Assembly & Whole Genome Alignment

Michael Schatz

Feb 4, 2019

Lecture 4: Applied Comparative Genomics

Assignment 1: Chromosome Structures
Due Feb 6 @ 11:59pm

Assignment 1: Chromosome Structures

Assignment Date: Wednesday, Jan 30, 2019
Due Date: Wednesday, Feb. 6, 2019 @ 11:59pm

Assignment Overview

In this assignment you will profile the overall structure of the genomes of several important species and then study the yeast genome in more detail. As a reminder, any questions about the assignment should be posted to Piazza

Question 1: Chromosome structures

Download the chomosome size files for the following genomes (Note these have been preprocessed to only include main chromosomes):

1. Arabidopsis thaliana (TAIR10) - An important plant model species [info]
2. Corn (Zea mays B73v4)) - The most widely grown crop in the world [info]
3. E. coli (Escherichia coli K12) - One of the most commonly studied bacteria [info]
4. Fruit Fly (Drosophila melanogaster, dm6) - One of the most important model species for genetics [info]
5. Human (hg38) - us :) [info]
6. Rice (Oryza sativa, IRGSP-1.0) - One of the most important crops in the world [info]
7. Worm (Caenorhabditis elegans, ce10) - One of the most important animal model species [info]
8. Yeast (Saccharomyces cerevisiae, sacCer3) - an important eukaryotic model species, also good for bread and beer [info]

Using these files, make a table with the following information per species:

- Question 1.1. Total genome size
- Question 1.2. Number of chromosomes
- Question 1.3. Largest chromosome size and name

https://github.com/schatzlab/appliedgenomics2019

Outline

1. ***Assembly theory***
   - Assembly by analogy
2. ***Practical Issues***
   - Coverage, read length, errors, and repeats
3. ***Next-next-gen Assembly***
   - Canu: recommended for PacBio/ONT project
4. ***Whole Genome Alignment***
   - MUMmer recommended

de Bruijn Graph Construction

- $G_k = (V,E)$
  - V = Length-*k* sub-fragments
  - E = Directed edges between consecutive sub-fragments
    - Sub-fragments overlap by *k*-1 words

Fragments |f|=5

It was the best of

was the best of times

Sub-fragment *k*=4

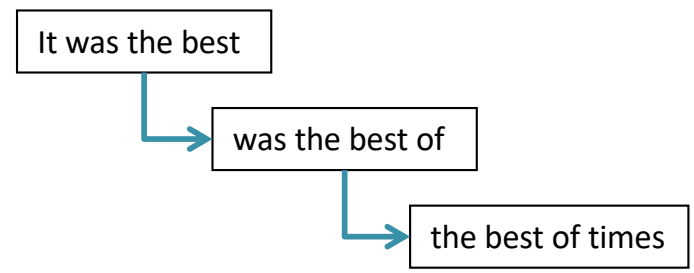
It was the best

was the best of

was the best of

the best of times

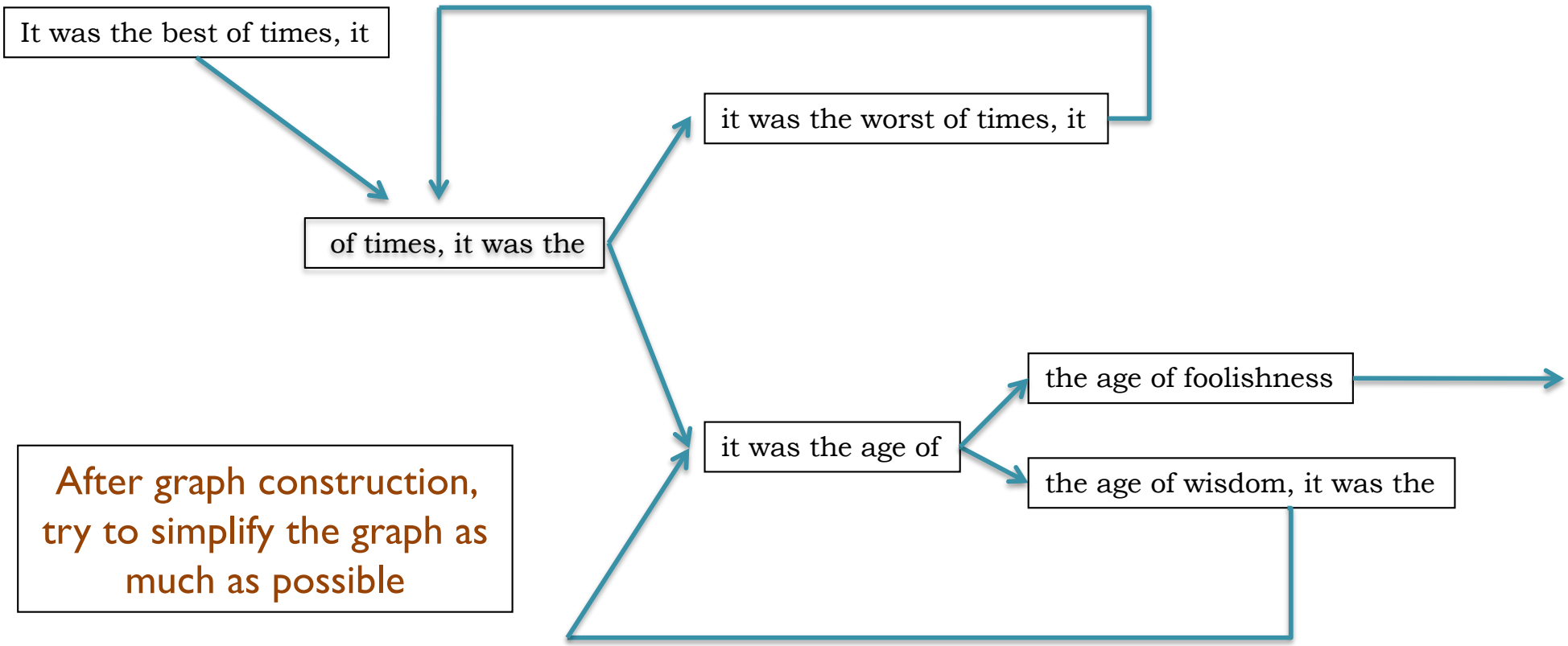
Directed edges (overlap by *k*-1)



