Elements of Bioinformatics Autumn 2010

VELI MÄKINEN

HTTP://WWW.CS.HELSINKI.FI/EN/COURSES/582606/2010/S/K/1

Lecture Mon 29.11

NEIGHBOR JOINING
AND
SEQUENCE ASSEMBLY

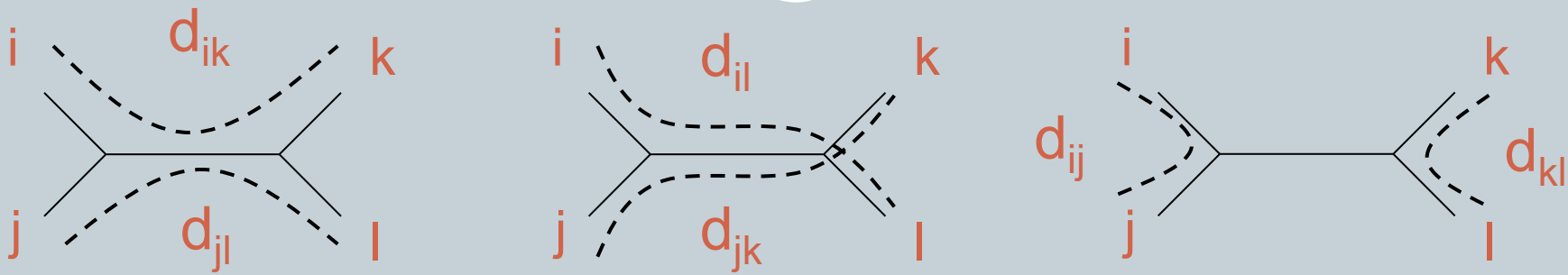
Part I

NEIGHBOR JOINING ALGORITHM AND PROOF THAT IT CONSTRUCTS AN ADDITIVE TREE

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 $n \times n$ matrix D is additive if and only if the four point condition holds for every 4 distinct elements $1 \leq i, j, k, l \leq n$
- See exercises for grounding of three-point (ultrametric) and four-point (additive) conditions.

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

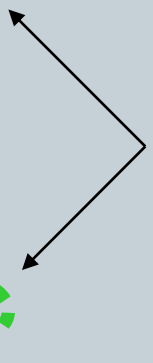
- Initialisation as in UPGMA
- Iteration
  - Find clusters i and j for which $d_{ij} - u(C_i) - u(C_j)$ is minimal
  - Define new cluster k by $C_k = C_i \cup C_j$, and define d_{kl} for all l
  - Define a node k with edges to i and j. Remove clusters i and j
  - Assign length $\frac{1}{2}\, d_{ij} +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



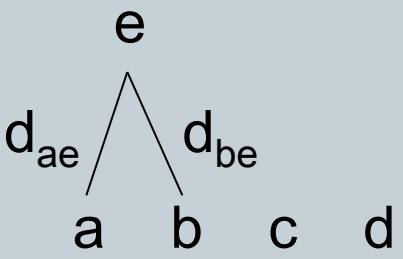
Choose either pair to join

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…

Neighbor joining algorithm: correctness

- ***Theorem***: *If D is an additive matrix, neighbor joining algorithm correctly constructs the corresponding additive tree.*
  Proof. (given on blackboard)
  Idea: Show that the leaves i and j joined must be neighbors in the additive tree (see Durbin et al., pp. 190-191). Then the theorem follows by induction.

