear time w.r.t. the length of permutation (Bader, Moret, Yan, 2001)

Cycle decomposition with signed data

- Consider the following two permutations that include *orientation* of markers
  - *J: +1 +5 -2 +3 +4*
  - *K: +1 -3 +2 +4 -5*

- We modify this representation a bit to include both endpoints of each marker:
  - J': 0 1a 1b 5a 5b 2b 2a 3a 3b 4a 4b 6
  - K': 0 1a 1b 3b 3a 2a 2b 4a 4b 5b 5a 6

Graph representation of J’ and K’

- Drawn online in lecture!

Multiple chromosomes

- In unichromosomal genomes, inversion (reversal) is the most common operation
- In multichromosomal genomes, inversions, translocations, *fissions* and *fusions* are most common

Multiple chromosomes

- Lets represent multichromosomal genome as a set of permutations, with $ denoting the boundary of a chromosome:

5 9 $	Chr 1
1 3 2 8 $	Chr 2
7 6 4 $	Chr 3

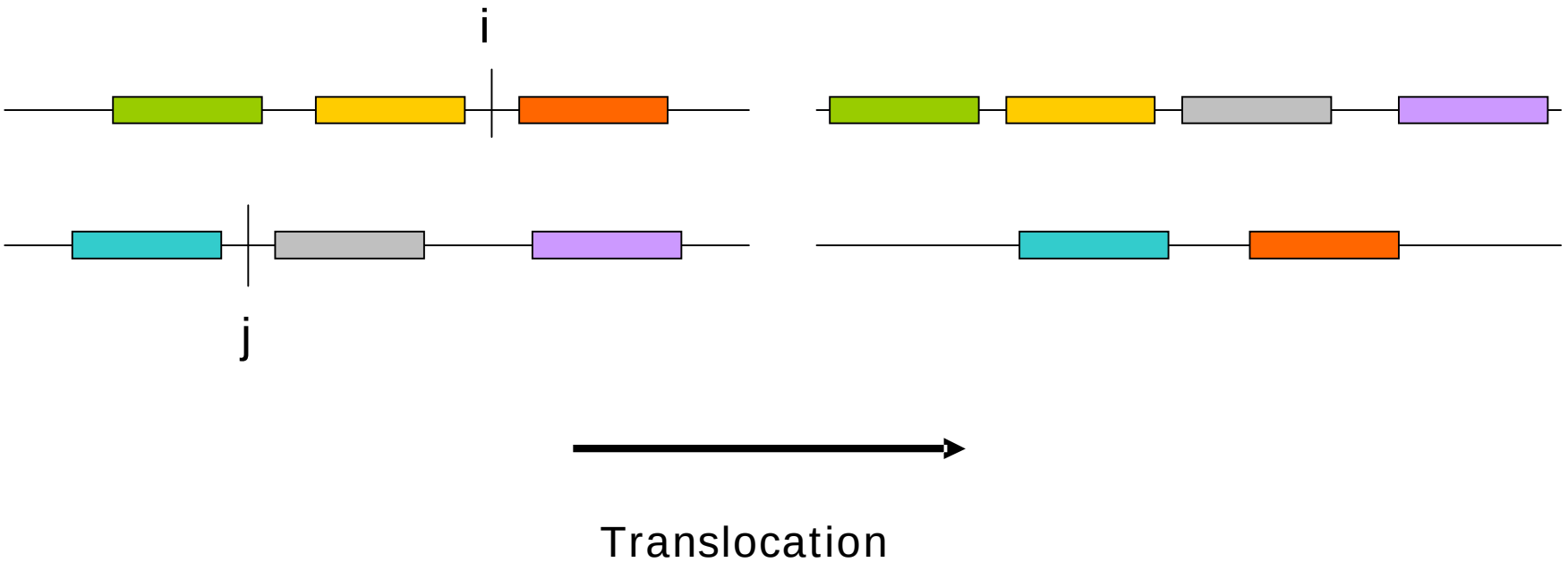
This notation is frequently used in software used to analyse genome rearrangements.

Multiple chromosomes

- Note that when dealing with multiple chromosomes, you need to specify numbering for elements on both genomes

Reversals & translocations

- Reversal p(Π, i, j)
- Translocation p(Π, σ, i, j)

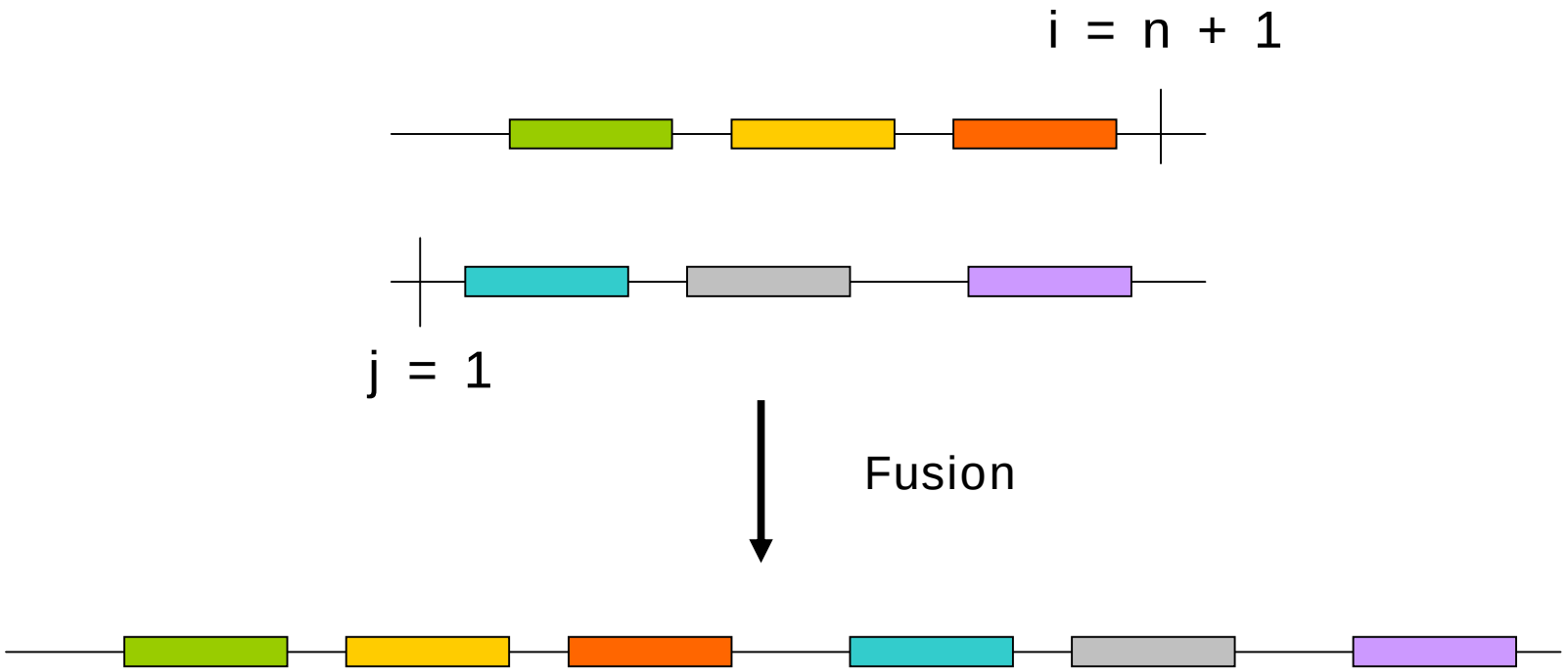


Fusions & fissions

- Fusion: merging of two chromosomes
- Fission: chromosome is split into two chromosomes
- Both events can be represented with a translocation

Fusion

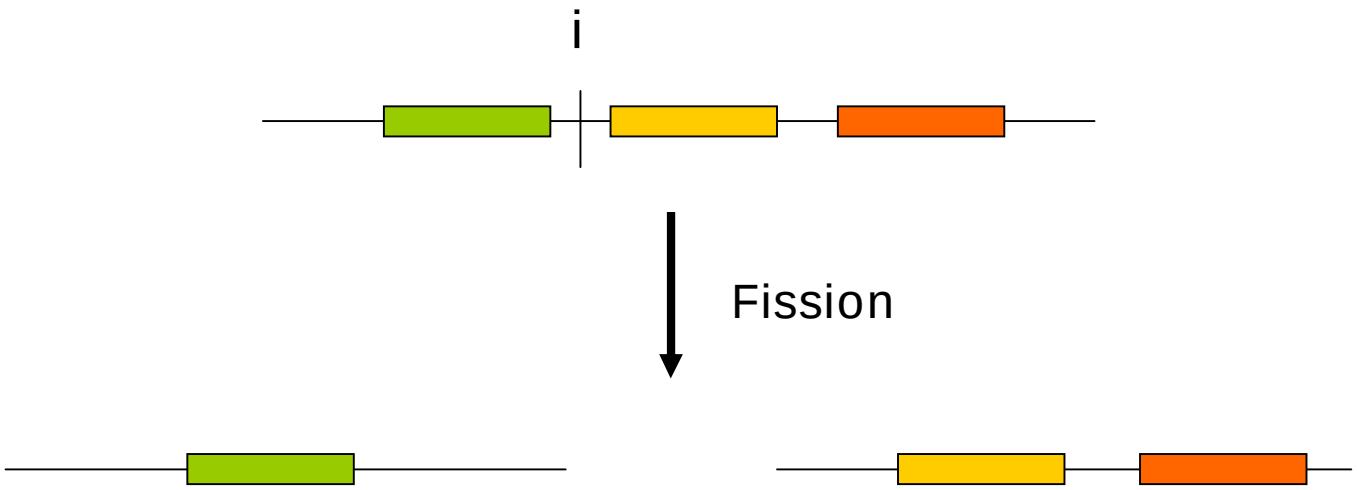
- Fusion by translocation $p(\Pi, \sigma, n+1, 1)$



Fission

Empty chromosome

- Fission by translocation p(Π, ∅, i, 1)



Algorithms for general genomic distance problem

- Hannenhalli, Pevzner: Transforming Men into Mice (polynomial algorithm for genomic distance problem), *36th Annual IEEE Symposium on Foundations of Computer Science*, 1995

Human & mouse revisited

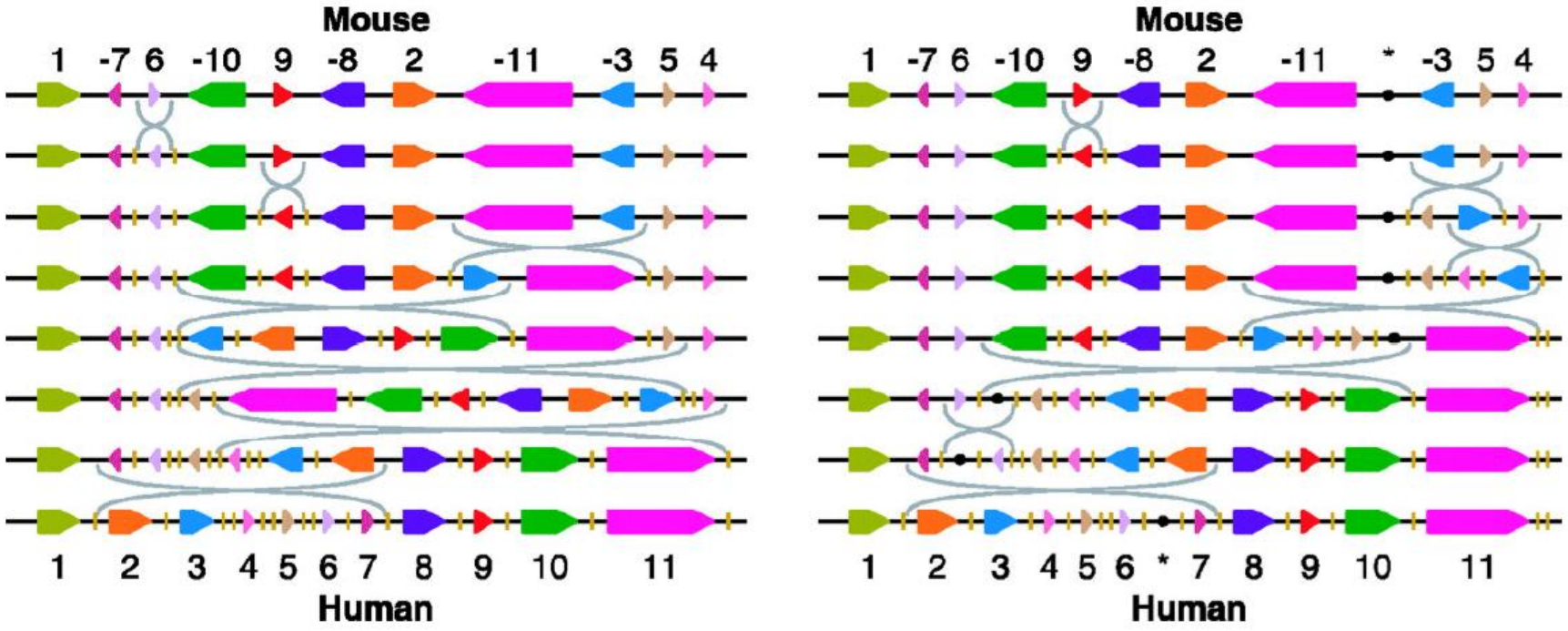
- Human and mouse are separated by about 75-83 million years of evolutionary history
- Only a few hundred rearrangements have happened after speciation from the common ancestory
- Pevzner & Tesler identified in 2003 for 281 synteny blocks a rearrangement from mouse to human with
  - 149 inversions
  - 93 translocations
  - 9 fissions

Discussion

- Genome rearrangement events are very rare compared to, e.g., point mutations
  - We can study rearrangement events further back in the evolutionary history
- Rearrangements are easier to detect in comparison to many other genomic events
- We cannot detect homologs 100% correctly so the input permutation can contain errors

