

# Sequencing and Sequence Assembly

---

Jiří Kléma

Department of Computer Science,  
Czech Technical University in Prague

Lecture based on Mark Craven's class at University of Wisconsin



<http://cw.felk.cvut.cz/wiki/courses/b4m36bin/start>

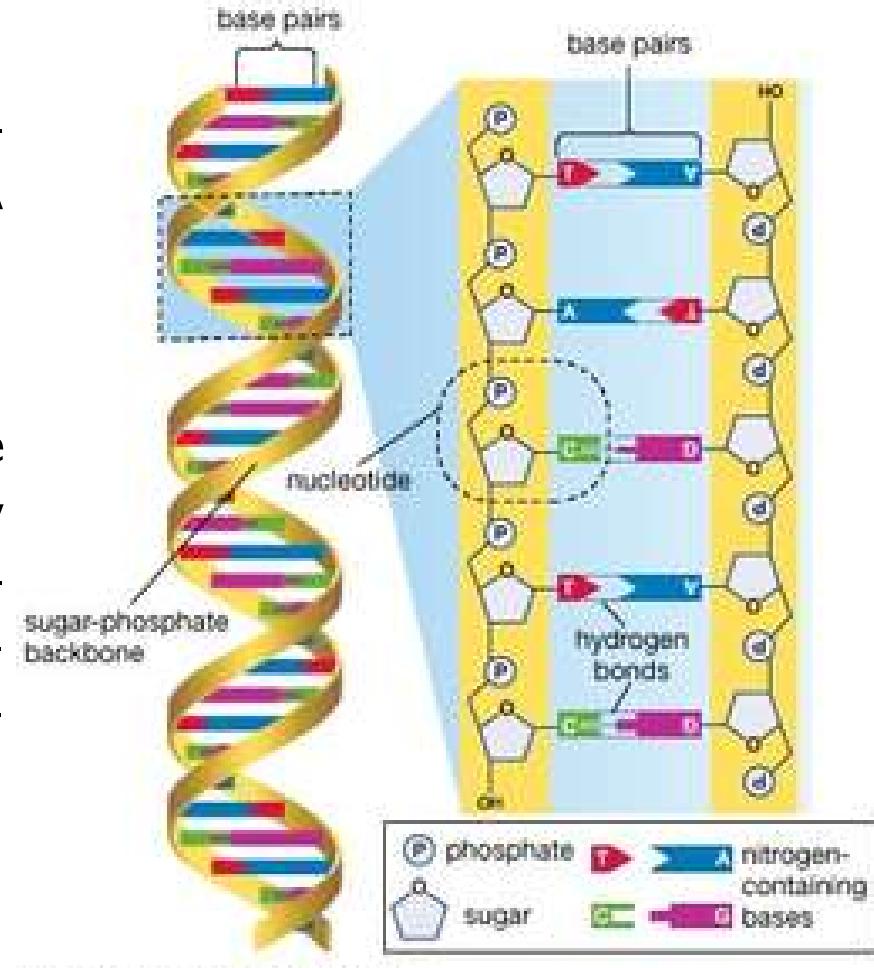
# Overview

---

- DNA sequencing
  - before – slow and expensive,
  - next-generation sequencing (NGS) – massively parallel, faster and cheaper,
- sequence assembly
  - we cannot read off the sequence of an entire molecule all at once,
  - the sequence has to be **assembled** from shorter reads,
  - stems from redundancy of the read set = overlaps between reads,
- assembly methods
  - greedy methods,
  - graph-based method
    - \* the de Bruijn graph method most popular in the age of NGS.

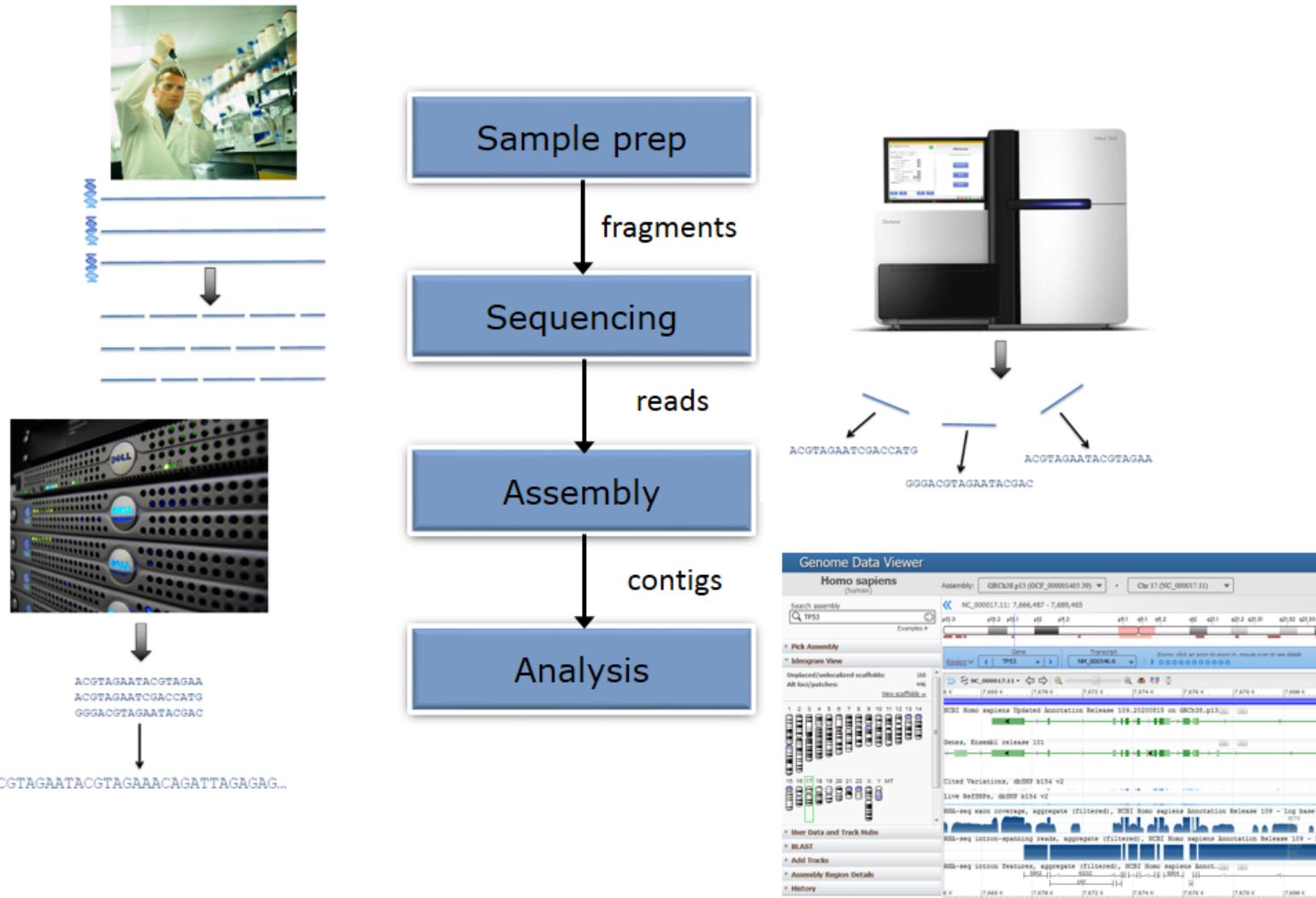
# DNA sequencing

- The basic objective
  - to determine the order of the nucleobases (A, C, G, T) in the DNA molecule,
- the subsequent analytical objectives
  - recognition of DNA structure (genes, introns, exons, regulatory regions, RNA genes), study of differences between species, organisms and individuals, understanding of function,
- sequencing methods
  - de novo vs resequencing.



© 2007 Encyclopædia Britannica, Inc.

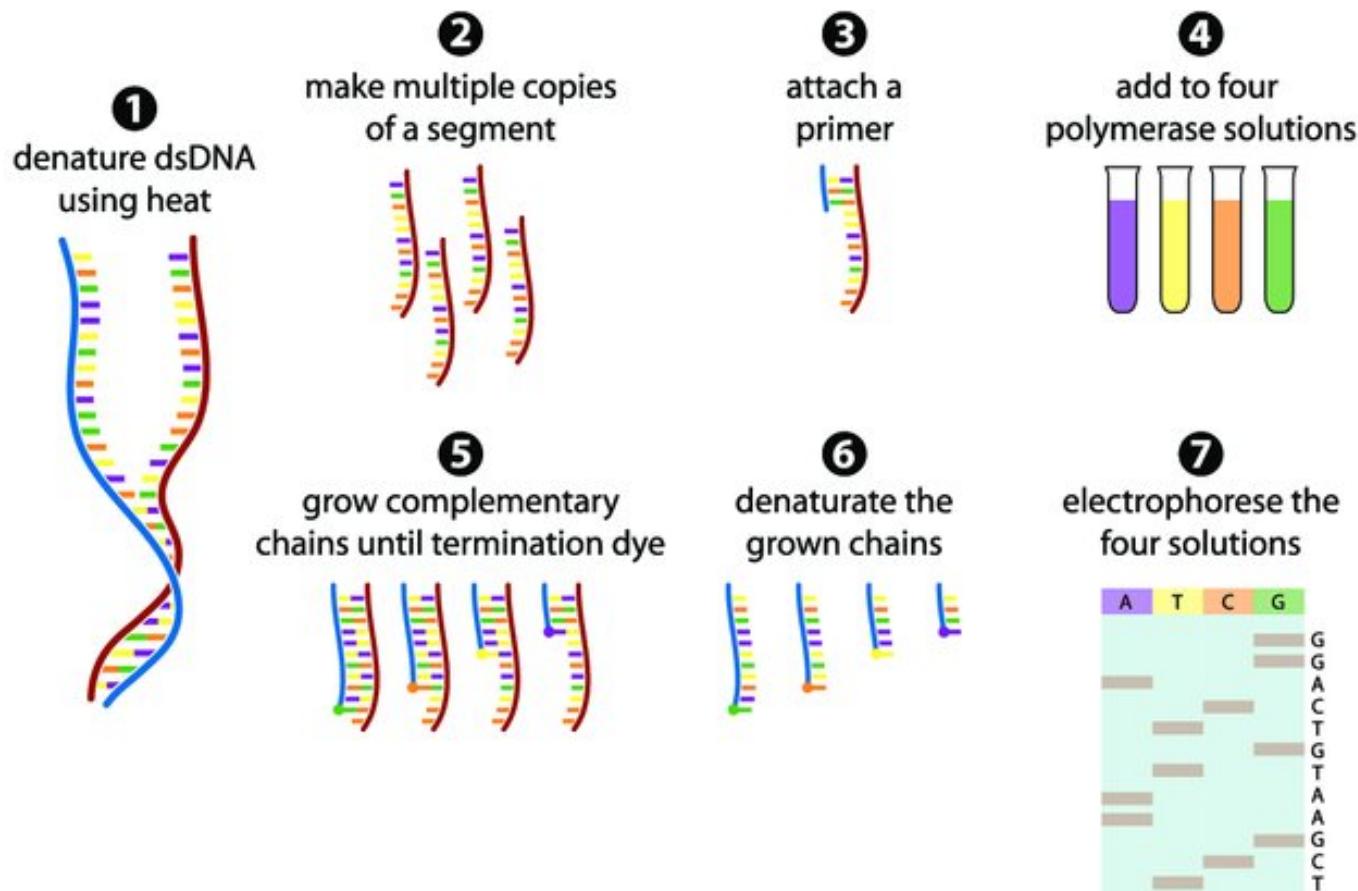
# DNA sequencing – the basic steps related with it



Bioinformatics Algorithms, Computer Science Department, Colorado State University, CS548.