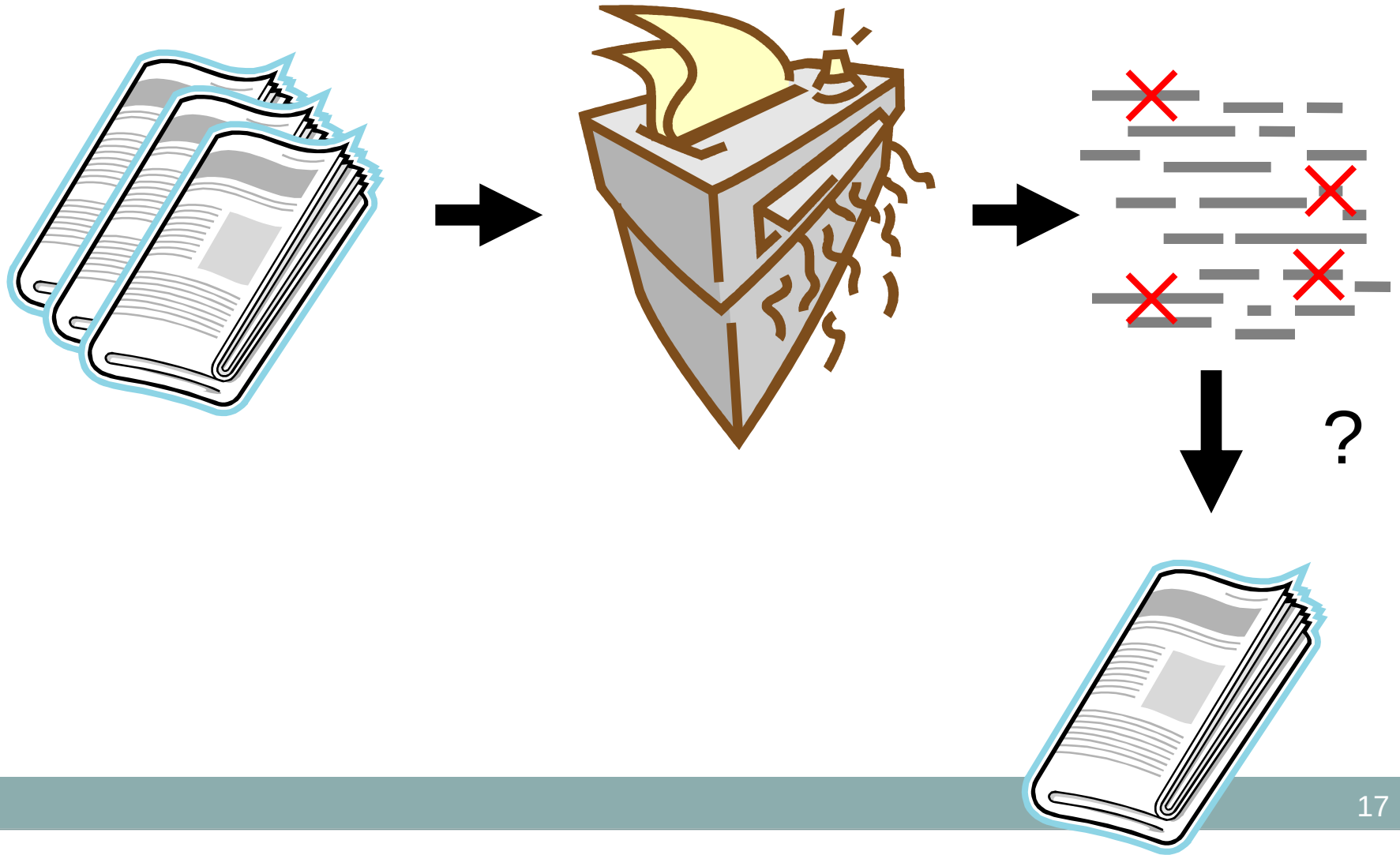
Part II

SEQUENCE 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

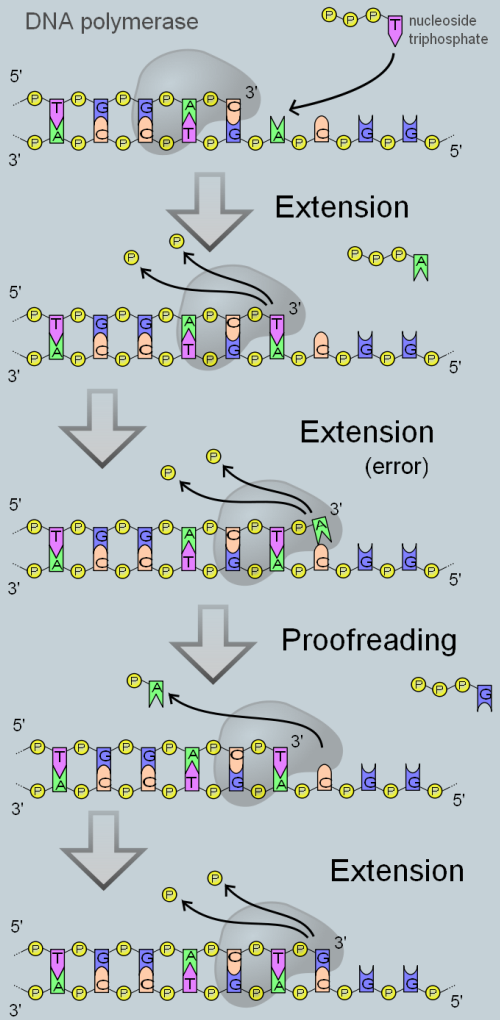
- DNA sequencing: resolving a nucleotide sequence (whole-genome or less)
- Many different methods developed
  - Maxam-Gilbert method (1977)
  - Sanger method (1977)
  - High-throughput methods