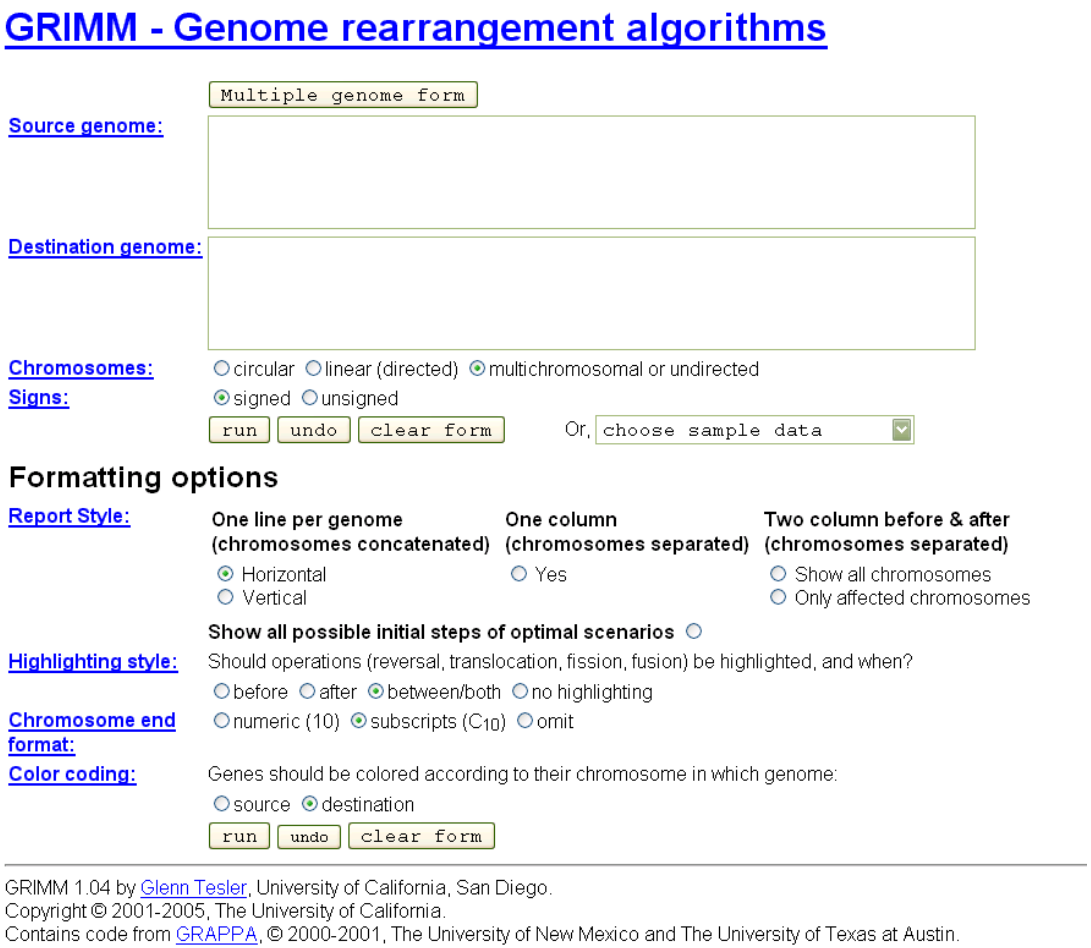
Discussion

- Genome rearrangement is to some degree constrained by the number and size of repeats in a genome
  - Notice how the importance of genomic repeats pops up once again
- Sequencing gives us (usually) signed data so we can utilize faster algorithms
- What if there are more than one optimal solution?



Two different genome rearrangement scenarios giving the same result.

GRIMM demonstration



Glenn Tesler, GRIMM: genome rearrangements web server. Bioinformatics, 2002,

GRIMM file format

```
# useful comment about first genome
# another useful comment about it
>name of first genome
1 -4 2 $ # chromosome  1
-3 5 6 # chromosome 2
>name of second genome
5 -3  $
6 $
2 -4 1 $
```

GRIMM supports analysis of one, two or more genomes

http://grimm.ucsd.edu/GRIMM/grimm_instr.html

Introduction to Bioinformatics

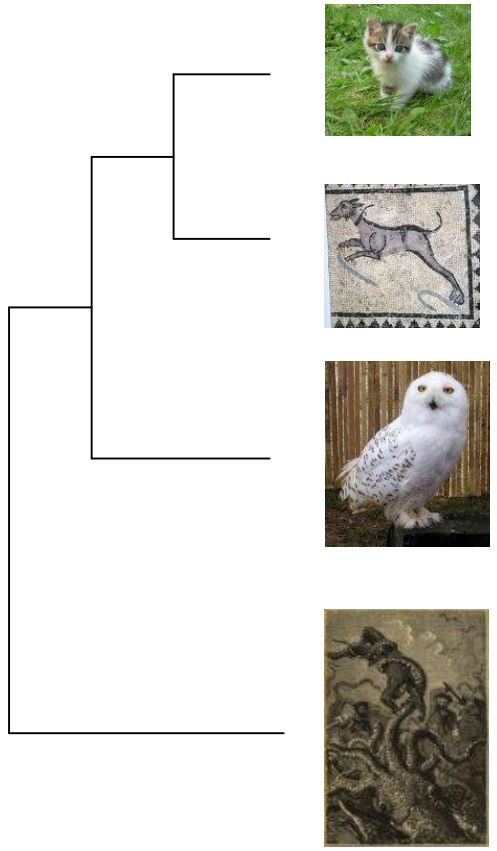
Phylogenetic trees

Inferring the Past: Phylogenetic Trees

- *The biological problem*
- Parsimony and distance methods
- Models for mutations and estimation of distances
- Maximum likelihood methods

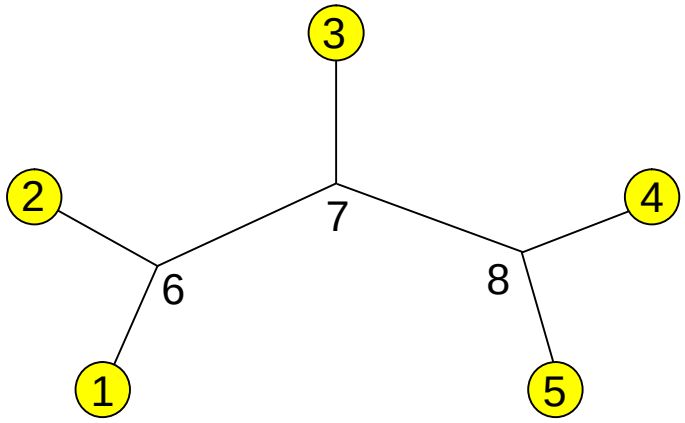
Phylogeny

- We want to study ancestor-descendant relationships, or *phylogeny*, among groups of organisms
- Groups are called *taxa* (singular: *taxon*)
- Organisms are usually called *operational taxonomic units* or *OTUs* in the context of phylogeny



Phylogenetic trees

- Leaves (external nodes) ~ species, observed (OTUs)
- Internal nodes ~ ancestral species/divergence events, not observed
- Unrooted tree does not specify ancestor-descendant relationships beyond the observation ”leaves are not ancestors”

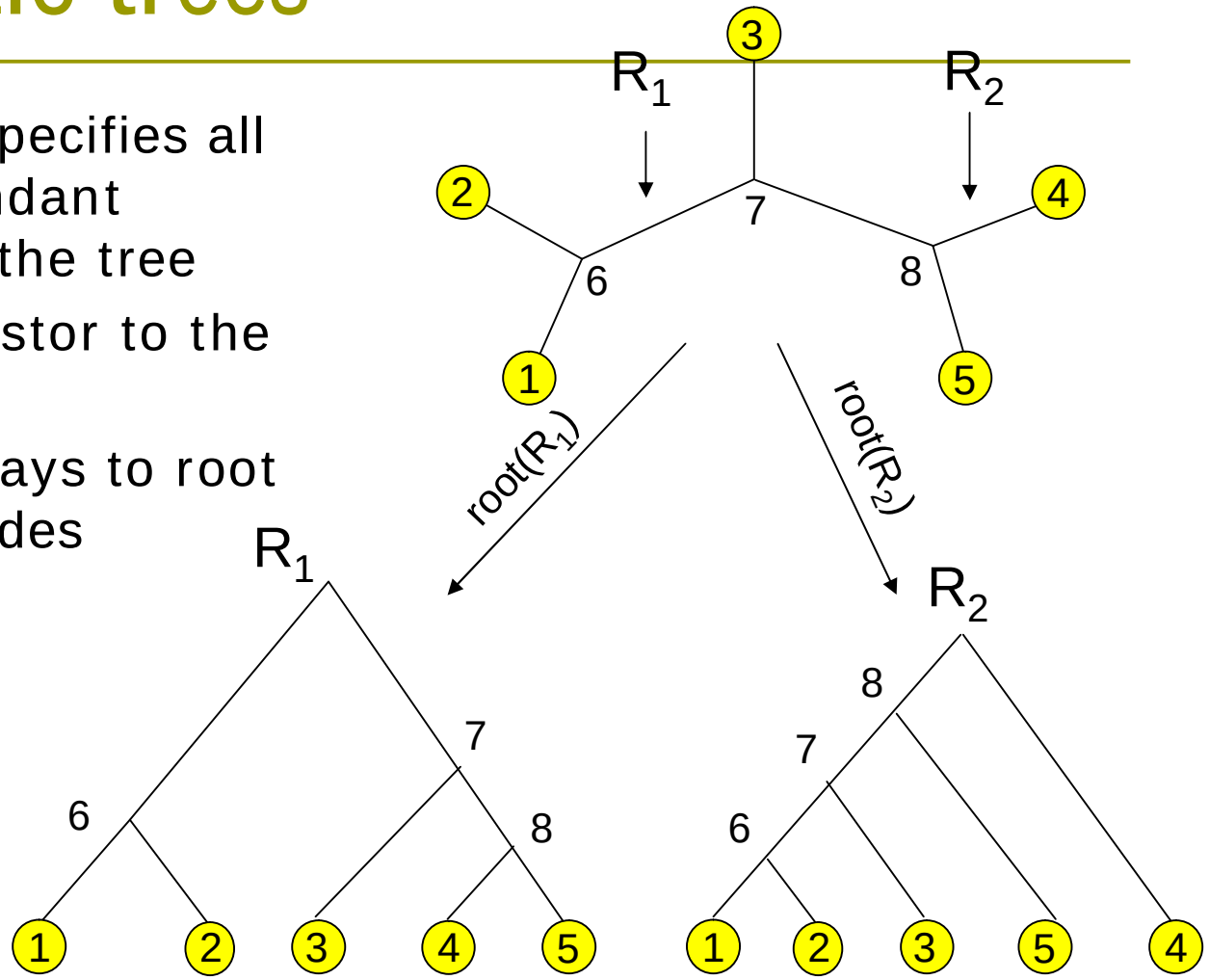


Unrooted tree with 5 leaves and 3 internal nodes.

Is node 7 ancestor of node 6?

Phylogenetic trees

- Rooting a tree specifies all ancestor-descendant relationships in the tree
- Root is the ancestor to the other species
- There are n-1 ways to root a tree with n nodes



Questions

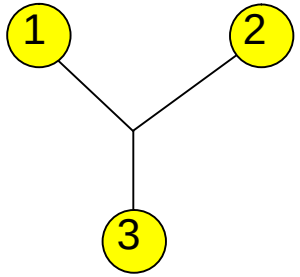
- Can we enumerate all possible phylogenetic trees for *n* species (or sequences?)
- How to score a phylogenetic tree with respect to data?
- How to find the best phylogenetic tree given data?

Finding the best phylogenetic tree: naive method

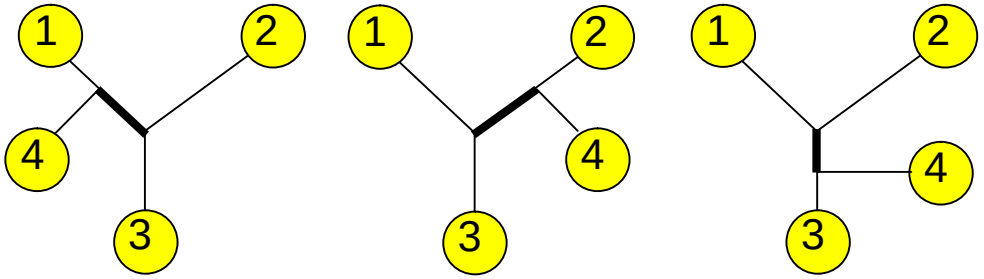
- How can we find the phylogenetic tree that best represents the data?
- Naive method: enumerate all possible trees
- How many different trees are there of *n* species?
- Denote this number by b_n

Enumerating unordered trees

- Start with the only unordered tree with 3 leaves ($b_3 = 1$)



- Consider all ways to add a leaf node to this tree



- Fourth node can be added to 3 different branches (edges), creating 1 new internal branch
- Total number of branches is n external and n – 3 internal branches
- Unrooted tree with n leaves has 2n – 3 branches

Enumerating unordered trees

- Thus, we get the number of unrooted trees

$$b_n = (2(n-1)-3)b_{n-1} = (2n-5)b_{n-1}$$
$$= (2n-5) * (2n-7) * \ldots * 3 * 1$$
$$= (2n-5)! / ((n-3)!2^{n-3}),\ n > 2$$

- Number of rooted trees b'_n is

$$b'_n = (2n-3)b_n = (2n-3)! / ((n-2)!2^{n-2}),\ n > 2$$

that is, the number of unrooted trees times the number of branches in the trees

Number of possible rooted and unrooted trees

n	B_n	b'_n
3	1	3
4	3	15
5	15	105
6	105	945
7	954	10395
8	10395	135135
9	135135	2027025
10	2027025	34459425
20	2.22E+020	8.20E+021
30	8.69E+036	4.95E+038

Too many trees?