# DNA sequencing before

- Sanger sequencing
  - 1977, still used for the Human Genome project until 2004.

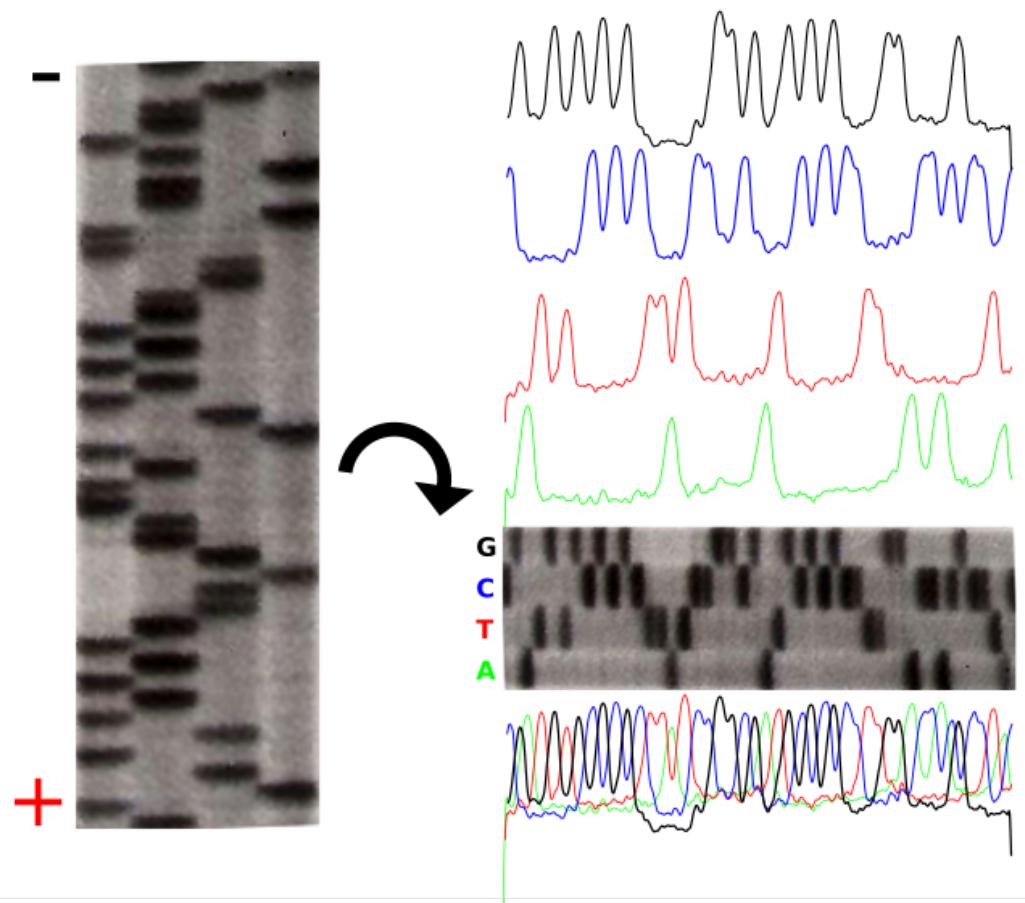


M. Gauthier: Simulation of polymer translocation through small channels, 2008.

# DNA sequencing before

---

- Sanger sequencing
  - the last step, electrophoresis.



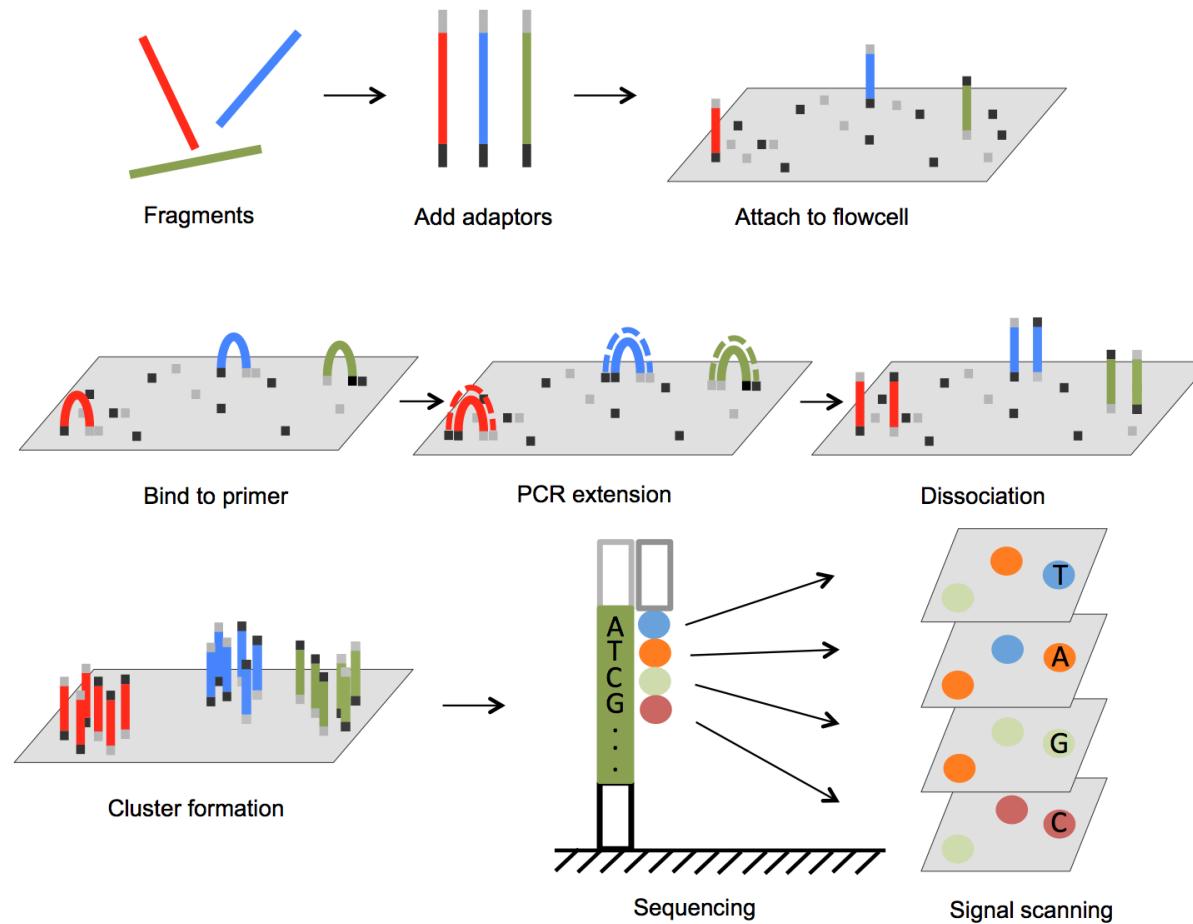
# Statistics for shotgun sequencing

---

- Shotgun sequencing
  - a general strategy to subdivide a long sequence into **random** fragments,
- Given:  $G$  – genome length ( $3 \times 10^9$  nts),  $L$  – read length (500 nts),  $N$  – number of reads (tbd)
- Calculate: coverage –  $a = NL/G$
- Questions tbd by stats (Lander-Waterman):
  - How many contigs are there?
  - How big are the contigs?
  - How many reads are in each contig?
  - How big are the gaps?
- Requirement: 99% in contigs, 1% in gaps
  - $a = 4.6$ ,  $N = 3 \times 10^7$ , mean contig length  $10^4$ ,
  - 100 reads/contig on average.

# DNA sequencing today

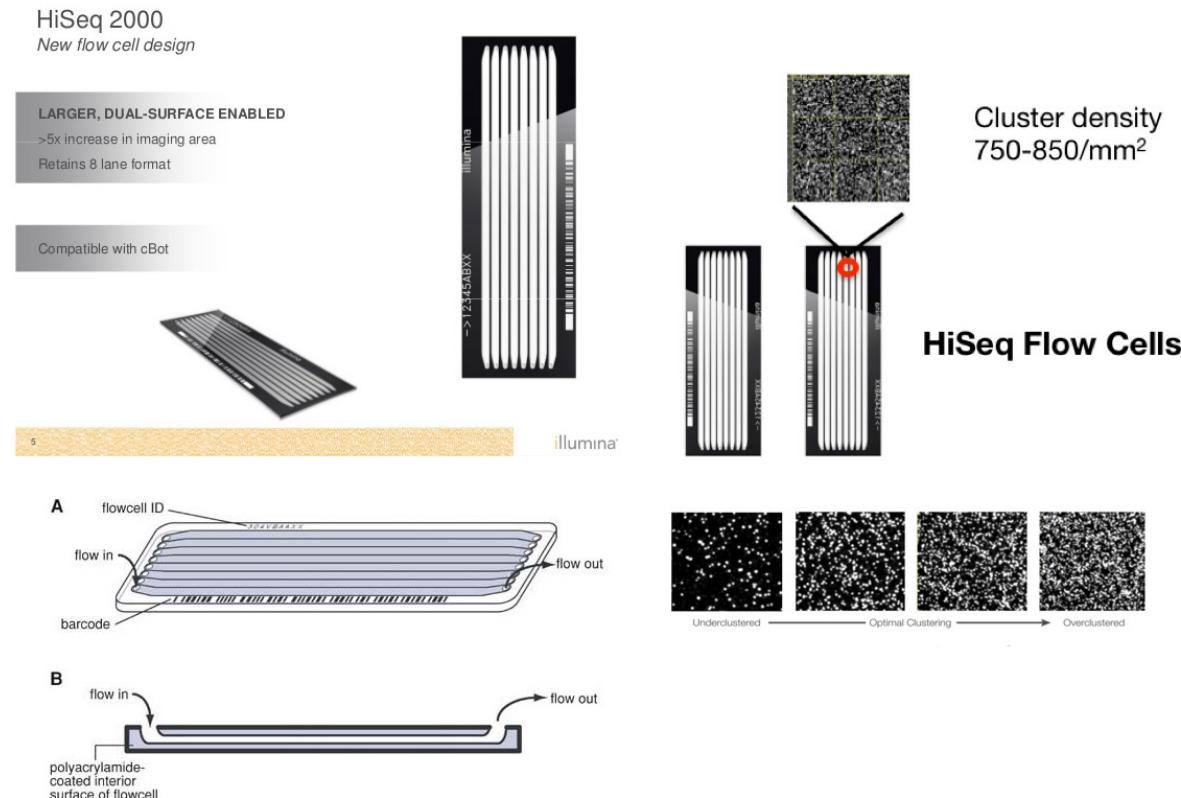
- Next/Second Generation sequencing
  - the main progress in massive parallelization (**high-throughput**).