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

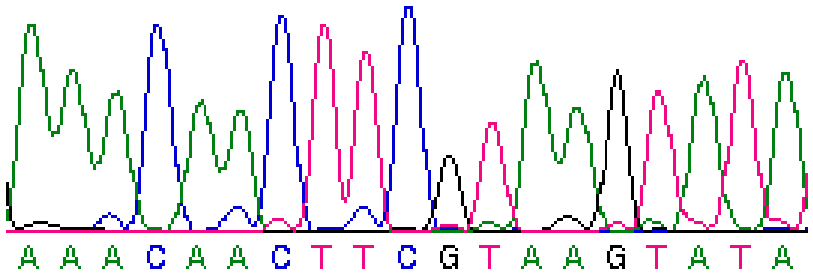
- In Sanger sequencing, chain-terminating dideoxynucleoside triphosphates (ddXTPs) are employed
  - ddATP, ddCTP, ddGTP, ddTTP lack the 3'-OH tail of dXTPs
- A mixture of dXTPs with small amount of ddXTPs is given to DNA polymerase with DNA template and primer
- ddXTPs are given fluorescent labels

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 repeats 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*
- See Algorithms for Bioinformatics course notes and exercises for studies on the shortest superstring problem (approximation algorithm, generalization to approximate case).

Shortest superstrings: issues

- NP-hard problem: unlikely to have an efficient (exact) algorithm; approximate solutions exist
- 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)

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

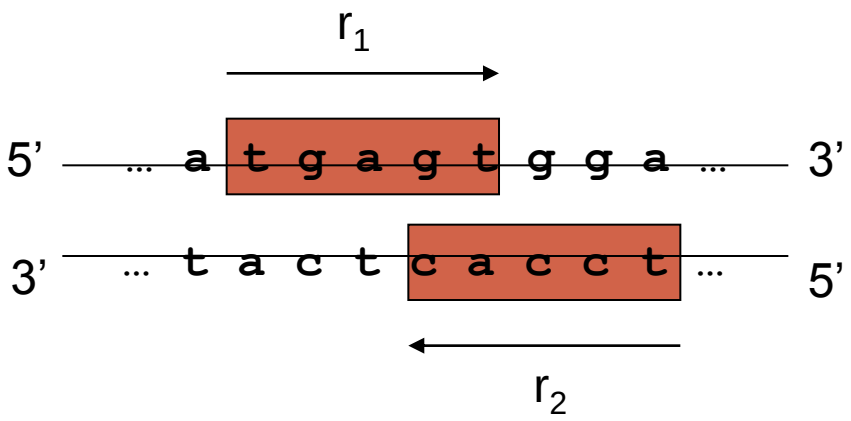
- Overlap
  - Finding potentially overlapping reads
- Layout
  - Finding the order of reads along DNA
- Consensus
  - Deriving the DNA sequence from the layout

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’ – C**AGCGCGCT**GCGTGACGAGTCTGACAAAGACGGTATGCGCATCG
TGATTGAAGTGAAACGCG**ATGCGGTC**GGTCGGTGAAGTTGTGCT - 3’

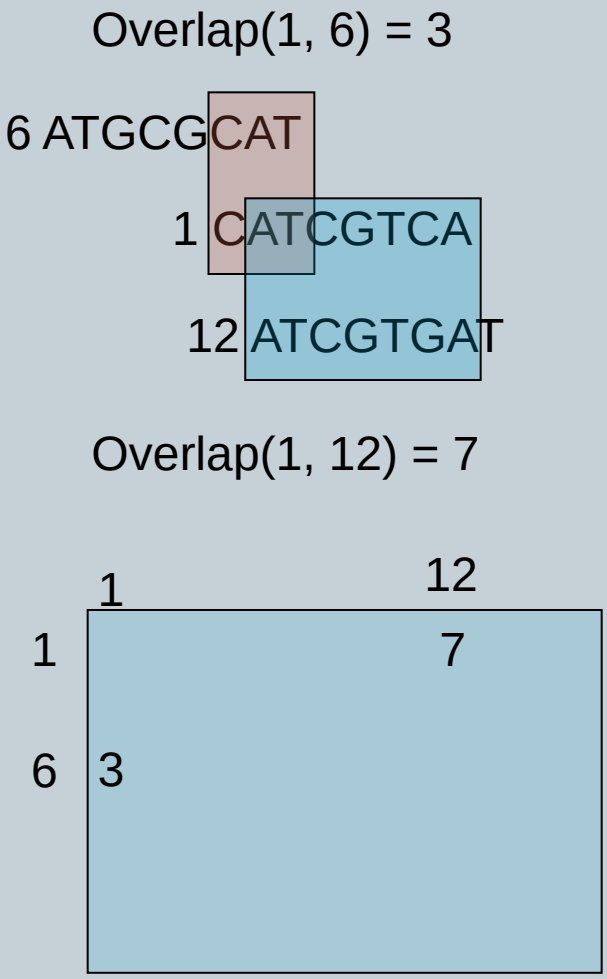
- 20 reads:

#	Read	Read*
	CATCGTCA	TCACGATG
	CGGTGAAG	CTTCACCG
	TATGCGCA	TGCGCATA
	GACGAGTC	GACTCGTC
	CTGACAAA	TTTGTCAG
	ATGCGCAT	ATGCGCAT
	ATGCGGTC	GACCGCAT
	CTGCGTGA	TCACGCAG
	GCGTGACG	CGTCACGC
	GTCGGTGA	TCACCGAC

#	Read	Read*
	GGTCGGTG	CACCGACC
	ATCGTGAT	ATCACGAT
	GCGCTGCG	CGCAGCGC
	GCATCGTG	CACGATGC
	AGCGCGCT	**AGCGCGCT**
	GAAGTTGT	ACAACTTC
	AGTGAAAC	GTTTCACT
	ACGCGATG	CATCGCGT
	GCGCATCG	CGATGCGC
	AAGTGAAA	TTTCACTT

Finding overlaps

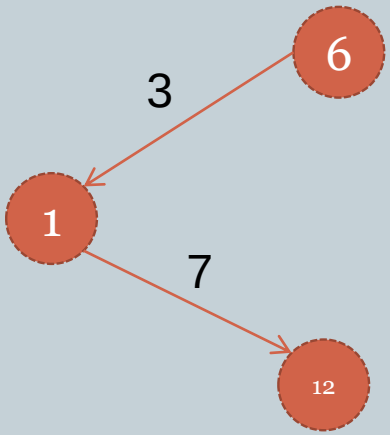
- Overlap between two reads can be found with a dynamic programming algorithm
  - Errors can be taken into account
  - See http://www.cs.helsinki.fi/u/nvalimak/opetus/afb10/ex4/prob1-4.pdf (solution for assignment 4.)
- Overlap scores stored into the overlap matrix
  - Entries (i,j) denote the *suffix* overlap of read r_i with *prefix* of r_j .
  - Each read corresponds to two rows and two columns; complements need to be considered as well.



Overlap graph

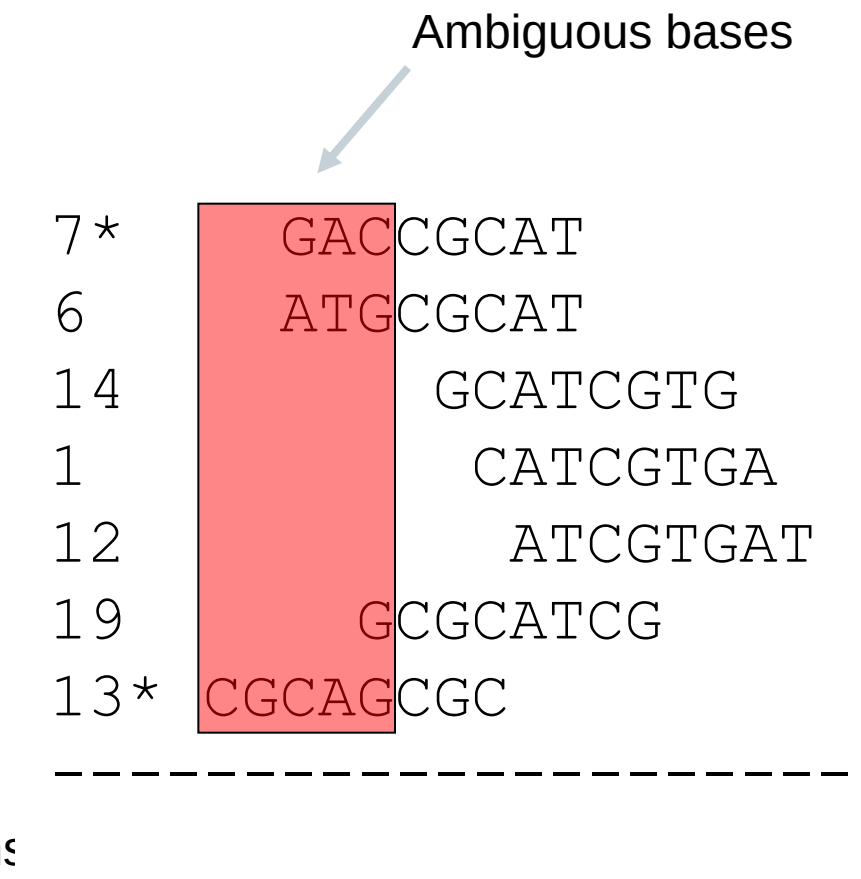
- In practice, computing pair-wise overlaps is time comsuming
  - Speed-up techniques required
    - Find a way to compute only significantly long overlaps
    - Instead of overlap matrix, one should directly construct its sparse version, *overlap graph*.
  - Exact significantly long overlaps can be computing efficiently using suffix trees, but suffix tree of all reads takes too much space for large genomes.
  - Suffix trees can be replaced by compressed data structures, like *FM-index* (see *Durbin's guest lecture on Wednesday 16-, B222*)

