Algorithms in Sequence Alignment

Lecture 4

Genome sequencing

Jaap Heringa
Vrije Universiteit Amsterdam

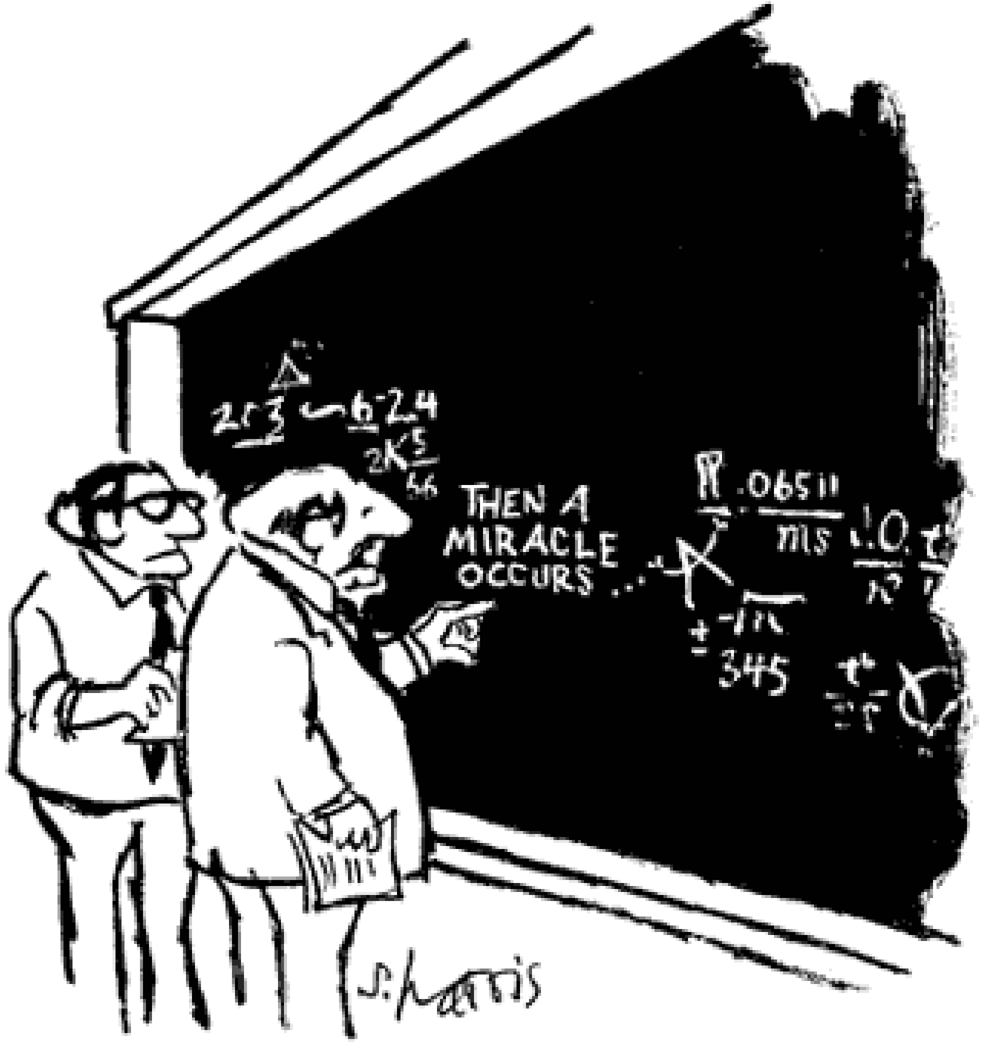
Content

- Human Genome Project – short history
- DNA sequencing
    - De Novo sequencing (de Bruijn graphs)
        - Paired-end reads
    - Reference-based sequencing (Burrows-Wheeler Aligner – BWA method)*

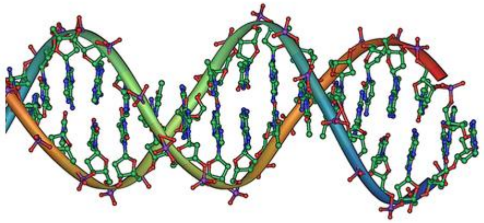
*Important for second assignment (BWA)

… the bioinformatics big bang

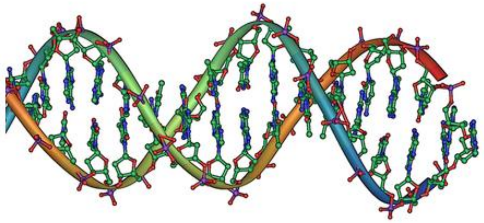
The Human Genome Project (HGP)



"I think you should be more explicit here in step two."



DNA sequence



OH BRAD, THEY SAY THERE'S DNA IN MY BODY!
WHO CARES, DARLING, ...WHO CARES...

Illustration by Ulla Mackensen

```
.....acctc ctgtgcaag acatgaaaca cccgatttac
nctgtggttc cccagatgg gtcctgtccc aggtgcacct
gcaggagtcg gcccaggac tggggaagcc tccagagctc
aaaaccccac tggtgacac aactcacaca tgcccacggt
gcccagagcc aaatcttgt gacacacctc ccccgtgccc
acggtgccca agcccaaat cttgtgacac acctccccca
tgcccacggt cccagagcc caaatcttgt gacacacctc
ccccgtgccc cggtgccca gcacctgaac tcttgggagg
accgtcagtc tcctcttcc ccccaaaacc caaggatacc
cttatgattt ccggacccc tgaggtcacg tgcgtggtgg
tggacgtgag cacgaagac cctnnngtcc agttcaagtg
gtacgtggac gcgtggagg tgcataatgc caagacaaag
ctgcgggagg gcagtacaa cagcacgttc cgtgtggtca
gcgtcctcac gtcctgcac caggactggc tgaacggcaa
ggagtacaag gcaaggtct ccaacaaagc aaccaagtca
gcctgacctg ctggtcaaa ggcttctacc ccagcgacat
cgccgtggag gggagagca atgggcagcc ggagaacaac
tacaacacca gcctcccat gctggactcc gacggctcct
tcttcctcta agcaagctc accgtggaca agagcaggtg
gcagcagggg acatcttct catgctccgt gatgcatgag
gctctgcaca ccgctacac gcagaagagc ctctc.....
```

Human genome project (1990 – 2003)

- ‘a milestone for humanity’
- performed using legacy sequencing techniques

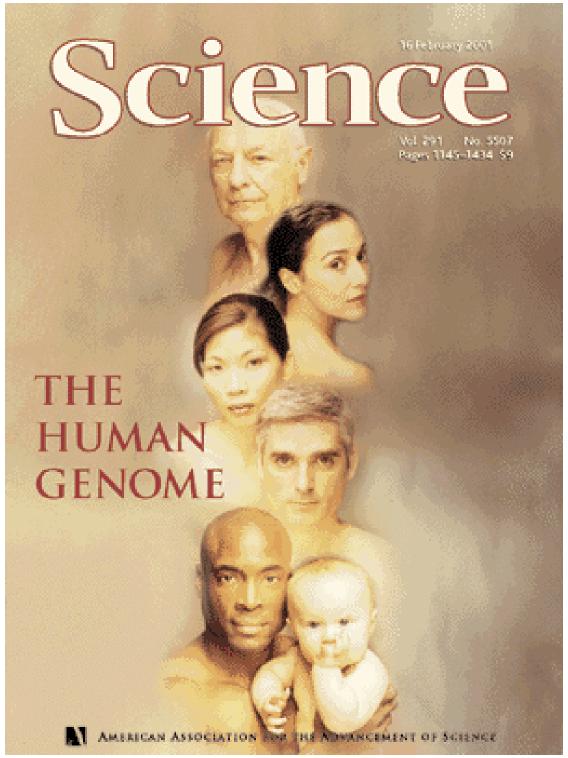
Human genome project (1990 – 2003)

- ‘a milestone for humanity’

- performed using legacy sequencing techniques

Craig Venter’s thread: human genome data might be made proprietary via patents by Celera Genomics

The Human Genome -- 26 June 2000



Dr. Craig Venter

Celera Genomics

-- Shotgun method

Francis Collins (USA) /

Sir John Sulston (UK)

Human Genome Project

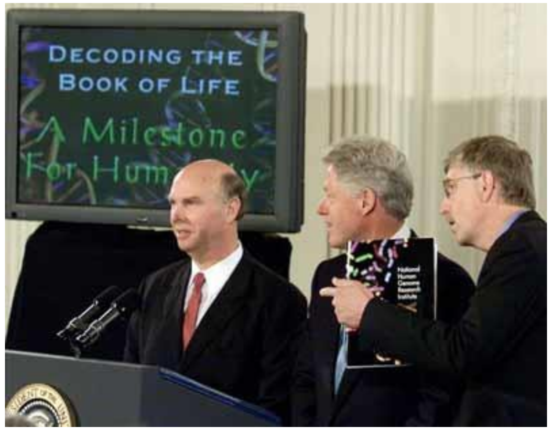
The Human Genome -- 26 June 2000

“Without a doubt, this is the most important, most wondrous map ever produced by humankind.”

U.S. President Bill Clinton on 26 June 2000
during a press conference at the White House.

26th June 2000 – announcing the first draft of the human genome

On 26 June 2000, the press conference at the white house took place, hosted by President Bill Clinton. Leaders of the public project and Celera announce completion of a working draft of the human genome sequence.



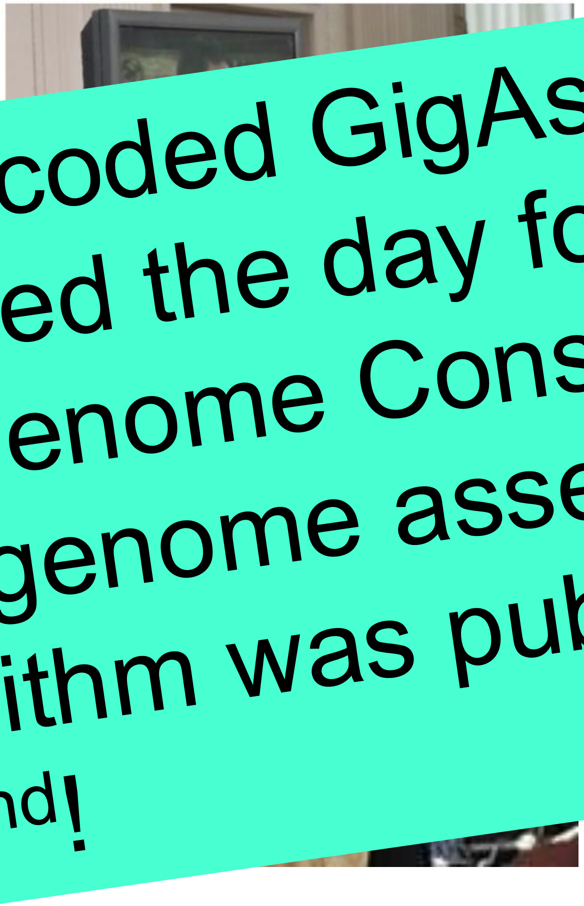
Craig Venter, Celera Genomics

Francis Collins, Human Genome Consortium

On hand at a press conference that followed the White House genome announcement are (from l) Dr. Craig Venter, Celera; and Dr. Francis Collins, director, NHGRI (NIH). Resolving the animosity between the rival projects was accomplished just in time to broker the joint announcement at the White House in Washington.

26th June 2000 – announcing the first draft of the human genome

On 26 June 2000, the press conference at the white house took place, hosted by President Bill Cl
Leaders o
and Celer
completio
of the hum
sequence.



Jim Kent coded GigAssembler, which saved the day for the Human Genome Consortium. The first genome assembly using the algorithm was published on June 22nd!

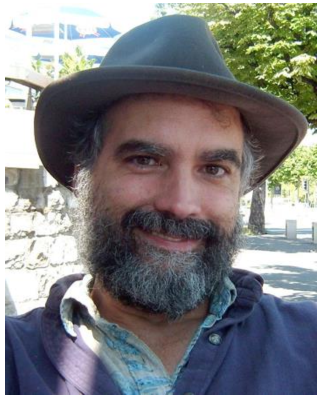
On h a press conference that followed the White House genome announcement are (from l) Dr. Craig Venter, Celera; and Dr. Francis Collins, director, NHGRI (NIH). Resolving the animosity between the rival projects was accomplished just in time to broker the joint announcement at the White House in Washington.

Jim Kent (UCSC)

- Wrote the program GigAssembler and built a small cluster computer to run it (together with David Haussler at UC Santa Cruz
  - Celera had the largest civilian cluster computer in the world at the time
- Kent is best known as the researcher who "saved" the human genome project, a feat chronicled in the *New York Times*. With little more than a month before the company Celera was to present a complete draft of the human genome to the White House in 2000, Kent wrote GigAssembler, a program that produced the first full working draft assembly of the human genome, which kept the data freely available in the public domain.
- Kent's first assembly on the human genome was released on June 22. Celera finished its assembly 3 days later on June 25, and the dual results were announced at the White House on June 26, pretending a friendly joint finish.
- The Santa Cruz data was made publicly available on the World Wide Web while the research paper describing this publicly funded genome was published in February 2001 special issue of *Nature*, in parallel (in that same week) with Celera's results in the journal *Science.*
- Kent received a number of prizes for his work, such as the Overton Prize by the ISCB.

GigAssembler

Report UCSC-CRL-00-17; December 27, 2000
WJ Kent, D Haussler, Genome Res. 2001. 11: 1541-1548



```
 extension                          tail

-----------------------------------
        |||||||||||||||||||||||
     -----------------------------------

    tail                                extension
```

```
A --------------------
      B ----------------------
   C---------------
           D ++++++?????????----------------
```

Add fragment D to ‘raft’ ABC based on alignment C-D (‘+’ region is aligned part of C and D, ‘?’ region needs to be checked between D and A, and D and B)

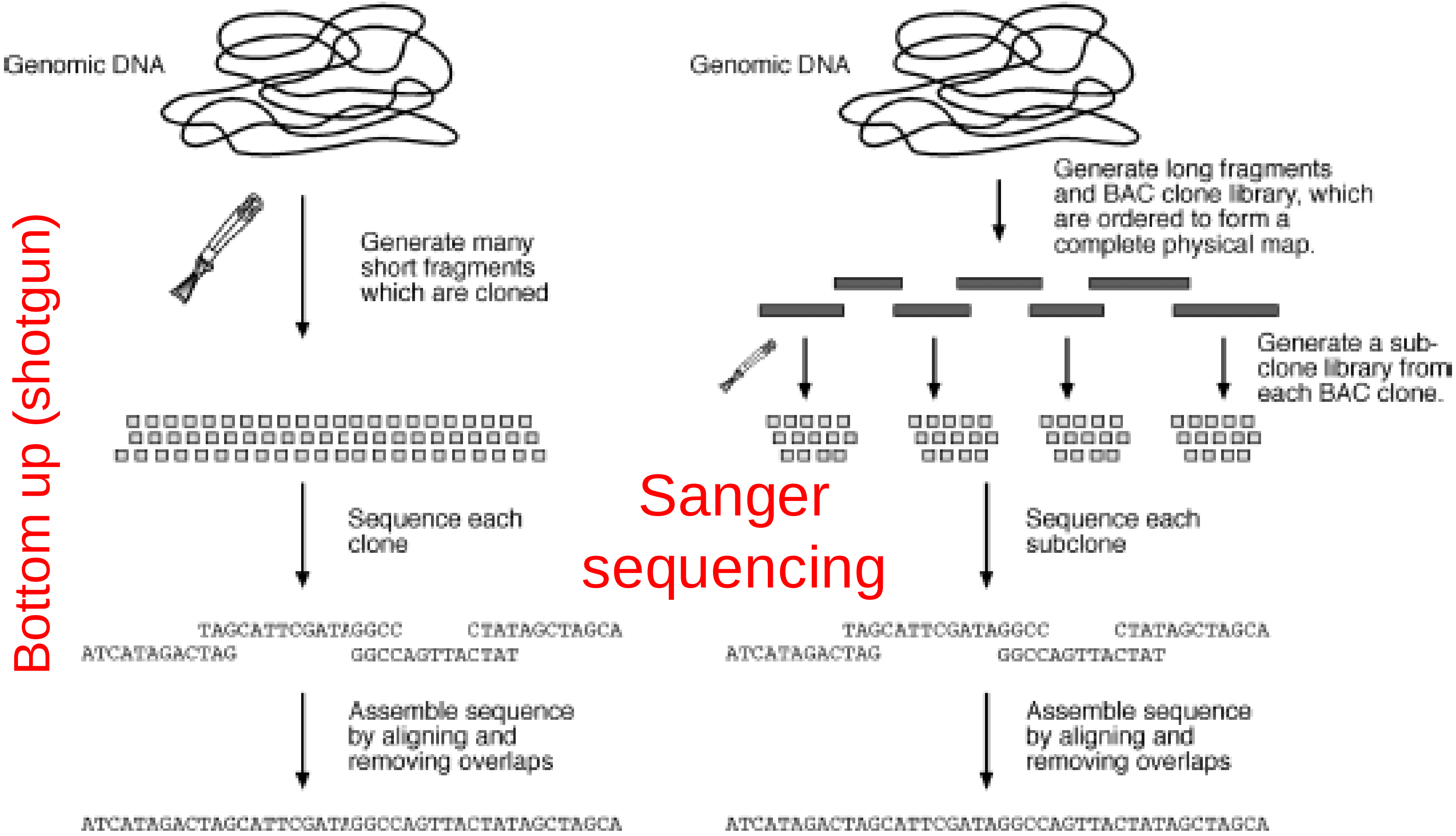
```
AAAAAAAAAAAAAAAAAAAAA
a1a1a1a1  a2a2a2a2a2
      BBBBBBBBBBBBBBBBBBBBB
      b1b1b1b1b1b1   b2b2b2
                        CCCCCCCCCCCCCCCCCCCCC
                        c1c1c1      c2c2c2c2
```

```
             /-->---b2c1----->------|
            /                       |
           /              /->-c2->-\ |
          /              /          \|
As-->--Bs-->--Ae-->--Cs-->--Be-->--Ce
 \                            /
  \                          /
   \--->---a1b1a2---->----/
```

Add rafts a1b1a2, b2c1 and c2 to initial order graph (showing begin and end of reads) to assess consistency

Finding most consistent solution (using graphs) for stringing fragments based on overlap