- We can't construct and evaluate every phylogenetic tree even for a smallish number of species
- Better alternative is to
  - Devise a way to evaluate an individual tree against the data
  - Guide the search using the evaluation criteria to reduce the search space

Inferring the Past: Phylogenetic Trees (chapter 12)

- The biological problem
- *Parsimony and distance methods*
- Models for mutations and estimation of distances
- Maximum likelihood methods

Parsimony method

- The parsimony method finds the tree that explains the observed *sequences* with a minimal number of substitutions
- Method has two steps
  - Compute smallest number of substitutions for a given tree with a *parsimony algorithm*
  - Search for the tree with the minimal number of substitutions

Parsimony: an example

- Consider the following short sequences

  1 ACTTT

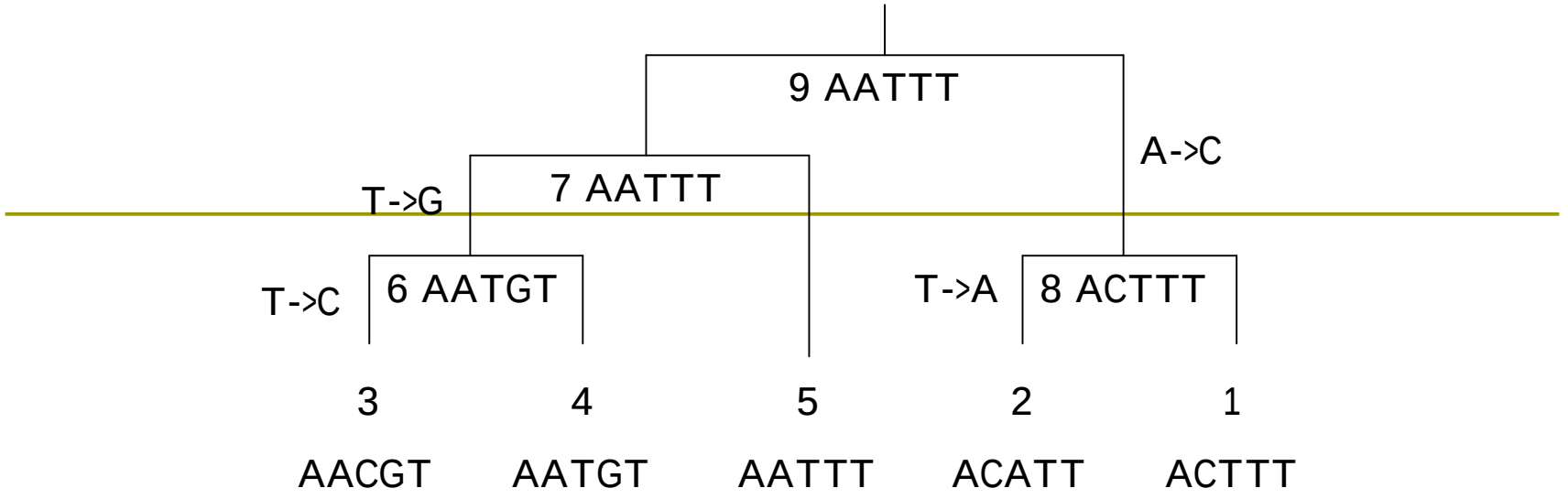
  2 ACATT

  3 AACGT

  4 AATGT

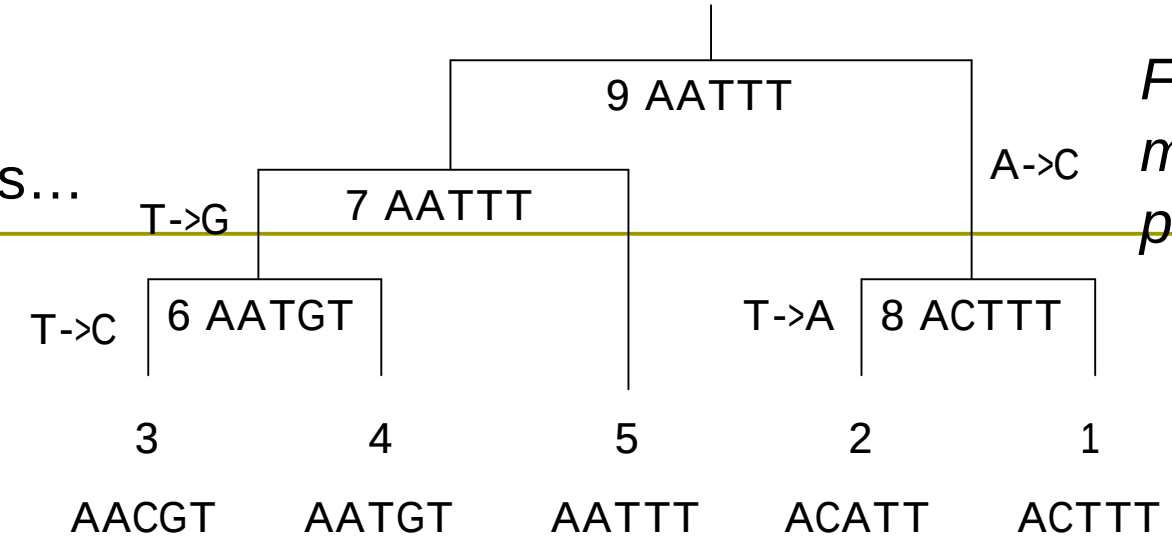
  5 AATTT

- There are 105 possible rooted trees for 5 sequences
- Example: which of the following trees explains the sequences with least number of substitutions?



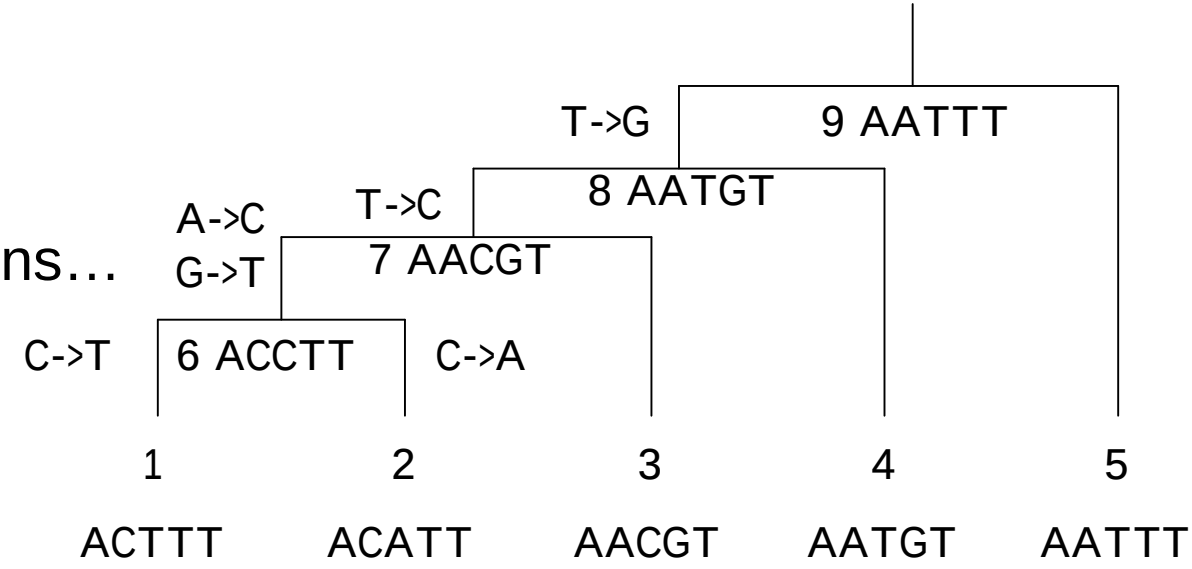
This tree explains the sequences with 4 substitutions

4 substitutions…



First tree is more parsimonious!

6 substitutions…

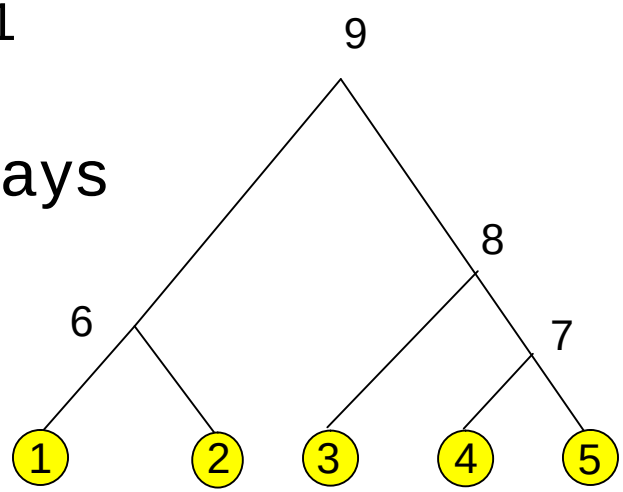


Computing parsimony

- Parsimony treats each site (position in a sequence) independently
- Total parsimony cost is the sum of parsimony costs (=required substitutions) of each site
- We can compute the minimal parsimony cost for a given tree by
  - First finding out possible assignments at each node, starting from leaves and proceeding towards the root
  - Then, starting from the root, assign a letter at each node, proceeding towards leaves

Labelling tree nodes

- An unrooted tree with *n* leaves contains 2n-1 nodes altogether
- Assign the following labels to nodes in a rooted tree
  - leaf nodes: 1, 2, …, n
  - internal nodes: n+1, n+2, …, 2n-1
  - root node: 2n-1
- The label of a child node is always smaller than the label of the parent node



Parsimony algorithm: first phase

- Find out possible assignments at every node for each site u independently. Denote site u in sequence i by $s_{i,u}$.

For i := 1, …, n do

  $F_i := \{s_{i,u}\}$ % possible assignments at node i

  $L_i := 0$ % number of substitutions up to node i

For i := n+1, …, 2n-1 do

  Let j and k be the children of node i

  If $F_j \cap F_k = \varnothing$

    then $L_i := L_j + L_k + 1$, $F_i := F_j \cup F_k$

    else $L_i := L_j + L_k$, $F_i := F_j \cap F_k$

Parsimony algorithm: first phase

Choose u = 3 (for example, in general we do this for all sites)

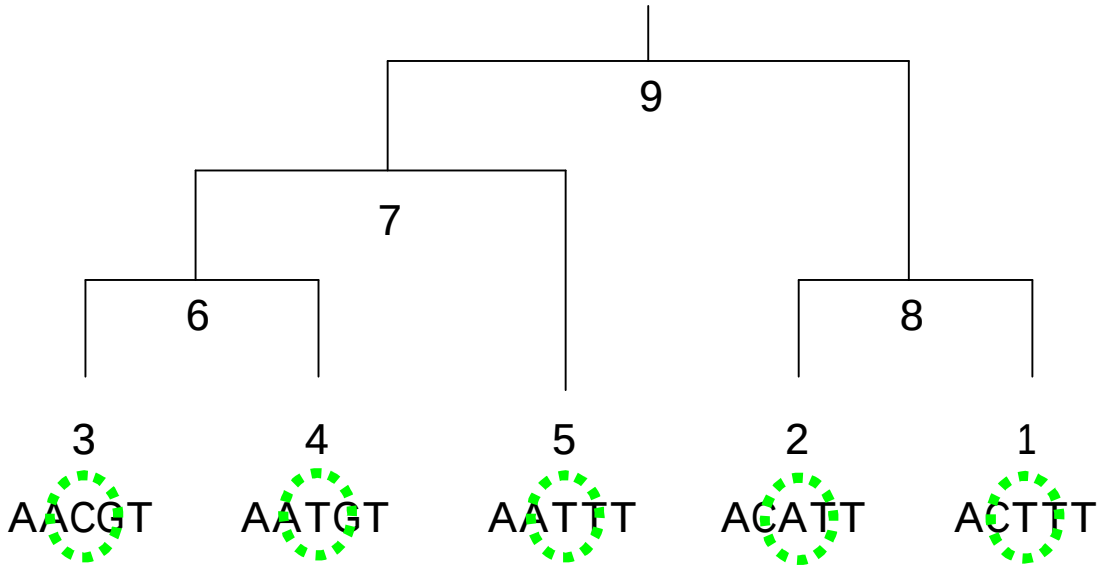
$F_1 := \{T\}$
$L_1 := 0$

$F_2 := \{A\}$
$L_2 := 0$

$F_3 := \{C\}$, $L_3 := 0$

$F_4 := \{T\}$, $L_4 := 0$

$F_5 := \{T\}$, $L_5 := 0$



Parsimony algorithm: first phase

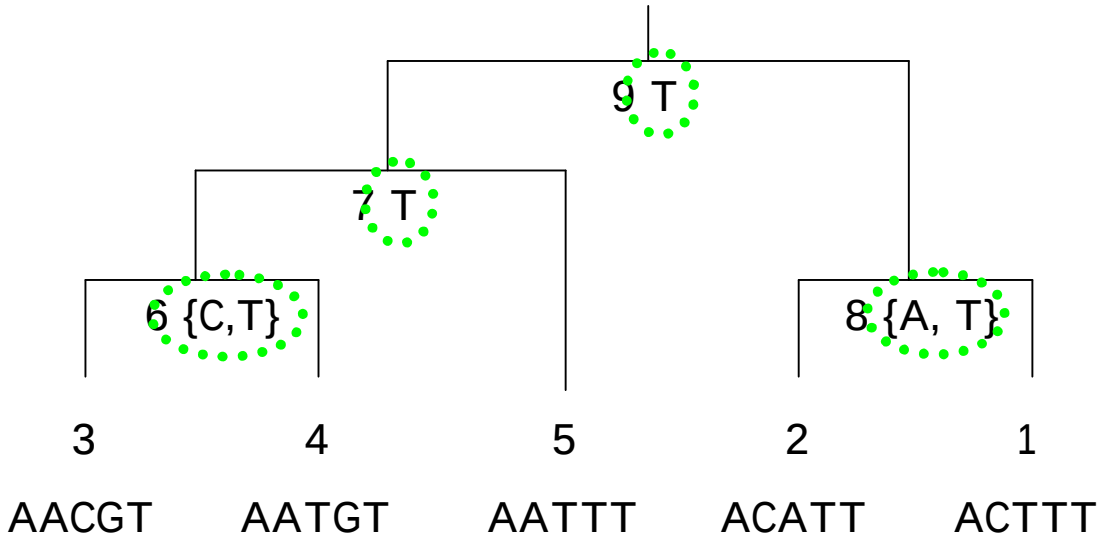
$F_6 := F_3 \cup F_4 = \{C, T\}$

$L_6 := L_3 + L_4 + 1 = 1$

$F_7 := F_5 \cap F_6 = \{T\}$
$L_7 := L_5 + L_6 = 1$

$F_8 := F_1 \cup F_2 = \{A, T\}$
$L_8 := L_1 + L_2 + 1 = 1$

$F_9 := F_7 \cap F_8 = \{T\}$
$L_9 := L_7 + L_8 = 2$



⇨ Parsimony cost for site 3 is 2

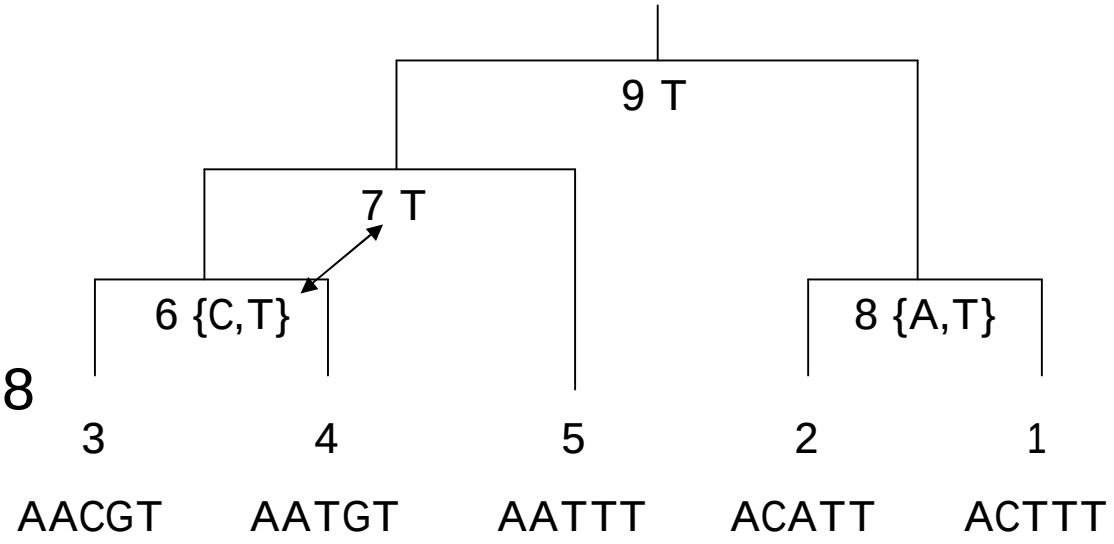
Parsimony algorithm: second phase

- Backtrack from the root and assign $x \in F_i$ at each node
- If we assigned y at parent of node i and $y \in F_i$, then assign y
- Else assign $x \in F_i$ by random

Parsimony algorithm: second phase

At node 6, the algorithm assigns T because T was assigned to parent node 7 and T $\in F_6$.