Four types of directed edges:
r_i overlaps r_j
r_i overlaps r_j*
r_i* overlaps r_j
r_i* overlaps r_j*



Finding layout & consensus

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

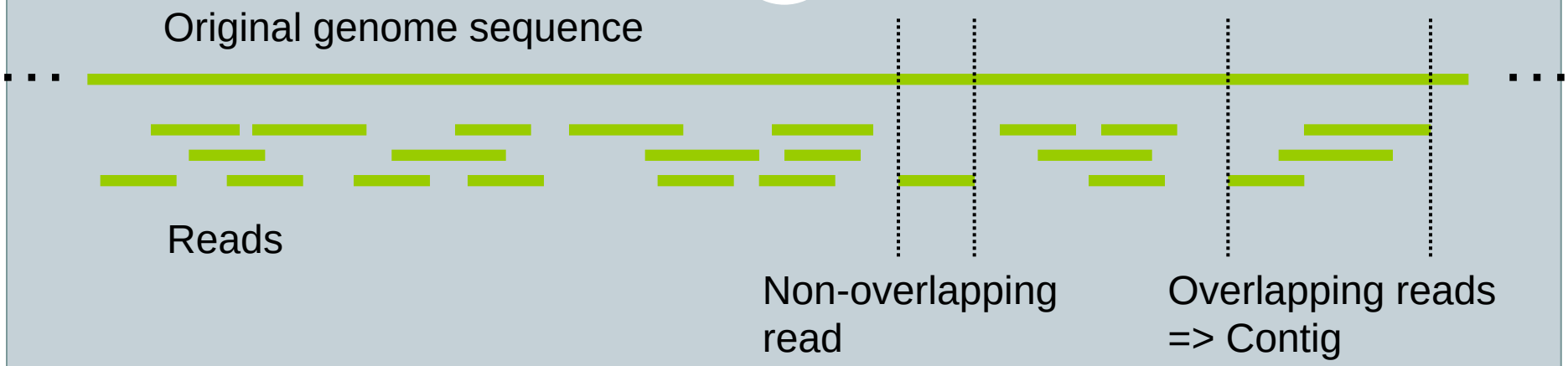


Finding layout & consensus

- 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

```
2                CGGTGAAG
10             GTCGGTGA
11           GGTCGGTG
7      ATGCGGTC
-----------------------
       ATGCGGTCGGTGAAG
```