HGP sequencing methods


Genomic DNA
Generate many short fragments which are cloned
Sequence each clone
TAGCATTCGATAGGCC
CTATAGCTAGCA
ATCATAGACTAG
GGCCAGTTACTAT
Assemble sequence by aligning and removing overlaps
ATCATAGACTAGCATTCGATAGGCCAGTTACTATAGCTAGCA
Genomic DNA
Generate long fragments and BAC clone library, which are ordered to form a complete physical map.
Generate a sub-clone library from each BAC clone.
Sequence each subclone
TAGCATTCGATAGGCC
CTATAGCTAGCA
ATCATAGACTAG
GGCCAGTTACTAT
Assemble sequence by aligning and removing overlaps
ATCATAGACTAGCATTCGATAGGCCAGTTACTATAGCTAGCA

Bottom up (shotgun)

Sanger sequencing

Top down (hierarchical)

Celera Genomics (Craig Venter)

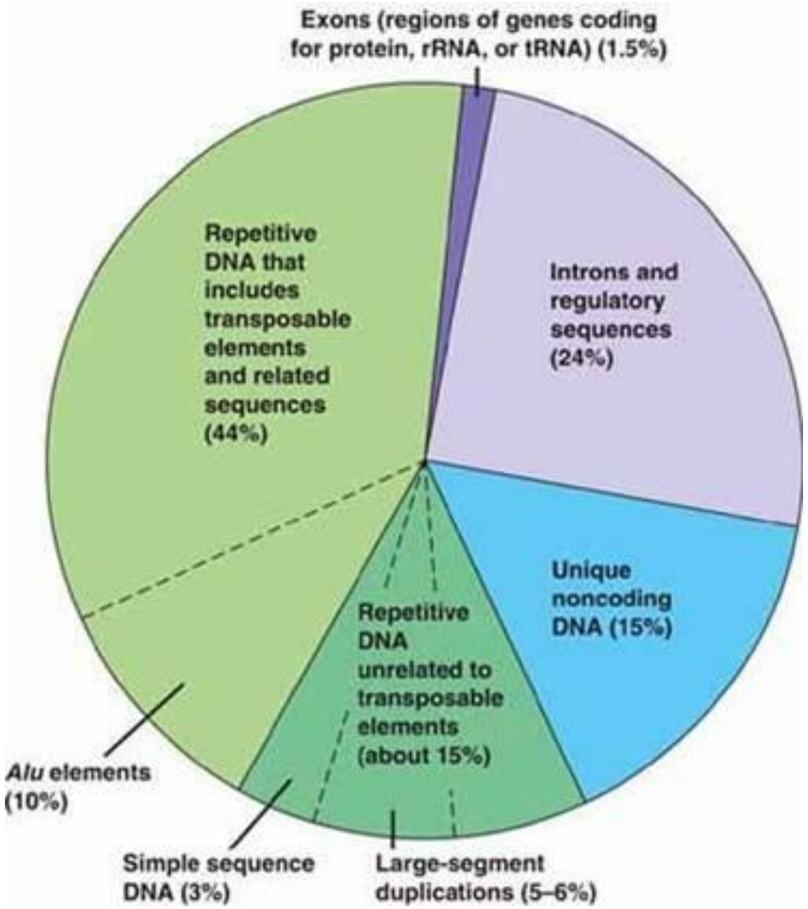
Human Genome Consortium

HGP sequencing methods

- Top-down (hierarchical) - The HGP consortium
  - Using top-down mapping, a single chromosome was cut into large pieces (chunks of ~150kb) that were amplified using bacterial artificial chromosome (BAC) clones. The chunks were carefully mapped such that the exact location of each chunk was known. The chunks were then cut up again -- each piece again 'shotgunned' into 2kb fragments and sequenced using Sanger's chain termination method. The resulting map depicts the order and distance between the original cleaves.
- Bottom-up (shotgun) - Celera (Craig Venter)
  - In shotgun sequencing, DNA is broken up randomly into numerous small segments, which are sequenced (using Sanger's chain termination method) to obtain *reads*. Multiple overlapping reads for the target DNA are obtained by performing several rounds of this fragmentation and sequencing. Computer programs then use the overlapping ends of different reads to assemble them into blocks of contiguous sequence (contigs). This is still the common sequencing strategy (but using modern sequencing technology)

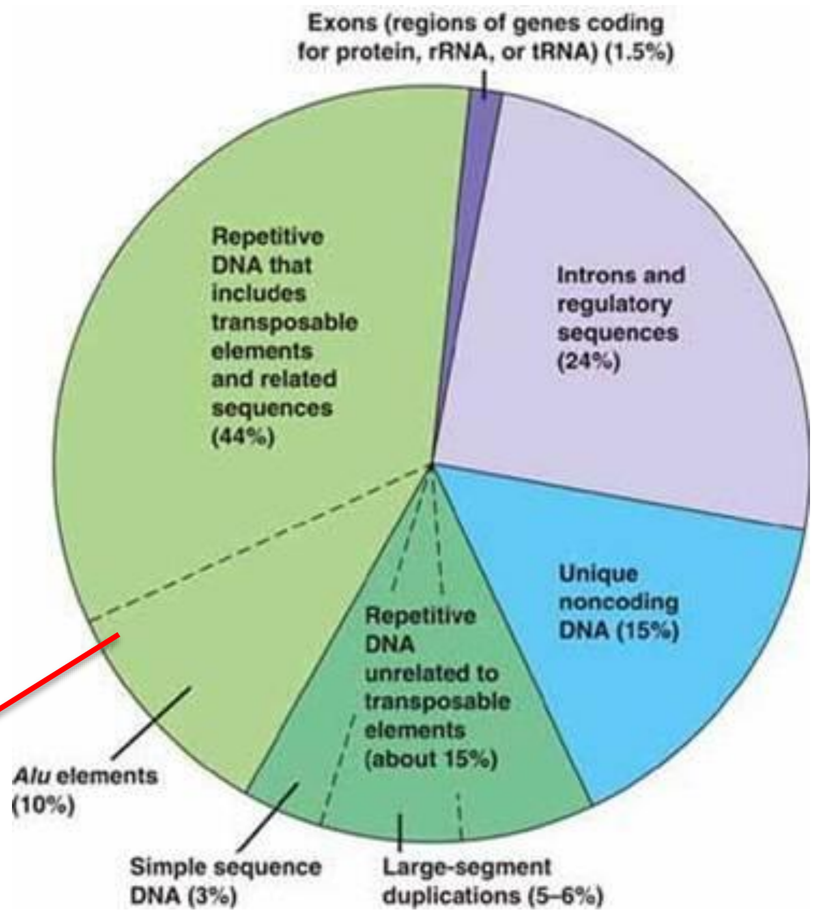
Human genome project - in numbers

- 23 chromosome pairs
- 20.000 genes
- 2.9 of 3.3 billion base pairs sequenced



Human genome project - in numbers

- 23 chromosome pairs
- 20.000 genes
- 2.9 of 3.3 billion base pairs sequenced

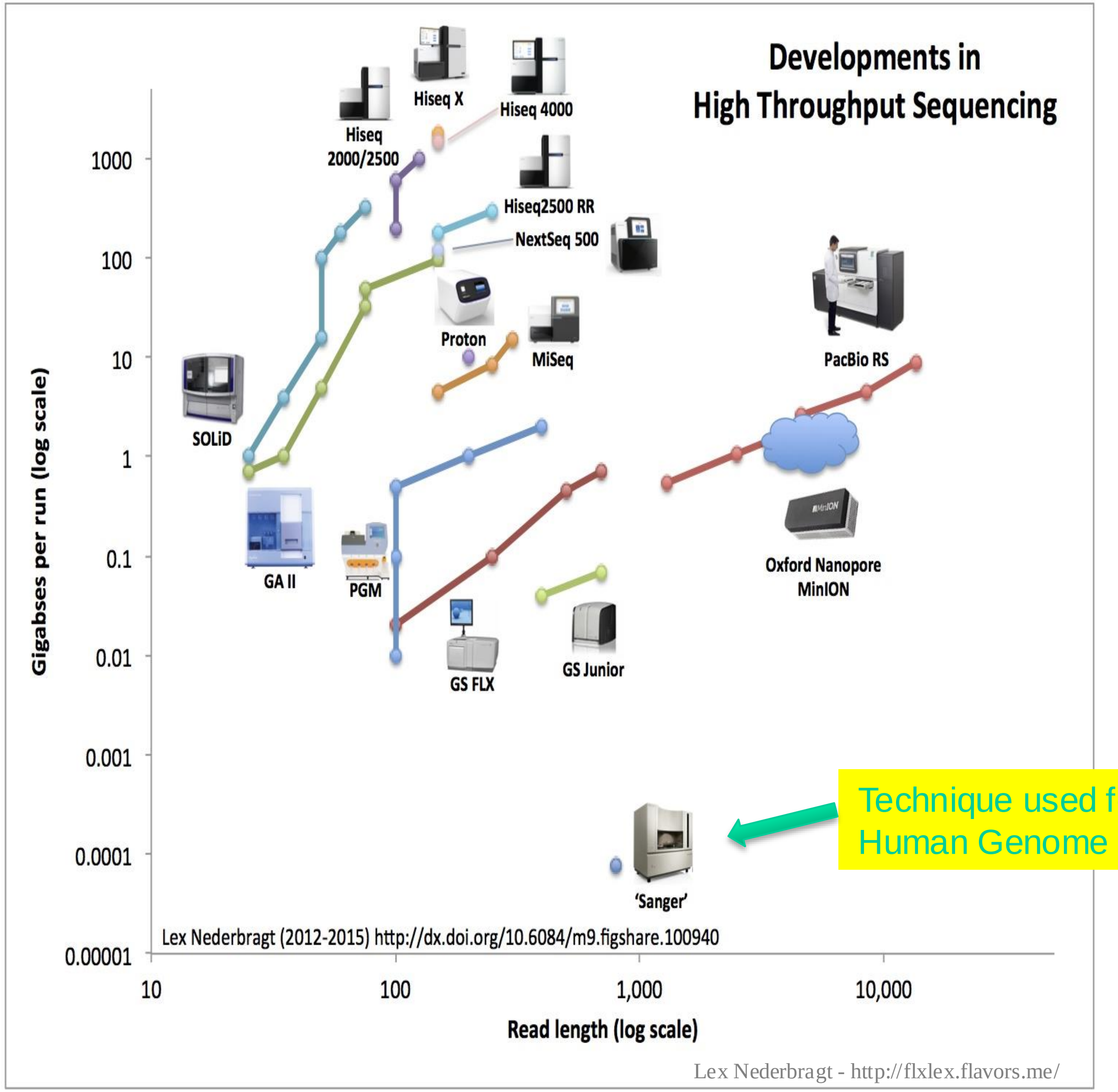


Easily more than half of the human genome comprises repetitive elements. This is a **challenge** for reconstruction (sequencing) algorithms

Next Generation Sequencing (NGS)

- Massively parallel sequencing of **millions to billions** of short fragments
- Very fast
  - E.g. compared to Sanger sequencing – exploited in HGP- max 384 DNA samples in a single batch (run) in up to 24 runs a day
- Huge amounts of data generated in single sequencing experiment (many TBs)
- Much reduced cost (1 human genome: HGP 3 billion $ *versus* current cost NGS $600)
- Shorter fragments (reads) than with Sanger sequencing
  - Many different techniques exist but based on approx. same principle. Differences reside mainly in chemical usage and the way fragments are stuck to the surface

Next Generation Sequencing



Lex Nederbragt (2012-2015) http://dx.doi.org/10.6084/m9.figshare.100940

Lex Nederbragt - http://flxlex.flavors.me/

Technique used for Human Genome Project

Source: Walter Pirovano, BaseClear

illumina®

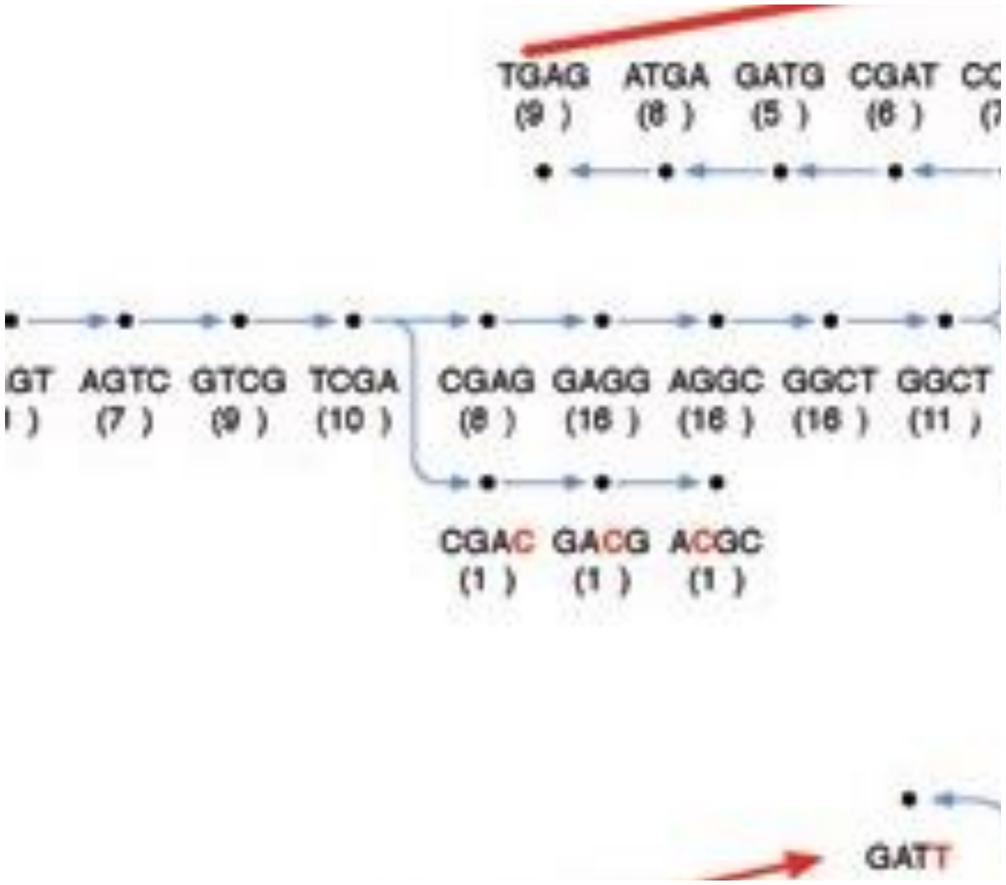
HiSeq 2500 System

MinION (2015)

Oxford NANOPORE Technologies

Source: Walter Pirovano, BaseClear

Sequencing (or assembly): The problem



Reconstructing a DNA sequence from many randomly selected short fragments (reads)

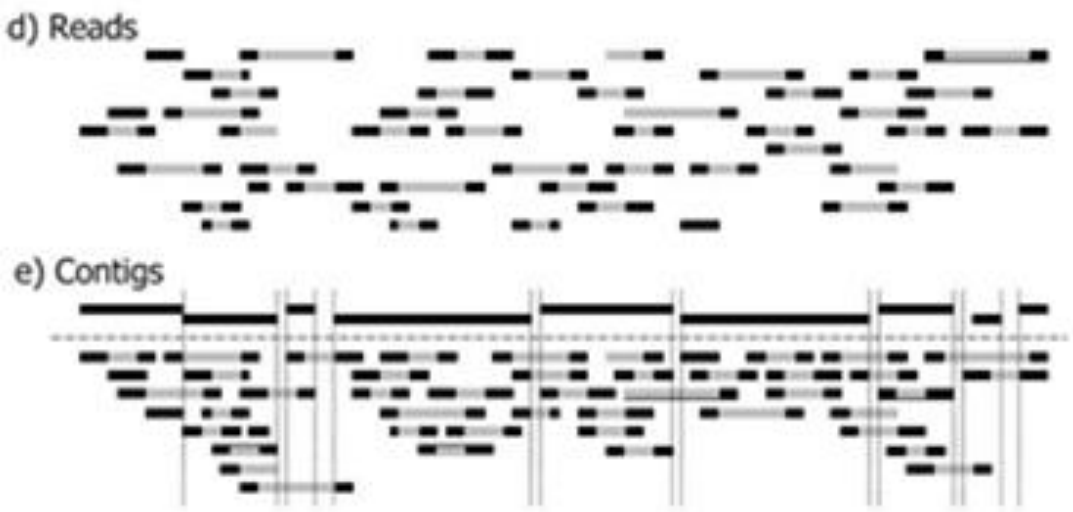
Reads may contain (experimental) **E**rrors…

Putting the reads together using bioinformatics

Two main ways of stringing together the very many small fragments into a complete genome sequence

- –*De novo* assembly of a genome
- –Assembly using alignment onto a **reference genome**

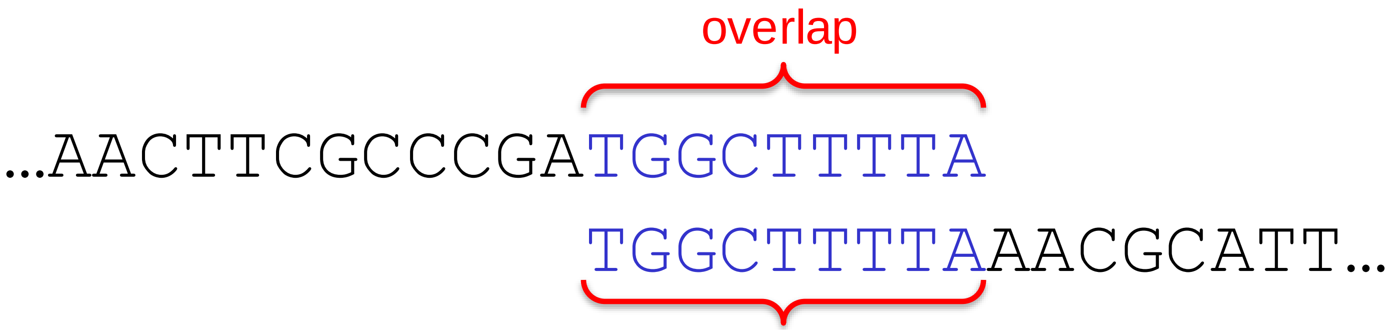
Assembly:
Puzzling reads into a genome



```
ACAGGAGGT                                        read1
         GAGGTCCAGA                     read2
        CCAGATGATGATA read3
-------------------------------------------------------
ACAGGAGGTCCAGATGATGATA         contig
```

De novo sequencing - a contig

- Reconstructing a complete genome *de novo* requires testing possible overlaps between all reads and then building the whole genome together according to some criterion:



overlap

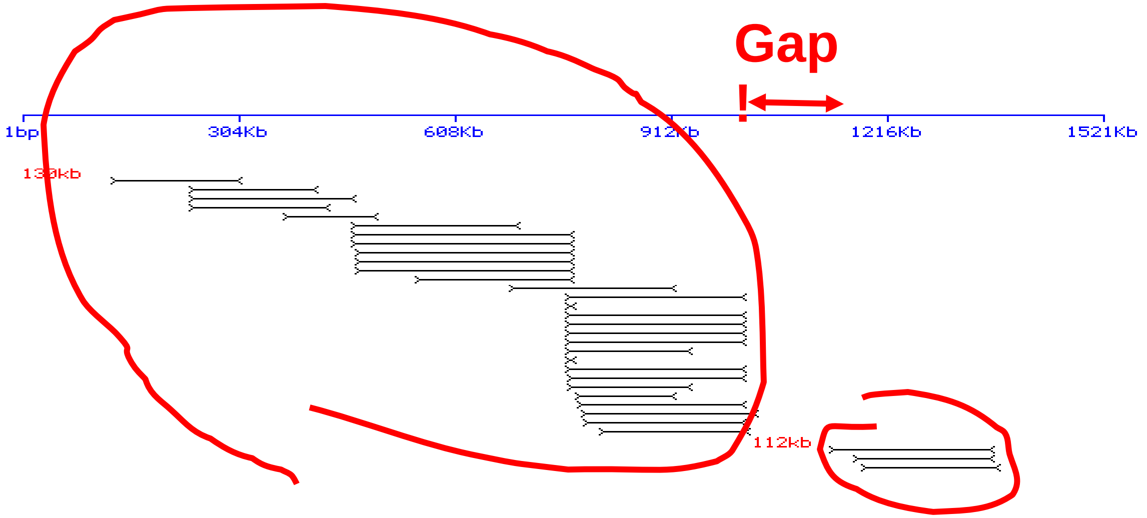
...AACTTCGCCCGATGGCTTTTA

TGGCTTTTAAACGCATT...