Lu et al.: Next Generation Sequencing in Aquatic Models, 2016.

# DNA sequencing today

- Next/Second Generation sequencing
  - technical equipment.



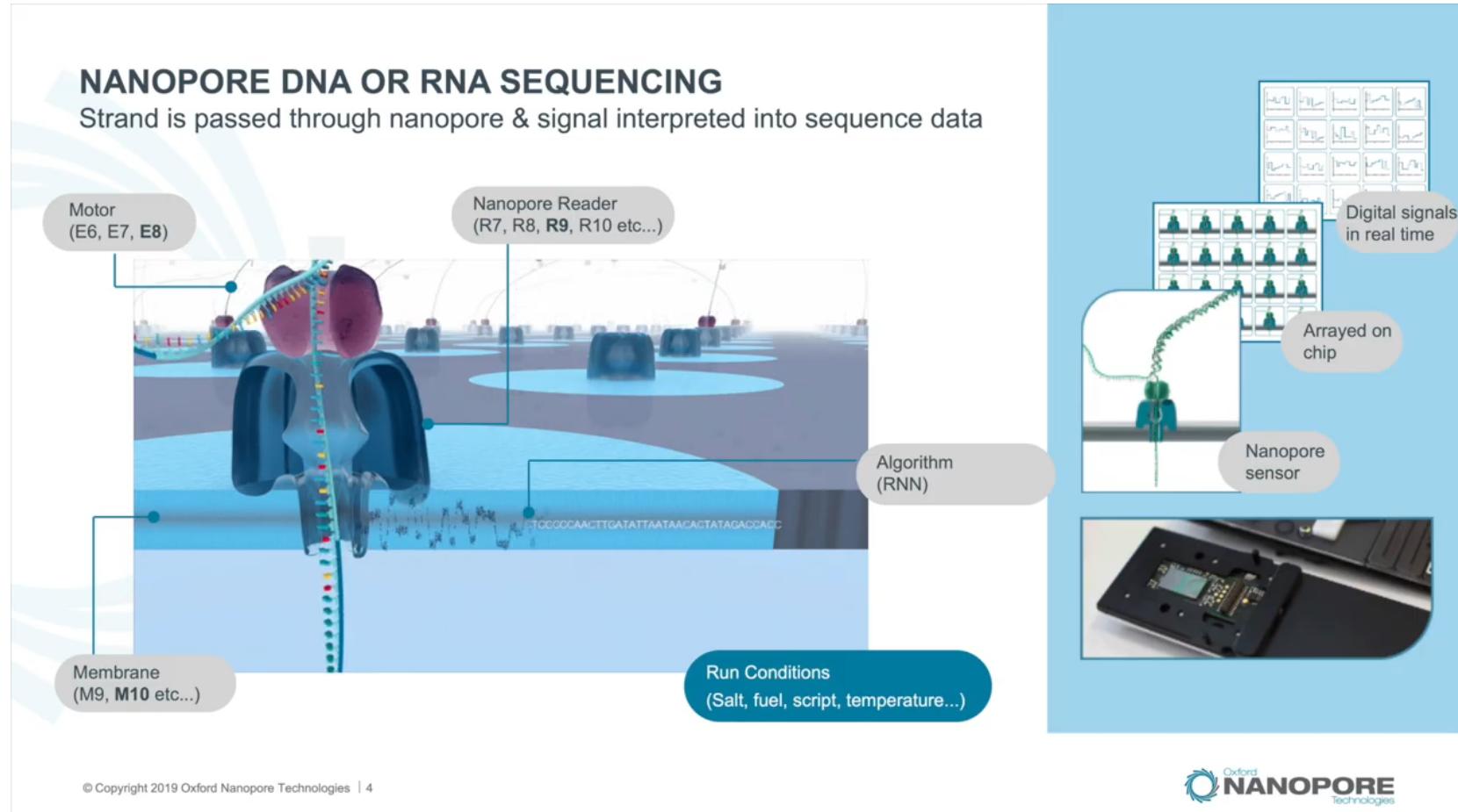
Illumina uses a glass 'flowcell', about the size of a microscope slide, with 8 separate 'lanes'.

The HiSeq instrument scans both upper and lower surfaces of each flowcell lane.

HiSeq2000 - Next Level Hacking.

# Modern methods under development

- Nanopore sequencing (the main idea from 2012, still evolving).



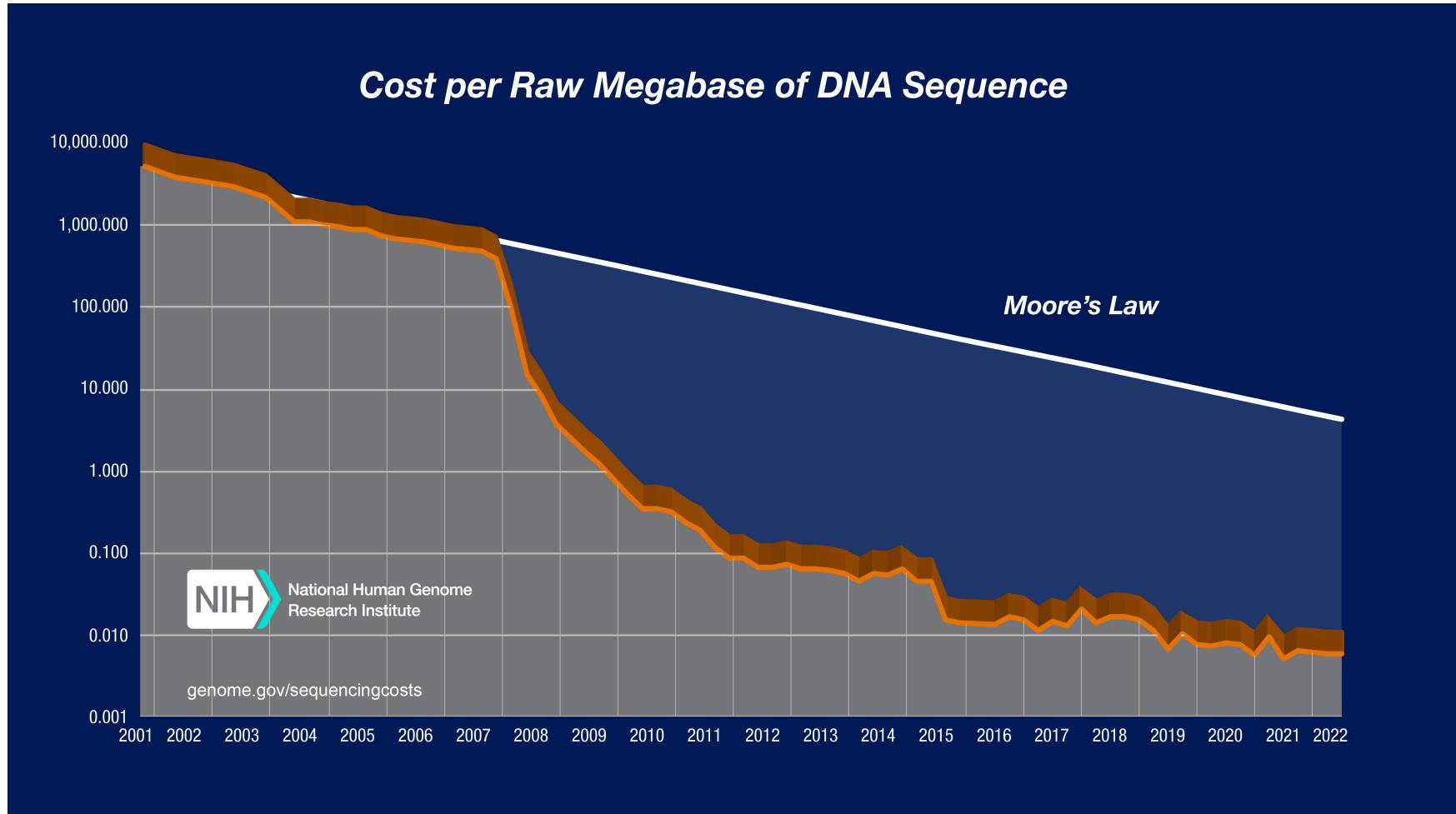
Oxford Nanopore video.

# Comparison of sequencing approaches

---

- The key parameters
  - length of read in base pairs (bps), error rate in %, price in \$ per million bps, reading speed in bps per day,
- Sanger sequencing
  - 500-800 bp, 1%, \$2400, ~1 Mbp/day,
  - very slow and very expensive,
- next generation technology
  - 454 Genome Sequencer: 250-600 bp, 1%, \$10, ~1 Gbp/day,
  - Illumina Genome Analyzer: 35-150 bp, 1%, \$0.15, ~100 Gbp/day,
  - breakthrough in mass usability, places demands on the assembly of DNA sequences,
- third generation sequencing
  - Oxford Nanopore: x10 kbp, initially up to 20%, since 2022 less than 5%,
  - a small portable sequencer with low acquisition costs.

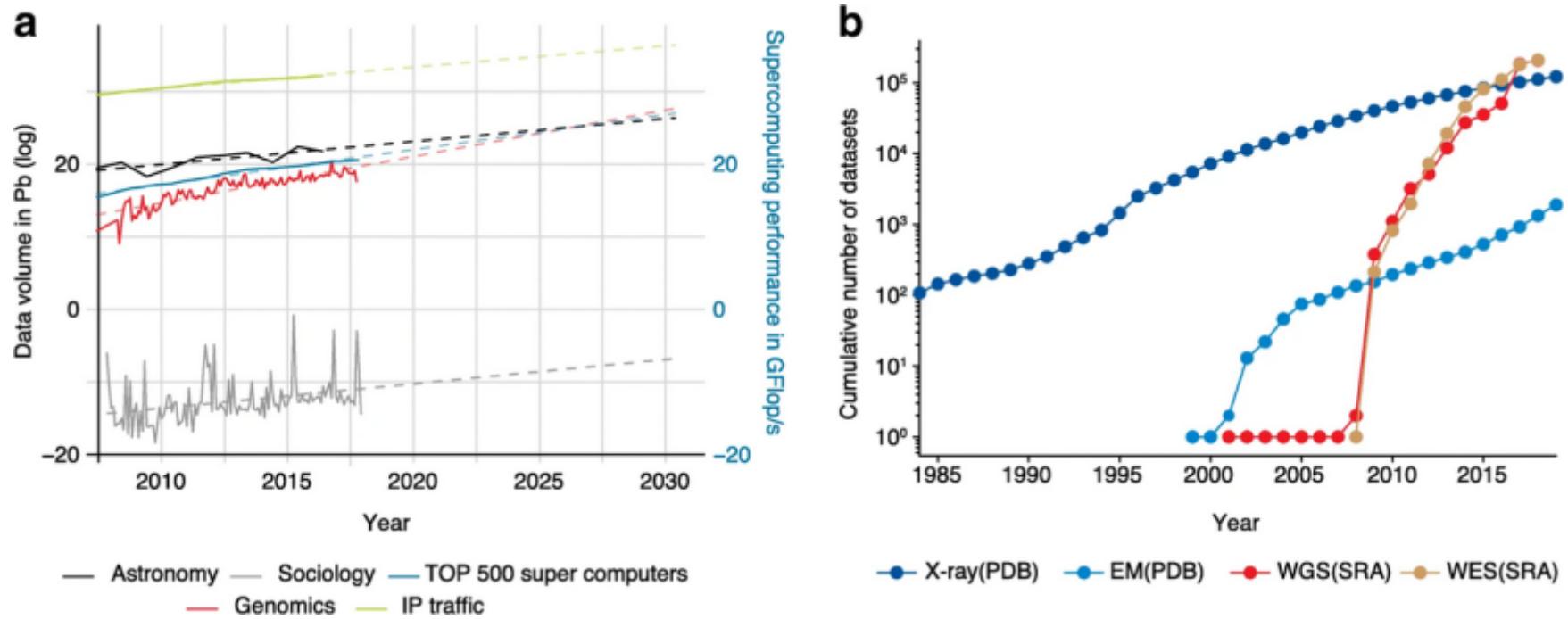
# DNA sequencing costs and volumes



National Human Genome Research Institute, 2022.