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

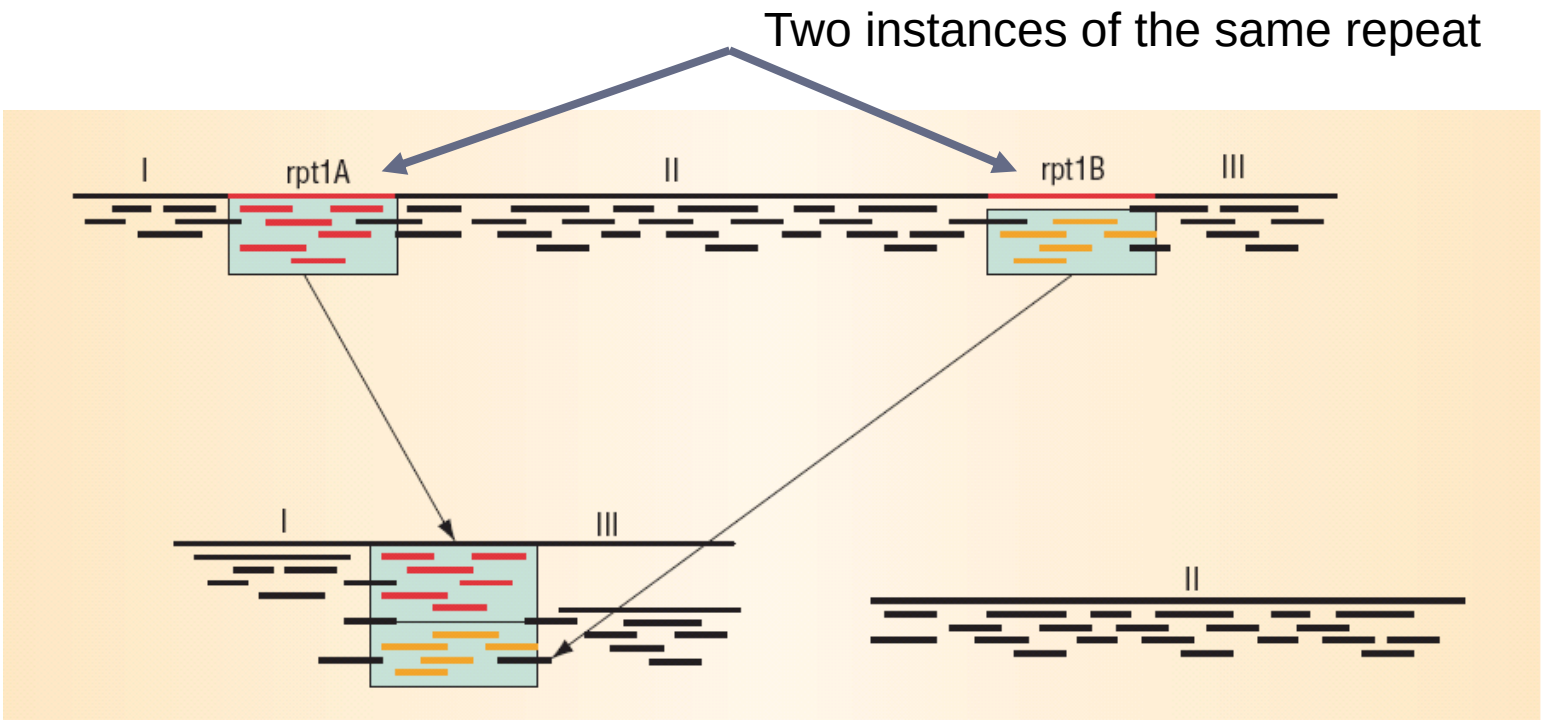
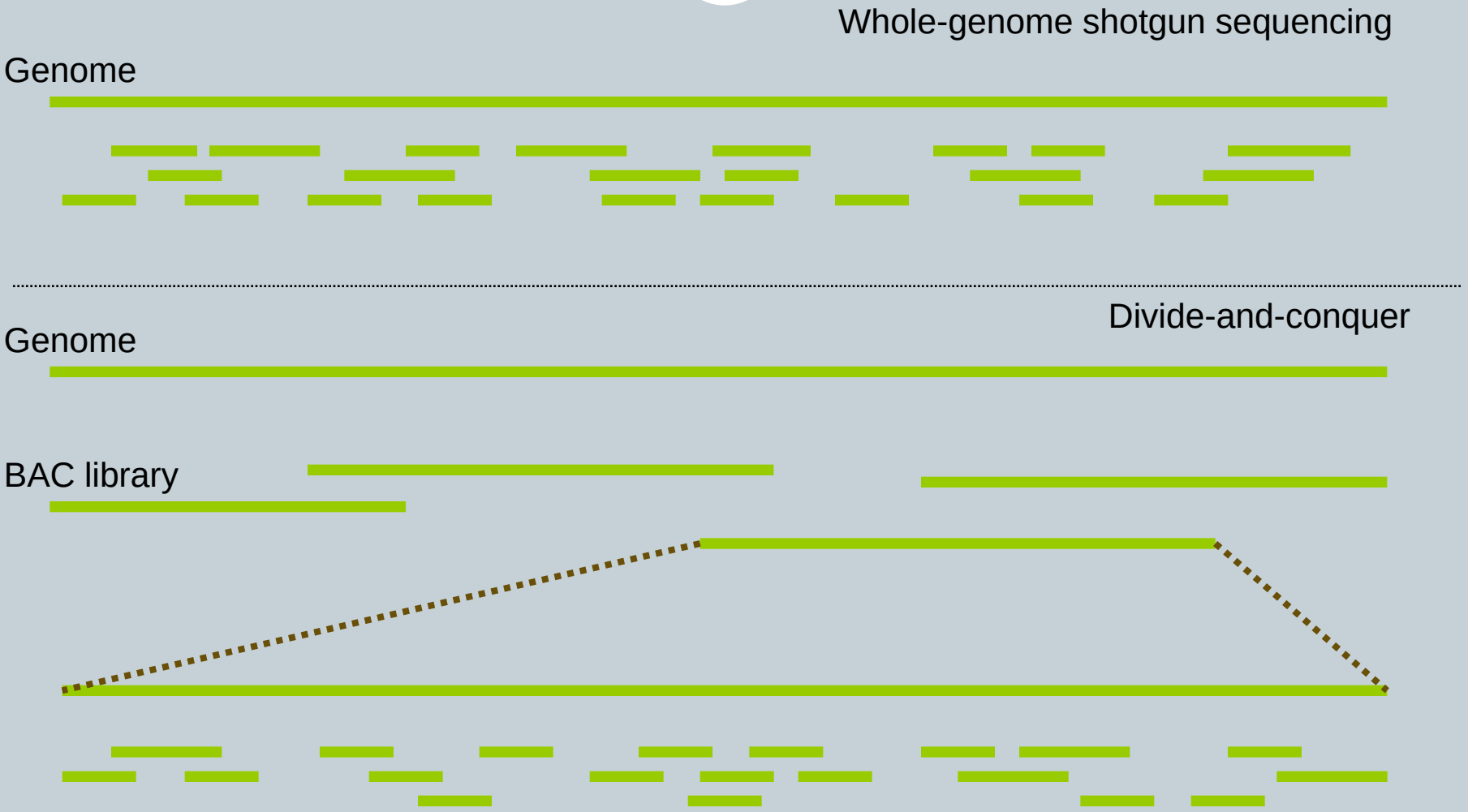