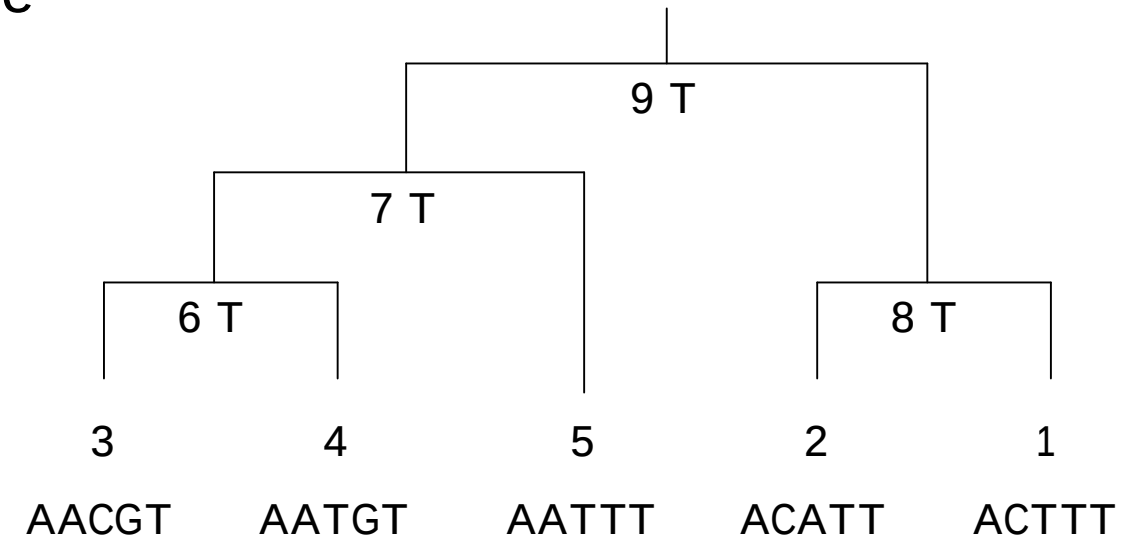
T is assigned to node 8 for the same reason.



The other nodes have only one possible letter to assign

Parsimony algorithm

First and second phase are repeated for each site in the sequences, summing the parsimony costs at each site



Properties of parsimony algorithm

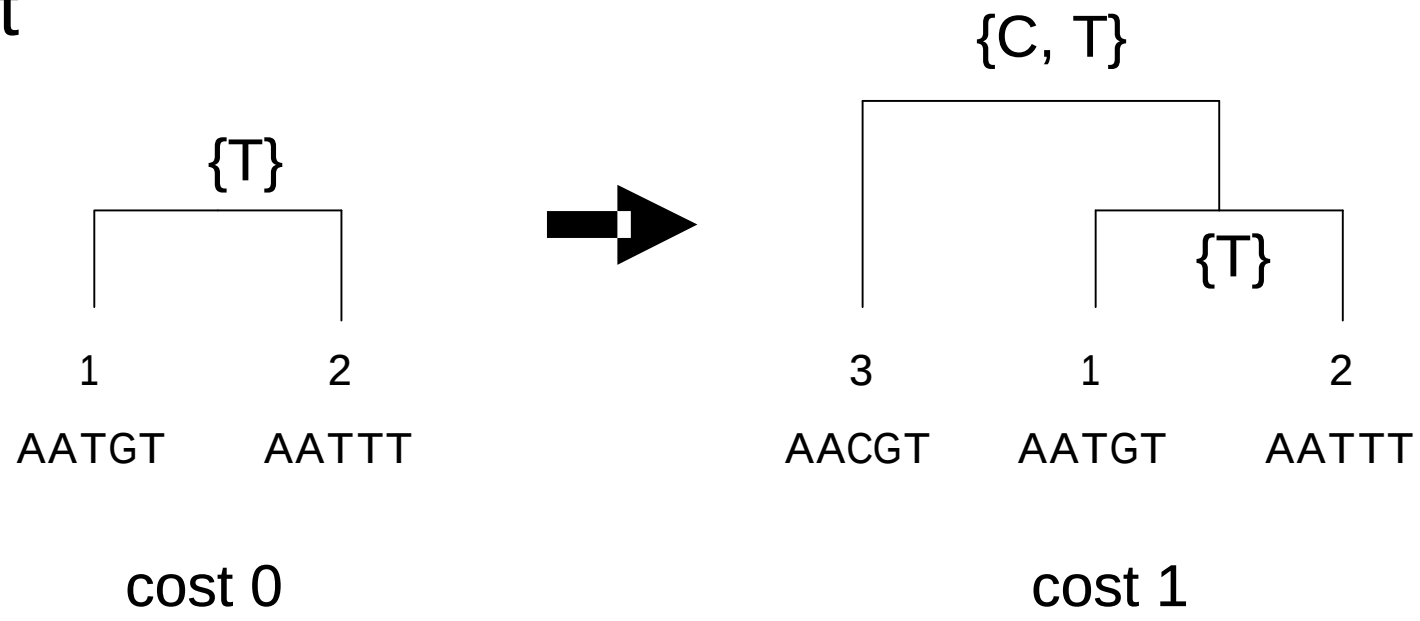
- Parsimony algorithm requires that the sequences are of same length
  - First align the sequences against each other and, optionally, remove indels
  - Then compute parsimony for the resulting sequences
  - Indels (if present) considered as characters
- Is the most parsimonious tree the correct tree?
  - Not necessarily but it explains the sequences with least number of substitutions
  - We can assume that the probability of having fewer mutations is higher than having many mutations

Finding the most parsimonious tree

- Parsimony algorithm calculates the parsimony cost for a given tree…
- …but we still have the problem of finding the tree with the lowest cost
- Exhaustive search (enumerating all trees) is in general impossible
- More efficient methods exist, for example
  - Probabilistic search
  - Branch and bound

Branch and bound in parsimony

- We can exploit the fact that adding edges to a tree can only increase the parsimony cost



Branch and bound in parsimony

Branch and bound is a general search strategy where

- Each solution is potentially generated
- Track is kept of the best solution found
- If a partial solution cannot achieve better score, we abandon the current search path

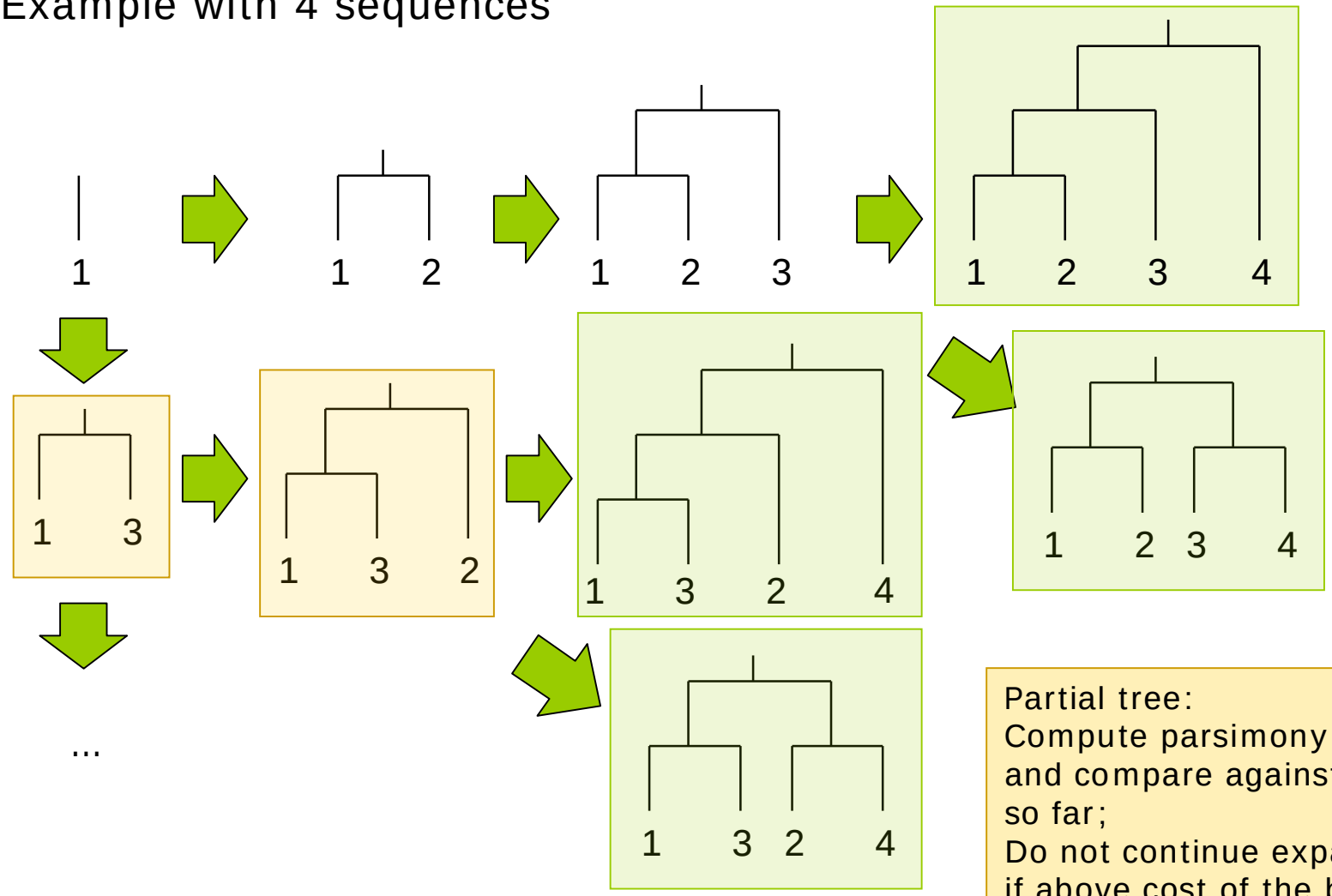
In parsimony…

- Start from a tree with 1 sequence
- Add a sequence to the tree and calculate parsimony cost
- If the tree is complete, check if found the best tree so far
- If tree is not complete and cost exceeds best tree cost, do not continue adding edges to this tree

Branch and bound example

Complete tree: compute parsimony cost

Example with 4 sequences



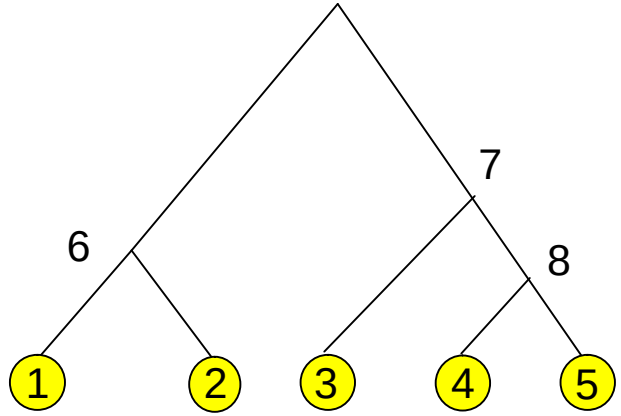
Partial tree:
Compute parsimony cost and compare against best so far;
Do not continue expansion if above cost of the best tree

Distance methods

- The parsimony method works on sequence (character string) data
- We can also build phylogenetic trees in a more general setting
- *Distance methods* work on a set of pairwise distances d_{ij} for the data
- Distances can be obtained from phenotypes as well as from genotypes (sequences)

Distances in a phylogenetic tree

- Distance matrix D = (d_{ij}) gives pairwise distances for *leaves* of the phylogenetic tree
- In addition, the phylogenetic tree will now specify distances between leaves and internal nodes
  - Denote these with d_{ij} as well



Distance d_{ij} states how far apart species i and j are evolutionary (e.g., number of mismatches in aligned sequences)

Distances in evolutionary context

- Distances d_{ij} in evolutionary context satisfy the following conditions
  - Symmetry: $d_{ij} = d_{ji}$ for each i, j
  - Distinguishability: $d_{ij} \neq 0$ if and only if $i \neq j$
  - Triangle inequality: $d_{ij} \leq d_{ik} + d_{kj}$ for each i, j, k
- Distances satisfying these conditions are called metric
- In addition, evolutionary mechanisms may impose additional constraints on the distances
  - *additive* and *ultrametric* distances

Additive trees

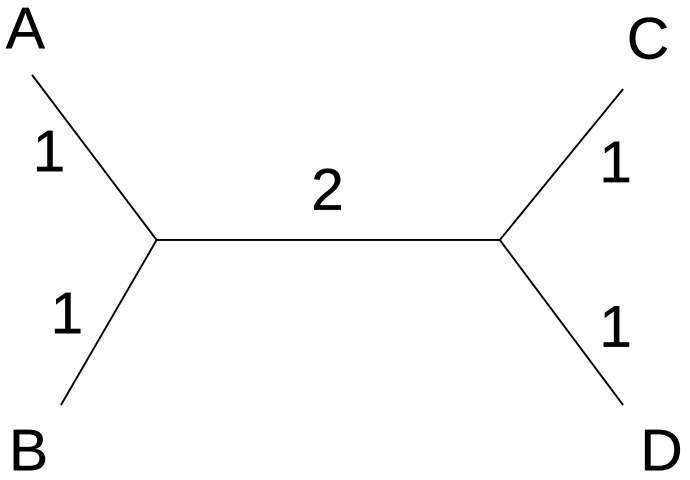
- A tree is called *additive*, if the distance between any pair of leaves (i, j) is the sum of the distances between the leaves and a node k on the shortest path from i to j in the tree

$$d_{ij} = d_{ik} + d_{jk}$$

- ”Follow the path from the leaf i to the leaf j to find the exact distance d_{ij} between the leaves.”