  - A known problem in Computer Science is the **Shortest Superstring Problem** (SSP), where all fragments are strung up to produce the shortest overall string *(i.e.* genome).
    - However, the shortest possible string is not an ideal criterion because genomes have many repeating fragments

De novo sequencing - a contig

- **Contig** – a continuous set of overlapping sequences



Computer programs have to check the overlap of all against all fragments to find the most likely order of the fragments, resulting in a completely reconstructed DNA sequence

De novo sequencing: two main problems

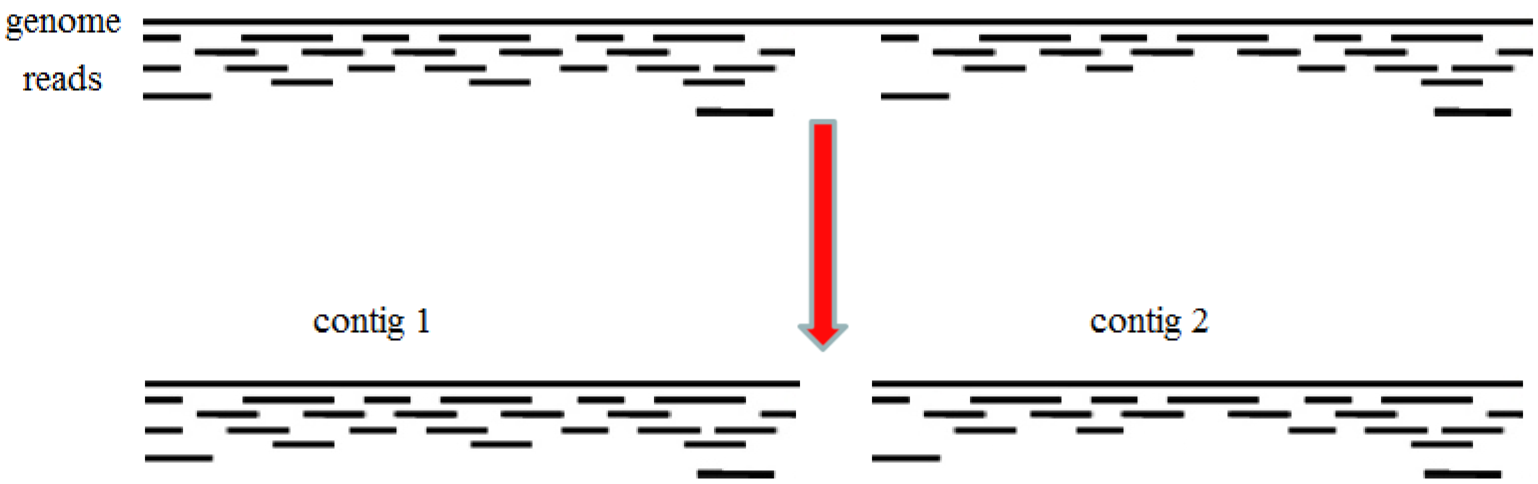
- Multiple contigs

  (due to lack of overlapping reads)

- Repeats

Lack of coverage

- Lack of coverage of the reads on the original genome causes multiple contigs
  - – due to randomness of shearing process there is a chance that some regions of the genome are unsequenced

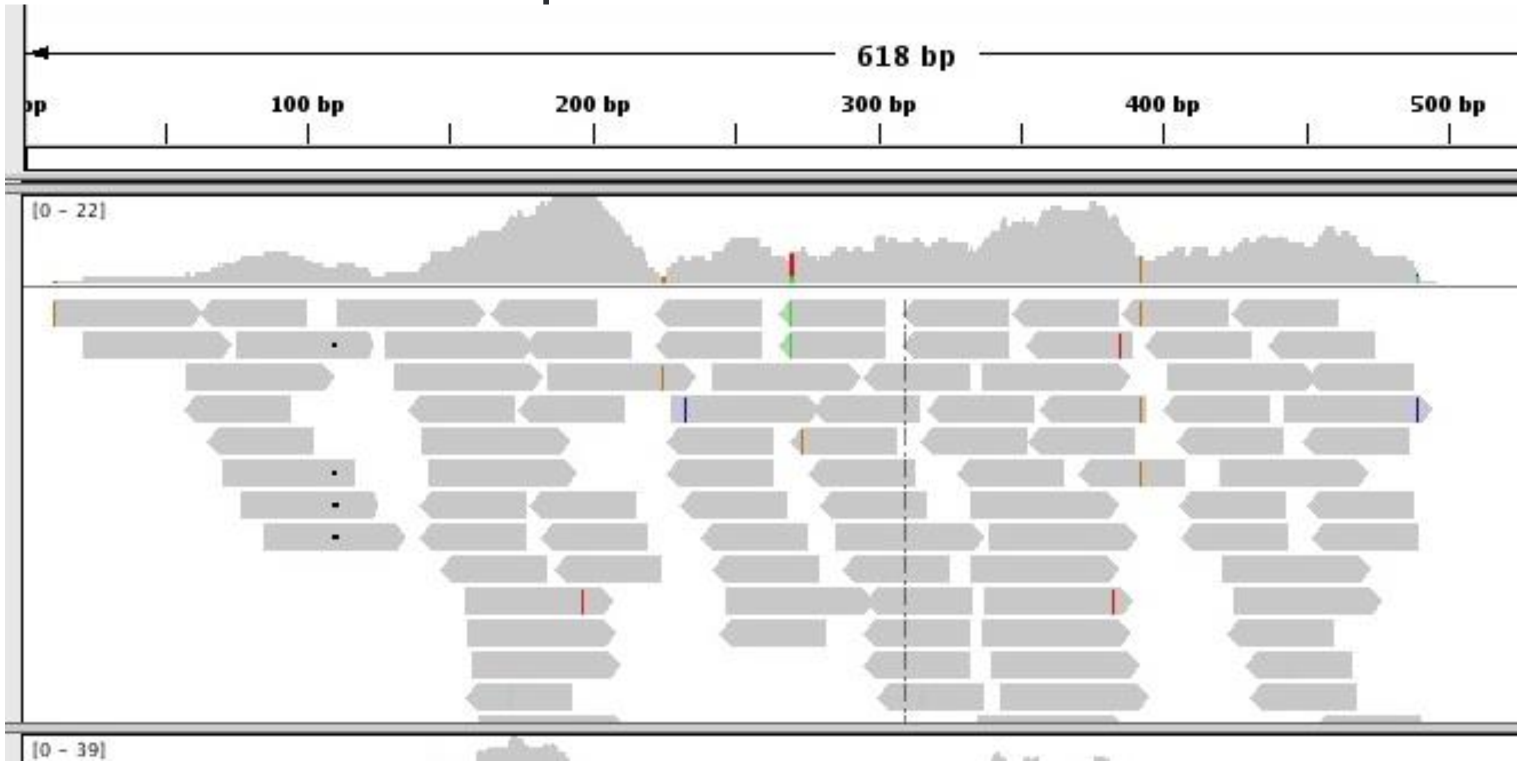


Depth of Coverage – Measure*

Usage of sequencing “depth” and sequencing “coverage”

•sequencing depth: (average) number of reads per base (often on entire sequencing sample)

•coverage: fraction of the genome (or targeted region) that has been sequenced at least once.

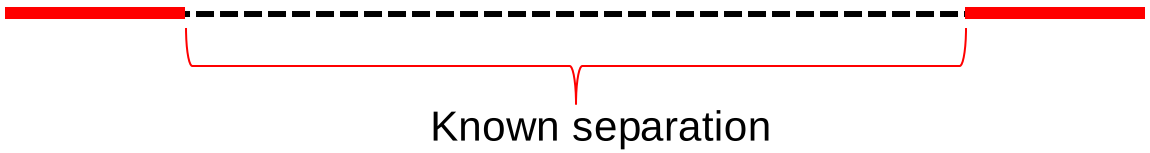


How to deal with gaps in between contigs

- If a number of unconnected contigs are present after building them using overlapping reads, the order of the contigs in the genome cannot be determined
- A way to alleviate this problem is by using paired-end reads or mate pairs
  - These are longer fragments of known length where only the (short) flanking regions are sequenced (i.e. the number of nucleotides in between the sequenced flanks is known)

Paired-end Reads and Mate Pairs

- High-molecular-weight DNA is sheared into random fragments, size-selected (usually 2, 5, 10 kb).
- Fragments are sequenced from both ends, yielding two short sequences.
  - Each sequence is called an *end-read* and two reads bridging a fragment are referred to as *mate pairs* (or *paired-end reads)*. Because fragments are size-selected, the distance between the two reads is known.



HGP: Celera basically used paired-end reads; much more than the HGC

For reference

Paired-end versus Mate Pair

‘length’ difference

Paired-end
Each "read" is two sequences (a pair) from each end of the same genomic DNA fragment. The distance between the reads on the original genome sequence (150-800 bp depending on NGS technique) is roughly equal to the length of the fragment that was sequenced at either end (minus the length of the ends)

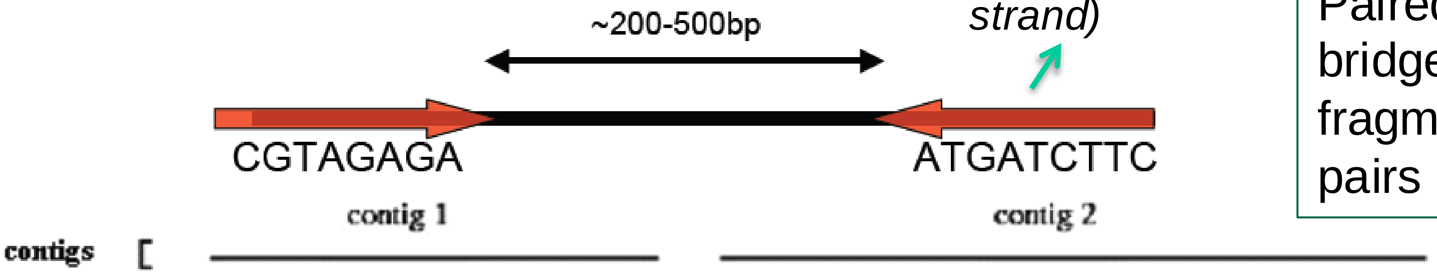
Mate-pair
Like paired-end reads, each "read" is two sequences from each end of the same DNA fragment, but the distance between the reads on the original genome sequence is much longer (3000-10000 bp) than a single read (200-800 bp). To get these longer “in-between” fragments, the trick here is that the DNA fragment has been engineered from a circularization process.

For reference

Paired-end versus Mate Pair

Paired-end strategy

Reverse compliment (of complementary strand)

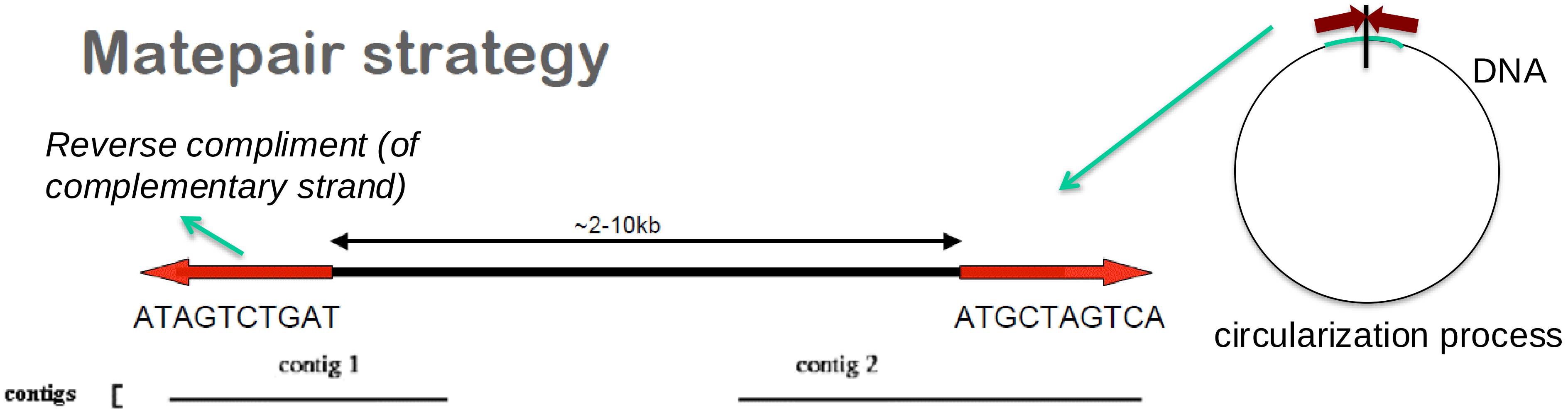


NGS fragments (see next slide) are shorter than Sanger fragments

Paired-end reads bridge shorter fragments than mate pairs

Matepair strategy

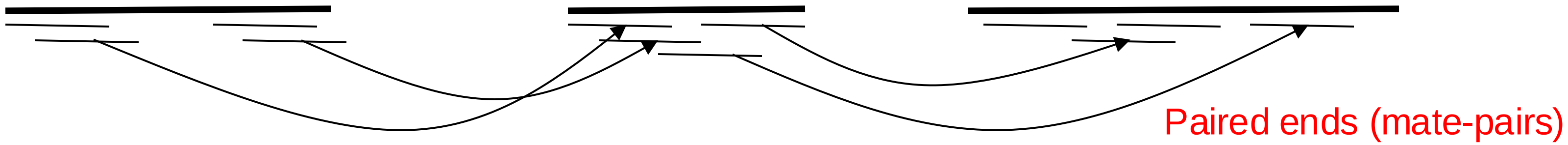
Reverse compliment (of complementary strand)



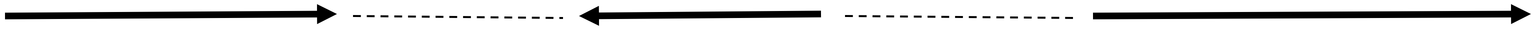
circularization process

Combining (linking) contigs with paired-end reads

- Align paired-end reads to contigs
- Combining contigs (into *scaffolds* - so contig order is known), if linked by paired ends, may be conditional on:
  - number of bridges
  - alignment accuracy
  - uniqueness of alignment (information content).