Figure 2. Repeat sequence. The top represents the correct layout of three DNA sequences. The bottom shows a repeat collapsed in a misassembly.

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

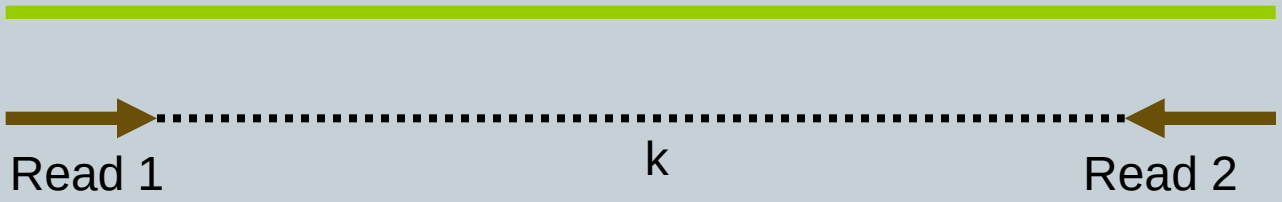
- Each BAC (Bacterial Artificial Chromosome) is about 150 kbp
- Covering the human genome requires ~30000 BACs
- BACs shotgun-sequenced separately
  - Number of repeats in each BAC is **significantly smaller** than in the whole genome…
  - …needs **much more manual work** compared to whole-genome shotgun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

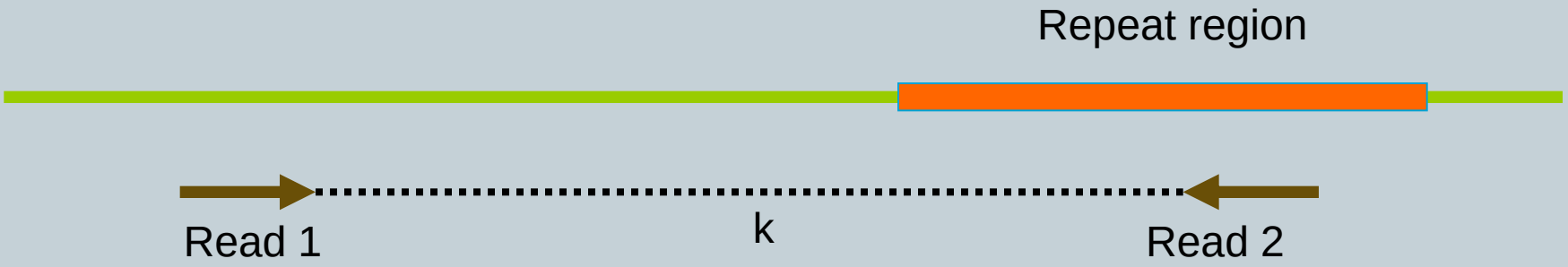
- *Paired end* (or *mate-pair*) sequencing is technique where
  - both ends of an insert are sequenced
  - For each insert, we get two reads
  - We know the distance between reads, and that they are in opposite orientation