Additive trees: example

	A	B	C	D
A	0	2	4	4
B	2	0	4	4
C	4	4	0	2
D	4	4	2	0



Ultrametric trees

- A rooted additive tree is called an *ultrametric tree*, if the distances between any two leaves i and j, and their common ancestor k are equal

$$d_{ik} = d_{jk}$$

- Edge length d_{ij} corresponds to the time elapsed since divergence of i and j from the common parent
- In other words, edge lengths are measured by a *molecular clock* with a constant rate

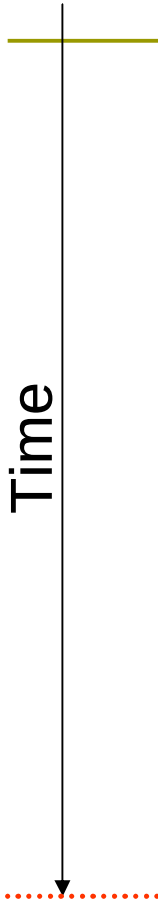
Identifying ultrametric data

- We can identify distances to be ultrametric by the three-point condition:

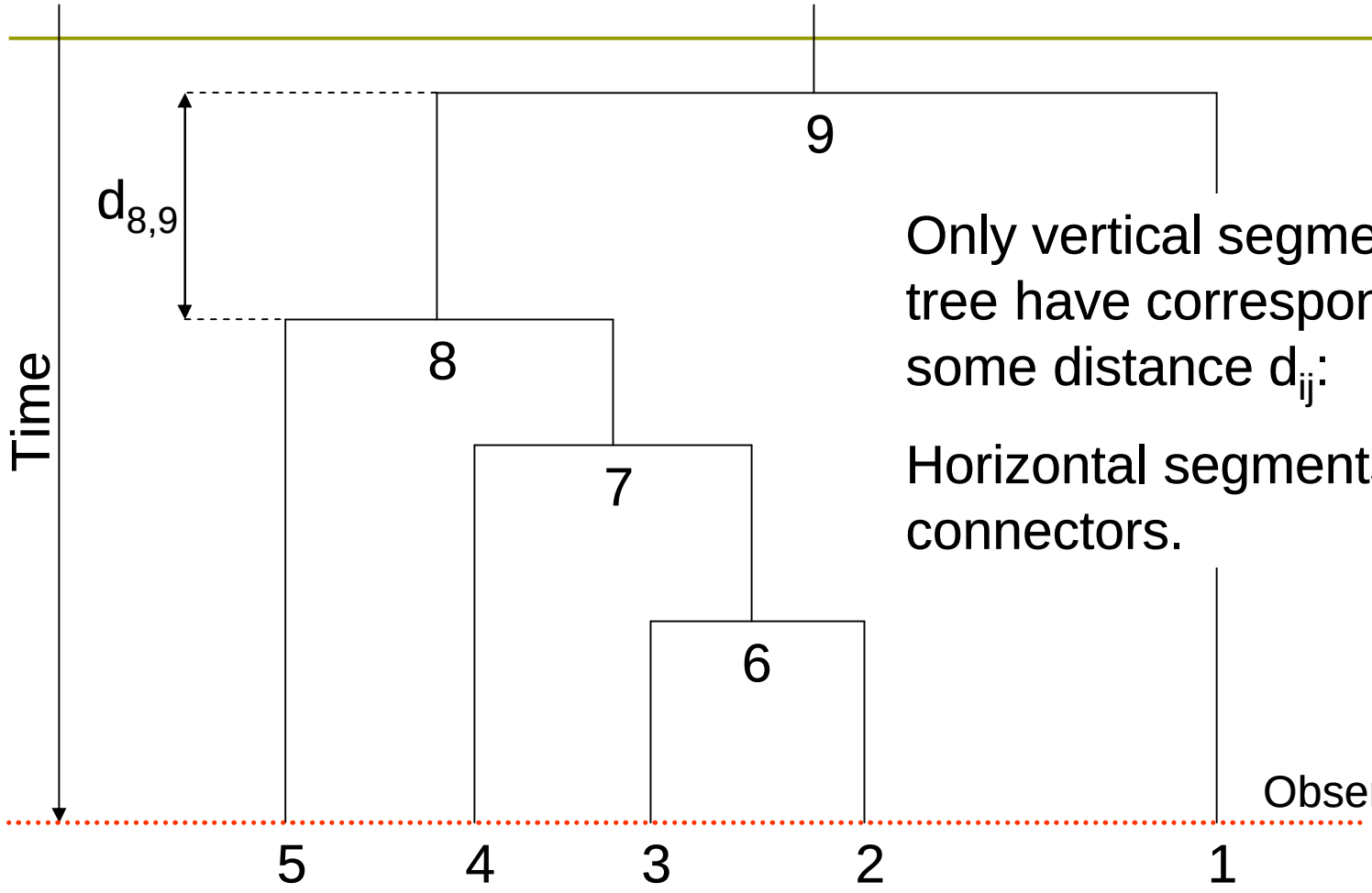
  D corresponds to an ultrametric tree if and only if for any three species i, j and k, the distances satisfy $d_{ij} \leq \max(d_{ik}, d_{kj})$

- If we find out that the data is ultrametric, we can utilise a simple algorithm to find the corresponding tree

Ultrametric trees



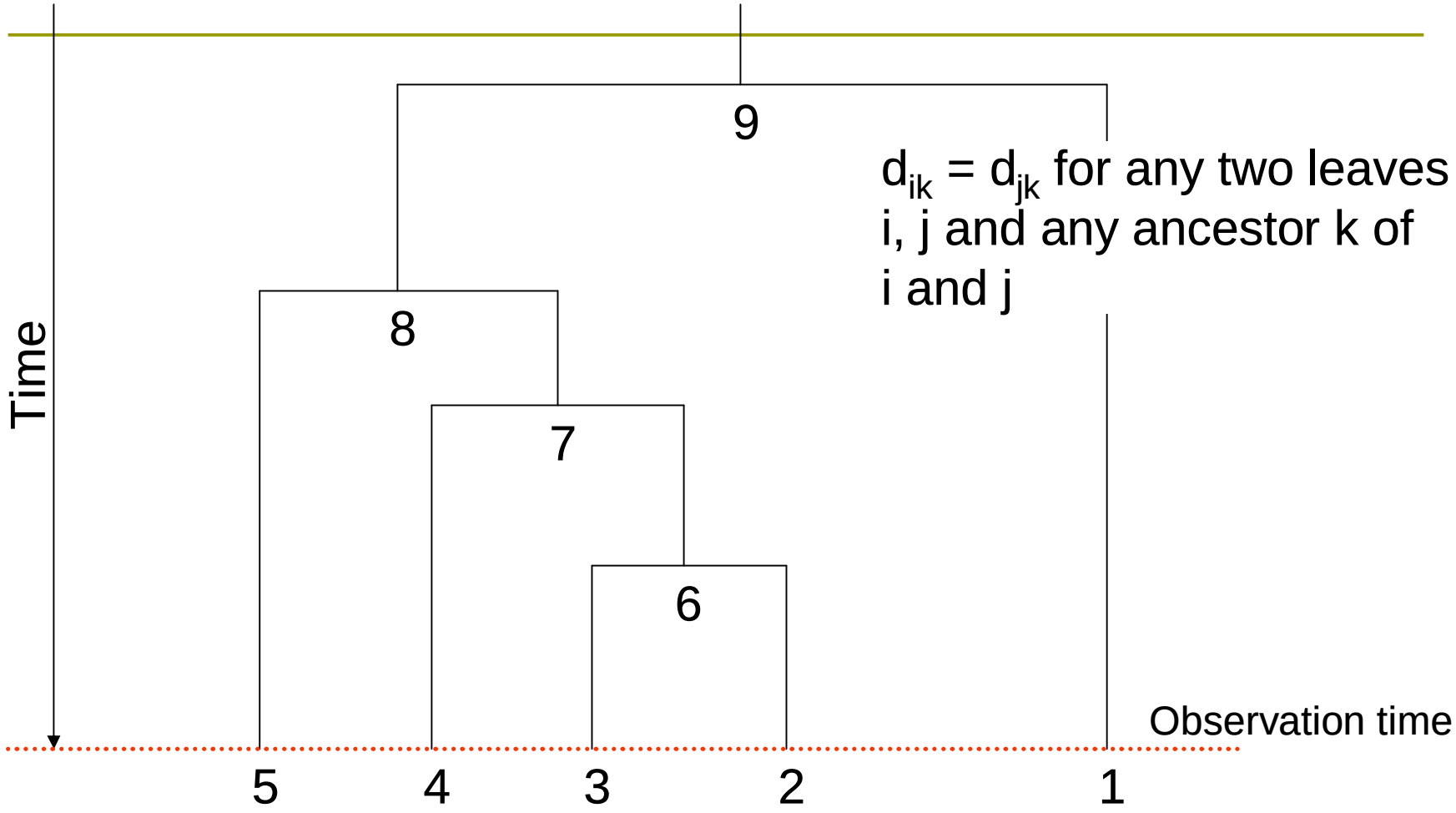
Ultrametric trees



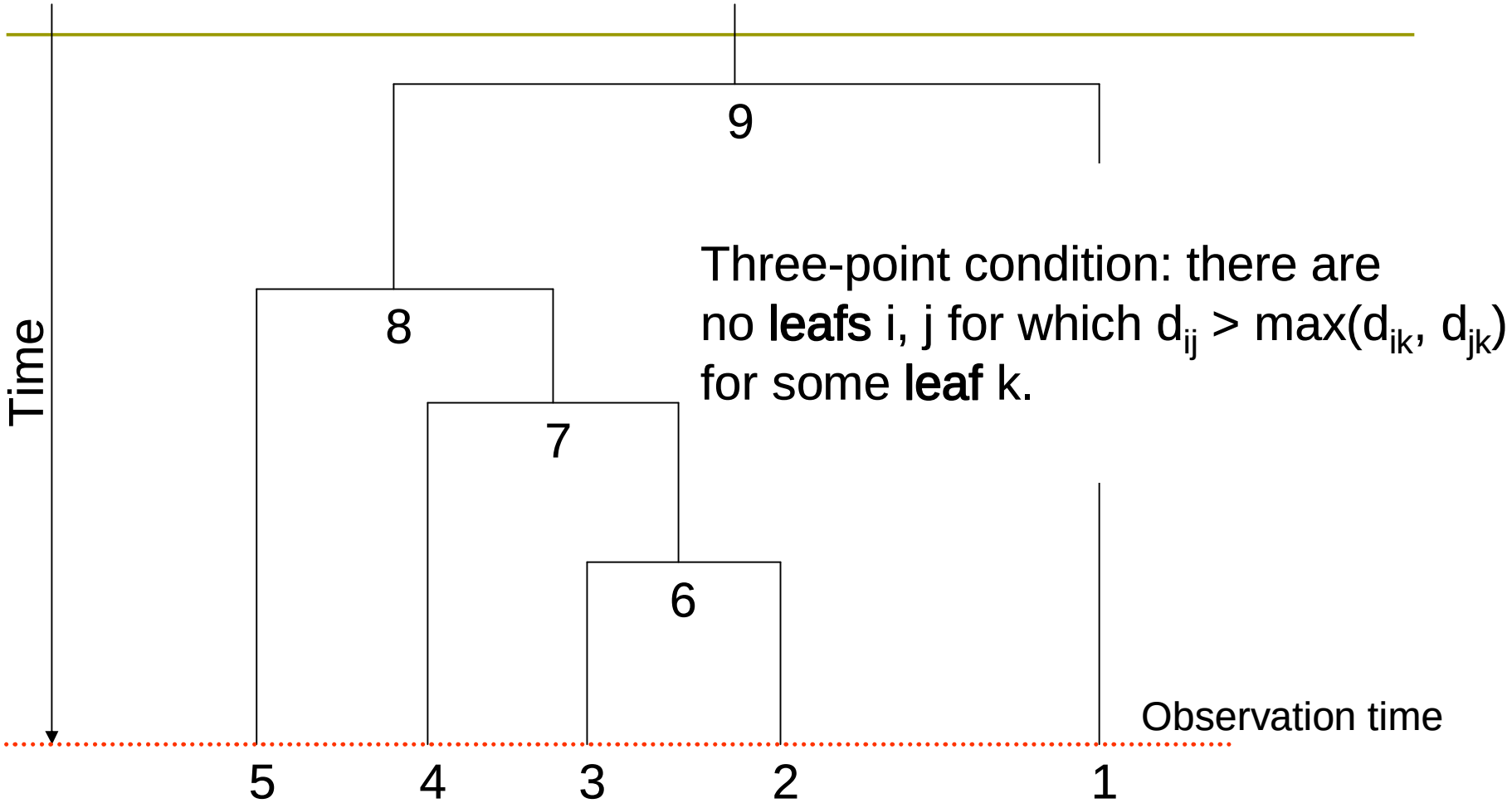
Only vertical segments of the tree have correspondence to some distance d_{ij}:

Horizontal segments act as connectors.

Ultrametric trees



Ultrametric trees



Three-point condition: there are no **leafs** i, j for which $d_{ij} > \max(d_{ik}, d_{jk})$ for some **leaf** k.