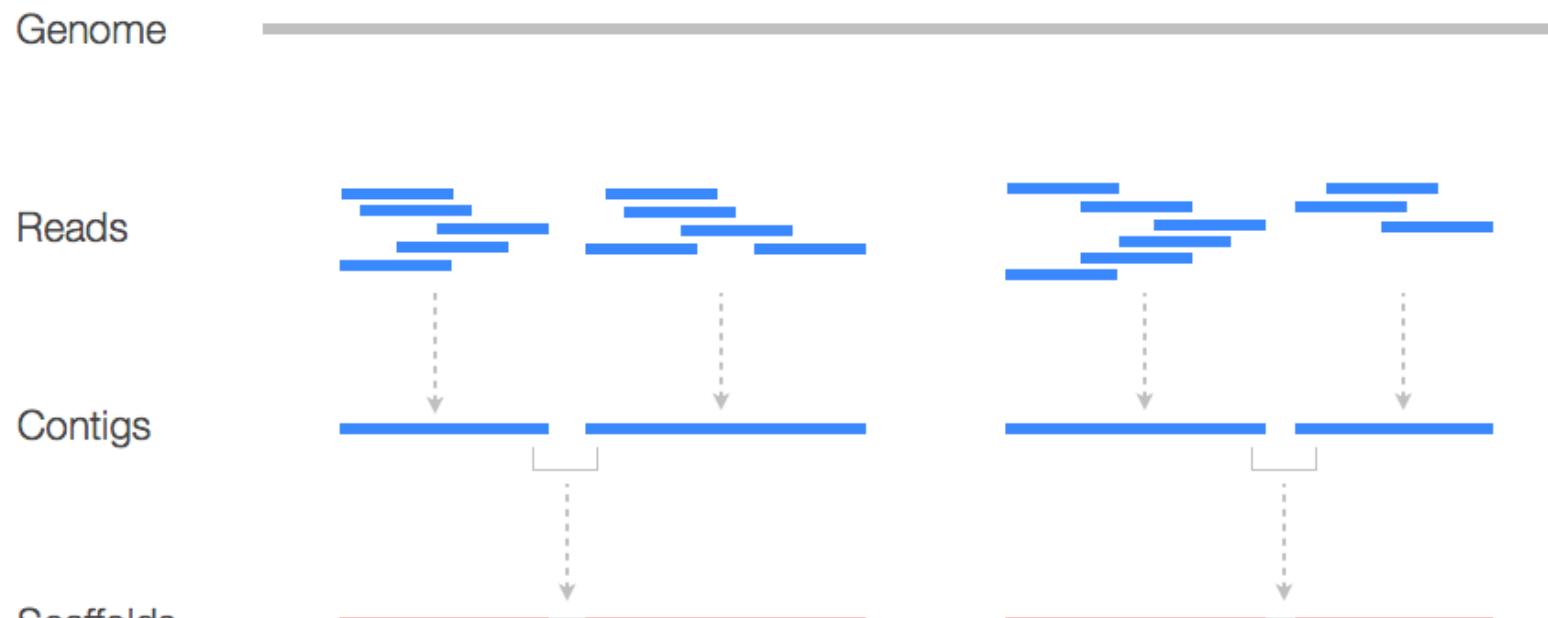
# DNA sequencing costs and volumes

- Genomics could become the largest data source already in this decade!
- Sequencing corresponds to WGS (whole genome sequencing) and WES (whole exome sequencing).



# Sequence assembly

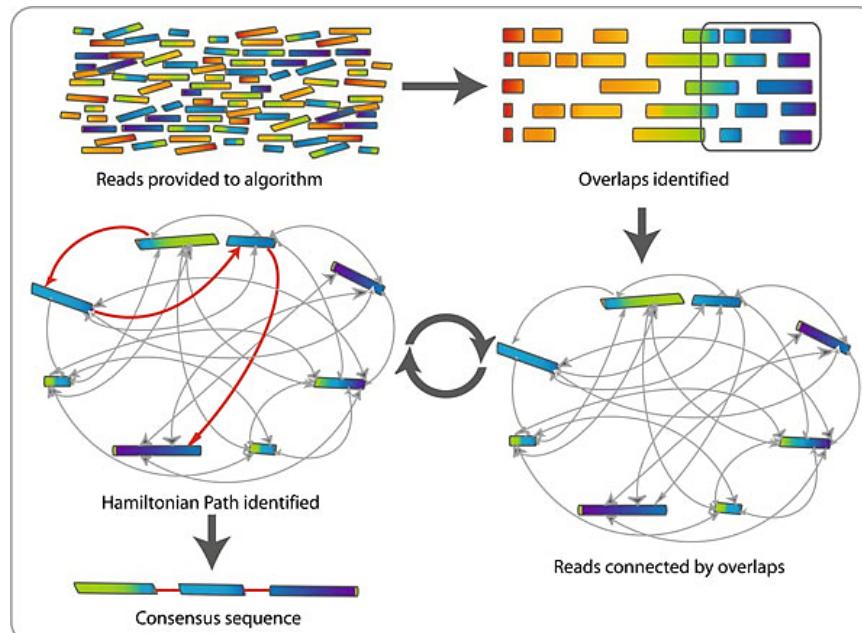
- assembles sequences whose length is close to the original sequence
  - contig = a set of concordant overlapping reads, a contiguous DNA stretch,
  - scaffold = links contiguous sections of DNA separated by gaps, the direction and length of the gaps are clear.



Neumann: Genome Assembly Tutorial.

# Sequence assembly

- you can simply create the shortest superstring for an existing read set
  - ideally assumes error-free reading and that identical reads come from the same position in the genome,
  - assumptions not met (read error rate, repeats), yet NP-hard problem,
  - can be solved hungrily, or using graph theory (see **overlap graphs** below).



Commins et al., Biological Procedures Online, 2009.

# Shortest superstring problem

---

- Objective: find a string  $s$  such that
  - all reads  $s_1, s_2, \dots, s_n$  are substrings of  $s$ ,
  - $s$  is as short as possible,
- assumptions:
  - “best” = “simplest”,
  - reads are 100% accurate,
  - identical reads must come from the same location on the genome,
- example:
  - given the reads:  $\{\text{ACG, CGA, CGC, CGT, GAC, GCG, GTA, TCG}\}$ ,
  - the shortest superstring is **TCGACGCGTA** (length 10).

# Algorithms for shortest superstring problem

---

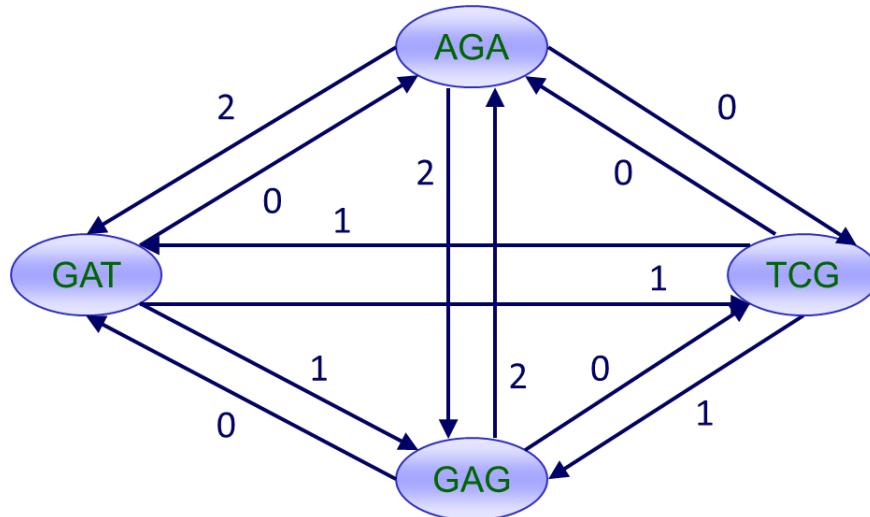
- This problem turns out to be NP-hard
- simple greedy strategy
  - uses a locally optimal problem-solving heuristic,
  - two strings are overlapping if prefix of one string is same suffix of other string or vice versa,

```
while # strings > 1 do
    merge two strings with maximum overlap
loop
```

- conjectured to give string with length  $\leq 2 \times$  minimum length,
- other approaches are based on graph theory . . . globally optimal solutions.