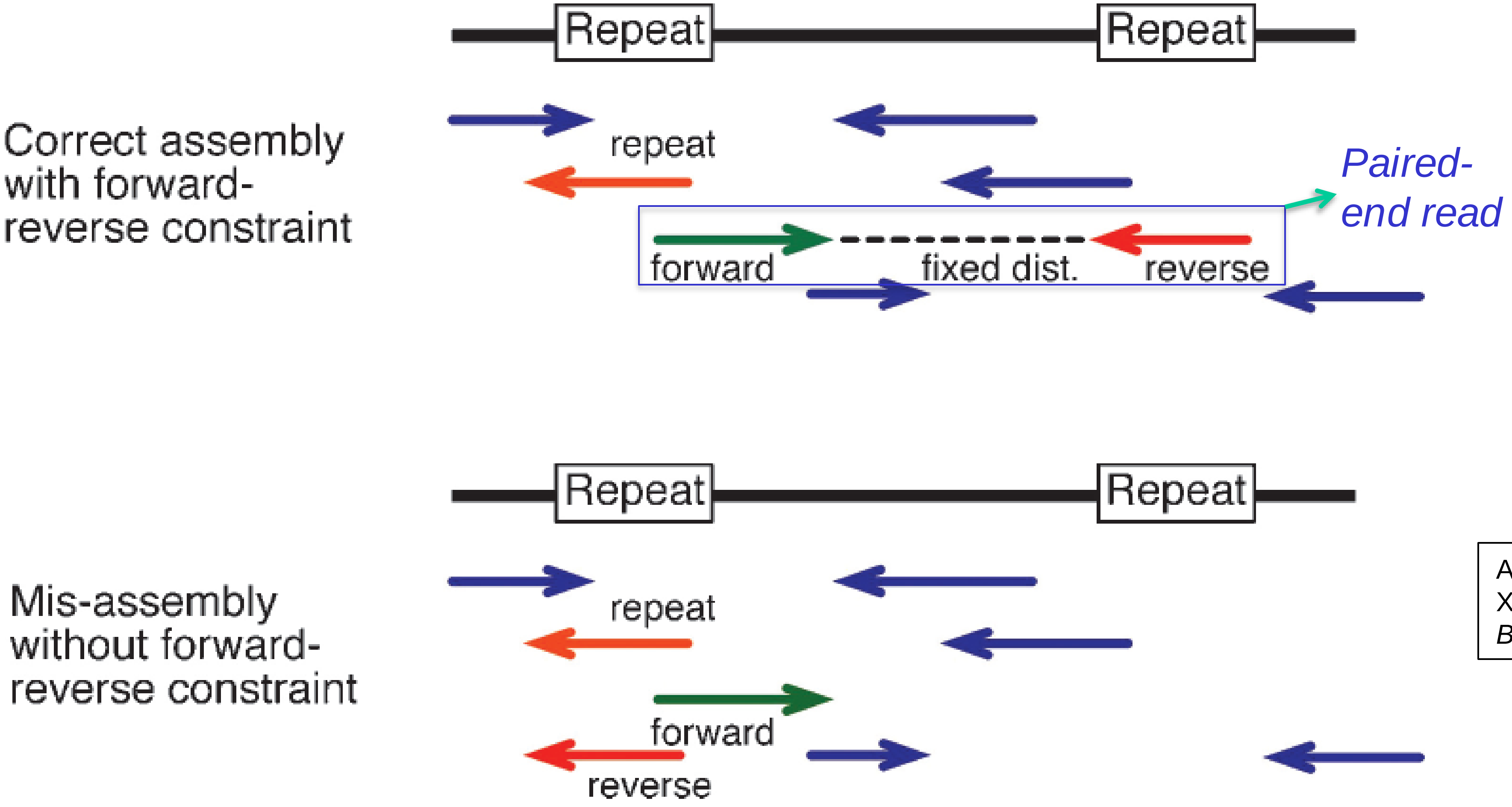


- Use paired-end insert size to estimate gap between the contigs



In addition to linking contigs, Paired-end reads can also reveal repeating fragments

The reverse looks like the repeat, and it can be aligned to the 1st or 2nd repeat



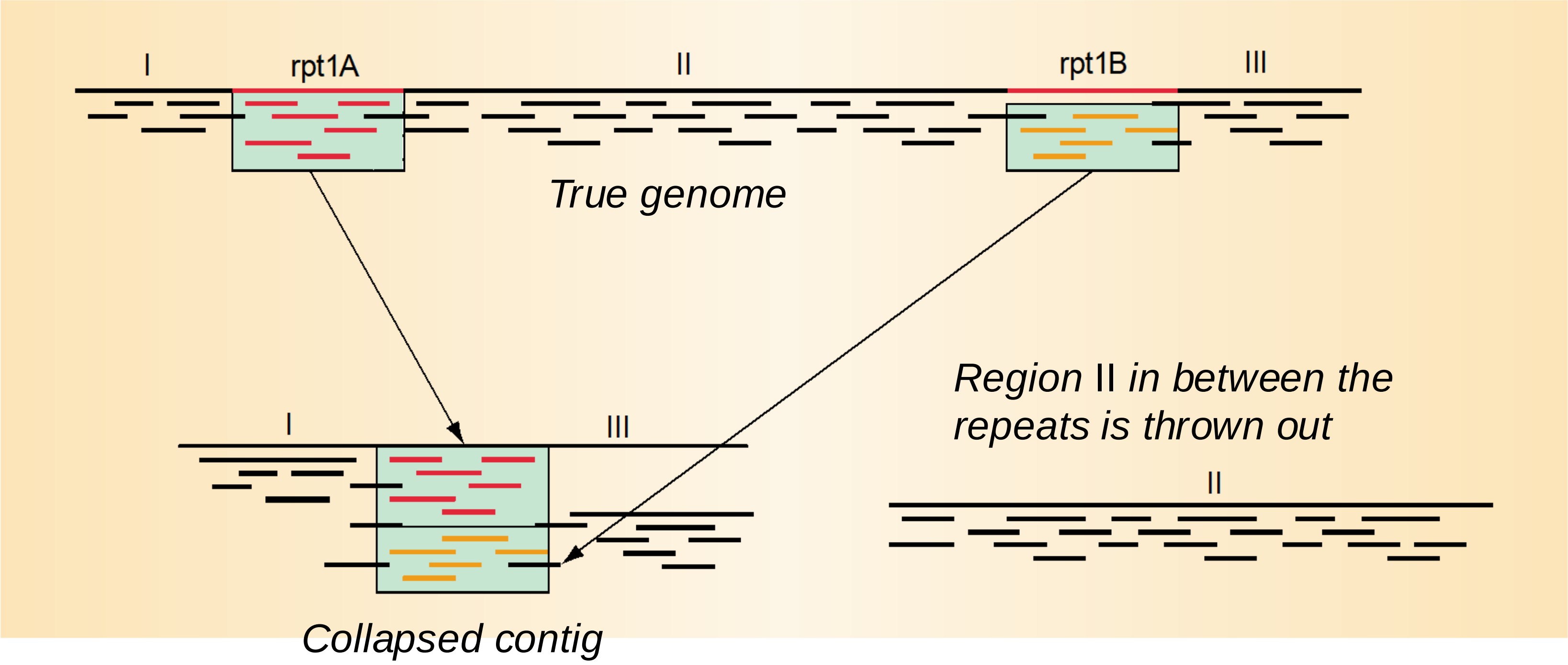
Repeat
Repeat
Correct assembly with forward-reverse constraint
repeat
Paired-end read
forward
fixed dist.
reverse
Repeat
Repeat
Mis-assembly without forward-reverse constraint
repeat
forward
reverse

Adapted from Xiong, *Essential Bioinformatics*

Example of sequence assembly with or without applying forward–reverse constraint (mate-pairs or paired-end reads), which fixes the sequence distance from both ends of a subclone. Without the restraint, the red fragment is misassembled due to matches of repetitive element in the middle of a fragment.

Problem with repetitive elements without paired end reads

- Repeats can cause major problems to the assembler;
  - Reads corresponding to two separate repeats may be collapsed in a single contig
  - Even with mate pairs, however, repeats with large intervening regions or multiple repeat regions (e.g. >600 repeats) cannot be resolved anymore



Carrying out *de novo* assembly

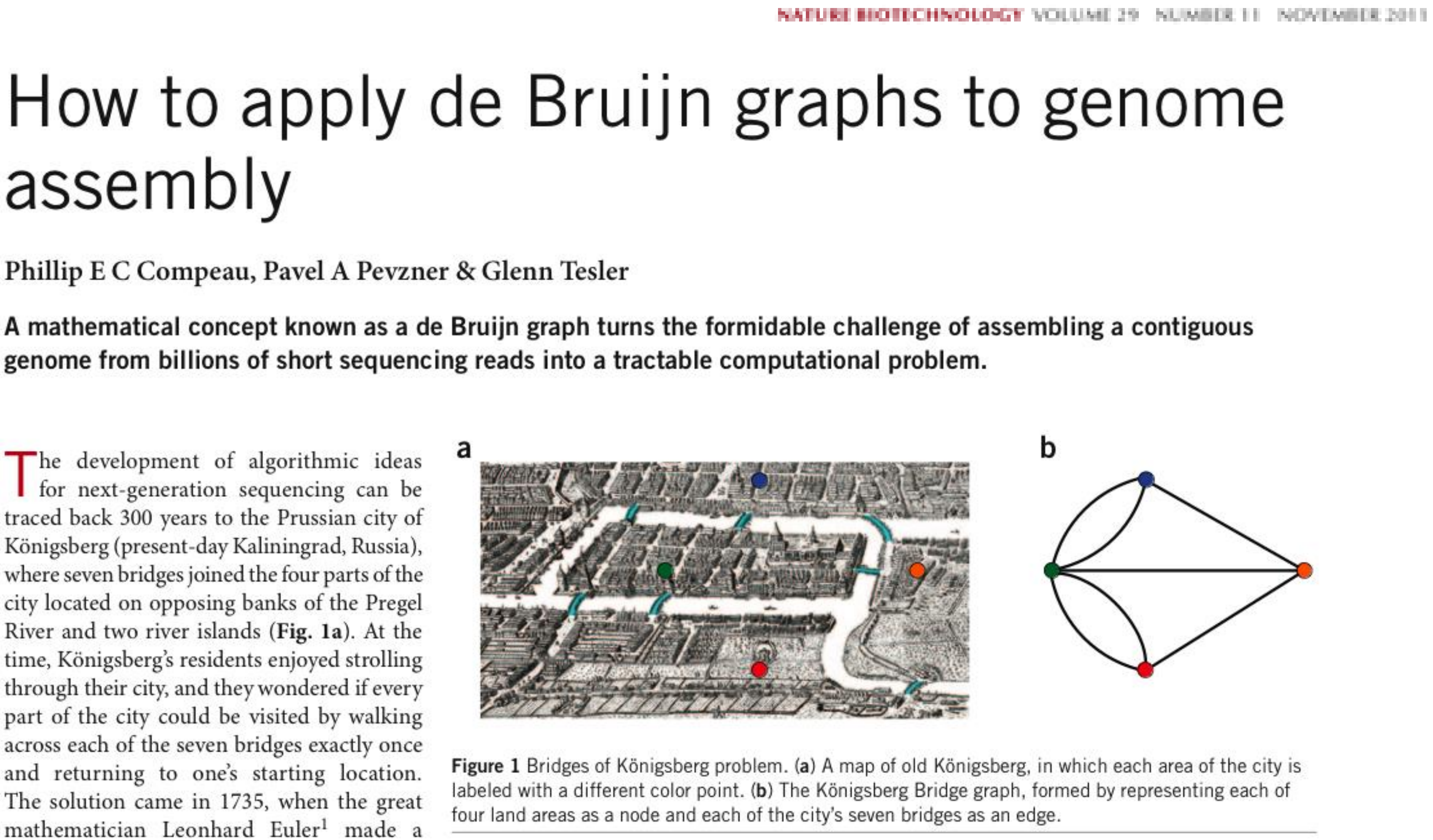
- We need an **algorithm** to construct the most likely DNA sequence (contigs) given the reads

- INPUT
  - Millions of sequenced fragments

- PROCESS
  - Cut reads in k-mers and determine overlap through string-matching (allow for small variations)
  - No reference needed

- OUTPUT
  - Alignment and sequence of new strain

For reference



NATURE BIOTECHNOLOGY VOLUME 29 NUMBER 11 NOVEMBER 2011

How to apply de Bruijn graphs to genome assembly

Phillip E C Compeau, Pavel A Pevzner & Glenn Tesler

A mathematical concept known as a de Bruijn graph turns the formidable challenge of assembling a contiguous genome from billions of short sequencing reads into a tractable computational problem.

The development of algorithmic ideas for next-generation sequencing can be traced back 300 years to the Prussian city of Königsberg (present-day Kaliningrad, Russia), where seven bridges joined the four parts of the city located on opposing banks of the Pregel River and two river islands (**Fig. 1a**). At the time, Königsberg's residents enjoyed strolling through their city, and they wondered if every part of the city could be visited by walking across each of the seven bridges exactly once and returning to one's starting location. The solution came in 1735, when the great mathematician Leonhard Euler[1] made a

Figure 1 Bridges of Königsberg problem. (**a**) A map of old Königsberg, in which each area of the city is labeled with a different color point. (**b**) The Königsberg Bridge graph, formed by representing each of four land areas as a node and each of the city's seven bridges as an edge.

Nature Biotechnology **29**, 987–991 (2011) doi:10.1038/nbt.2023

For reference

Königsberg Problem:

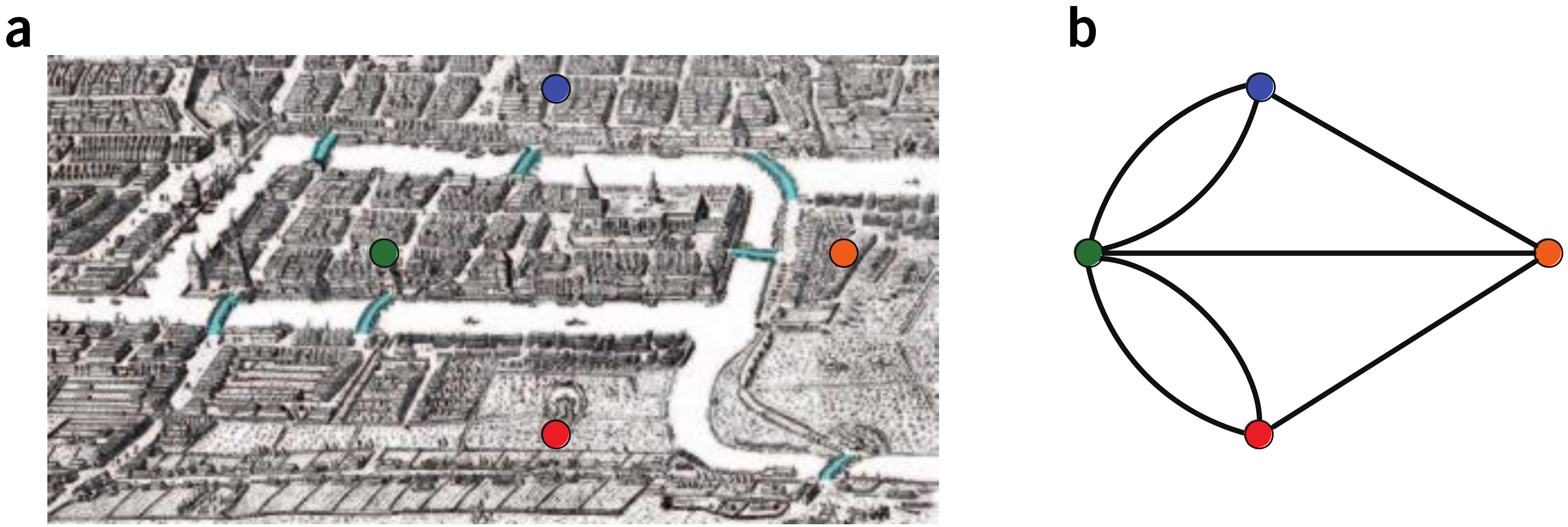


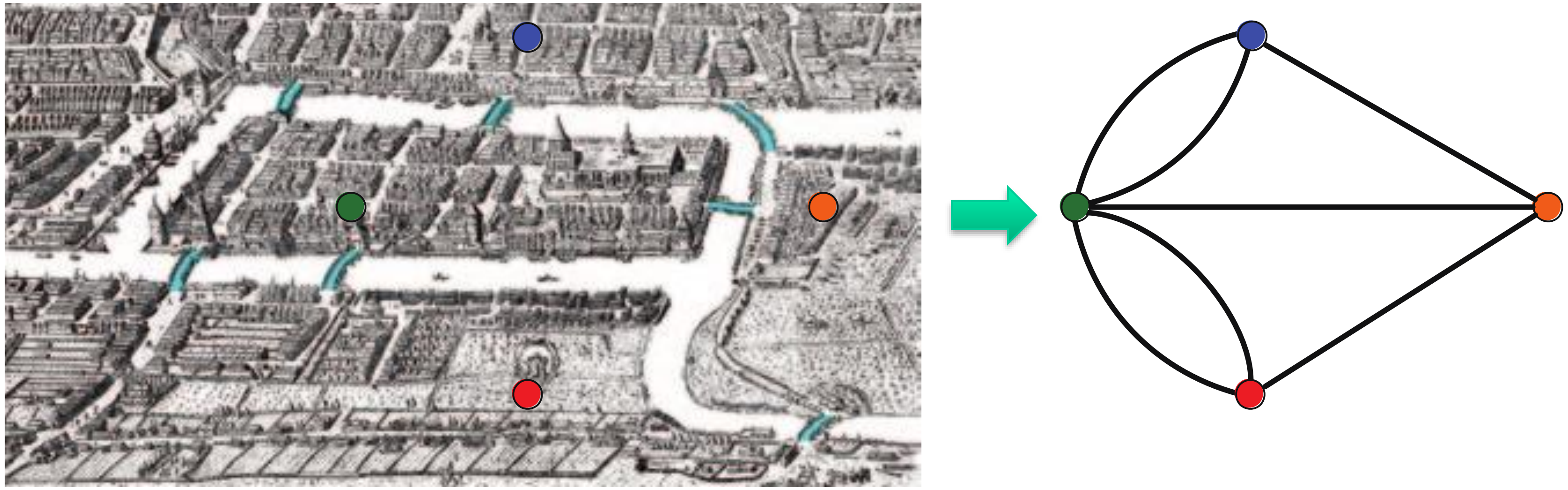
Figure 1 Bridges of Königsberg problem. (**a**) A map of old Königsberg, in which each area of the city is labeled with a different color point. (**b**) The Königsberg Bridge graph, formed by representing each of four land areas as a node and each of the city's seven bridges as an edge.

Euler's problem:
Can we visit every bridge (edge) once ending in the same place (making a cycle)?