UPGMA algorithm

- UPGMA (unweighted pair group method using arithmetic averages) constructs a phylogenetic tree via clustering
- The algorithm works by at the same time
  - Merging two clusters
  - Creating a new node on the tree
- The tree is built from leaves towards the root
- UPGMA produces a ultrametric tree

Cluster distances

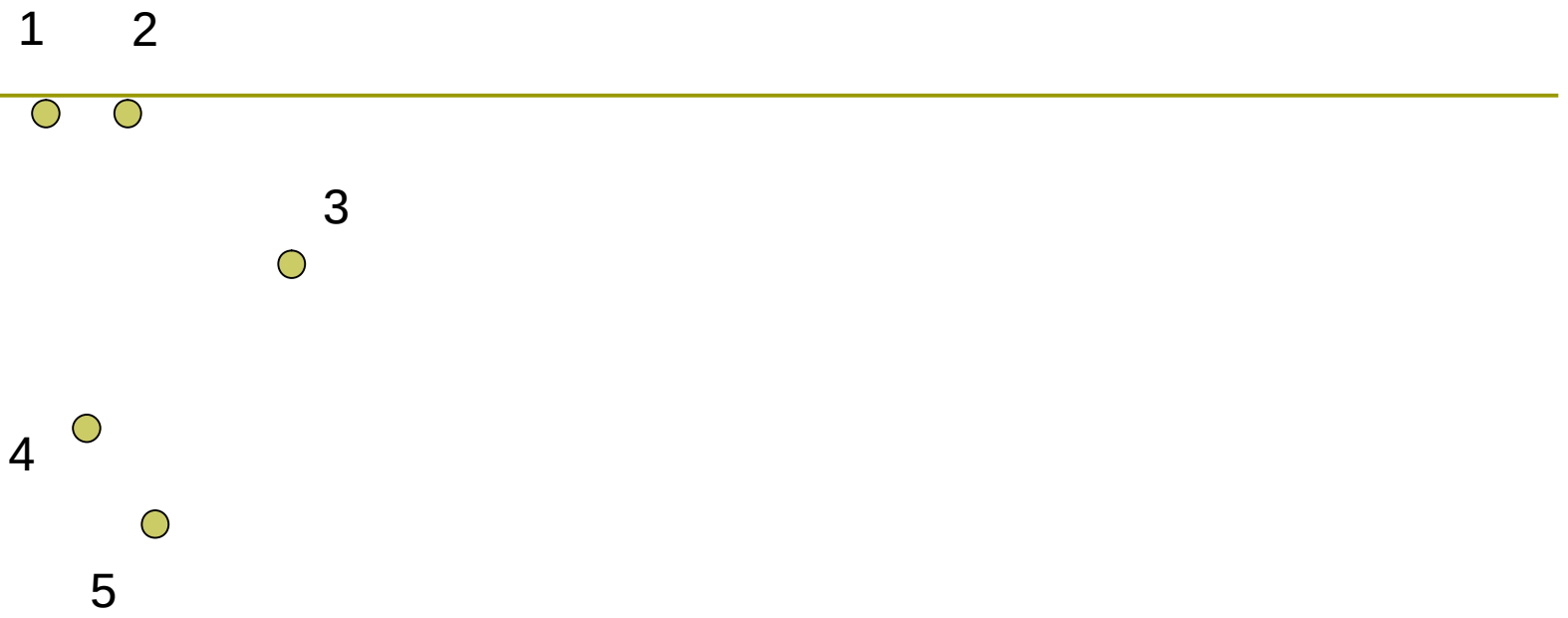
- Let distance d_{ij} between clusters C_i and C_j be

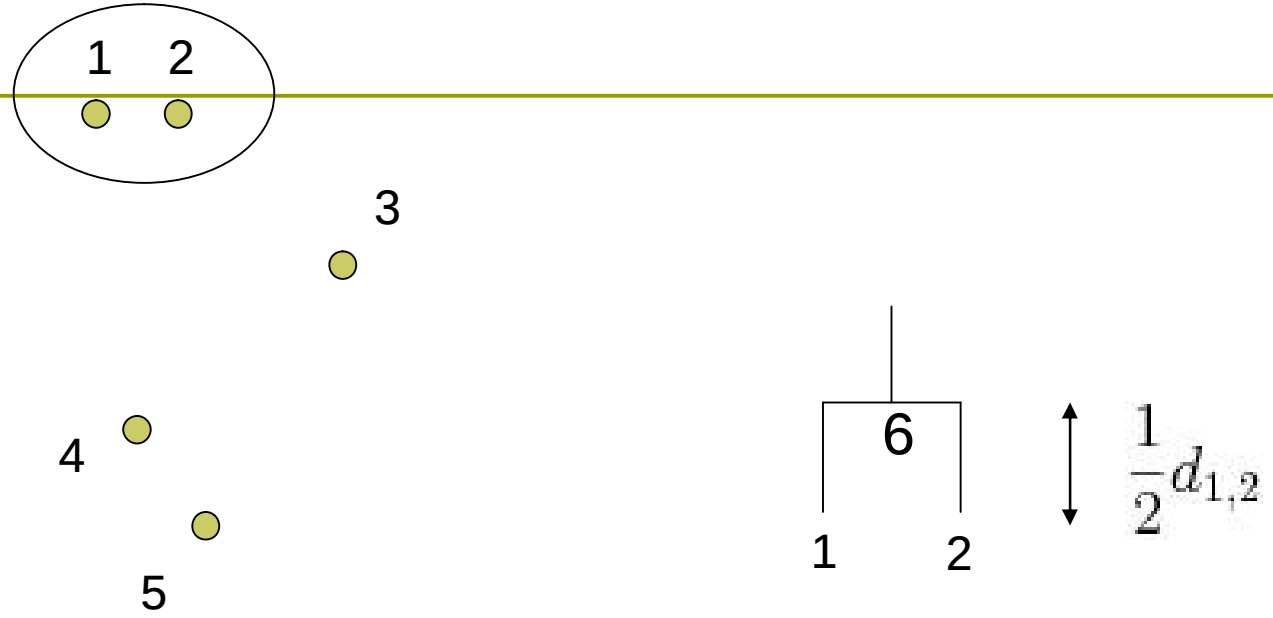
$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

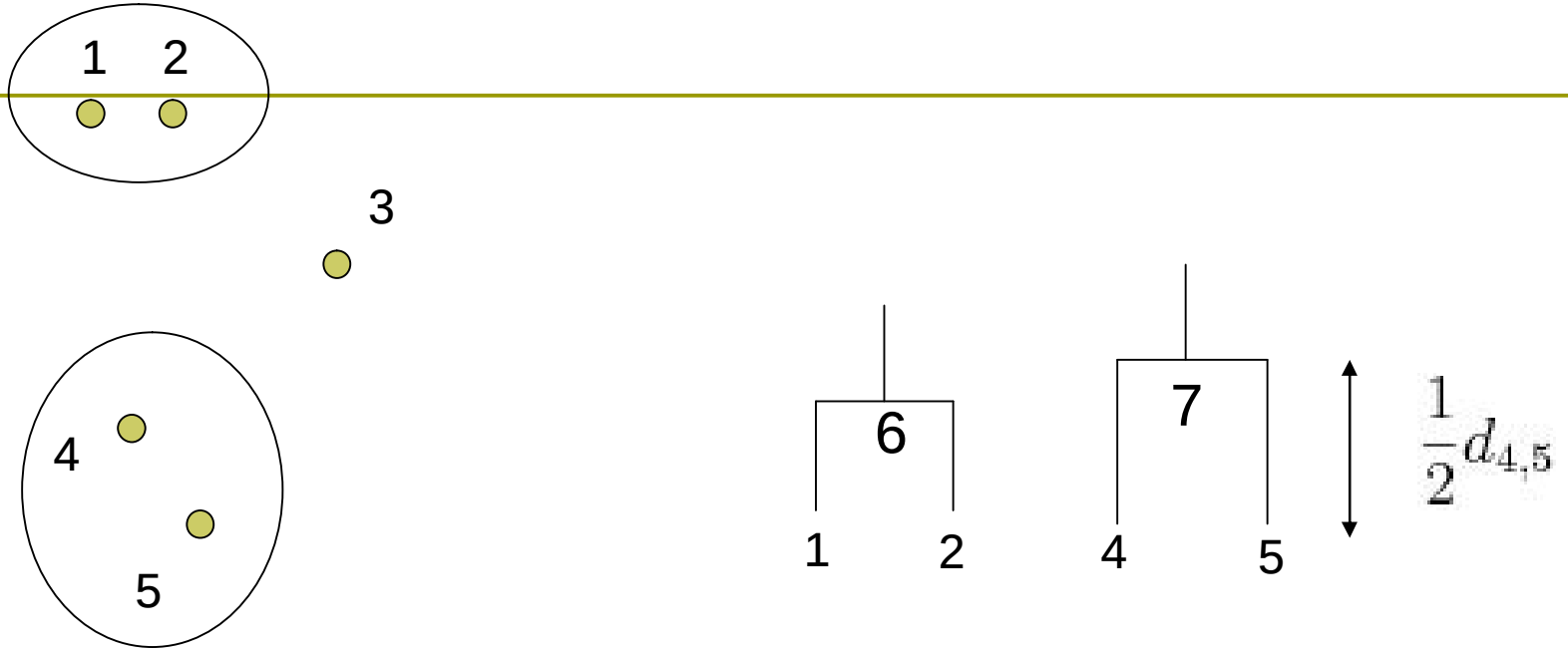
that is, the average distance between points (species) in the cluster.

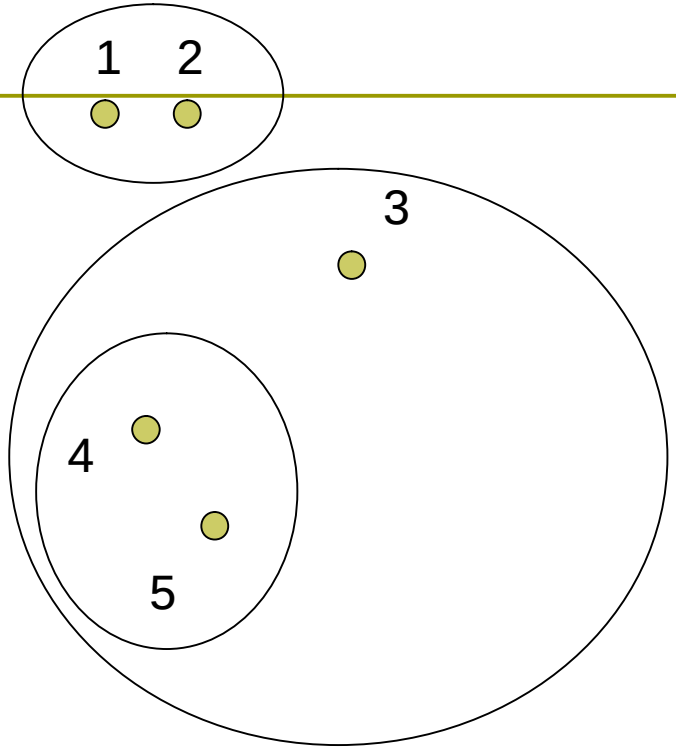
UPGMA algorithm

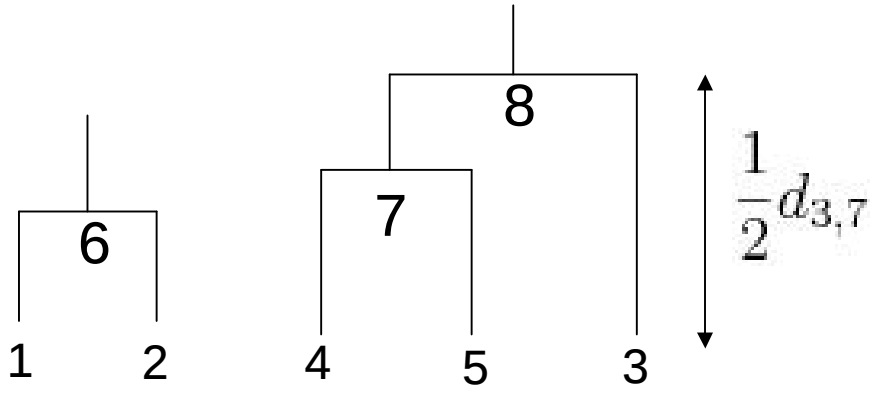
- Initialisation
  - Assign each point i to its own cluster C_i
  - Define one leaf for each sequence, and place it at height zero
- Iteration
  - Find clusters i and j for which d_{ij} is minimal
  - Define new cluster k by $C_k = C_i \cup C_j$, and define d_{kl} for all l
  - Define a node k with children i and j. Place k at height $d_{ij}/2$
  - Remove clusters i and j
- Termination:
  - When only two clusters i and j remain, place root at height $d_{ij}/2$