# Overlap graph

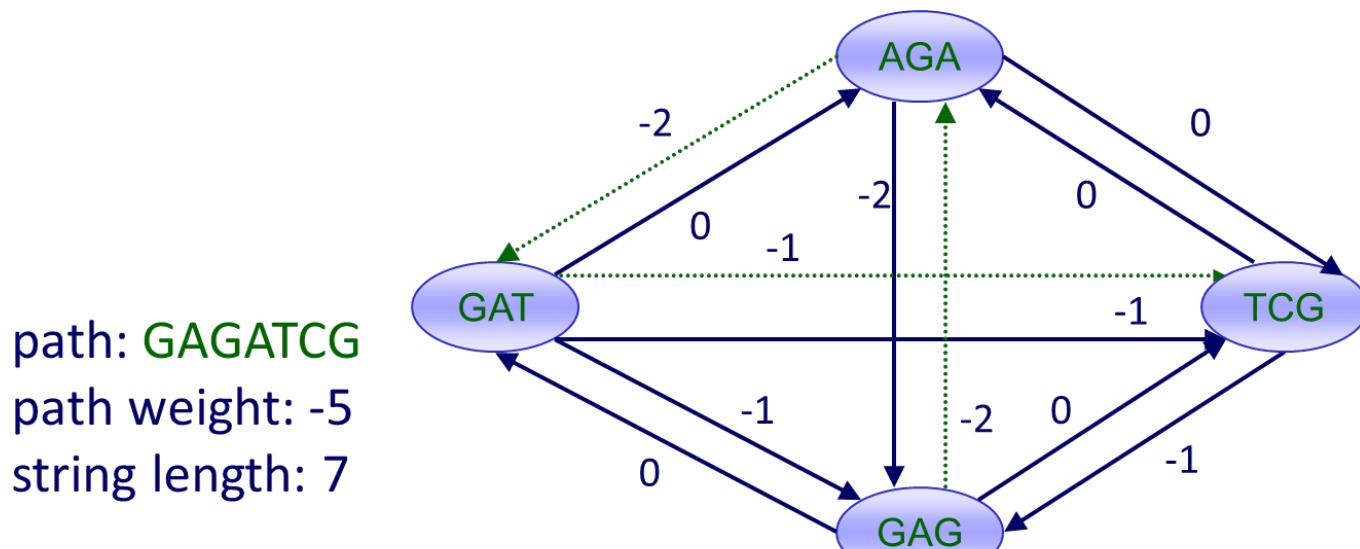
- For a set of reads  $S$ , construct a directed weighted graph  $G = (V, E, w)$ 
  - with one vertex per read ( $v_i \in V$  corresponds to  $s_i \in S$ ),
  - edges between all vertices (a complete graph),
  - $w(v_i, v_j) = \text{overlap}(s_i, s_j)$ ,
  - $\text{overlap}(s_i, s_j) = \text{length of longest suffix of } s_i \text{ that is a prefix of } s_j$ ,
- overlap graph example: let  $S = \{\text{AGA, GAT, TCG, GAG}\}$ .



Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

# Assembly as finding a Hamiltonian path

- Hamiltonian path: path through graph that visits each vertex exactly once,
- minimize superstring length
  - minimize weight of Hamiltonian path in overlap graph with edge weights negated,
  - this is essentially the Traveling Salesman Problem (also NP-complete),



Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

# Assembly as finding a Hamiltonian path

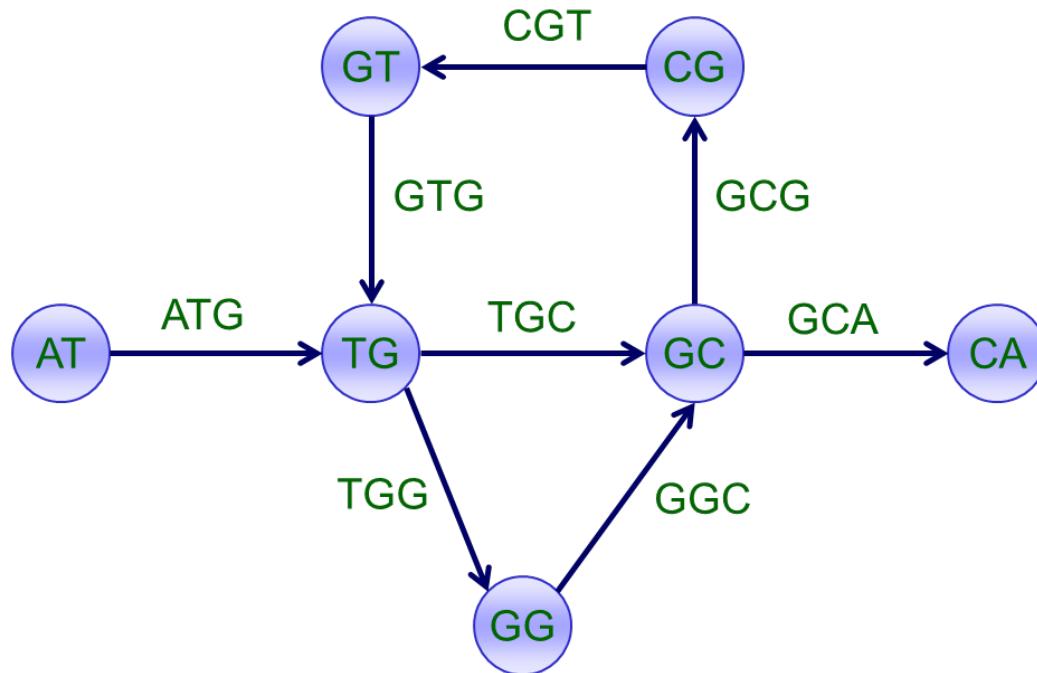
---

- Finding Hamiltonian path is an NP-complete problem,
- nevertheless overlap graphs are often used for sequence assembly
  - can detect repeats,
  - heuristical hierarchical decomposition
    - \* unitigs (no forks, no conflicts) solved first,
  - mate-pairs to scaffold.

# de Bruijn graph

---

- $\text{spectrum}(s, k) = \text{set of all } k\text{-mers (substrings of length } k\text{) from a string } s,$
- in a de Bruijn graph
  - edges =  $k$ -mers that occur in  $\text{spectrum}(s, k)$ , vertices =  $(k-1)$ -mers,
- example:  $\text{spectrum} = \{\text{ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT}\}.$



Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

# de Bruijn graph

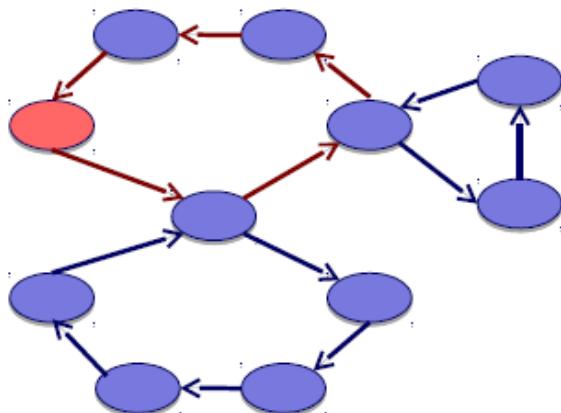
---

- Can we find a DNA sequence containing all k-mers?
  - in a de Bruijn graph, can we find a path that visits every edge of the graph exactly once?
- the theory of Eulerian graphs
  - cycle: a path in a graph that starts/ends on the same vertex,
  - Eulerian cycle: a path that visits every edge of the graph exactly once,
  - theorem: a connected graph has an Eulerian cycle if and only if each of its vertices are balanced,
  - a vertex  $v$  is balanced if  $\text{indegree}(v) = \text{outdegree}(v)$ ,
  - there is a **linear-time** algorithm for finding Eulerian cycles!
- We have reads, not k-mers . . .
  - reads are immediately split into shorter k-mers,
  - certain information loss (read coherence, overlaps).

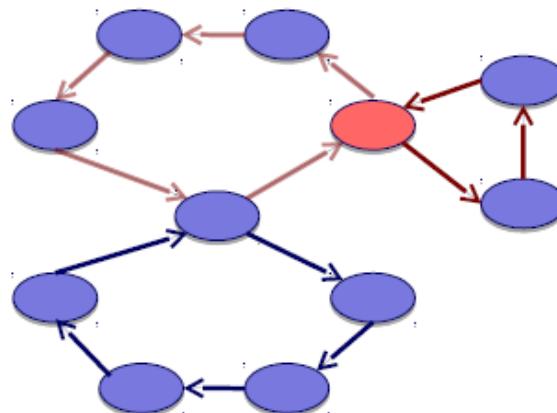
# Eulerian cycle algorithm

- Start at any vertex  $v$ , traverse unused edges until returning to  $v$ ,
  - while the cycle  $c$  is not Eulerian
    - pick a vertex  $w$  along  $c$  for which there are untraversed outgoing edges,
    - traverse unused edges until ending up back at  $w$ ,
    - join two cycles into one cycle  $c$ .