– Overlaps between fragments are implicitly computed

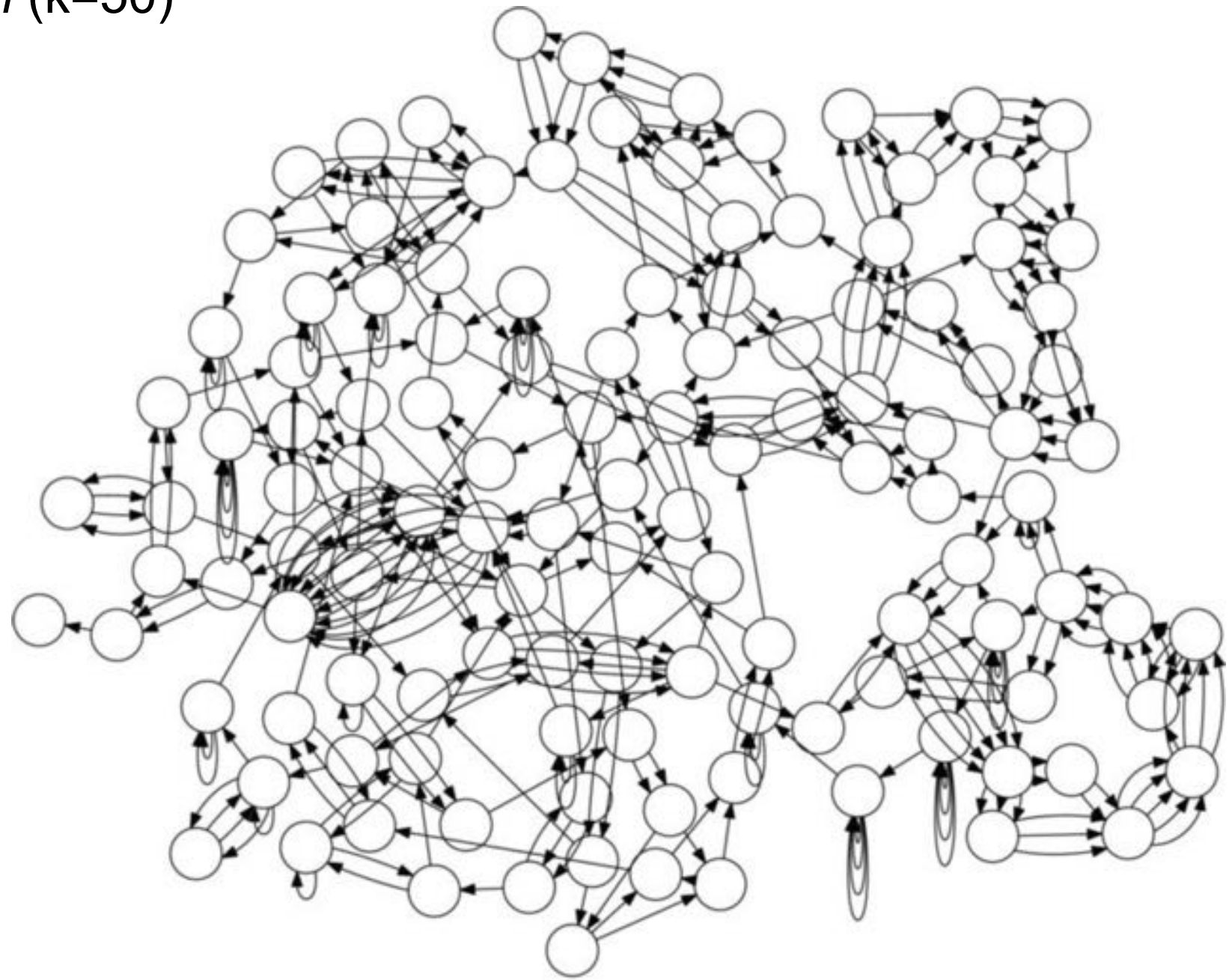
de Bruijn, 1946
Idury et al., 1995
Pevzner et al., 2001