For reference

Euler Cycle

- A cycle that visits every edge exactly once

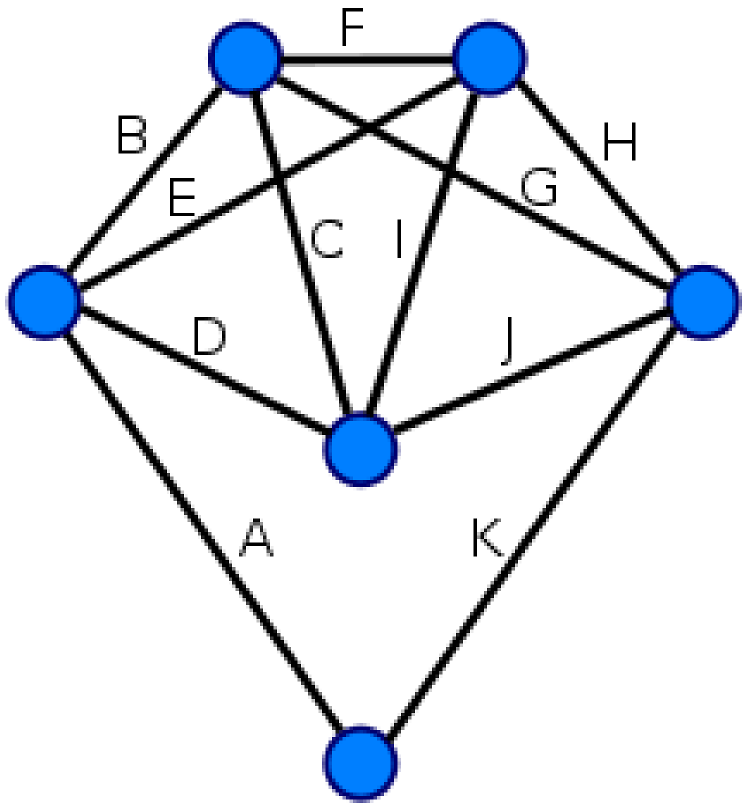


- Is this possible in Königsberg?
- What are the conditions for a graph to contain an Eulerian cycle? (think about the degree of a node)
  - The degree of a node (vertex) is the number of incoming/outgoing edges of that node

For reference

Finding the Eulerian Path

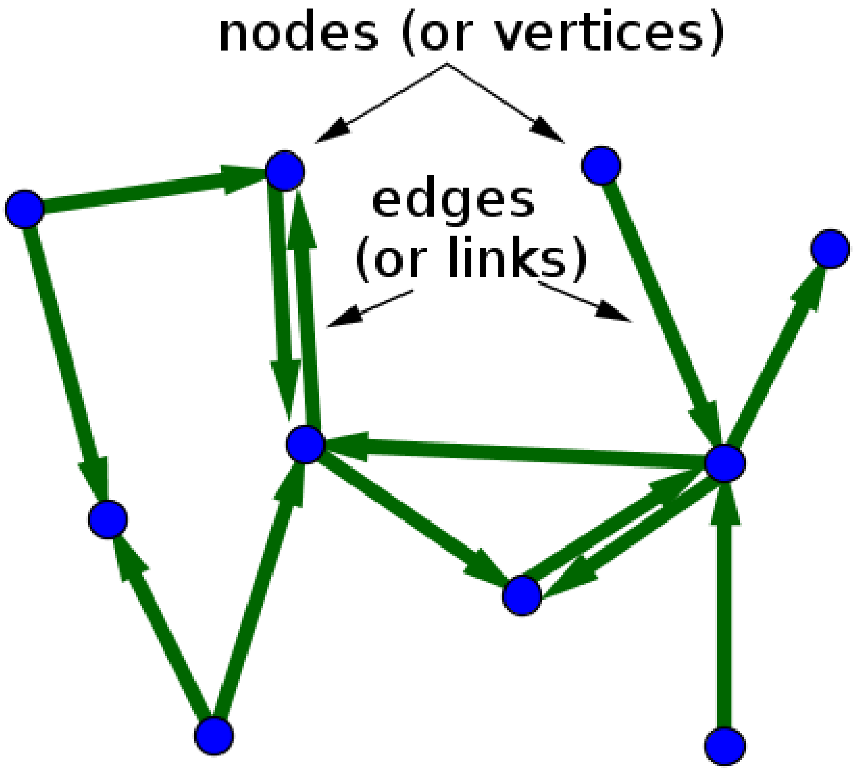
```
1. Start from any node
2. Remove edge after visit
3. when back at original
   node
   if no edges left:
         terminate
   else
         start at 1
4. Join all cycles
   together
```



Does this give you a single solution?
What about Eulerian cycles in directed graphs?

For reference

Directed Graphs



K-mers

- K-mers in assembly algorithms, are used in many bioinformatics algorithms optimised for speed, e.g. “words” in the widely-used BLAST suite (later lecture)

- Example:
  - ACTGTTA contains the 4-mers:
    - ACTG
    - CTGT
    - TGTT
    - GTTA

De Bruijn Graphs for Genome Assembly

- Nodes are k-mers

- Edges are (k+1)-mers, connecting two nodes

- Such nodes need to have an overlap of (k-1) symbols

- Directed edges, such that the suffix of the outgoing node overlaps with the prefix of the incoming node

$$\boxed{\mathrm{ATG} \xrightarrow{\mathrm{ATGT}} \mathrm{TGT}}$$

- connecting 4-mer: ATGT

- Overlap: TG

$$\mathrm{aTG} \longrightarrow \mathrm{TGt}$$

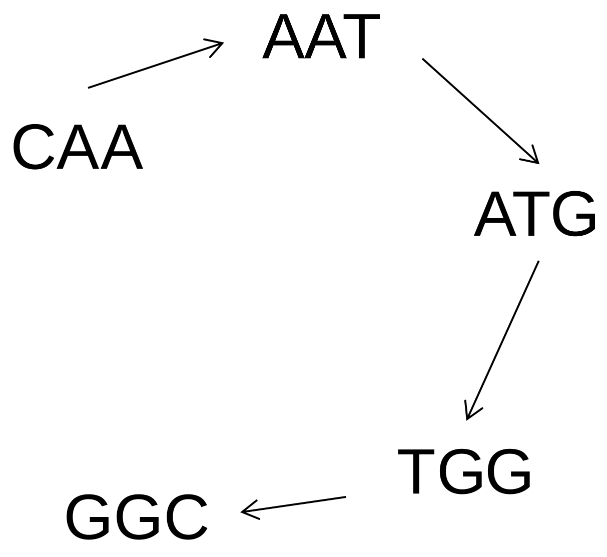
- TG is the suffix of aTG
- TG is the prefix of TGt

Reads connect nodes

- Given a read

CAATGGC

- 3-mers can be connected if connecting 4-mers are present in the read



CAA

AAT

ATG

TGG

GGC

- What happens if we add:

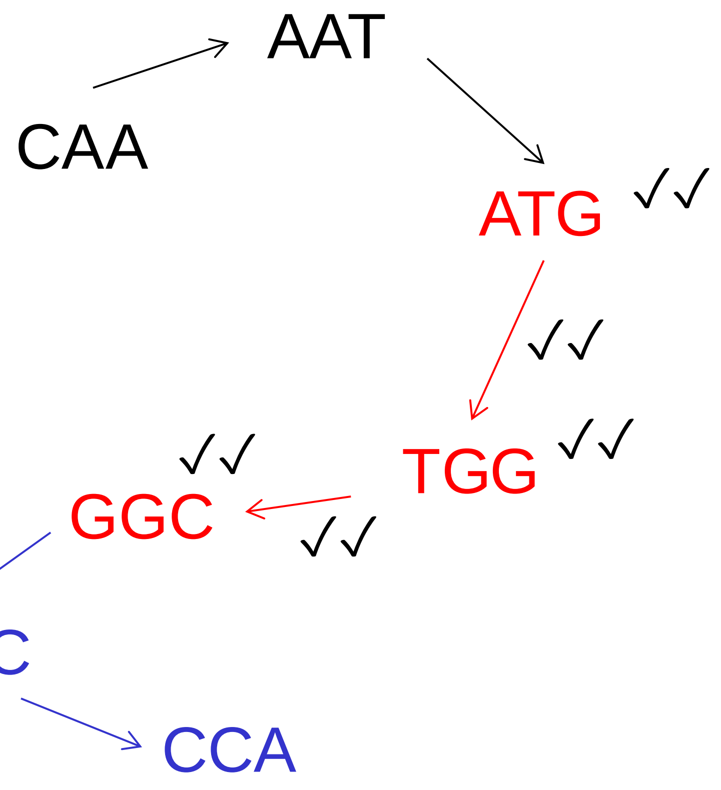
ATGGCGA

Reads connect nodes

- Given a read

CAATGGC

- 3-mers can be connected if connecting 4-mers are present in the read



AAT

CAA

ATG ✓✓

✓✓

TGG ✓✓

✓✓

GGC

✓✓

GCC

CCA

- What happens if we add:

ATGGCCA

De Bruijn graph algorithm

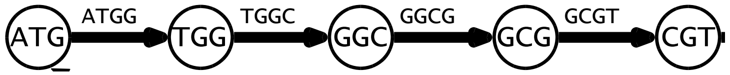
- Build the *k*-mer chain for each read

- Build *k*-mer graph: Before you add a node to the graph, check each time whether a node has been seen before
  - If so, add 1 to number of visits for the node
  - If not, introduce the new node

- Traverse graph to get the optimal genome sequence

Tallying how often each node and edge is traversed (used) will help decide about the optimal path

Constructing “de Bruijn” graphs ** - paper exercise



1) Given de read `ATGGCGT`, we can draw the 3-mer graph above. Finish the graph for the following reads

```
ATGGCGT
GGCGTGC
CAATGGC
CGTGCAA
TGCAATG
```

1) Indicate in the graph how often each edge and node occurs.

2) What was the genome? Are there any alternatives?

4) Add the following read – with a mismatch – to the graph

GG**A**GTGC

4) Given the new graph, how could you deduce what the actual sequence was?

Home exercise..

Reconstructed genome (5 alternatives in red)

CAATGGC

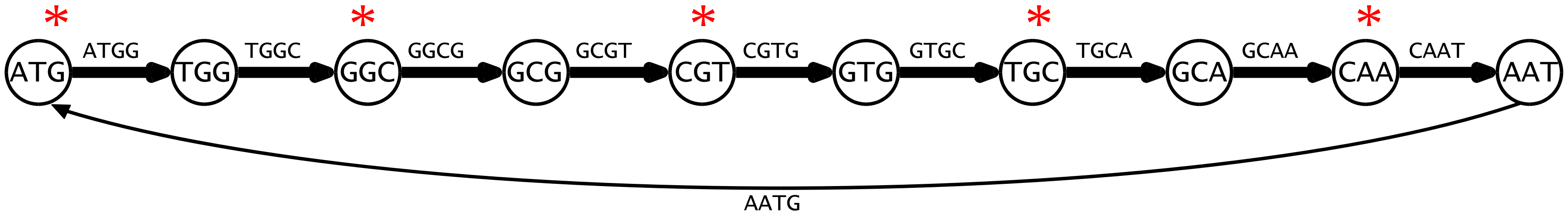
ATGGCGT

GGCGTGC

CGTGCAA

TGCAATG

Since the graph is circular in this case, the reconstructed genome may start at each of the five read starting 3-mers (i.e. red asterisks)



```
ATGGCGT
  GGCGTGC
    CGTGCAA
      TGCAATG
        CAATGGC
ATGGCGTGCAATGGC
```

```
GGCGTGC
  CGTGCAA
    TGCAATG
      CAATGGC
        ATGGCGT
GGCGTGCAATGGCGT
```

```
CGTGCAA
  TGCAATG
    CAATGGC
      ATGGCGT
        GGCGTGC
CGTGCAATGGCGTGC
```

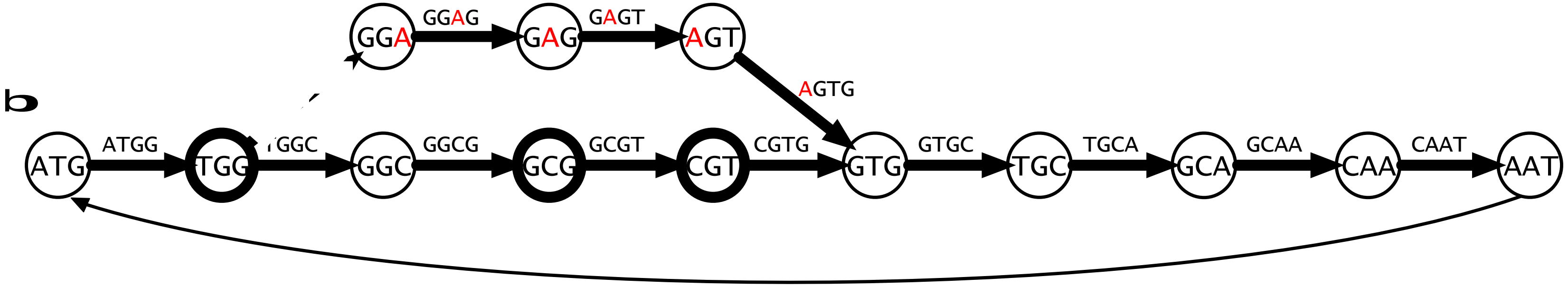
```
TGCAATG
  CAATGGC
    ATGGCGT
      GGCGTGC
        CGTGCAA
TGCAATCGCGTGCAA
```

```
CAATGGC
  ATGGCGT
    GGCGTGC
      CGTGCAA
        TGCAATG
CAATCGCGTGCAATG
```

Home exercise

Can you (re)draw the graph if we add another read with a mismatch

CAATGGC	CGTGCAA
ATGGCGT	TGCAATG
GGCGTGC	GGAGTGC



Given this graph:
how would we decide what the actual sequence was?

Home exercise..

Same reads, 2-mer graph

- Same solution as with 3-mers?

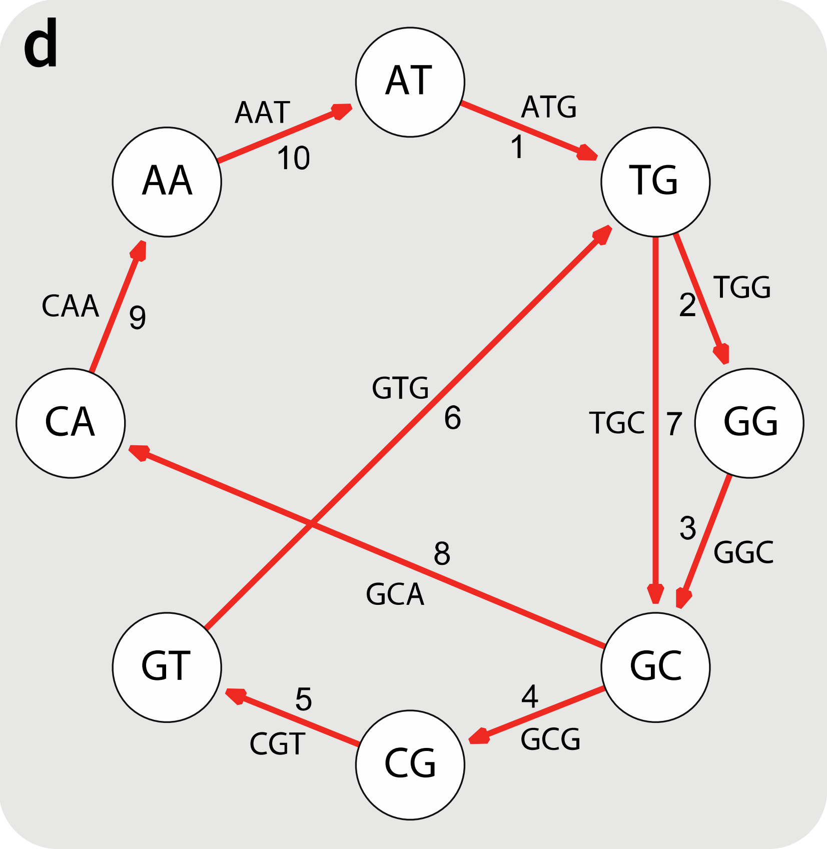
CAATGGC

ATGGCGT

GGCGTGC

CGTGCAA

TGCAATG



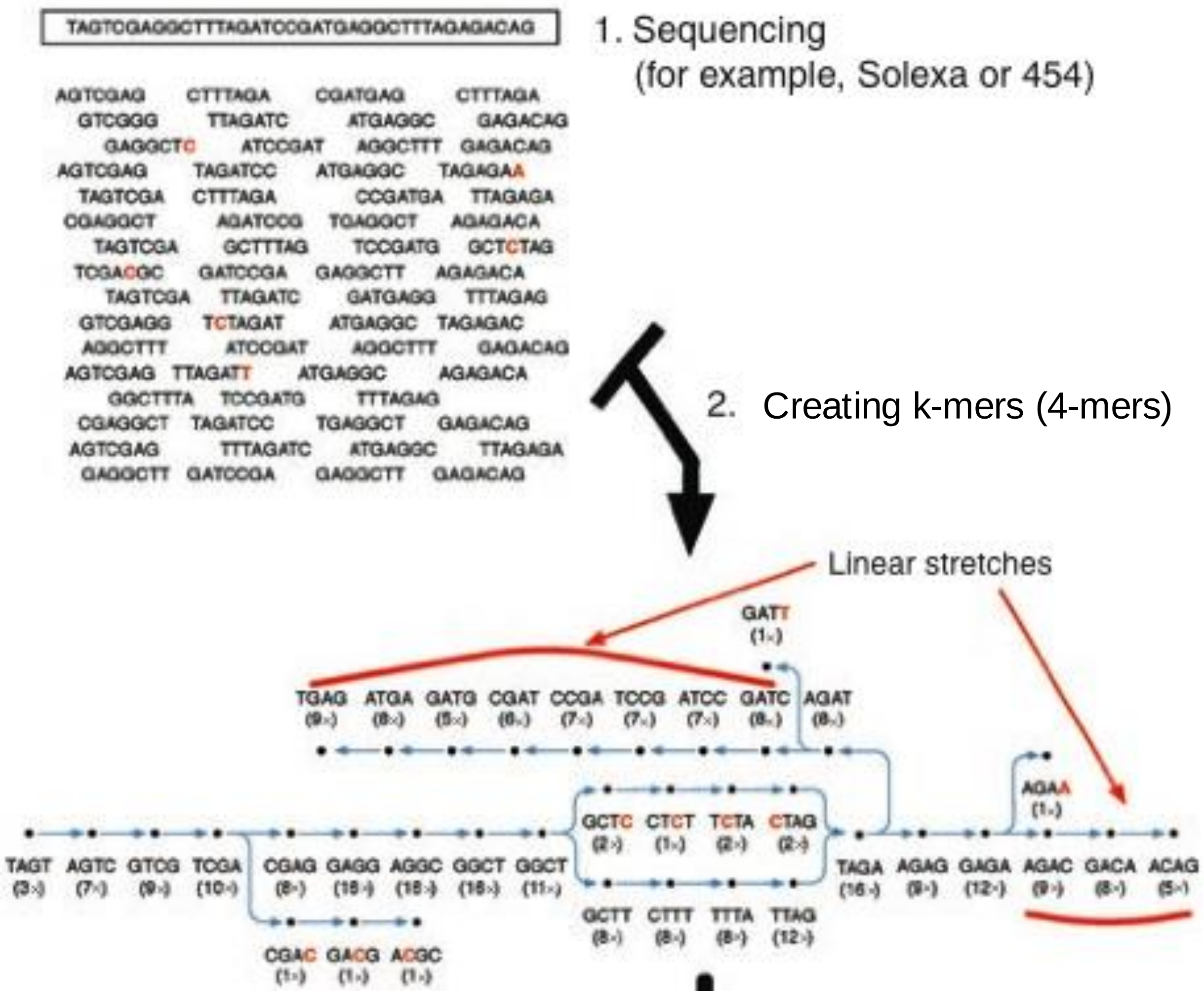
Eulerian cycle
Visit each edge once

Building the 2-mer graph for the above 5 reads:

- What is the genome?
- Is the solution unique?

Home exercise..