1) start at arbitrary vertex



2) start at vertex along cycle with untraversed edges

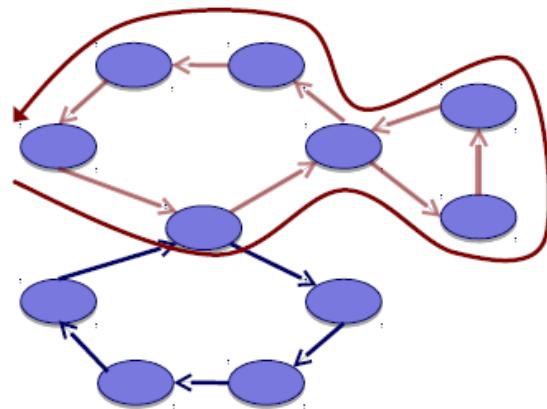


Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

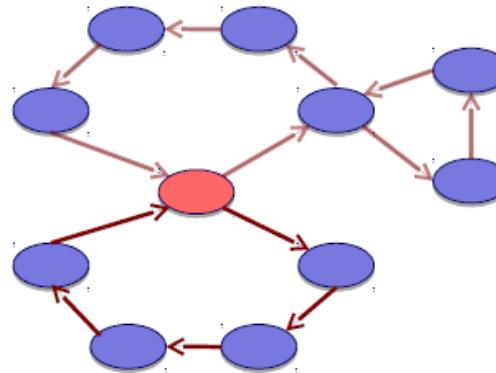
# Eulerian cycle algorithm

---

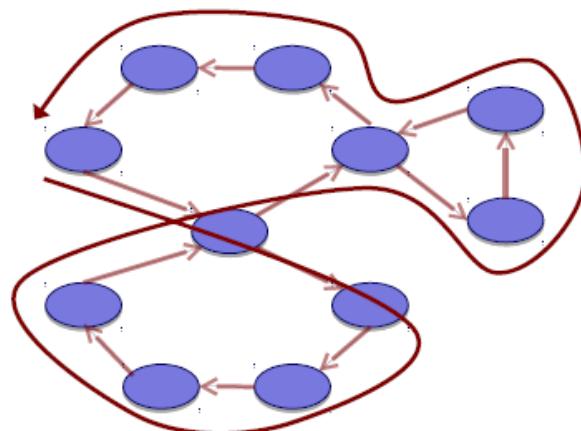
3) join cycles



4) start at vertex along cycle with untraversed edges



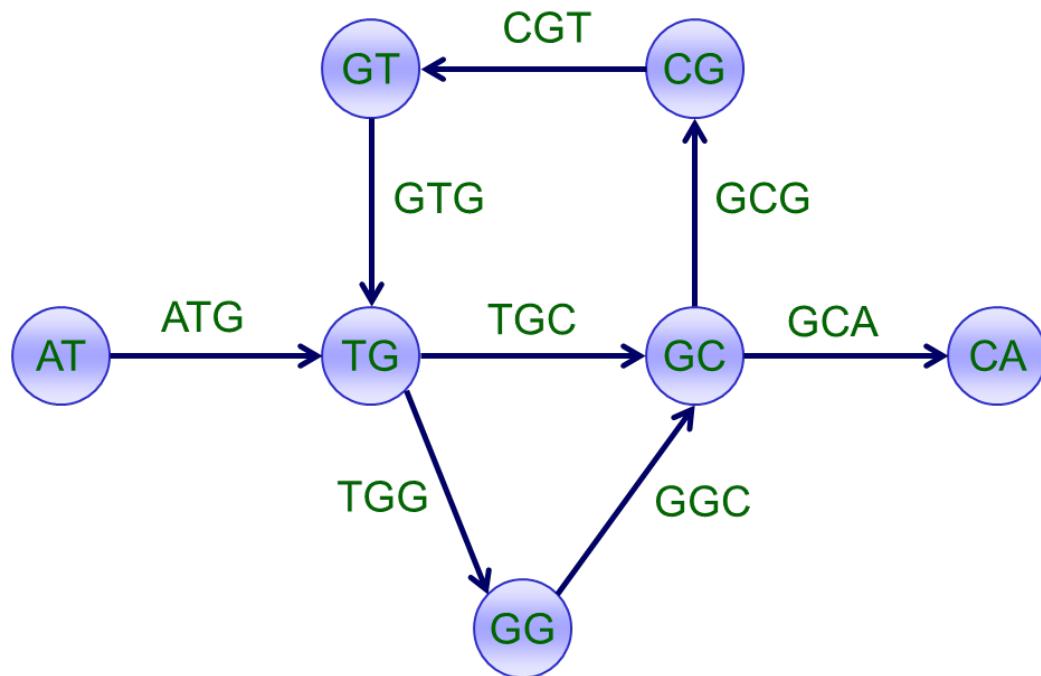
5) join cycles



Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

# Assembly as finding Eulerian paths

- Eulerian path: path that visits every edge exactly once (actually, a trail),
- the assembly problem = finding Eulerian paths in a de Bruijn graph,
- resulting sequences contain all k-mers.
- example assembly: **ATGGCGTGCA** or **ATGCGTGGCA**.

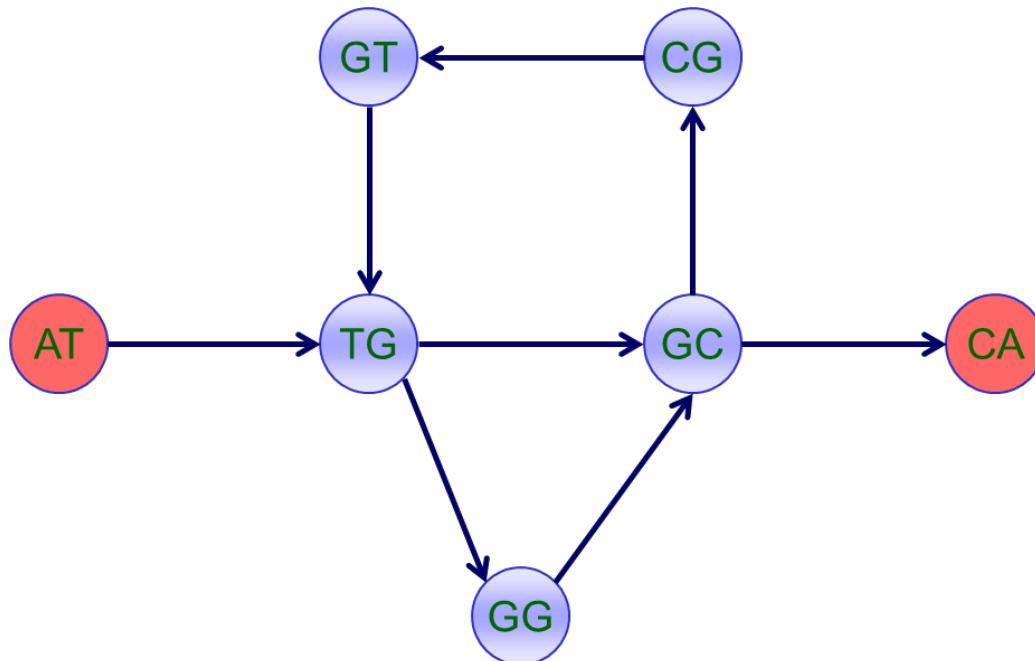


Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

## Eulerian paths

---

- a vertex  $v$  is semibalanced if  $|\text{indegree}(v) - \text{outdegree}(v)| = 1$ ,
- a connected graph has an Eulerian path if and only if it contains at most two semibalanced vertices.

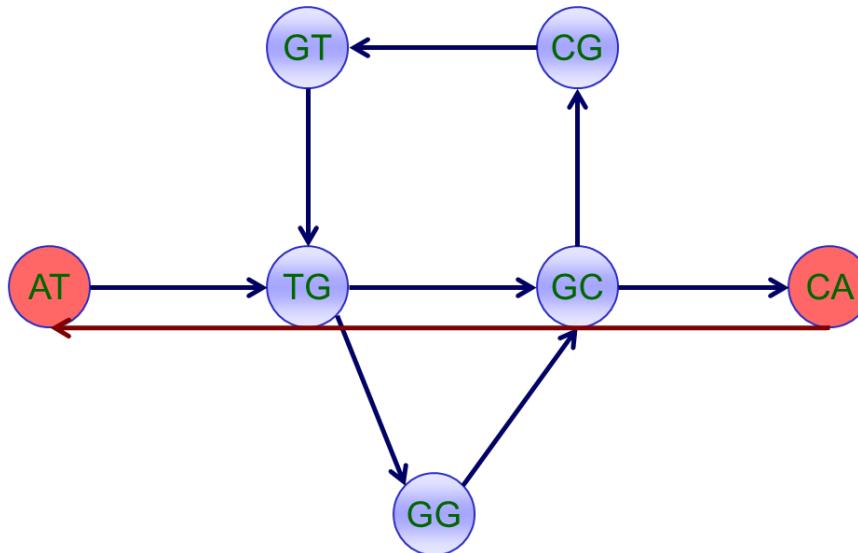


Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

## Eulerian path → Eulerian cycle

---

- If a graph has an Eulerian Path starting at  $w$  and ending at  $x$  then
  - all vertices must be balanced, except for  $w$  and  $x$  which may have  $|\text{indegree}(v) - \text{outdegree}(v)| = 1$ ,
  - if  $w$  and  $x$  are not balanced, add an edge between them to balance,
  - graph now has an Eulerian cycle which can be converted to an Eulerian path by removal of the added edge.



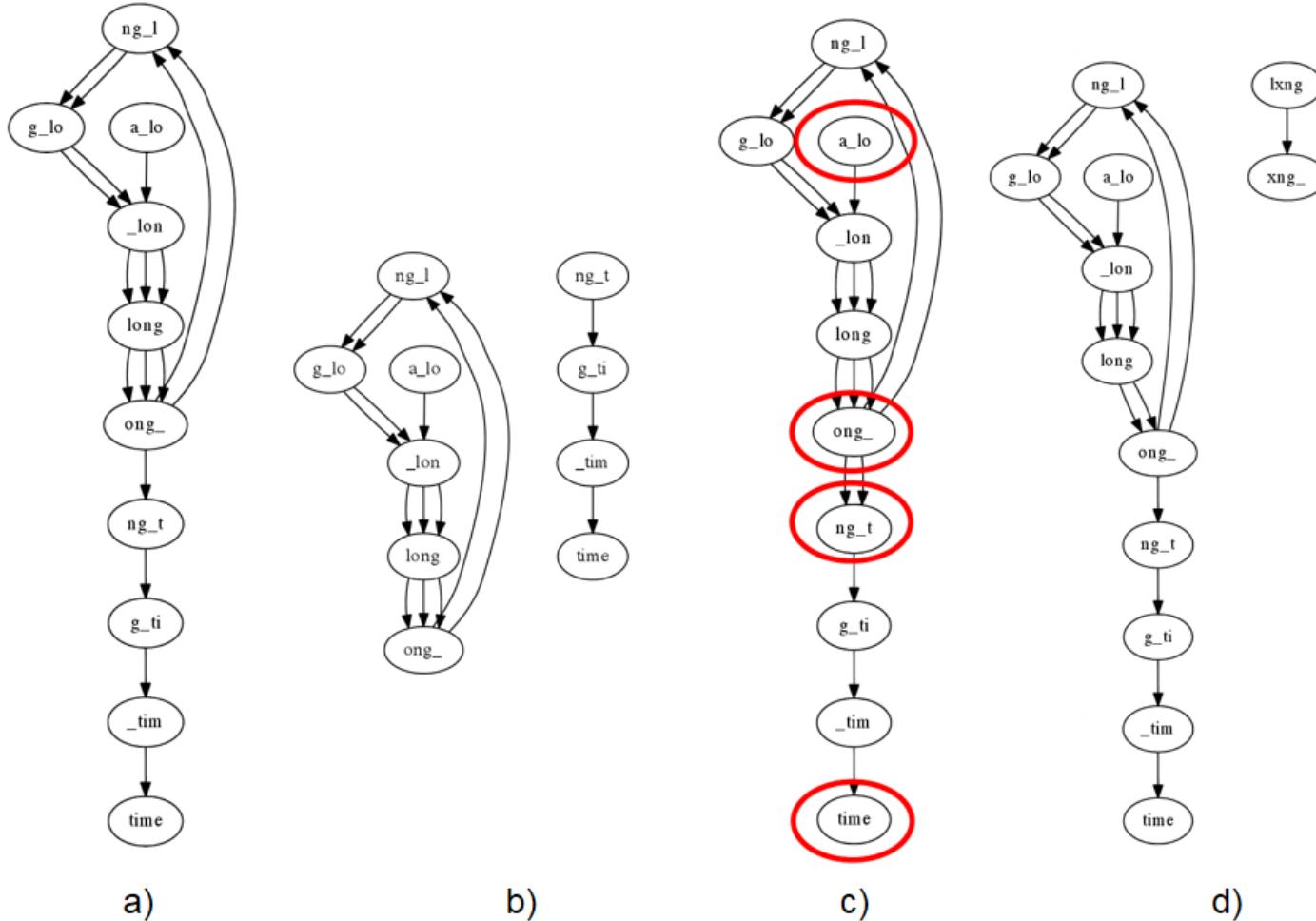
Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

# Violating assumptions in de Bruijn graphs

---

- Assume a sequence: **a\_long\_long\_long\_time**
  - length  $m=21$ , the sequence contains repeats,
  - choose  $k=5$ , number of 5-mers  $n=m-k+1=17$ ,
  - taken from [Langmead: de Bruijn graph assembly, 2014].
- Assume different sets of k-mers (see the next slide):
  - (a) all 5-mers  $\rightarrow$  the correct assembly,
  - (b) omitting **ong\_t**  $\rightarrow$  two graph components, the overall graph not Eulerian,
  - (c) extra copy of **ong\_t**  $\rightarrow$  4 semi-balanced nodes, graph not Eulerian,
  - (d) errors and differences between chromosomes, turn a copy of **long\_** into **lxng\_**  $\rightarrow$  graph not connected, largest component not Eulerian.