de Bruijn Graph Assembly



After graph construction, try to simplify the graph as much as possible

E. coli (k=50)

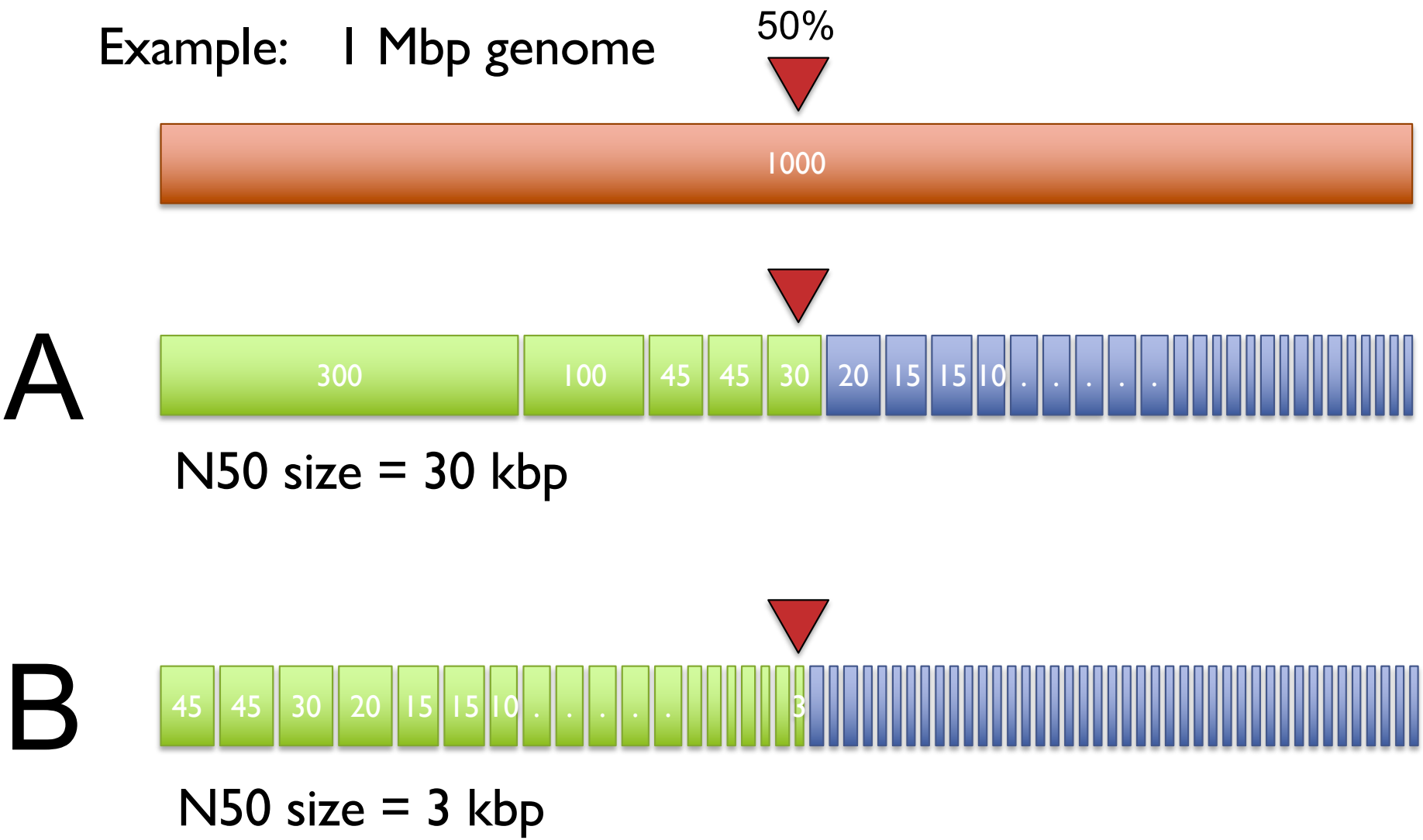


Reducing assembly complexity of microbial genomes with single-molecule sequencing
Koren et al (2013) Genome Biology. **14**:R101 https://doi.org/10.1186/gb-2013-14-9-r101