  - Typically read length < insert length

Paired end sequencing

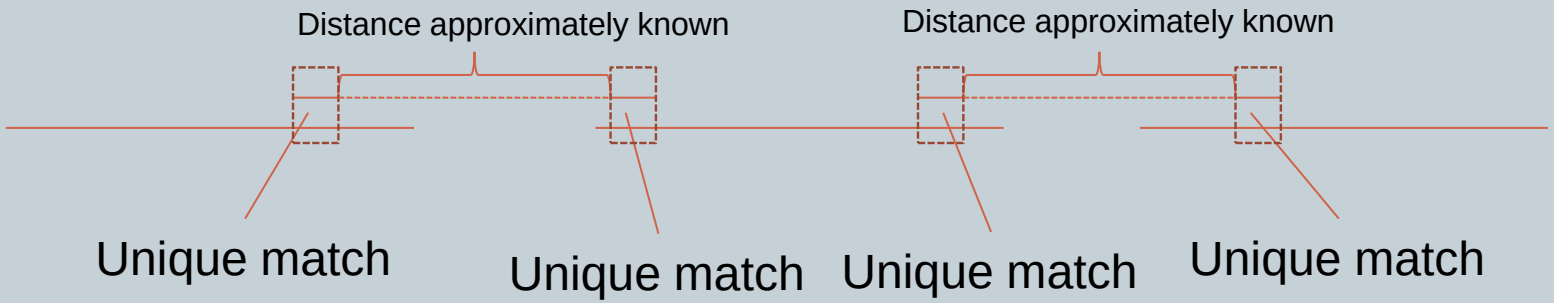
- The key idea of paired end sequencing:
  - **Both reads** from an insert are unlikely to be in repeat regions
  - If we know where the first read is, we know also second's location



- This technique helps to WGSS higher organisms

Scaffolding

- Paired end reads help to order the contigs into *scaffolds*.



- Even non-unique matches can be exploited
  - Each contig pair receives votes from paired end reads that suggest ordering them to distance [min,max] apart.
  - Find a global contig ordering that maximizes satisfied votes.
    - Not an easy optimization problem.

Alternative approach: Virtual sequencing by hybridization

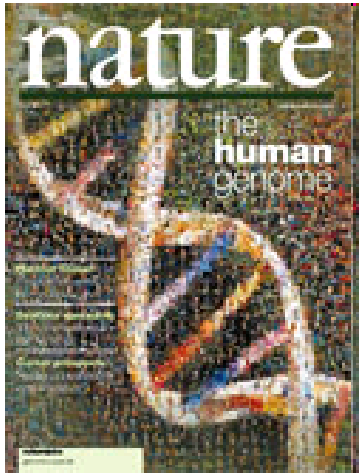
- Consider all k-mers of all reads.
- Create a graph with each (k-1)-mer as a node and k-mers as edges:
  - There is an edge between nodes $X=x_1x_2...x_{k-1}$ and $Y=y_1y_2...y_{k-1}$ if and only if $x_2...x_{k-1}=y_1...y_{k-2}$ and $x_1...x_{k-1}y_{k-1}$ is a k-mer inside at least one read.
  - Subgraph of *de Bruijn* graph.
- If coverage would be identical through the genome and there would be no errors in the reads, *Eulerian path* on the graph would give the solution (see Algorithms for Bioinformatics course).
- Some assemblers try to correct this graph in order to use the Eulerian path approach.

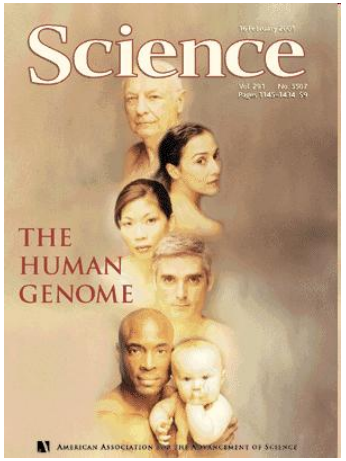
First whole-genome shotgun sequencing project: Drosophila melanogaster

- 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

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
- EULER
- Velvet
- Newbler
- SOAPdenovo
- …

Next generation sequencing techniques

- Sanger sequencing is the prominent first-generation sequencing method
- Many new sequencing methods have emerged
  - 454 (~400 bp reads)
  - Illumina Solexa (35-150 bp reads)
  - SOLiD (~50 bp reads, colour codes)
  - Helicos (~55 bp reads from single molecule!)
  - Pacific Biosciences ("**thousands of nucleotides**", TBA 201?, third-generation sequencer)
- See Lars Paulin's lecture on Thursday