# Violating assumptions in de Bruijn graphs



Langmead: de Bruijn graph assembly, 2014.

## de Bruijn graphs – short k-mers

---

- Only short k-mers guarantee that none is missed,
- still, the number of k-mers remains  $O(N)$ 
  - $N$  is the total length of reads,
- de Bruijn graph with  $O(N)$  edges and  $O(N)$  nodes too
  - can be constructed in  $O(N)$ ,
  - Euler cycle found in  $O(N)$ .

Genome: `a_long_long_long_time`

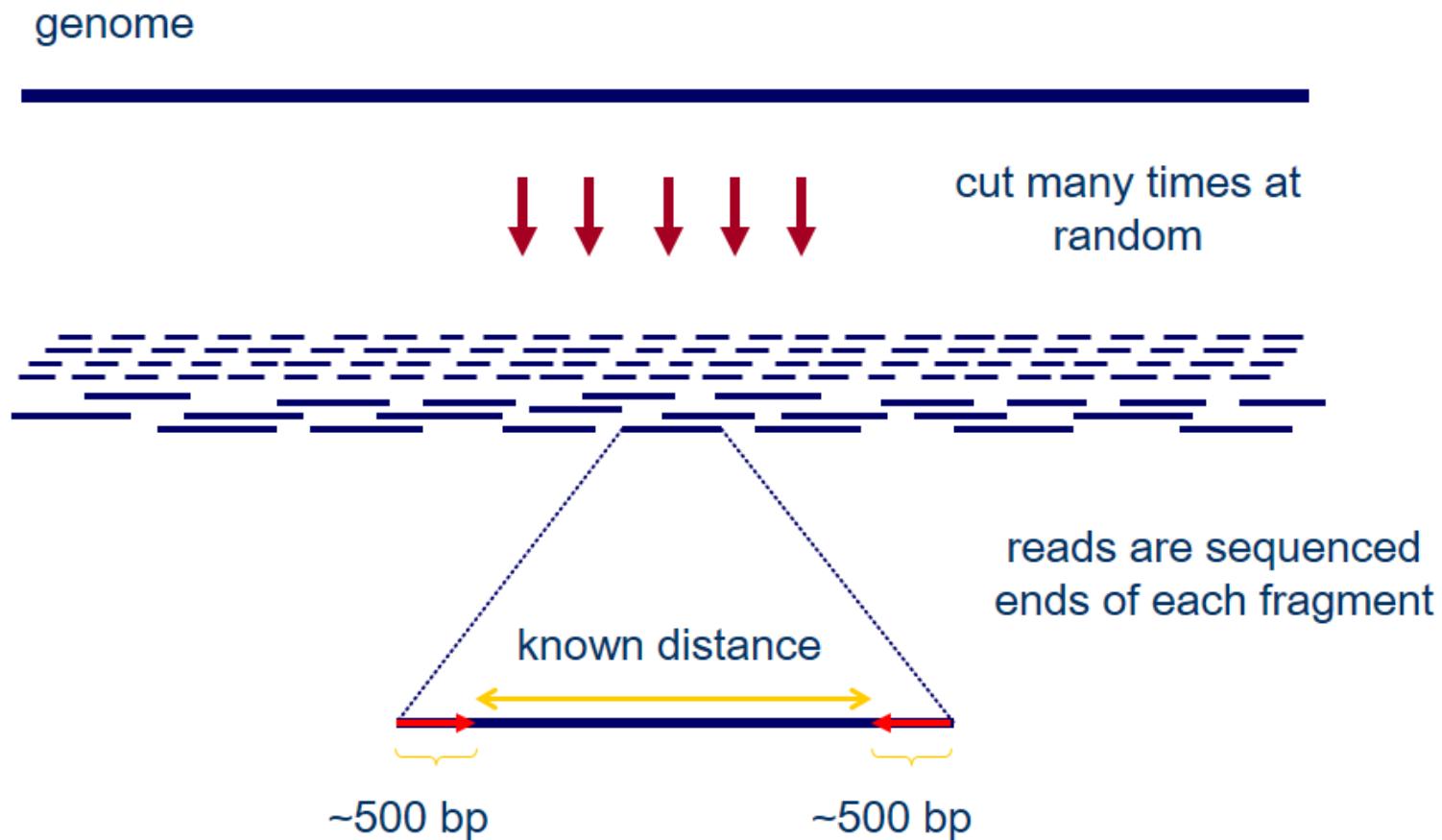
Reads: `a_long_long_long`, `ng_long_l`, `g_long_time`

k-mers: `a_long_l`                    `ng_long_`    `g_long_t`  
`_long_lo`                            `g_long_l`    `_long_ti`  
`_long_lon`                            `long_tim`  
`ong_long`                            `ong_time`  
`ng_long_`  
`g_long_l`  
`_long_lo`  
`_long_lon`  
`ong_long`

Langmead: de Bruijn graph assembly, 2014.

## Paired end reads

- One approach to reducing ambiguity in assembly is to use paired end reads.



Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

# Repetitive sequences (repeats)

---

- Most common source of assembly errors,
- if sequencing technology produces reads  $>$  repeat size, impact is much smaller,
- most straightforward solution:
  - mate pairs with spacing  $>$  largest known repeat.

True structure of genomic region

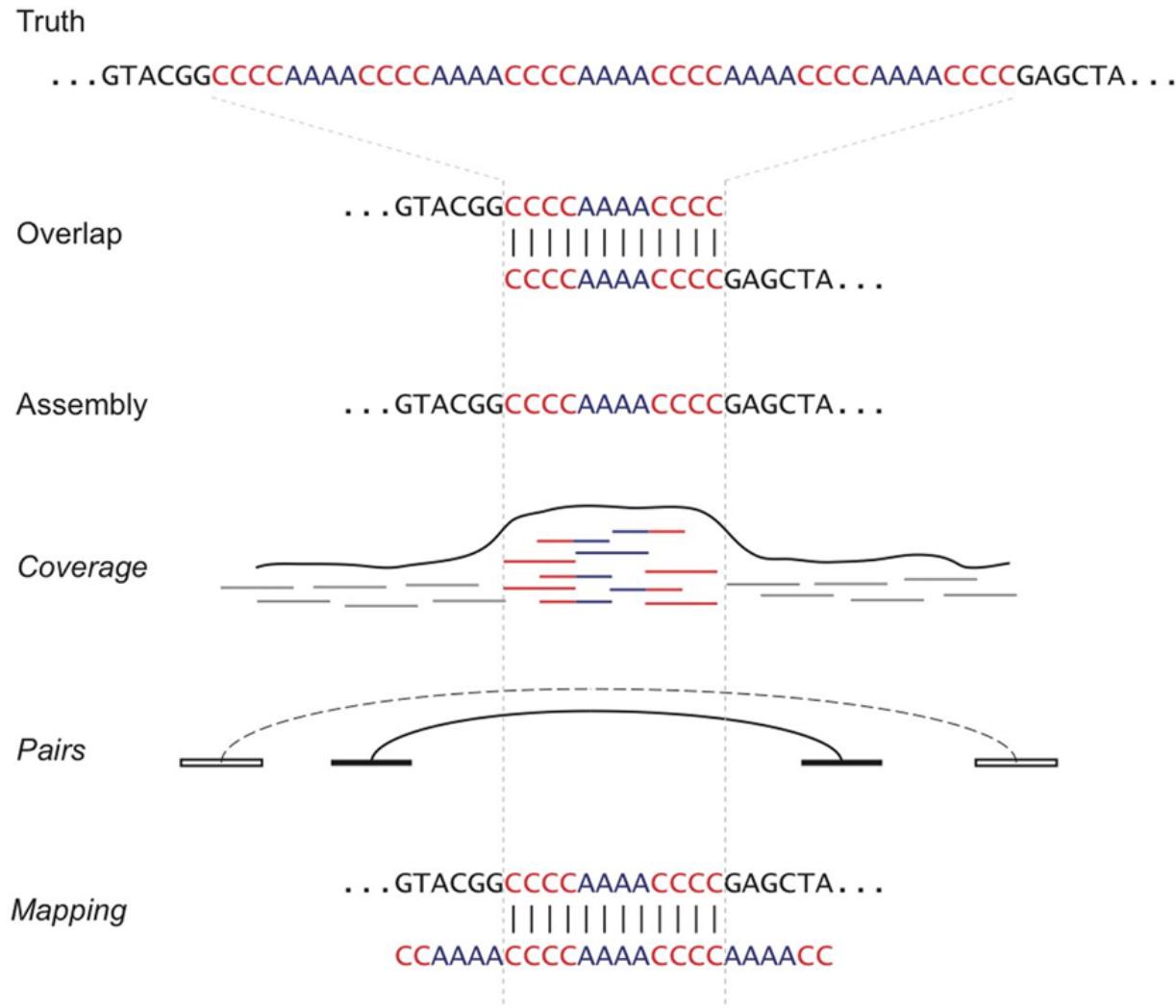


Incorrect assembly with “orphan” contig (red)



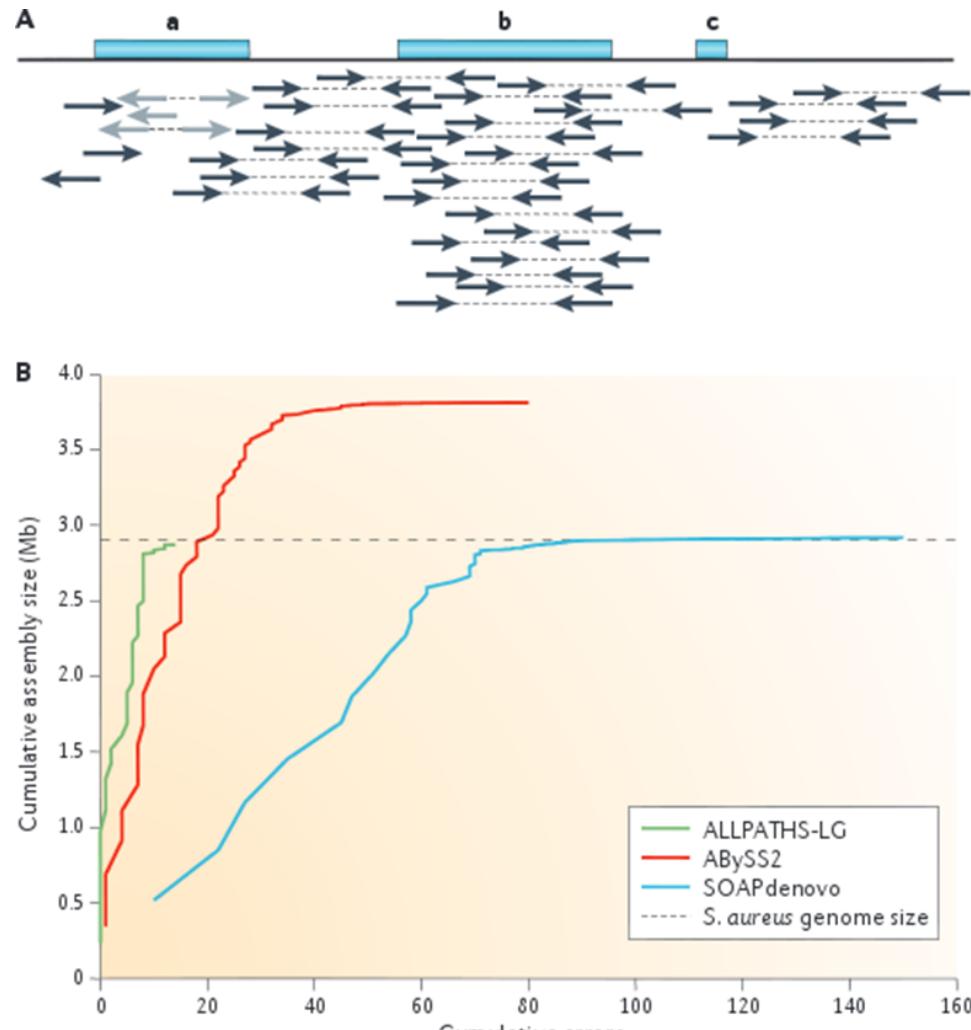
Salzberg and Yorke: Beware of mis-assembled genomes, Bioinformatics, 2005.