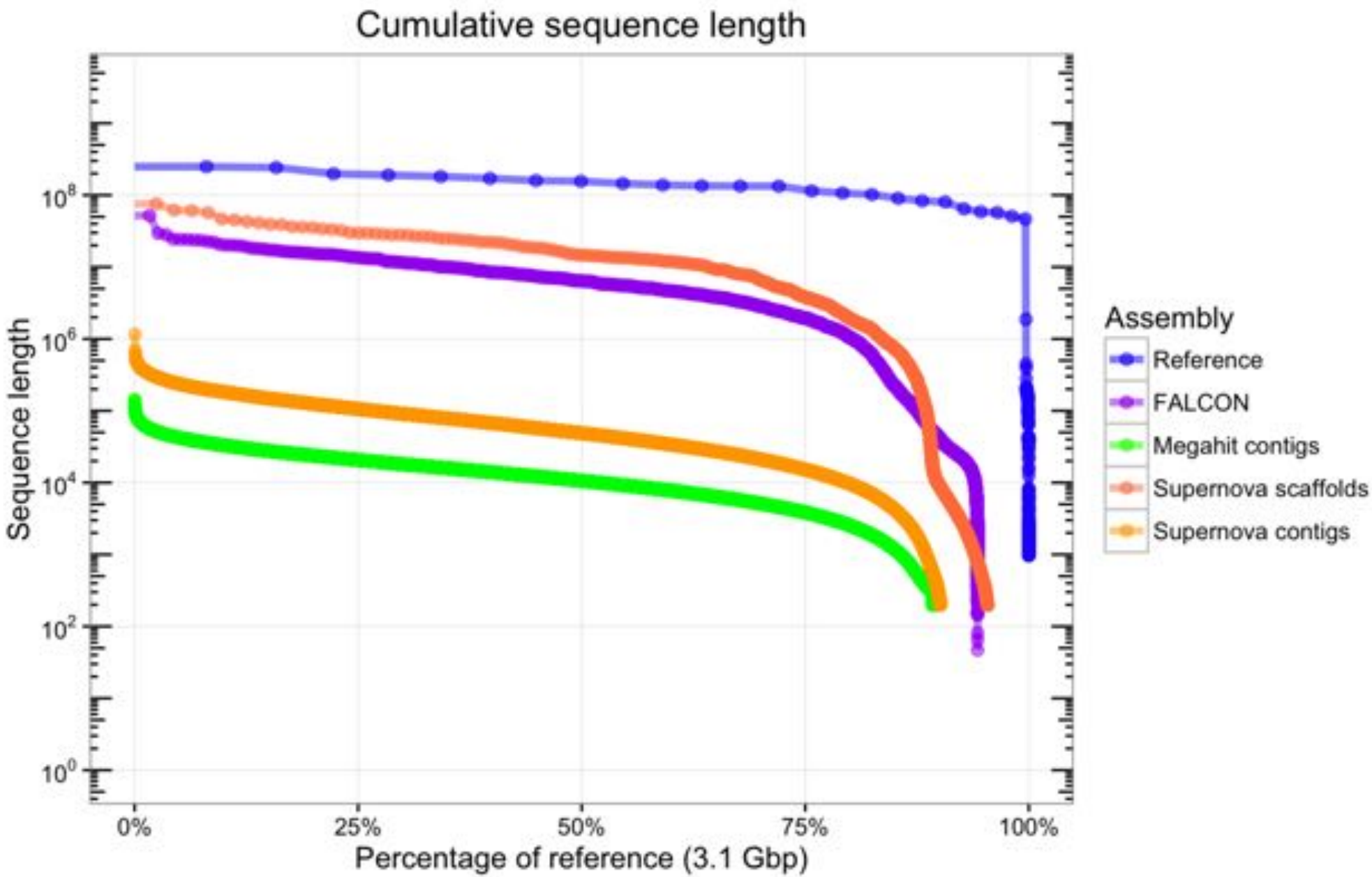
Contig N50

Def: 50% of the genome is in contigs as large as the N50 value

Example: 1 Mbp genome



Contig Nchart



Outline

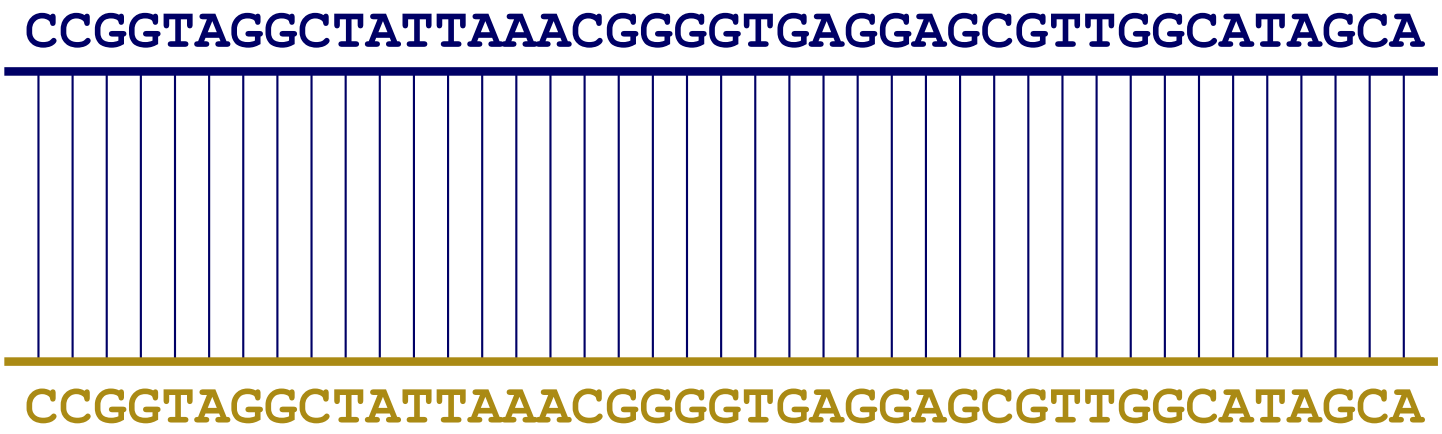
1. ***Assembly theory***
   - Assembly by analogy
2. ***Practical Issues***
   - Coverage, read length, errors, and repeats
3. ***Next-next-gen Assembly***
   - Canu: recommended for PacBio/ONT project
4. ***Whole Genome Alignment***
   - MUMmer recommended