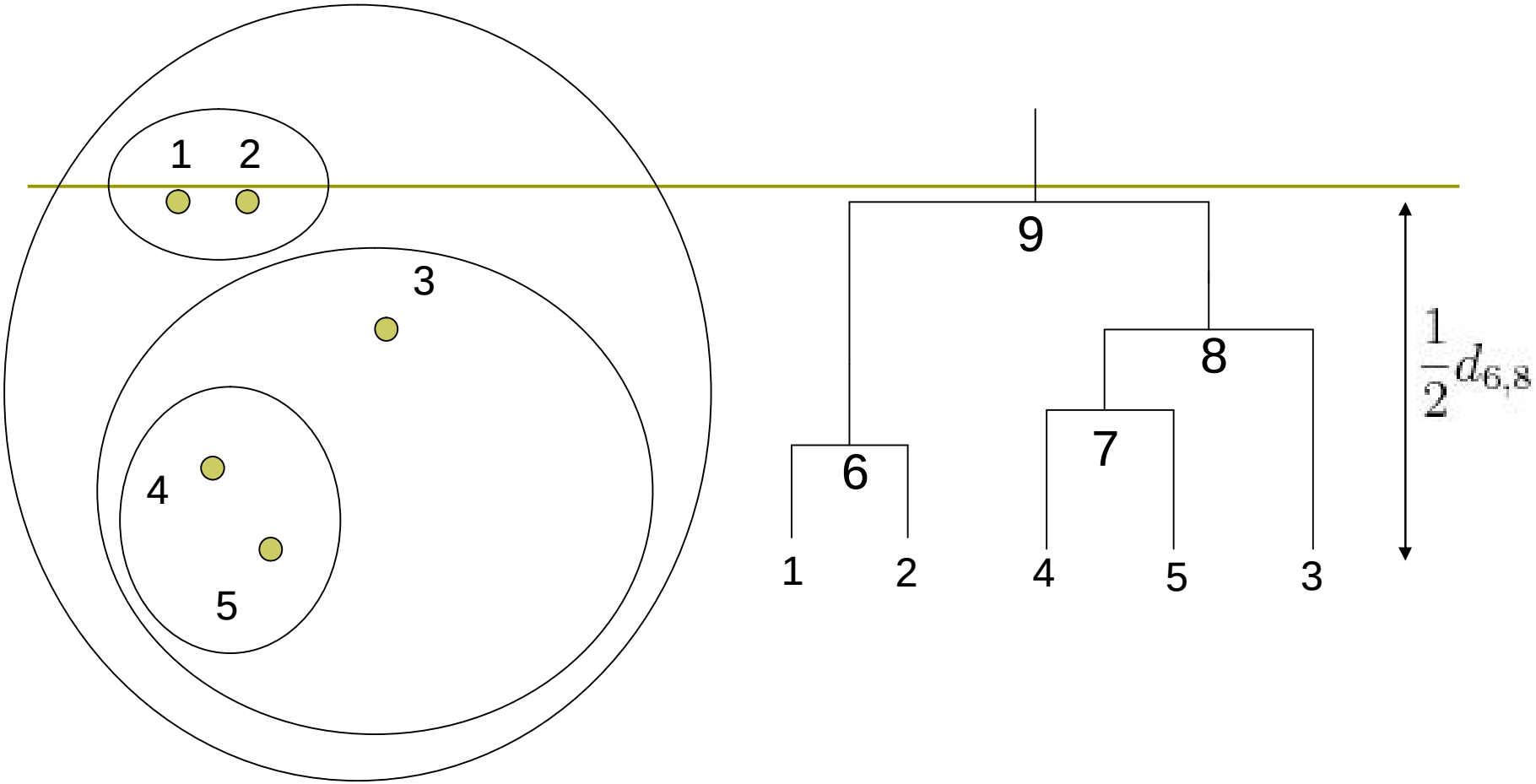

1
2
3
4
5


1
2
3
4
5
6
1
2
$\frac{1}{2}d_{1,2}$


1
2
3
4
5
6
7
1
2
4
5
$\frac{1}{2}d_{4,5}$


1
2
3
4
5


6
1
2
8
7
4
5
3
$\frac{1}{2}d_{3,7}$


1
2
3
4
5
9
8
7
6
1
2
4
5
3
$\frac{1}{2}d_{6,8}$

UPGMA implementation

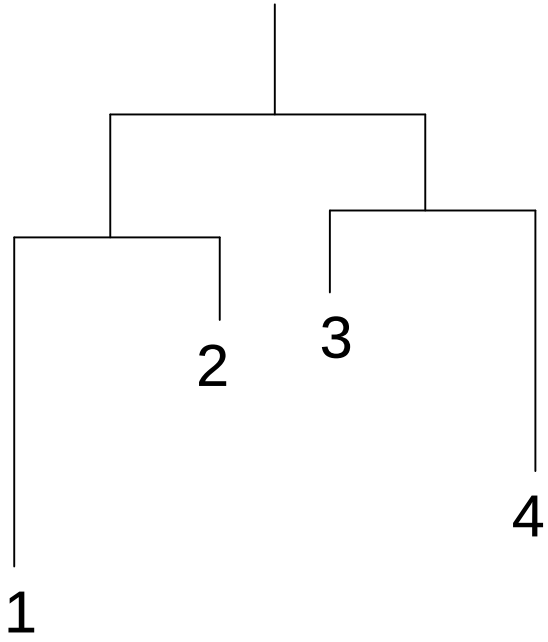
- In naive implementation, each iteration takes $O(n^2)$ time with n sequences => algorithm takes $O(n^3)$ time

- The algorithm can be implemented to take only $O(n^2)$ time (see Gronau & Moran, 2006, for a survey)

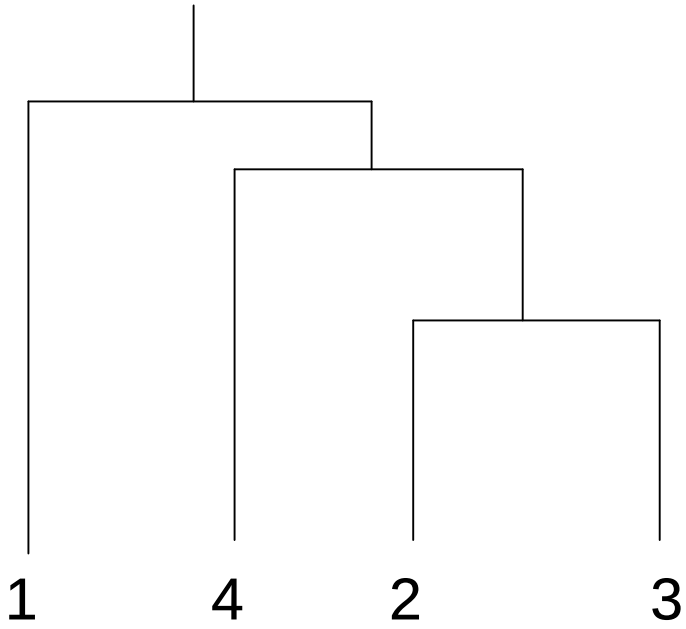
Problem solved?

- We now have a simple algorithm which finds a ultrametric tree
  - If the data is ultrametric, then there is exactly one ultrametric tree corresponding to the data (we skip the proof)
  - The tree found is then the ”correct” solution to the phylogeny problem, if the assumptions hold
- Unfortunately, the data is not ultrametric in practice
  - Measurement errors distort distances
  - *Basic assumption of a molecular clock does not hold usually very well*

Incorrect reconstruction of non-ultrametric data by UPGMA



Tree which corresponds to non-ultrametric distances

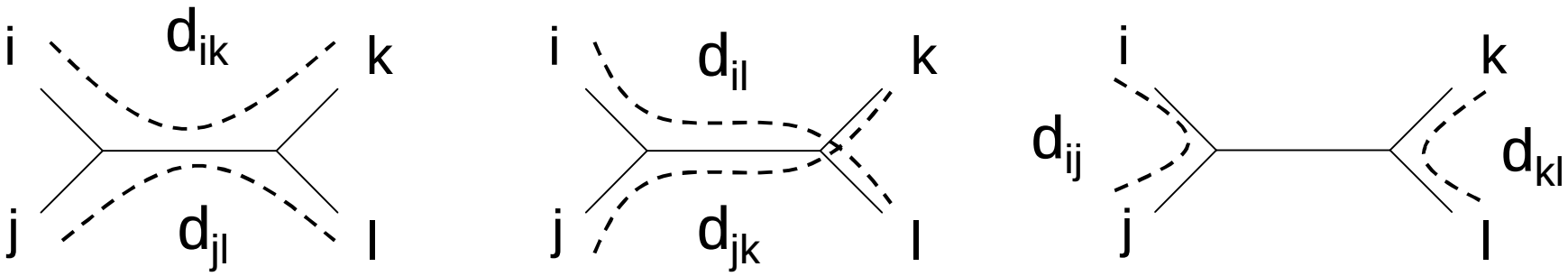


Incorrect ultrametric reconstruction by UPGMA algorithm

Checking for additivity

- How can we check if our data is additive?
- Let i, j, k and l be four *distinct* species
- Compute 3 sums: $d_{ij} + d_{kl}$, $d_{ik} + d_{jl}$, $d_{il} + d_{jk}$

Four-point condition



- The sums are represented by the three figures
  - Left and middle sum cover all edges, right sum does not
- *Four-point condition*: i, j, k and l satisfy the four-point condition if two of the sums $d_{ij} + d_{kl}$, $d_{ik} + d_{jl}$, $d_{il} + d_{jk}$ are the same, and the third one is smaller than these two

Checking for additivity

- An n x n matrix D is additive if and only if the four point condition holds for every 4 distinct elements $1 \leq i, j, k, l \leq n$

Finding an additive phylogenetic tree

- Additive trees can be found with, for example, the neighbor joining method (Saitou & Nei, 1987)
- The neighbor joining method produces unrooted trees, which have to be rooted by other means
  - A common way to root the tree is to use an outgroup
  - Outgroup is a species that is known to be more distantly related to every other species than they are to each other
  - Root node candidate: position where the outgroup would join the phylogenetic tree
- However, in real-world data, even additivity usually does not hold very well

Neighbor joining algorithm

- Neighbor joining works in a similar fashion to UPGMA
  - Find clusters C_1 and C_2 that minimise a function $f(C_1, C_2)$
  - Join the two clusters C_1 and C_2 into a new cluster C
  - Add a node to the tree corresponding to C
  - Assign distances to the new branches
- Differences in
  - The choice of function $f(C_1, C_2)$
  - How to assign the distances

Neighbor joining algorithm

- Recall that the distance d_{ij} for clusters C_i and C_j was

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

- Let $u(C_i)$ be the separation of cluster C_i from other clusters defined by

$$u(C_i) = \frac{1}{n-2} \sum_{C_j} d_{ij}$$

where n is the number of clusters.

Neighbor joining algorithm

- Instead of trying to choose the clusters C_i and C_j closest to each other, neighbor joining at the same time
  - Minimises the distance between clusters C_i and C_j and
  - Maximises the separation of both C_i and C_j from other clusters

Neighbor joining algorithm

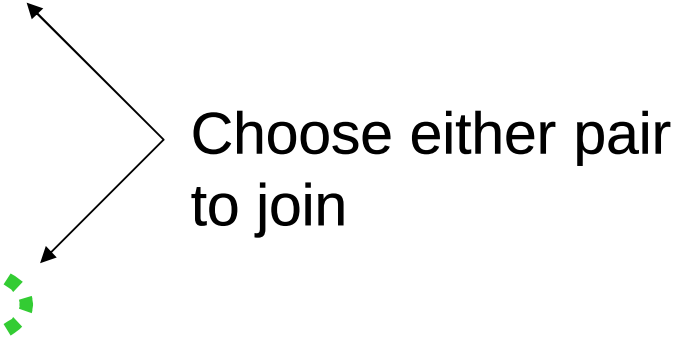
- Initialisation as in UPGMA
- Iteration
  - Find clusters i and j for which $d_{ij} - u(C_i) - u(C_j)$ is minimal
  - Define new cluster k by $C_k = C_i \cup C_j$, and define d_{kl} for all l
  - Define a node k with edges to i and j. Remove clusters i and j
  - Assign length $\frac{1}{2} d_{ij} + \frac{1}{2} (u(C_i) - u(C_j))$ to the edge i -> k
  - Assign length $\frac{1}{2} d_{ij} + \frac{1}{2} (u(C_j) - u(C_i))$ to the edge j -> k
- Termination:
  - When only one cluster remains

Neighbor joining algorithm: example

	a	b	c	d
a	0	6	7	5
b		0	11	9
c			0	6
d				0

i	u(i)
a	(6+7+5)/2 = 9
b	(6+11+9)/2 = 13
c	(7+11+6)/2 = 12
d	(5+9+6)/2 = 10

i,j	d_{ij} – $u(C_i)$ – $u(C_j)$	
a,b	6 - 9 - 13	= -16
a,c	7 - 9 - 12	= -14
a,d	5 - 9 - 10	= -14
b,c	11 - 13 - 12	= -14
b,d	9 - 13 - 10	= -14
c,d	6 - 12 - 10	= -16

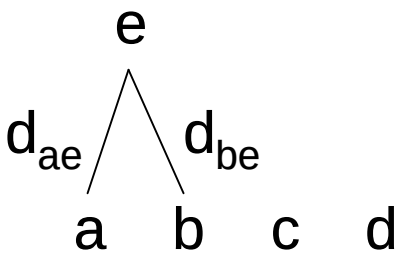


Neighbor joining algorithm: example

	a	b	c	d
a	0	6	7	5
b		0	11	9
c			0	6
d				0

i	u(i)
a	(6+7+5)/2 = 9
b	(6+11+9)/2 = 13
c	(7+11+6)/2 = 12
d	(5+9+6)/2 = 10

i,j	d_{ij} – $u(C_i)$ – $u(C_j)$	
a,b	6 - 9 - 13	= -16
a,c	7 - 9 - 12	= -14
a,d	5 - 9 - 10	= -14
b,c	11 - 13 - 12	= -14
b,d	9 - 13 - 10	= -14
c,d	6 - 12 - 10	= -16



d_{ae} = ½ 6 + ½ (9 – 13) = 1

d_{be} = ½ 6 + ½ (13 – 9) = 5

This is the first step only…

Inferring the Past: Phylogenetic Trees (chapter 12)

- The biological problem
- Parsimony and distance methods
- *Models for mutations and estimation of distances*
- *Maximum likelihood methods*
  - These parts of the book is skipped on this course (see slides of 2007 course for material on these topics)
  - No questions in exams on these topics!

Problems with tree-building

- Assumptions
  - Sites evolve independently of one other
  - (Sites evolve according to the same stochastic model; not really covered this year)
  - The tree is rooted
  - The sequences are aligned
  - Vertical inheritance

Additional material on phylogenetic trees

- Durbin, Eddy, Krogh, Mitchison: Biological sequence analysis
- Jones, Pevzner: An introduction to bioinformatics algorithms
- Gusfield: Algorithms on strings, trees, and sequences

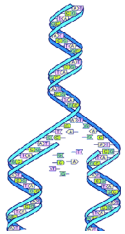
- Course on phylogenetic analyses in Spring 2009

Introduction to Bioinformatics

Wrap-up

2010: Human genome in 15 minutes!

Topics



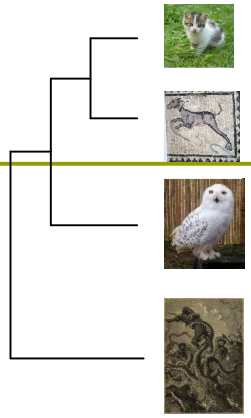
Sequencing
-Sanger
-High-throughput

```
atgagccaag ttccgaacaa ggattcgcgg
gtcggtaaag agcattggaa cgtcggagat
aagaagcgga tgaatttccc cataacgcca
gtggaagaga aggaggcggg cctcccgatc
actccggccc gaagggttga gagtacccca
gaaatcacct ccagaggacc ccttcagcga
catagcgata ggaggggatg ctaggagttg
```

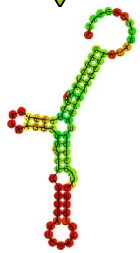
Reads

Sequence assembler

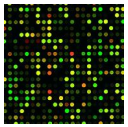
Contigs
=> Genomes



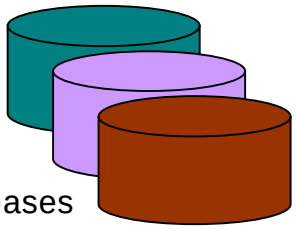
Comparative genomics
- *Phylogenetics*
- *Genome rearrangements*
- *...*



Gene expression analysis



DNA chips



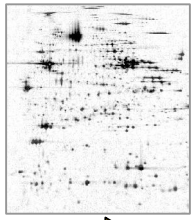
Biodatabases

BLAST query

gtggaagagaaggaggcgg...

gaagggttgagagtacccca...
ccagaggacccccttcagcga...
ggaggggatgctaggagttg...