# Mis-assembly of repetitive sequence



Schatz et al.: Hawkeye and AMOS: visualizing and assessing the quality of genome assemblies, Brief Bioinform 2013.

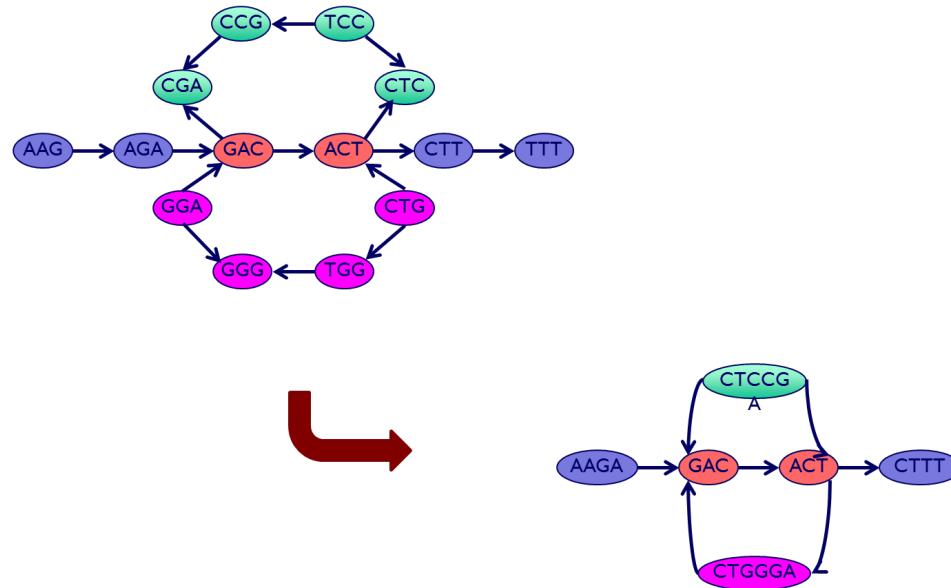
# Methods for assembly validation



Nagarajan and Pop: Sequence Assembly Demystified, Nature Reviews, 2013.

# The Velvet assembler

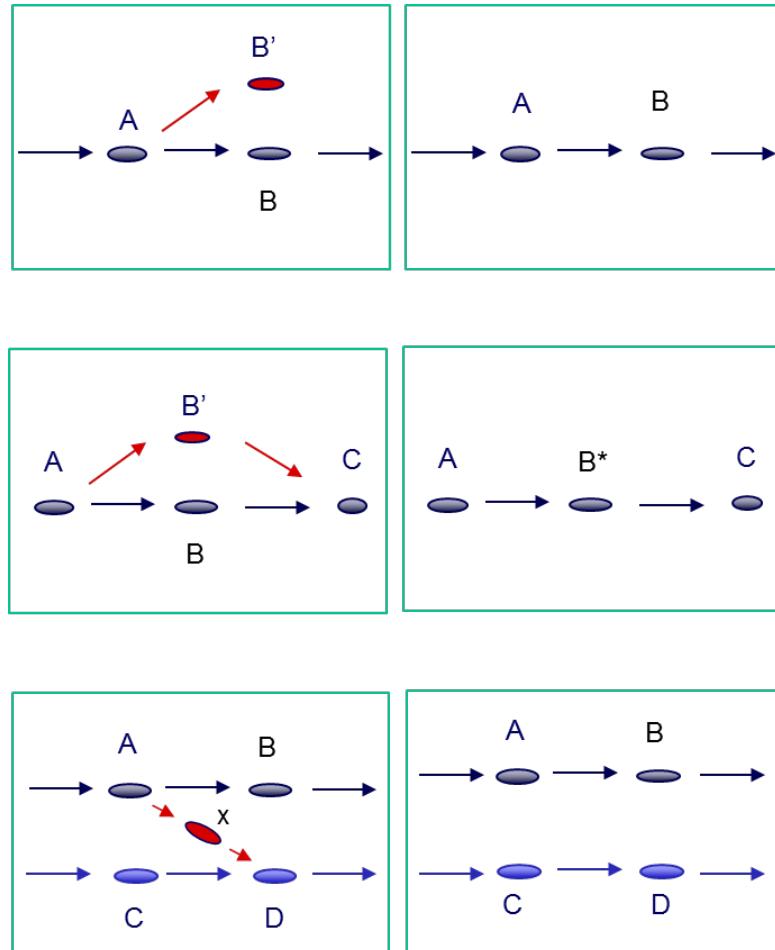
- Based on de Bruijn graphs, includes additional tricks for
  - reducing the size of the graph,
  - trying to correct for errors in sequences,
  - taking advantage of paired-end reads,
- compress the graph, collapse linear subgraphs:



Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

# Error correction in Velvet

- errors at end of read
  - trim off “dead-end” tips,
- errors in middle of read
  - pop bubbles,
- chimeric edges
  - clip short, low coverage nodes.



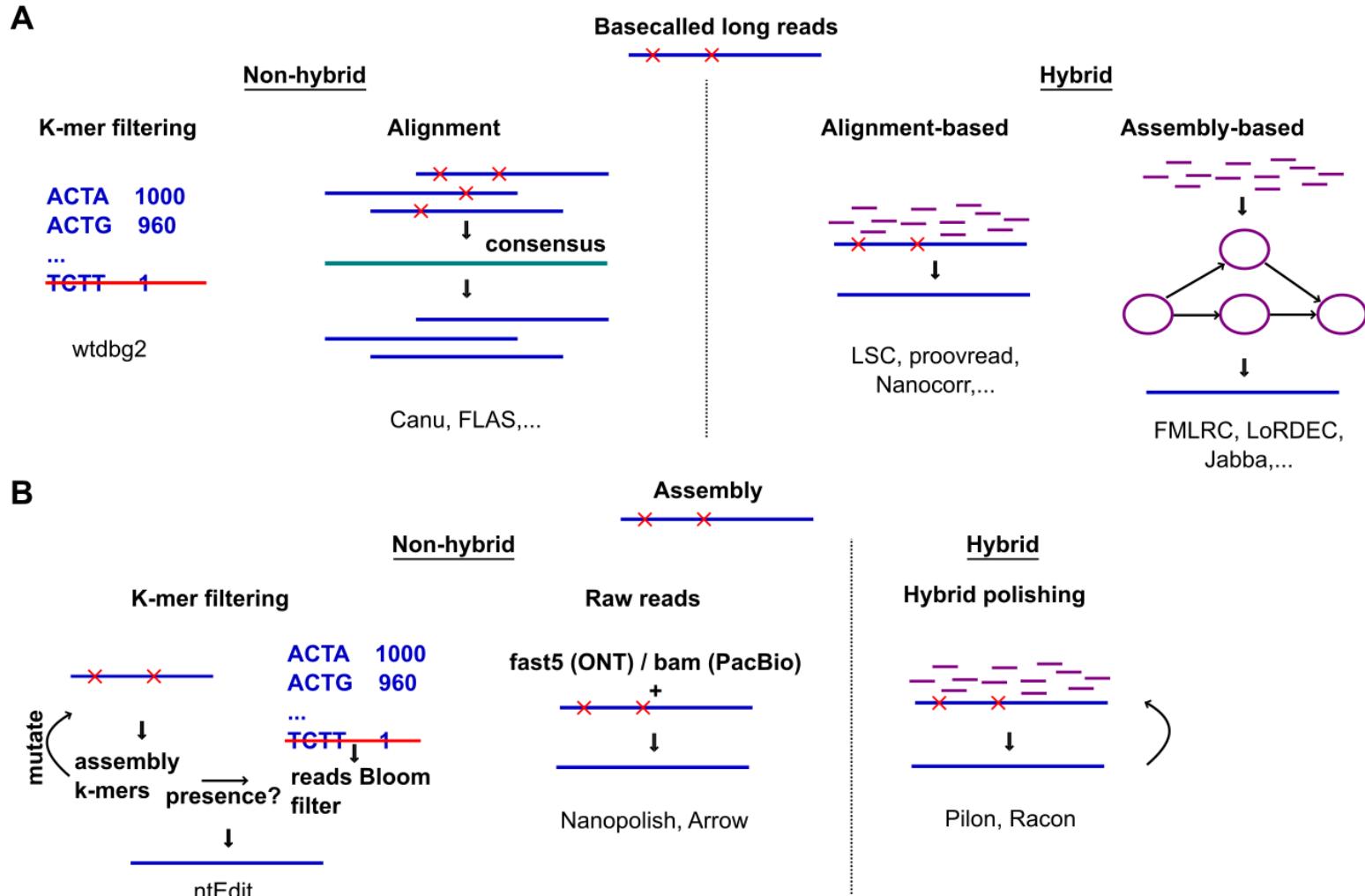
Marc Craven, BMI/CS 576, [www.biostat.wisc.edu/bmi576](http://www.biostat.wisc.edu/bmi576).

## Short vs long read assembly

---

- short read assembly
  - assembly most often based on de Bruijn graphs,
  - the main issues
    - \* large number of reads, efficiency issues,
    - \* repeat resolution,
- long read assembly
  - assembly based on overlap graphs,
  - the main issue is read correction
    - \* requires high coverage (50-100x), could be expensive,
    - \* or hybrid assembly with shorter reads to error-correct.

# Long read correction and polishing



Amarasinghe et al.: Opportunities and challenges in long-read sequencing data analysis, *Genome Biology*, 2020.

# Summary

---

- The sequencing problem
  - sequencing in vitro,
  - sequence assembly in silico,
    - \* de novo versus resequencing,
    - \* approaches: greedy, overlap graph, Euler trail,
    - \* elements: reads, contigs, scaffolding,
  - assembly validation
    - \* statistical, viewers, comparative methods,
- still open problem
  - costs, efficiency, reliability,
  - changes in sequencing imply changes in assembly.