Whole Genome Alignment with MUMmer

Slides Courtesy of Adam M. Phillippy
NHGRI

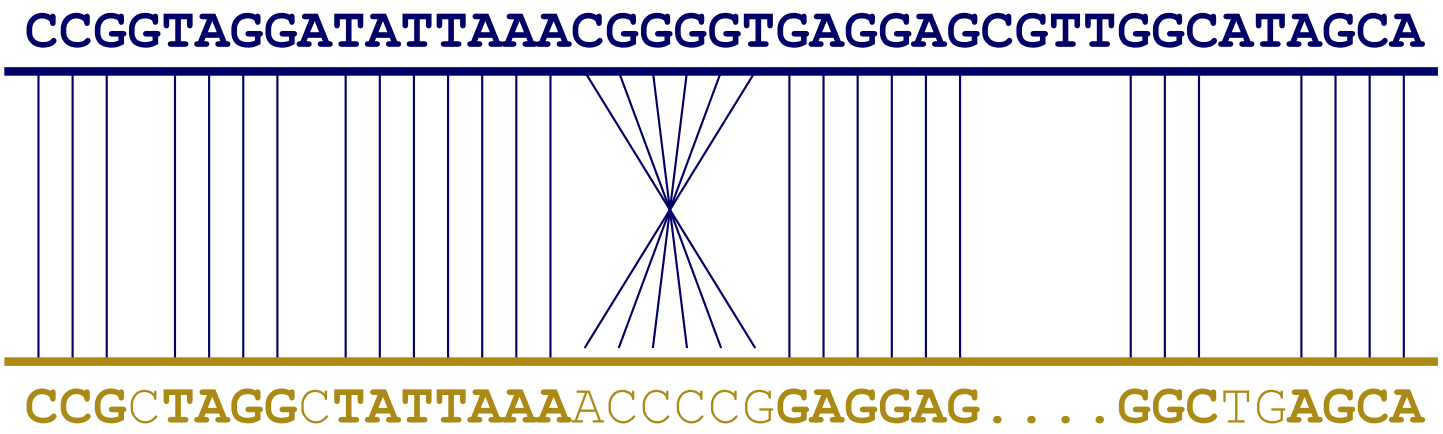
Goal of WGA

- For two genomes, *A* and *B*, find a mapping from each position in *A* to its corresponding position in *B*



Not so fast...

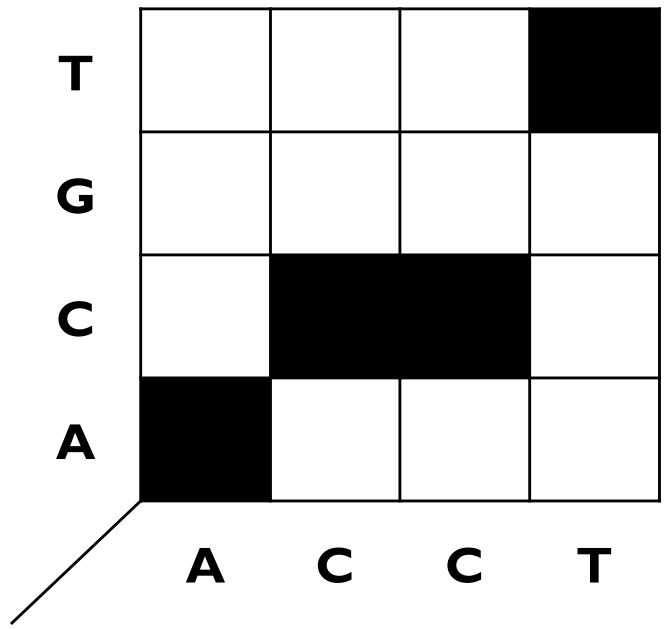
- Genome *A* may have insertions, deletions, translocations, inversions, duplications or SNPs with respect to *B* (sometimes all of the above)