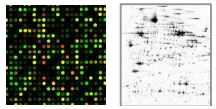
Similar sequences = homologs?

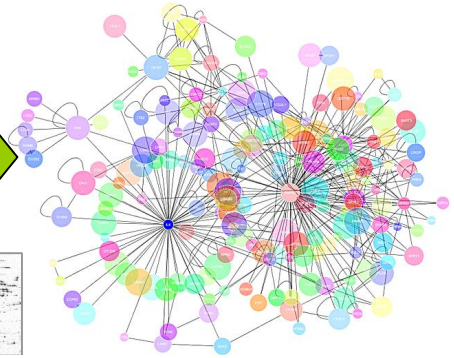


Proteomics

Protein gels
MS/MS techniques

```
atgagccaag
aagaagcgga
cagcggaaga
```





Systems Biology

Exams

- Course exam Wednesday 15 October 16.00-19.00 Exactum A111
- Separate exams
  - Tue 18 November 16.00-20.00 Exactum A111
  - Fri 16 January 16.00-20.00 Exactum A111
  - Tue 31 March 16.00-20.00 Exactum A111
- Check exam date and place before taking the exam! (previous week or so)

Exam regulations

- If you are late more than 30 min, you cannot take the exam
- You are not allowed to bring material such as books or lecture notes to the exam
- Allowed stuff: blank paper (distributed in the exam), pencils, pens, erasers, calculators, snacks
- Bring your student card or other id!

Grading

- Grading: on the scale 0-5
  - To get the lowest passing grade 1, you need to get at least 30 points out of 60 maximum
- Course exam gives you maximum of 48 points
- Note: if you take the first separate exam, the best of the following options will be considered:
  - Exam gives you max 48 points, exercises max 12 points
  - Exam gives you max 60 points
- In second and subsequent separate exams, only the 60 point option is in use

Exercise points

- Max. marks: 31
- 80% of 31 ~= 24 marks -> 12 points
- 2 marks = 1 point

Topics covered by exams

- Exams cover everything presented in lectures (exception: biological background not covered)

- Word distributions and occurrences (course book chapters 2-3)
- Genome rearrangements (chapter 5)
- Sequence alignment (chapter 6)
- Rapid alignment methods: FASTA and BLAST (chapter 7)
- Sequencing and sequence assembly (chapter 8)

Topics covered by exams

- Similarity, distance and clustering (chapter 10)
- Expression data analysis (chapter 11)
- Phylogenetic trees (chapter 12)
- Systems biology: modelling biological networks (no chapter in course book)

Bioinformatics courses in 2008

- Biological sequence analysis (II period, Kumpula)
  - Focus on probabilistic methods: Hidden Markov Models, Profile HMMs, finding regulatory elements, …
- Modeling of biological networks (20-24.10., TKK)
  - Biochemical network modelling and parameter estimation in biochemical networks using mechanistic differential equation models.

Bioinformatics courses in autumn 2008

- Bayesian paradigm in genetic bioinformatics (II period, Kumpula)
  - Applications of Bayesian approach in computer programs and data analysis of
    - genetic past,
    - phylogenetics,
    - coalescence,
    - relatedness,
    - haplotype structure,
    - disease gene associations.

Bioinformatics courses in autumn 2008

- Statistical methods in genetics (II period, Kumpula)
  - Introduction to statistical methods in gene mapping and genetic epidemiology.
  - Basic concepts of linkage and association analysis as well as some concepts of population genetics will be covered.

Bioinformatics courses in Spring 2009

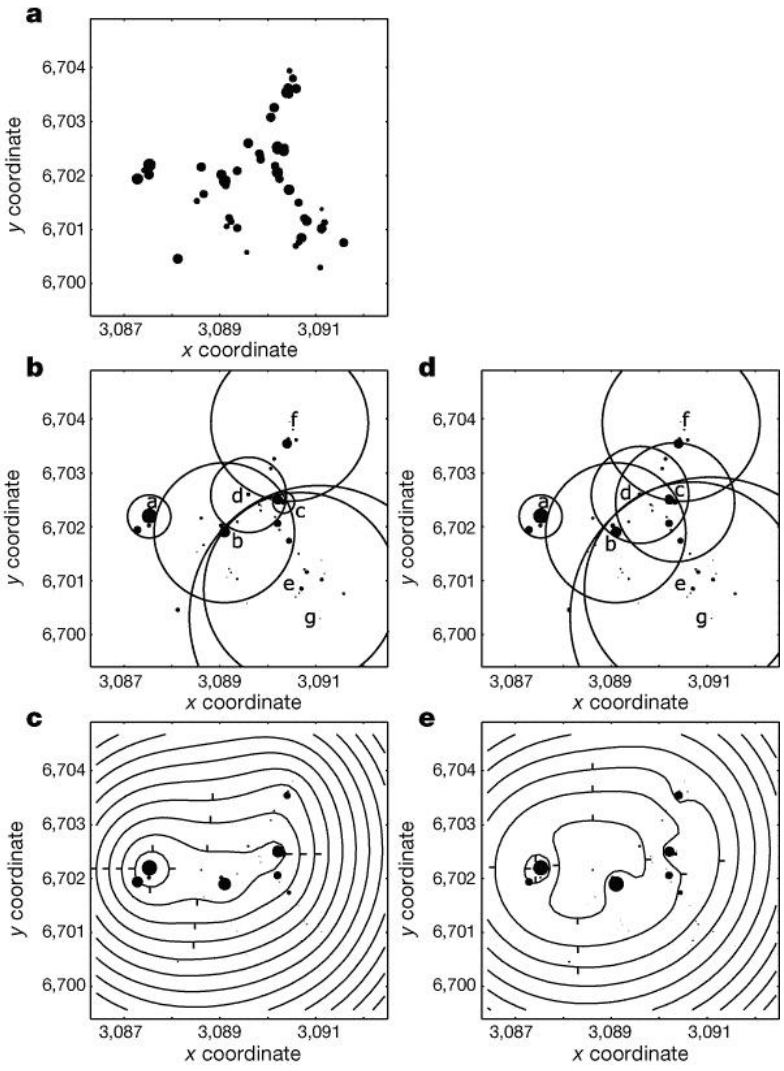
- Practical Course in Biodatabases (III period, Kumpula)
- High-throughput bioinformatics (III-IV periods, TKK)
- Phylogenetic data analyses (IV period, Kumpula)
  - Maximum likelihood methods, Bayesian methods, program packages
- Metabolic modelling (IV period, Kumpula)

Genomes sequenced – all done?

- Sequencing is just the beginning
  - What do genes and proteins do?
    - Functional genomics
  - How do they interact with other genes and proteins?
    - Systems biology

Two sides of the same question!

Bioinformatics (at least mathematical biology) can exist outside molecular biology



Melitaea cinxia, Glanville Fritillary butterfly

The metapopulation capacity of a fragmented landscape
Ilkka Hanski and Otso Ovaskainen
Nature 404, 755-758(13 April 2000)

Metagenomics

- Metagenomics or environmental genomics
  - ”At the last count 1.8 million species were known to science. That sounds like a lot, but in truth it's no big deal. We may have done a reasonable job of describing the larger stuff, but the fact remains that an average teaspoon of water, soil or ice contains millions of micro-organisms that have never been counted or named. ”

    -- Henry Nicholls

Omics

- Genome
- Transcriptome
- Metabolome
- Metallome
- Lipidome
- Glycome
- Interactome
- Spliceome
- ORFeome
- Speechome
- Mechanome
- Phenome
- Exposome
- Textome
- Receptorome
- Kinome
- Neurome
- Cytome
- Predictome
- Omeome
- Reactome
- Connectome

http://en.wikipedia.org/wiki/-omics

Take-home messages

- Don’t trust biodatabases blindly!
  - Annotation errors tend to accumulate
- Consider
  - Statistical significance
  - Sensitivity

  of your results
- Think about the whole ”bioinformatics workflow”:
  - Biological phenomenon -> Modelling -> Computation -> Validation of results
- Results from bioinformatics tools and methods must be validated!
- Actively seek cooperation with experts

Bioinformatics journals

- Bioinformatics, http://bioinformatics.oupjournals.org/
- BMC Bioinformatics, http://www.biomedcentral.com/bmcbioinformatics
- Journal of Bioinformatics and Computational Biology (JBCB), http://www.worldscinet.com/jbcb/jbcb.shtml
- Journal of Computational Biology, http://www.liebertpub.com/CMB/
- IEEE/ACM Transactions on Computational Biology and Bioinformatics , http://www.computer.org/tcbb/
- PLoS Computational Biology, www.ploscompbiol.org
- In Silico Biology, http://www.bioinfo.de/isb/
- Nature, Science (bedtime reading)

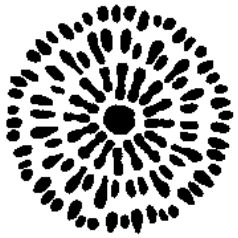
Bioinformatics conferences

- ISMB, Intelligent Systems for Molecular Biology (Toronto, July 2008)
- ICSB, International Conference on Systems Biology (Göteborg, Sweden; 22-28 August)
- RECOMB, Research in Computational Molecular Biology
- ECCB, European Conference on Computational Biology
- WABI, Workshop on Algorithms in Bioinformatics
- PSB, Pacific Symposium on Biocomputing

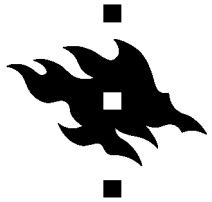
January 5-9, 2009
The Big Island of Hawaii

Master's degree in bioinformatics?

- You can apply to MBI during the application period November '08 – 2 February '09
  - Bachelor's degree in suitable field
  - At least 60 ECTS credits in CS or mathstat
  - English language certificate
- Passing this course gives you the first 4 credits for Bioinformatics MSc!



MBI MASTER'S DEGREE
PROGRAMME IN BIOINFORMATICS



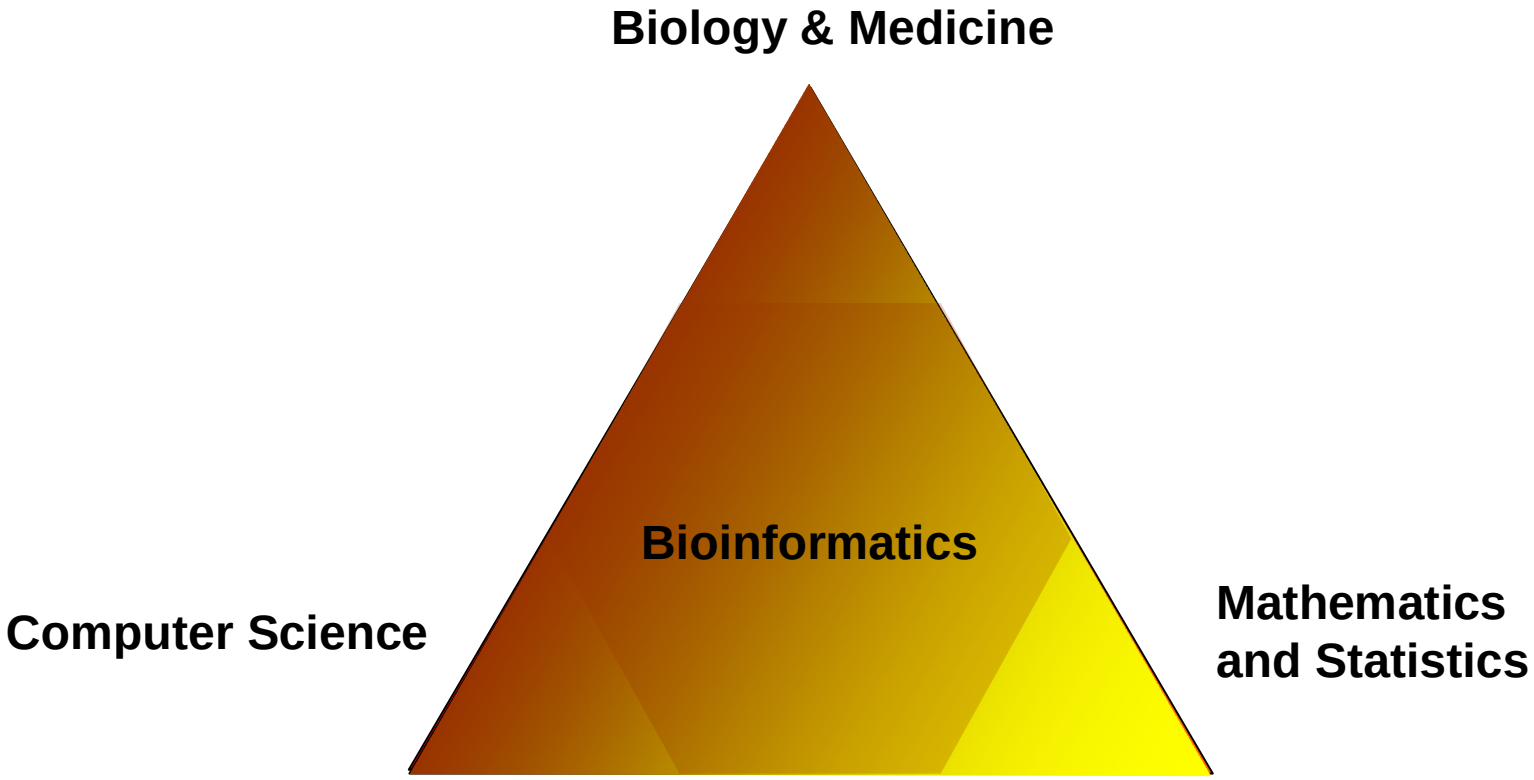
Information session on MBI

- Wednesday 19.11. 13.00-15.00 Exactum D122
- www.cs.helsinki.fi/mbi/events/info08
- Talks in Finnish

Mailing list for bioinformatics courses and events

- MBI maintains a mailing list for announcement on bioinformatics courses and events
- Send email to bioinfo a t cs.helsinki.fi if you want to subscribe to the list (you can unsubscribe in the same way)
- List is moderated

The aim of this course



Where would you be in this triangle?

Has your position shifted during the course?

Feedback

- Please give feedback on the course!
  - https://ilmo.cs.helsinki.fi/kurssit/servlet/Valinta?kieli=en
- Don’t worry about your grade – you can give feedback anonymously

Thank you!

- I hope you enjoyed the course!

Halichoerus grypus, Gray seal or *harmaahylje* in Finnish