De Bruijn Graphs for larger read sets

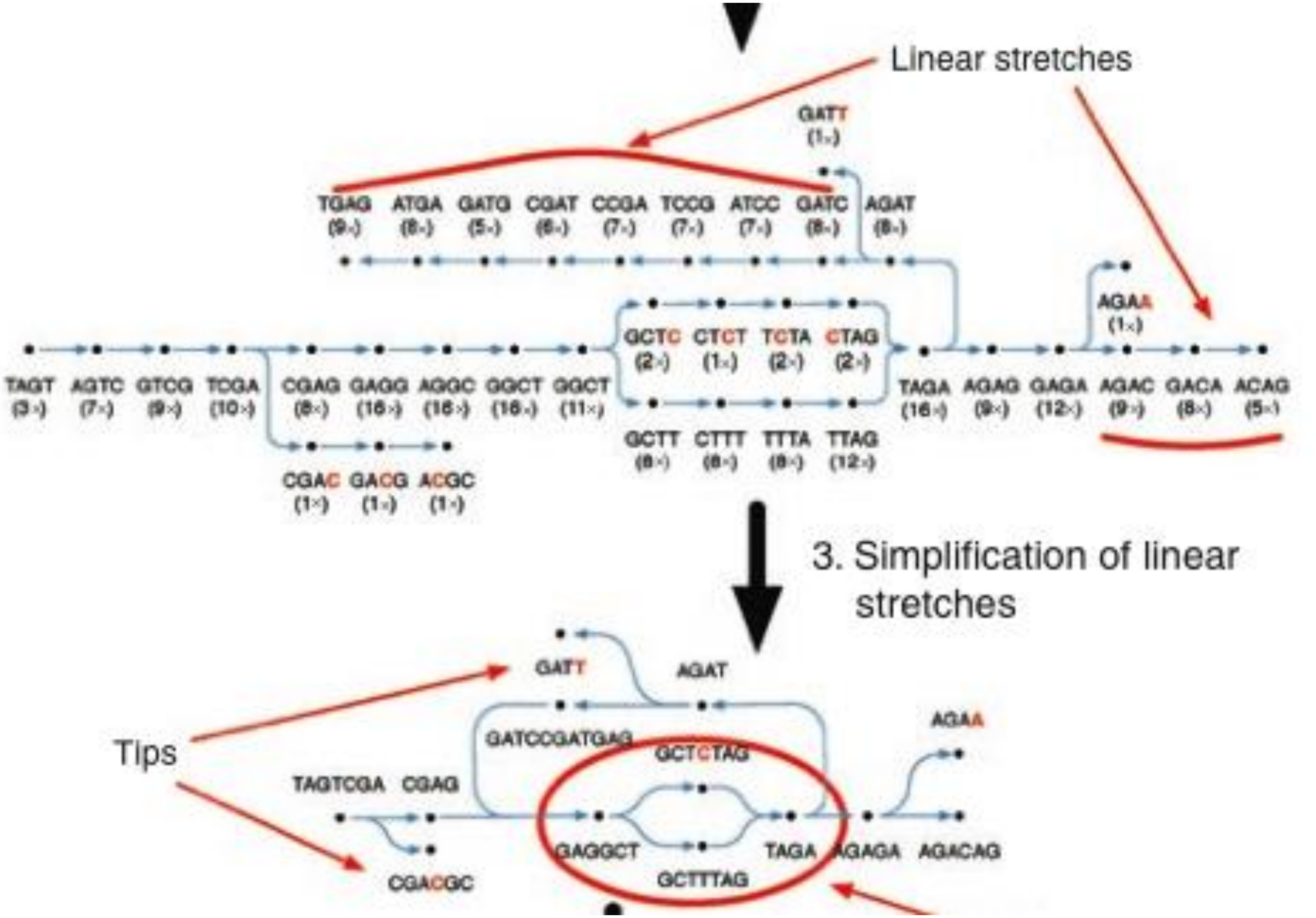

TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG
AGTCGAG CTTTAGA CGATGAG CTTTAGA
GTCGGG TTAGATC ATGAGGC GAGACAG
GAGGCTC ATCCGAT AGGCTTT GAGACAG
AGTCGAG TAGATCC ATGAGGC TAGAGAA
TAGTCGA CTTTAGA CCGATGA TTAGAGA
CGAGGCT AGATCCG TGAGGCT AGAGACA
TAGTCGA GCTTTAG TCCGATG GCTCTAG
TCGACGC GATCCGA GAGGCTT AGAGACA
TAGTCGA TTAGATC GATGAGG TTTAGAG
GTCGAGG TCTAGAT ATGAGGC TAGAGAC
AGGCTTT ATCCGAT AGGCTTT GAGACAG
AGTCGAG TTAGATT ATGAGGC AGAGACA
GGCTTTA TCCGATG TTTAGAG
CGAGGCT TAGATCC TGAGGCT GAGACAG
AGTCGAG TTTAGATC ATGAGGC TTAGAGA
GAGGCTT GATCCGA GAGGCTT GAGACAG
GATT (1×)
TGAG (9×)
ATGA (8×)
GATG (5×)
CGAT (6×)
CCGA (7×)
TCCG (7×)
ATCC (7×)
GATC (8×)
AGAT (8×)
AGAA (1×)
GCTC (2×)
CTCT (1×)
TCTA (2×)
CTAG (2×)
TAGT (3×)
AGTC (7×)
GTCG (9×)
TCGA (10×)
CGAG (8×)
GAGG (18×)
AGGC (18×)
GGCT (18×)
GGCT (11×)
TAGA (16×)
AGAG (9×)
GAGA (12×)
AGAC (9×)
GACA (8×)
ACAG (5×)
GCTT (8×)
CTTT (8×)
TTTA (8×)
TTAG (12×)
CGAC (1×)
GACG (1×)
ACGC (1×)

1. Sequencing
(for example, Solexa or 454)

2. Creating k-mers (4-mers)

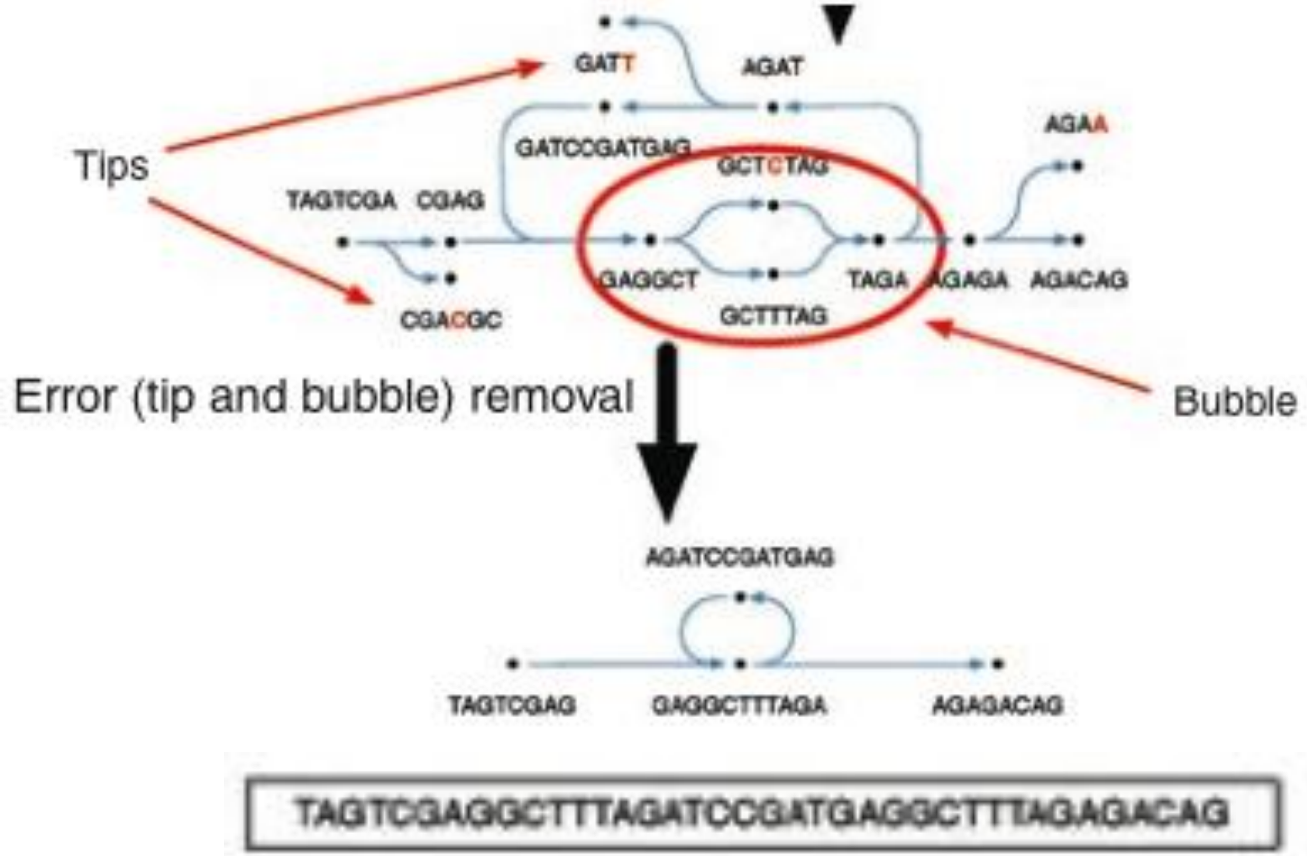
Linear stretches

Linear stretches:
continuous sequence

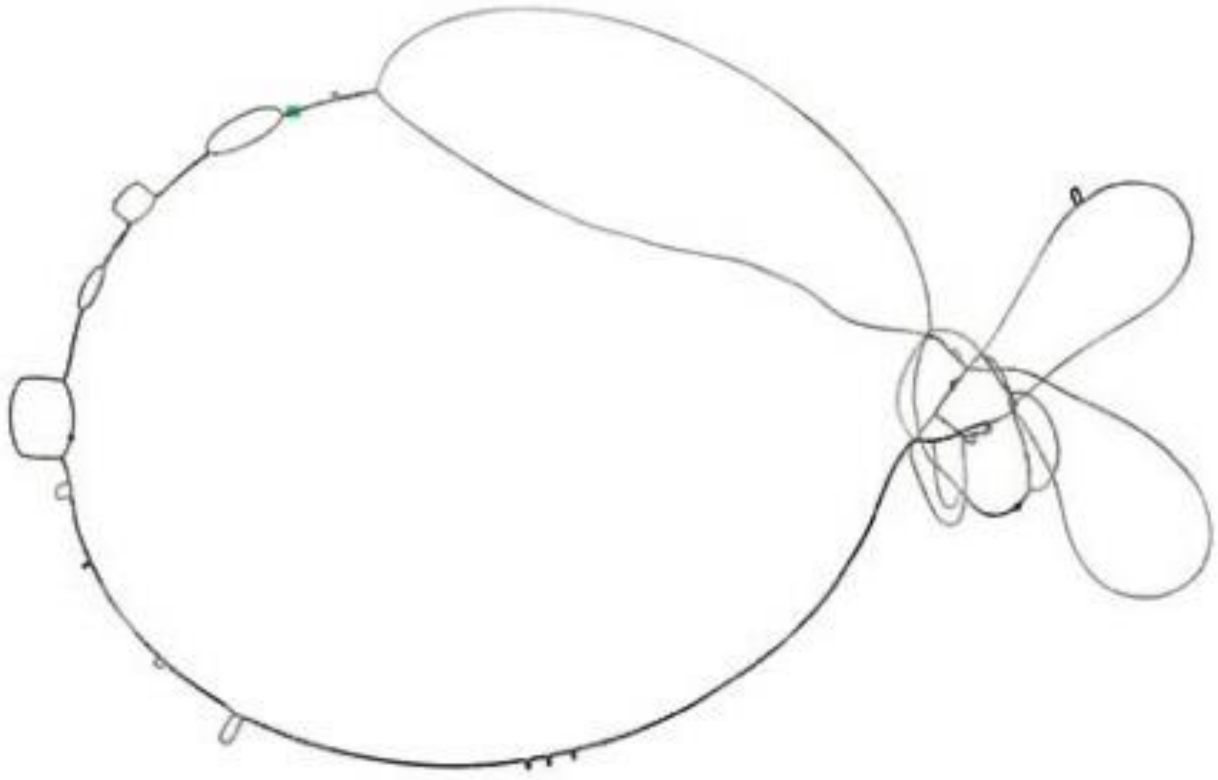


Tip & Bubble removal

tip: just delete the tip
bubble: keep the path with more coverage or some other metric



Full de Bruijn Graph



A full de Bruijn graph of two related plasmids that have a locus in common. The de Bruijn graph was created with 30-bp k-mers. The open loops are regions that differ between the two plasmids, whereas the heavier lines indicate common regions.

Sequencing if a **reference genome** is available

a lucky thing

- INPUT
  - – Millions of sequenced fragments
  - – A reference genome sequence

- PROCESS
  - – String matching of the sequence reads against the reference genome sequence (allowing for small variations)
  - – Reference and sequenced organism need to be closely related (at least the same species)

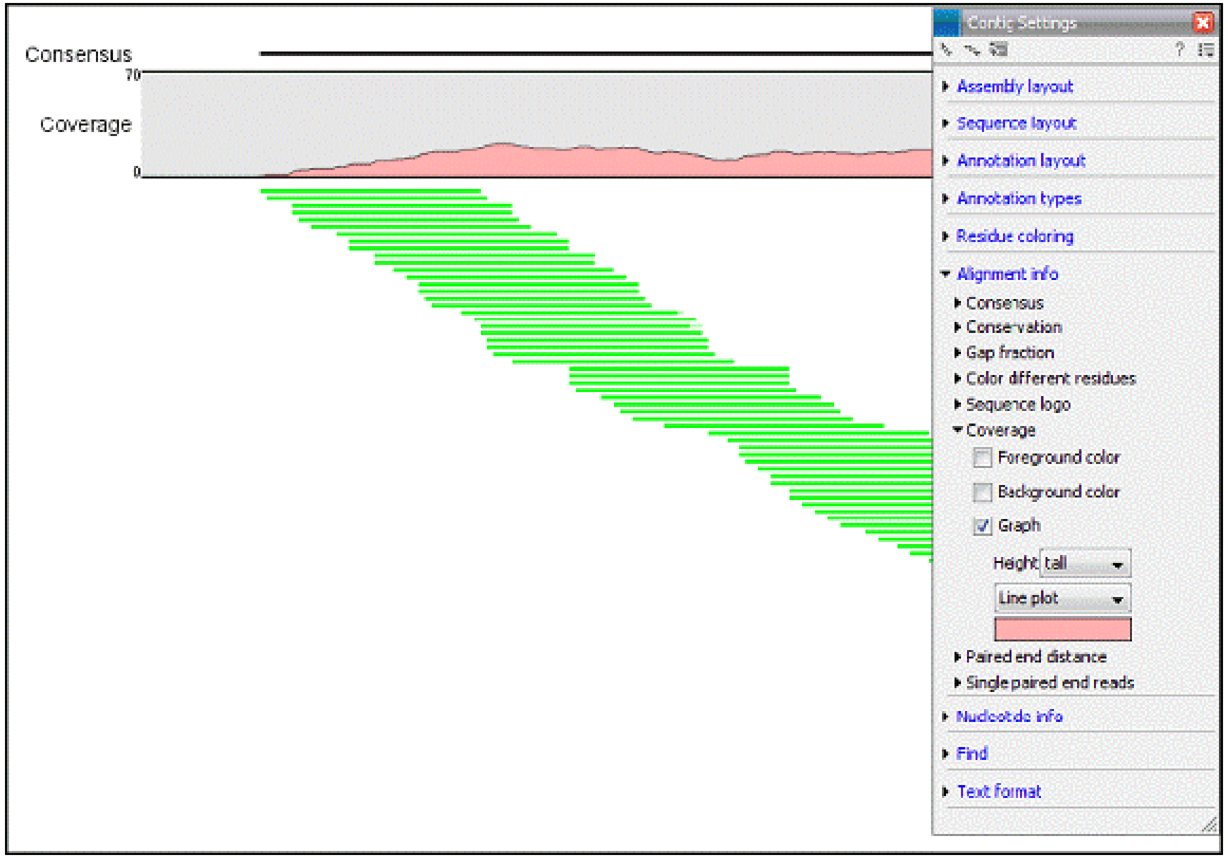
- OUTPUT
  - – Alignment and sequence of newly sequenced genome

Reference alignment

Match sequence to a given genome:

-Same species different individual

-Closely related species



□ Foreground color

□ Background color

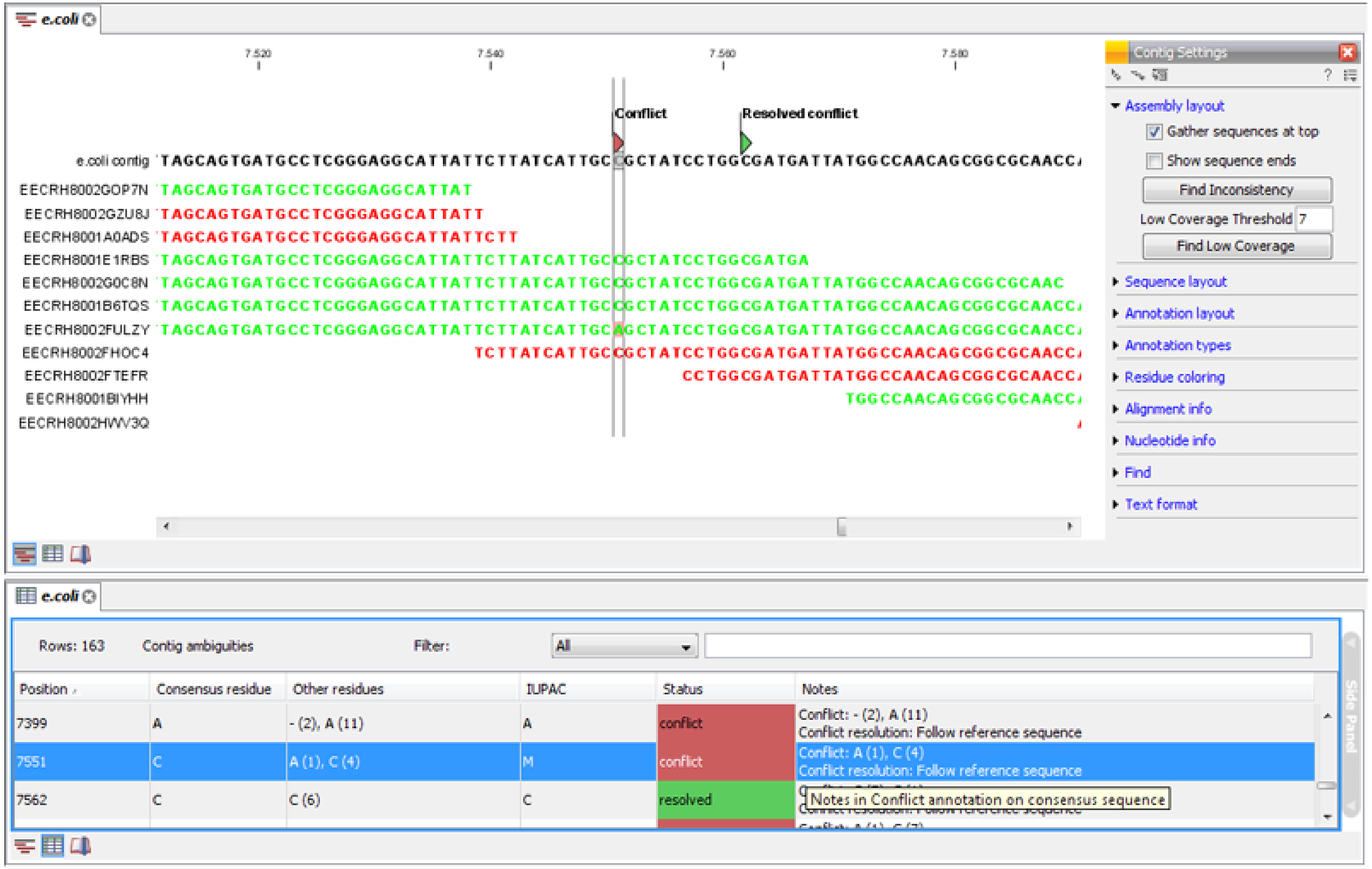
☑ Graph

Height tall

Line plot

Reference alignment

Match sequence to a ‘similar’ genome



```
e.coli contig   TAGCAGTGATGCCTCGGGAGGCATTATTCTTATCATTGCCGCTATCCTGGCGATGATTATGGCCAACAGCGGCGCAACC
EECRH8002GOP7N  TAGCAGTGATGCCTCGGGAGGCATTAT
EECRH8002GZU8J  TAGCAGTGATGCCTCGGGAGGCATTATT
EECRH8001A0ADS  TAGCAGTGATGCCTCGGGAGGCATTATTCTT
EECRH8001E1RBS  TAGCAGTGATGCCTCGGGAGGCATTATTCTTATCATTGCCGCTATCCTGGCGATGA
EECRH8002G0C8N  TAGCAGTGATGCCTCGGGAGGCATTATTCTTATCATTGCCGCTATCCTGGCGATGATTATGGCCAACAGCGGCGCAAC
EECRH8001B6TQS  TAGCAGTGATGCCTCGGGAGGCATTATTCTTATCATTGCCGCTATCCTGGCGATGATTATGGCCAACAGCGGCGCAACC
EECRH8002FULZY  TAGCAGTGATGCCTCGGGAGGCATTATTCTTATCATTGCAGCTATCCTGGCGATGATTATGGCCAACAGCGGCGCAACC
EECRH8002FHOC4                             TCTTATCATTGCCGCTATCCTGGCGATGATTATGGCCAACAGCGGCGCAACC
EECRH8002FTEFR                                               CCTGGCGATGATTATGGCCAACAGCGGCGCAACC
EECRH8001BIYHH                                                             TGGCCAACAGCGGCGCAACC
EECRH8002HWV3Q
```

☑ Gather sequences at top

☐ Show sequence ends

Find Inconsistency

Low Coverage Threshold 7

Find Low Coverage

Rows: 163 Contig ambiguities Filter: All

Position	Consensus residue	Other residues	IUPAC	Status	Notes
7399	A	- (2), A (11)	A	conflict	Conflict: - (2), A (11) Conflict resolution: Follow reference sequence
7551	C	A (1), C (4)	M	conflict	Conflict: A (1), C (4) Conflict resolution: Follow reference sequence
7562	C	C (6)	C	resolved	Notes in Conflict annotation on consensus sequence

How to search through millions of reads?

- **Input:** reference genome and a set of reads (BAM file)
- Need fast way to look up reads that will potentially align well
- Most current methods use “Burrows-Wheeler transform”
  - Also used in data compression (like “zipping”)
  - Here it helps aligning the reads against the genome in a fast way

Represent a string into its BWT. The string can be reconstructed from the BWT.

Reference-based sequencing

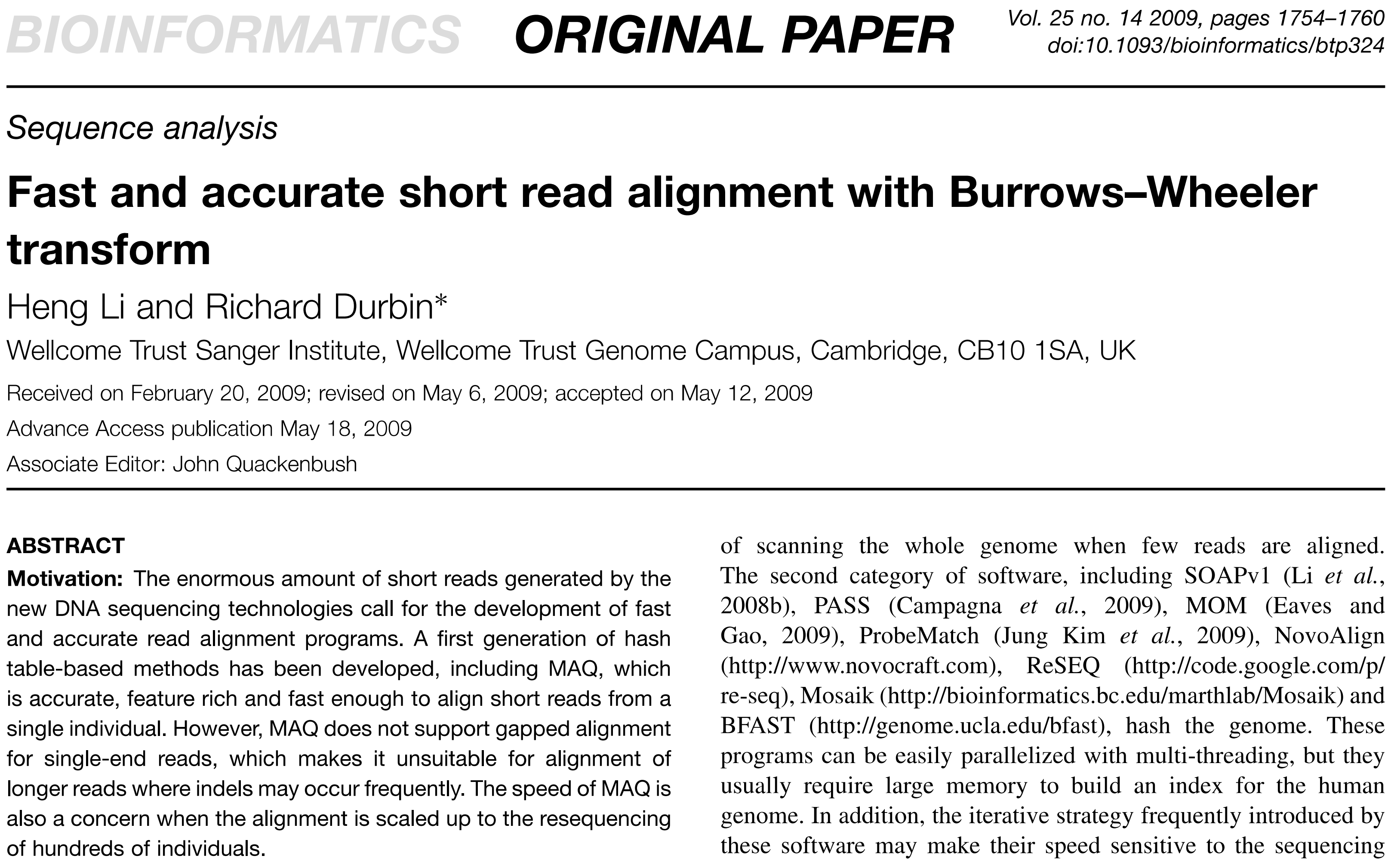
The problem of mismatches

- The reference genome will contain mismatches relative to the reads that should be aligned against it.
- There are various strategies to deal with matching fragments containing mismatches.
- However, compared to exact string matching, looking for alignments where symbols may differ will increase processing times.

The BWA can handle mismatches at the cost of longer processing time

Burrows-Wheeler Aligner (BWA)

Li & Durbin, 2009 - doi:10.1093/bioinformatics/btp324



BIOINFORMATICS ORIGINAL PAPER

Vol. 25 no. 14 2009, pages 1754–1760
doi:10.1093/bioinformatics/btp324

Sequence analysis

Fast and accurate short read alignment with Burrows–Wheeler transform

Heng Li and Richard Durbin*

Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Cambridge, CB10 1SA, UK

Received on February 20, 2009; revised on May 6, 2009; accepted on May 12, 2009

Advance Access publication May 18, 2009

Associate Editor: John Quackenbush

ABSTRACT

Motivation: The enormous amount of short reads generated by the new DNA sequencing technologies call for the development of fast and accurate read alignment programs. A first generation of hash table-based methods has been developed, including MAQ, which is accurate, feature rich and fast enough to align short reads from a single individual. However, MAQ does not support gapped alignment for single-end reads, which makes it unsuitable for alignment of longer reads where indels may occur frequently. The speed of MAQ is also a concern when the alignment is scaled up to the resequencing of hundreds of individuals.

of scanning the whole genome when few reads are aligned. The second category of software, including SOAPv1 (Li *et al.*, 2008b), PASS (Campagna *et al.*, 2009), MOM (Eaves and Gao, 2009), ProbeMatch (Jung Kim *et al.*, 2009), NovoAlign (http://www.novocraft.com), ReSEQ (http://code.google.com/p/re-seq), Mosaik (http://bioinformatics.bc.edu/marthlab/Mosaik) and BFAST (http://genome.ucla.edu/bfast), hash the genome. These programs can be easily parallelized with multi-threading, but they usually require large memory to build an index for the human genome. In addition, the iterative strategy frequently introduced by these software may make their speed sensitive to the sequencing

Burrows-Wheeler Aligner (BWA)

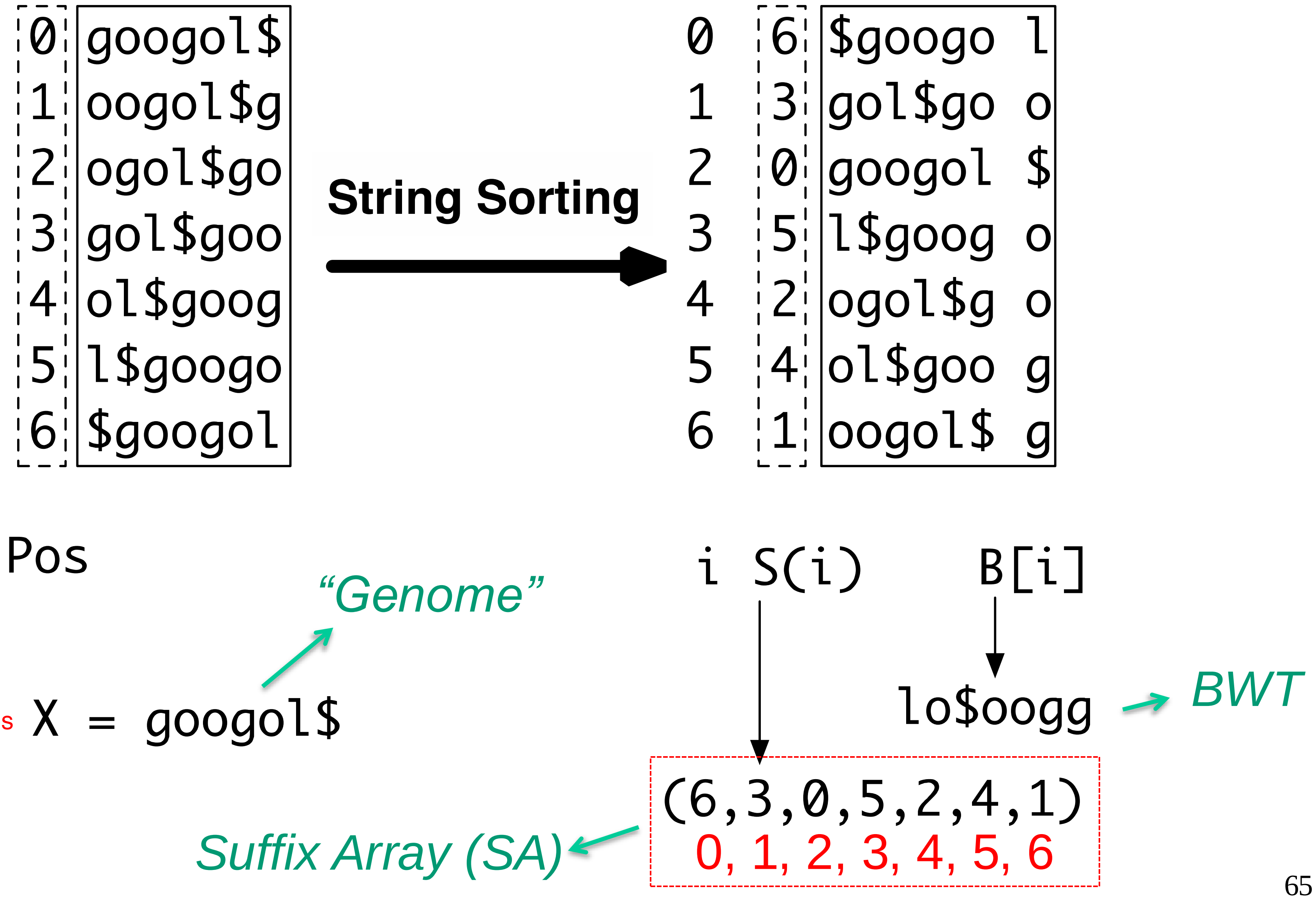
Li & Durbin, 2009 - doi:10.1093/bioinformatics/btp324

“Fast and accurate short read alignment with Burrows–Wheeler transform”

- Fast aligner
    - Based on **indexing** technique using **suffix tree formalism**
- Can handle inexact repeats with a defined maximum number of differences (mismatches or gaps)
- Li & Durbin (2009) describe BWA principle, definitions, algorithm and example
    - BWA is widely used: BWA paper is cited 47360 times (Nov 2024)

Burrows-Wheeler Transform (BWT)

suffix+prefix of the suffix
- i.e. full string googol$
suffix: oogol$, prefix g



Pos	
0	googol$
1	oogol$g
2	ogol$go
3	gol$goo
4	ol$goog
5	l$googo
6	$googol

i	S(i)		B[i]
0	6	$googo	l
1	3	gol$go	o
2	0	googol	$
3	5	l$goog	o
4	2	ogol$g	o
5	4	ol$goo	g
6	1	oogol$	g

Suffix array, order of suffixes after sorting

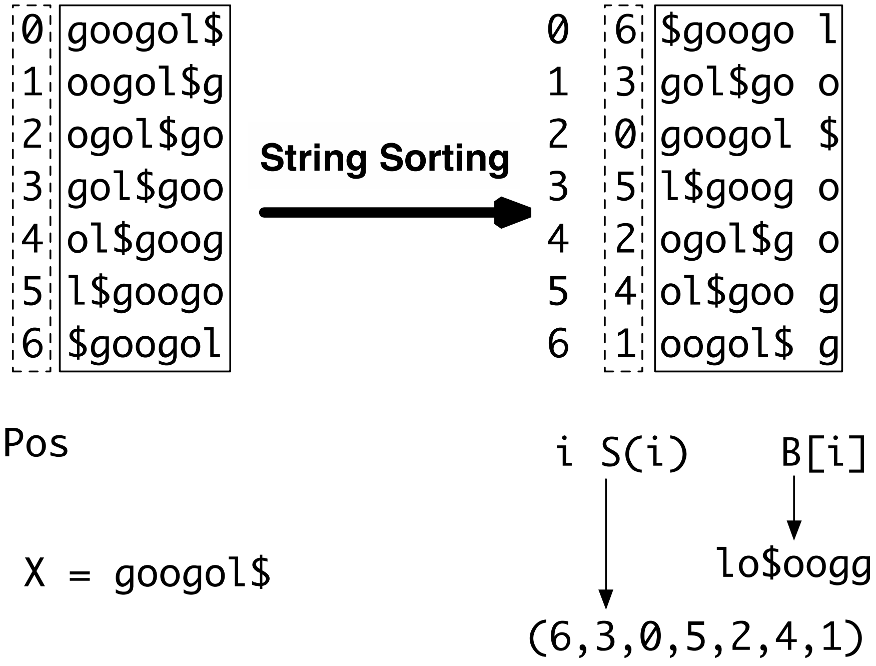
X = googol$

lo$oogg

(6,3,0,5,2,4,1)
0, 1, 2, 3, 4, 5, 6

Example Burrows-Wheeler Aligner (BWA)

Li and Durbin, 2009



Pos	
0	googol$
1	oogol$g
2	ogol$go
3	gol$goo
4	ol$goog
5	l$googo
6	$googol

String Sorting

i	S(i)		B[i]
0	6	$googo	l
1	3	gol$go	o
2	0	googol	$
3	5	l$goog	o
4	2	ogol$g	o
5	4	ol$goo	g
6	1	oogol$	g

X = googol$

lo$oogg

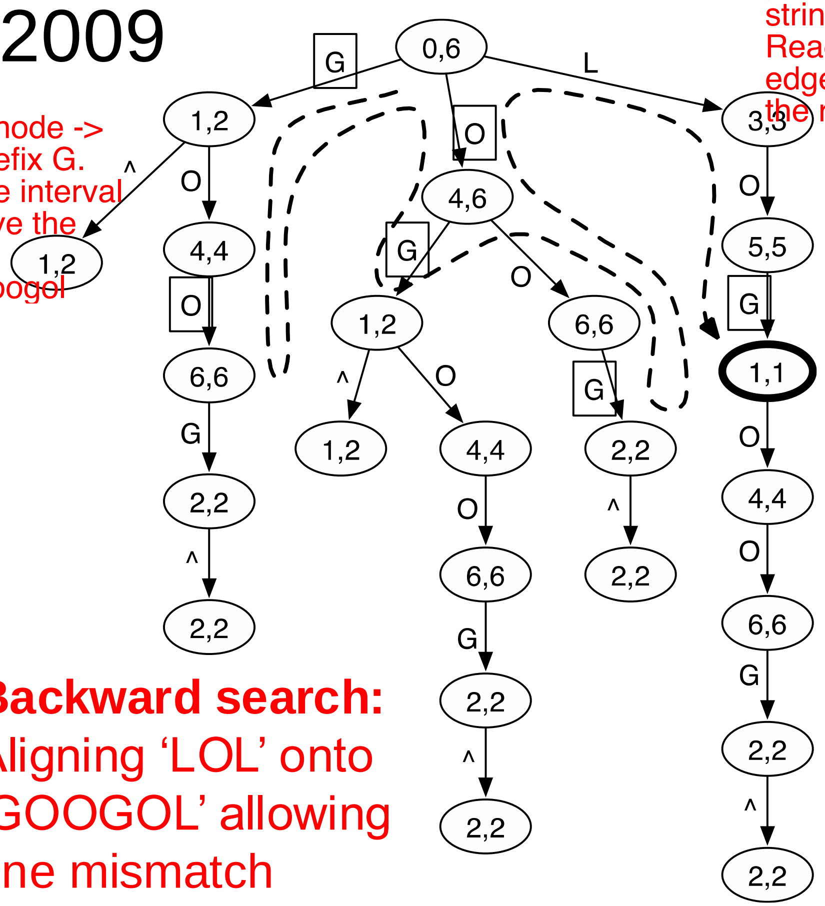
(6,3,0,5,2,4,1)

Constructing **suffix array and BWT string** for *X* = googol$. String *X* is circulated to generate seven strings, which are then lexicographically sorted. After sorting, the positions of the first symbols form the suffix array (6,3,0,5,2,4,1) and the concatenation of the last symbols of the circulated strings gives the BWT string lo$oogg.