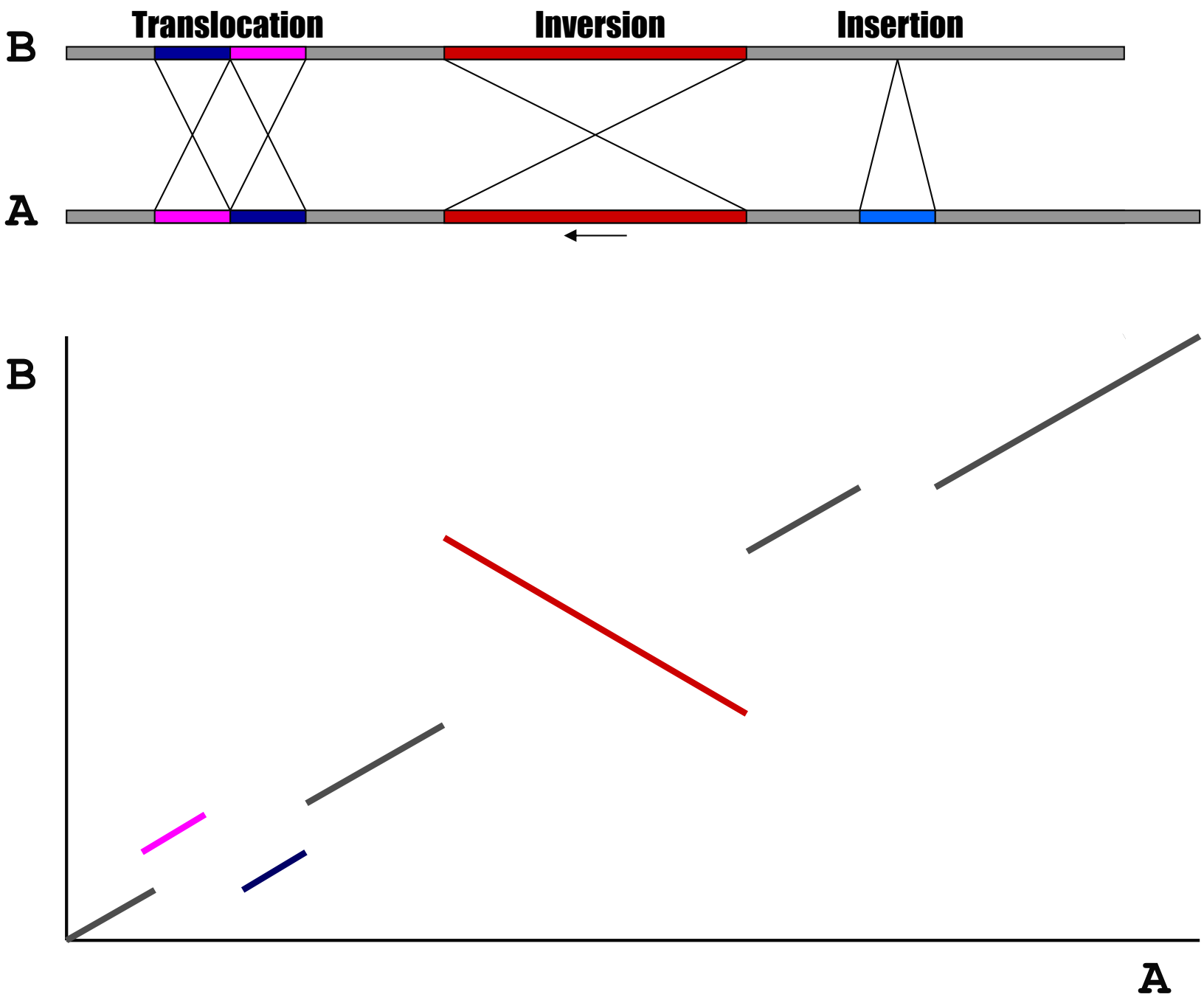
WGA visualization

- How can we visualize *whole* genome alignments?

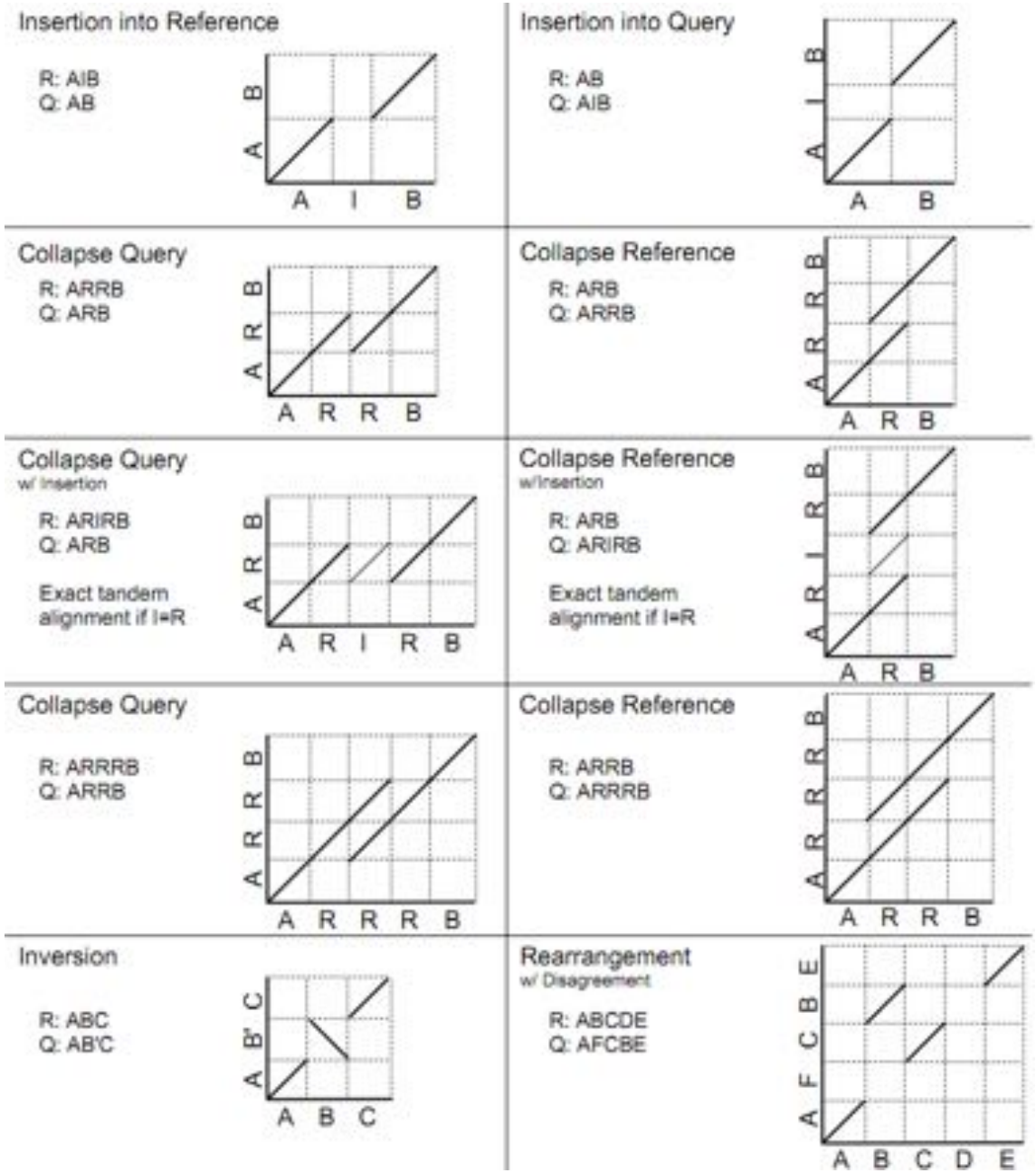
- With an alignment dot plot
  - *N* x *M* matrix
    - Let *i* = position in genome *A*
    - Let *j* = position in genome *B*
    - Fill cell *(i,j)* if A_i shows similarity to B_j



  - A perfect alignment between *A* and *B* would completely fill the positive diagonal