These are the prefixes of the suffixes. Not prefixes of the original string

Read each node using edges from the leaf to the root

For example, this node -> Represents the prefix G. Its values gives the interval of suffixes that have the prefix G: 1-2: gol$go and googol



For example, this represents the prefix OL

This is GOOGOL

This is ^GOOGOL

Backward search:
Aligning 'LOL' onto 'GOOGOL' allowing one mismatch

Prefix trie (digital tree) of string 'GOOGOL'. Symbol '^' marks the start of the string. The two numbers in a node give the suffix array interval of the string represented by the node. The dashed line shows the route of the brute-force search for query string 'LOL', allowing at most one mismatch. Edge labels in squares mark the mismatches to the query in searching. The only hit is the bold node [1,1] which represents string 'GOL'.

Example Burrows-Wheeler Aligner (BWA)

Aligning 'LOL' onto 'GOOGOL' allowing one mismatch

0	6	$googo	l
1	3	gol$go	o
2	0	googol	$
3	5	l$goog	o
4	2	ogol$g	o
5	4	ol$goo	g
6	1	oogol$	g

i S(i)

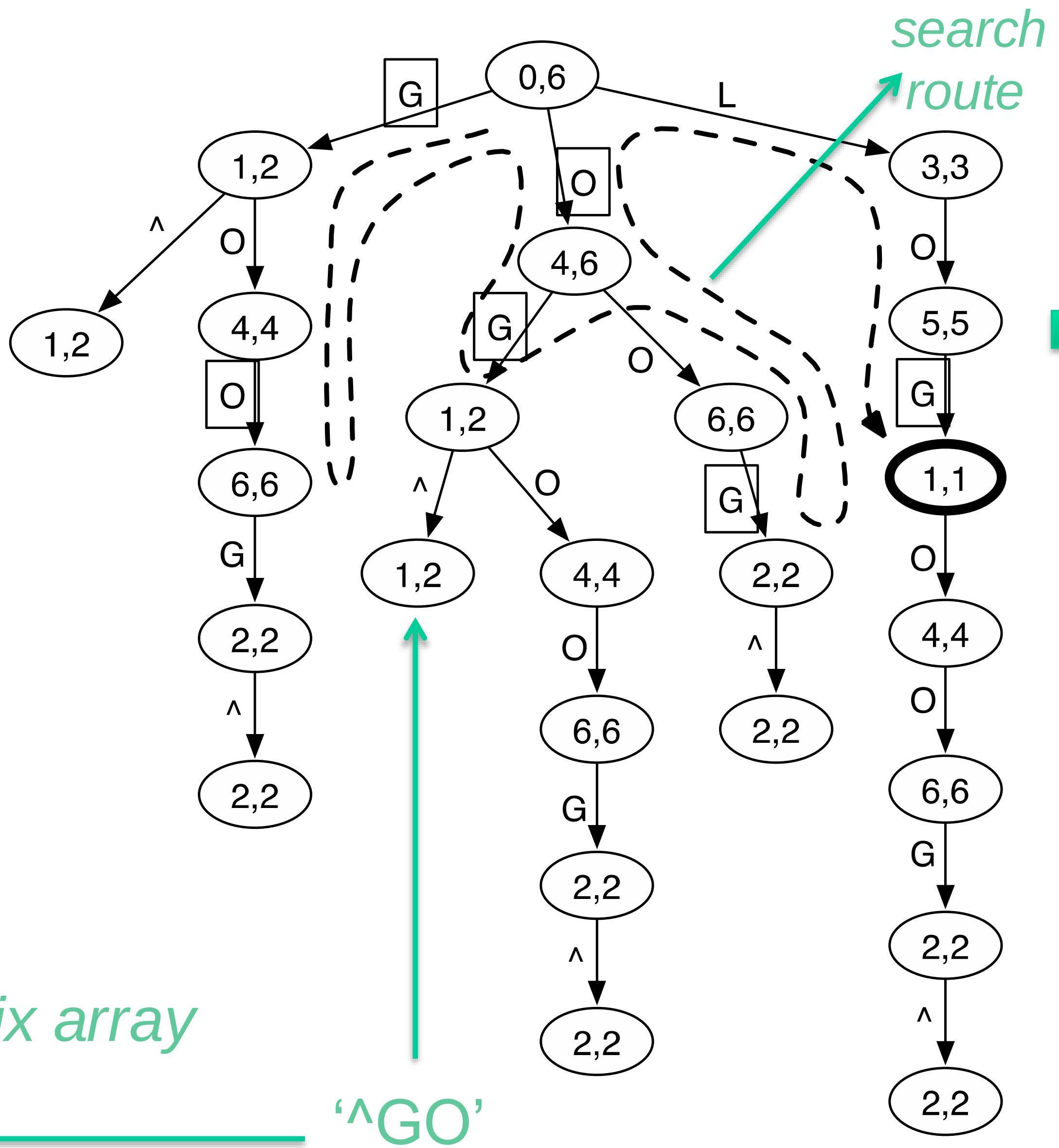
B[i]

lo$oogg

(6,3,0,5,2,4,1)

0, 1, 2, 3, 4, 5, 6

suffix array



'^GO'

GOOGOL

LOL

Building the Prefix Trie

1. Create a root node and label (*begin*, *end*) with the full stretch of the suffix array (0, L), where L is the length of the reference genome sequence (without the '$' end symbol)

   For the previous example, 0 to 6

2. Create a first layer of nodes using the elements (letters) of the BWT; For each node, create and edge (pointing from root to the node), label the edge with the associated element, and label the node with the *begin* and *end* suffix array positions corresponding to the element.

   For the previous example, the nodes are G, O, and L

3. For each node, if there is a prefix (preceding element) for the string represented by the node (the string being defined by the labels of the edges leading back to the root), then create a node and label the node and its edge as for 2. Do this for all possible prefixes including the artificial start symbol '^'.

   O comes before L
   O and G comes before O
   G and $ comes before G
   Seeing a $ means we are at the beginning of the string, stop and add a ^ node. The interval for ^ node is the same as its previous letter

Burrows-Wheeler Aligner (BWA)

- Having the prefix trie (digital tree), BWA searches for an exact substring in O(*W*) time, where *W* is the (average) read length.

- For inexact searches (allowing up to *k* mismatches), the search times go up, but are kept under control by defining lower bounds of the number of differences for each substring W[0, *i*].
  - This limits the search space by avoiding searches across subtrees in the prefix trie.

BWA supports paired-end mapping

- BWA first finds positions of all good hits (one end of paired-end read), sorts them according to chromosomal coordinates and then does a linear scan through all potential hits to pair the two ends.

Backtrack to find coordinates of the matches

- To calculate all the chromosomal coordinates requires looking up the suffix array frequently.

- BWA performs Smith–Waterman alignment for unmapped reads whose mates can be reliably aligned. BWA thus employs SW alignment to rescue some reads with excessive differences.

When one of the paired end reads is not aligned, using the mapped read and its length information, BWA can perform local alignment of the unmapped read in region n=length nucleotides away from the mapped read.

BWA performance (Li & Durbin, 2009)

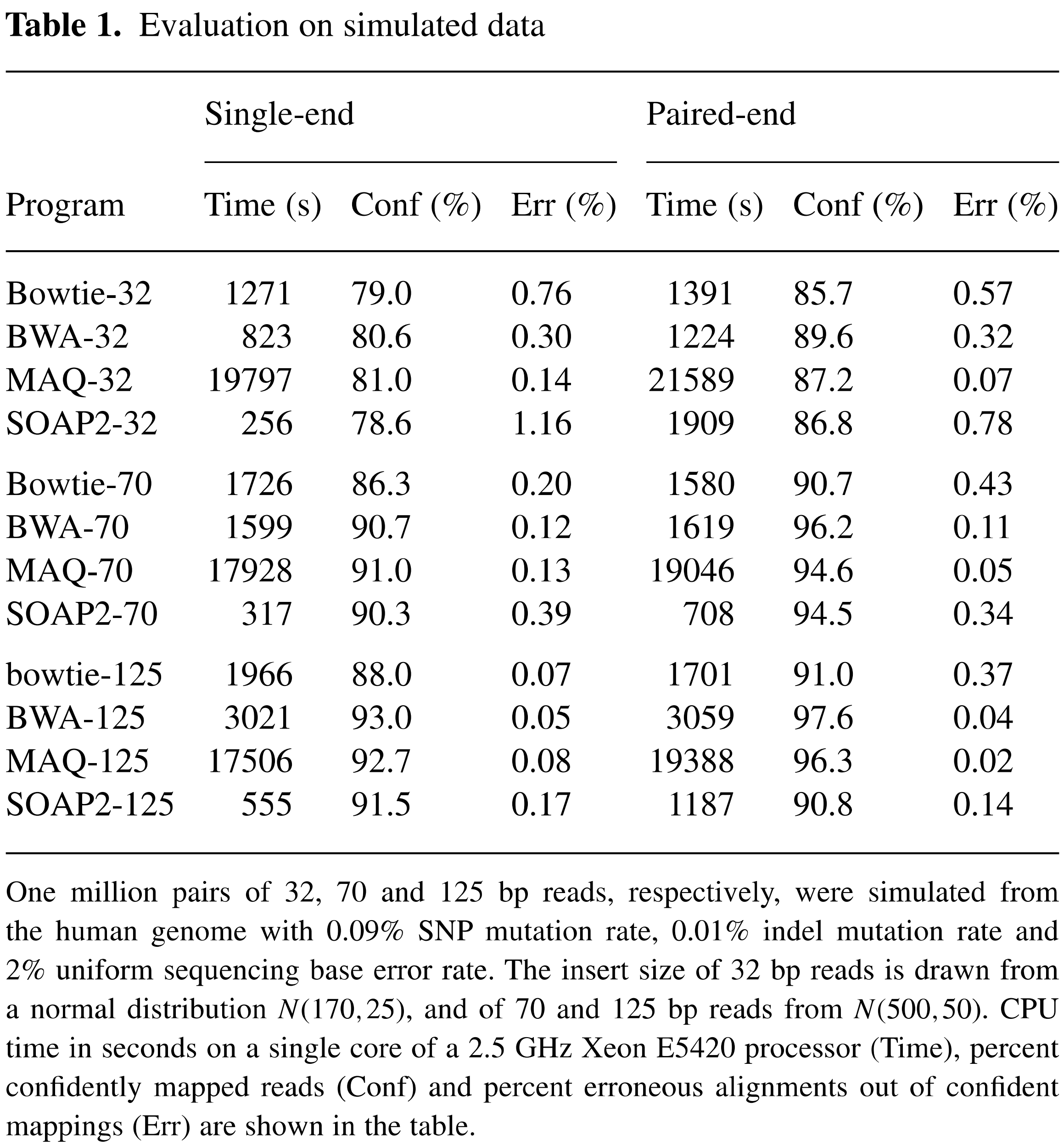


Table 1. Evaluation on simulated data

	Single-end			Paired-end		
Program	Time (s)	Conf (%)	Err (%)	Time (s)	Conf (%)	Err (%)
Bowtie-32	1271	79.0	0.76	1391	85.7	0.57
BWA-32	823	80.6	0.30	1224	89.6	0.32
MAQ-32	19797	81.0	0.14	21589	87.2	0.07
SOAP2-32	256	78.6	1.16	1909	86.8	0.78
Bowtie-70	1726	86.3	0.20	1580	90.7	0.43
BWA-70	1599	90.7	0.12	1619	96.2	0.11
MAQ-70	17928	91.0	0.13	19046	94.6	0.05
SOAP2-70	317	90.3	0.39	708	94.5	0.34
bowtie-125	1966	88.0	0.07	1701	91.0	0.37
BWA-125	3021	93.0	0.05	3059	97.6	0.04
MAQ-125	17506	92.7	0.08	19388	96.3	0.02
SOAP2-125	555	91.5	0.17	1187	90.8	0.14

One million pairs of 32, 70 and 125 bp reads, respectively, were simulated from the human genome with 0.09% SNP mutation rate, 0.01% indel mutation rate and 2% uniform sequencing base error rate. The insert size of 32 bp reads is drawn from a normal distribution $N(170, 25)$, and of 70 and 125 bp reads from $N(500, 50)$. CPU time in seconds on a single core of a 2.5 GHz Xeon E5420 processor (Time), percent confidently mapped reads (Conf) and percent erroneous alignments out of confident mappings (Err) are shown in the table.

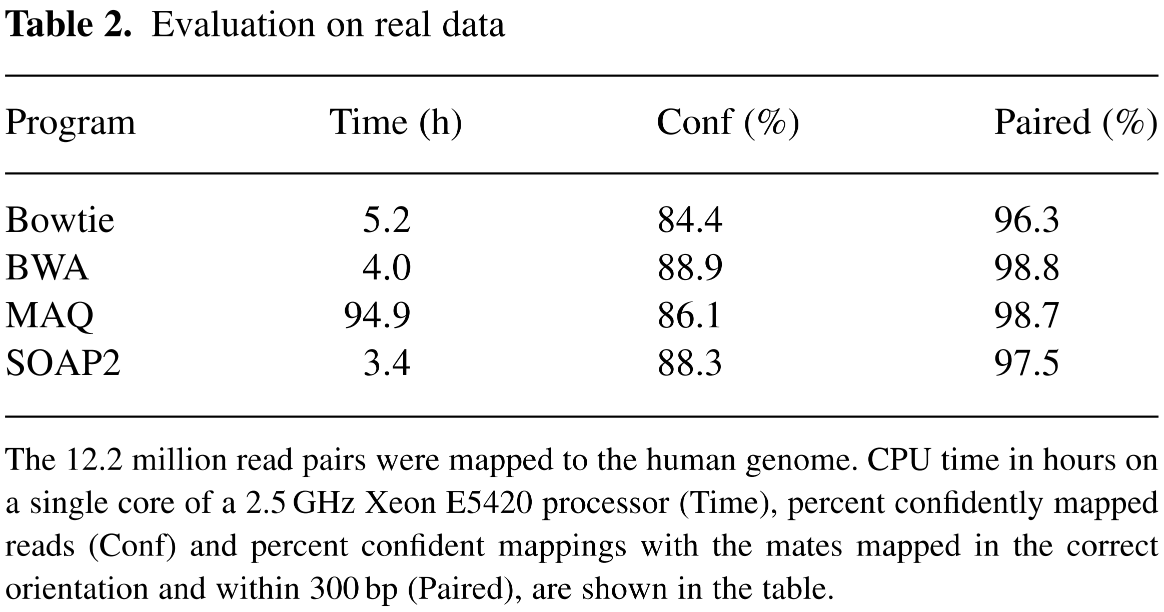


Table 2. Evaluation on real data

Program	Time (h)	Conf (%)	Paired (%)
Bowtie	5.2	84.4	96.3
BWA	4.0	88.9	98.8
MAQ	94.9	86.1	98.7
SOAP2	3.4	88.3	97.5

The 12.2 million read pairs were mapped to the human genome. CPU time in hours on a single core of a 2.5 GHz Xeon E5420 processor (Time), percent confidently mapped reads (Conf) and percent confident mappings with the mates mapped in the correct orientation and within 300 bp (Paired), are shown in the table.

Question – Mismatches **

- What may be the cause of any mismatches in the alignment between the sequence read and the reference genome?
  - Think both about evolution and the experimental procedure

- Would it be useful if we could differentiate between different sources of noise? How could this be done?

Sequencing errors

- Incorrect information within the reads because of errors of sequencing machines
- Sometimes error rate > 1%

- ACAG**GAGGT** read1
- **GAGGT**CCAGA read2
- **GAAGT**CCAGA read3
- **GAGGT**CCCGA read4

Major Problems with NGS data

- Huge amounts of data to process
- High error rate
- Lack of coverage
- **Repeat sequences**
  - this is the largest general problem for (de novo) assembly

Practical 2 - BWA

- Coding assignment to create and use the Burrows-Wheeler Transform (BWT) and derived indexes (rank and fmap) to reconstruct the reference genome sequence.
- Paper exercise to construct the BWT, Suffix Array, and Prefix Trie from a reference genome sequence.

Take home

- Human Genome Project (HGP) saga
  - Sanger sequencing
- *De novo* and reference-based sequencing
  - Make sure you are able to carry out the De Bruijn graph algorithm
- Problems with *de novo* sequencing
  - multiple contigs
  - repeats
- Paired-end and mate pair reads
  - Make sure you are able to carry out the algorithm on paper
- De Bruijn graphs and *de novo* sequencing
- BWA for reference-based sequencing
- Sequencing depth