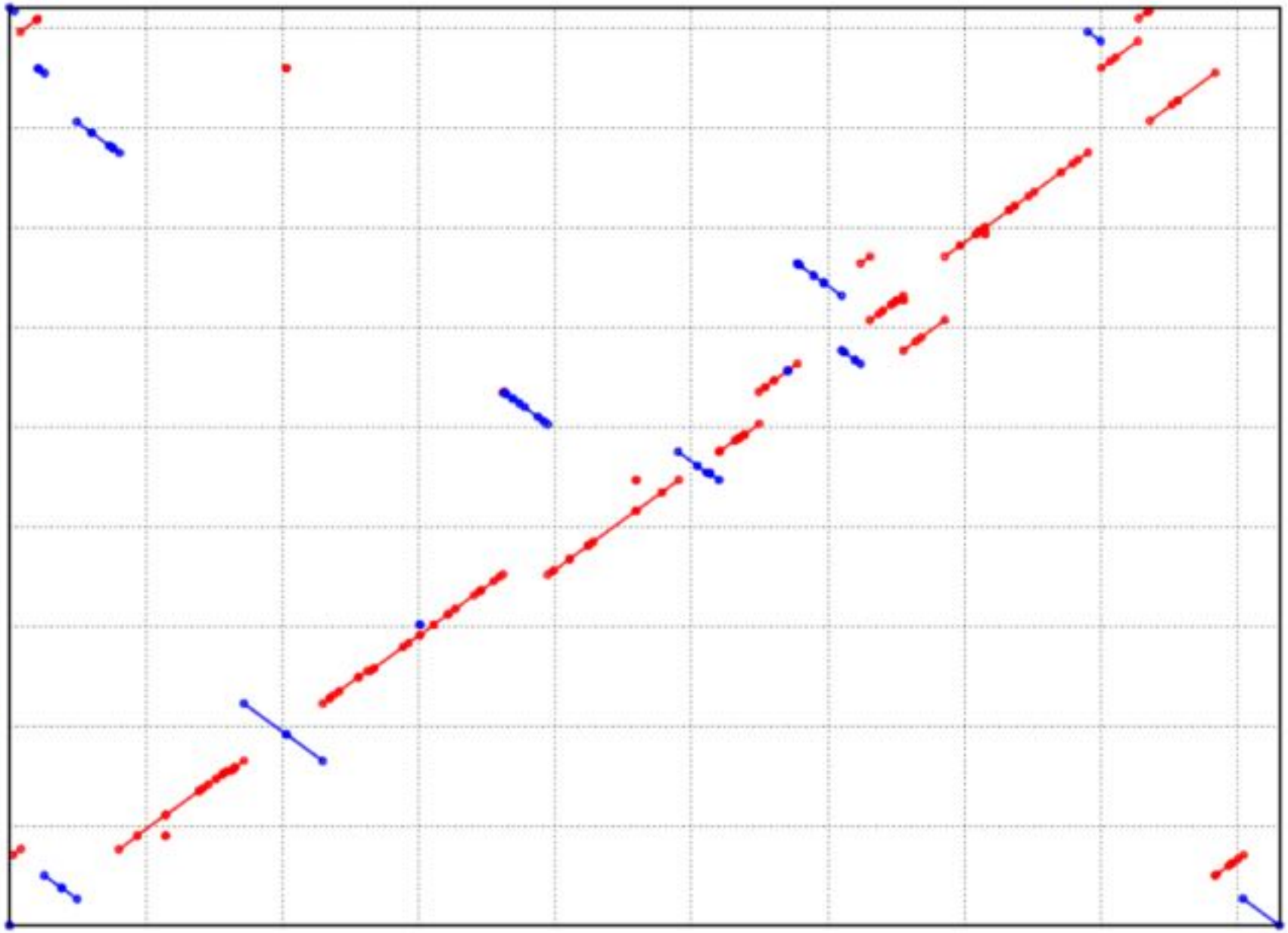

Translocation
Inversion
Insertion
B
A
B
A

SV Types



- Different structural variation types / misassemblies will be apparent by their pattern of breakpoints
- Most breakpoints will be at or near repeats
- Things quickly get complicated in real genomes

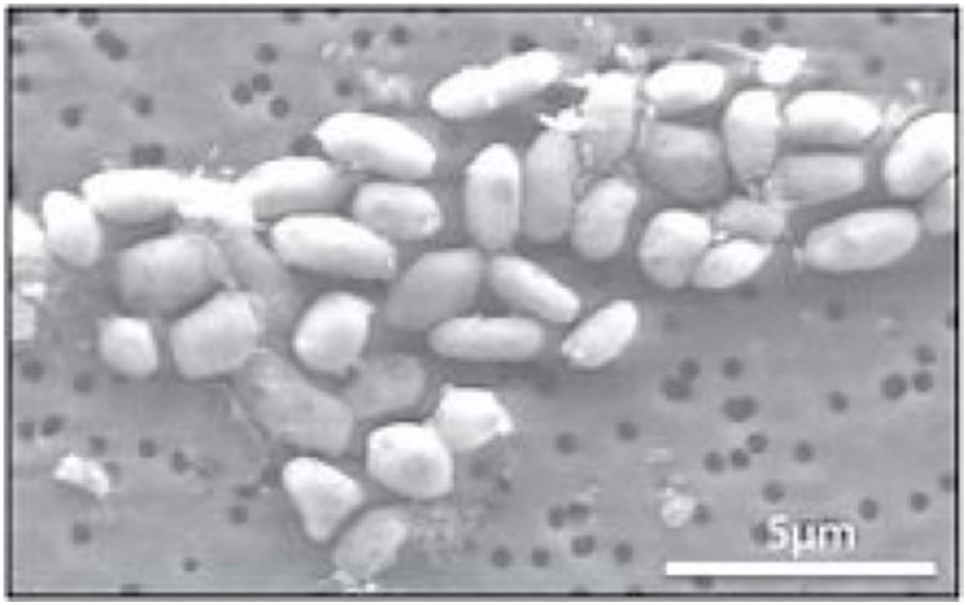
http://mummer.sf.net/manual/AlignmentTypes.pdf



Alignment of 2 strains of *Y. pestis*

http://mummer.sourceforge.net/manual/

Halomonas sp. GFAJ-1



<u>Library 1: Fragment</u>

Avg Read length: 100bp

Insert length: 180bp

<u>Library 2: Short jump</u>

Avg Read length: 50bp

Insert length: 2000bp

A Bacterium That Can Grow by Using Arsenic Instead of Phosphorus
Wolfe-Simon et al (2010) *Science. 332(6034)1163-1166.*

Digital Information Storage

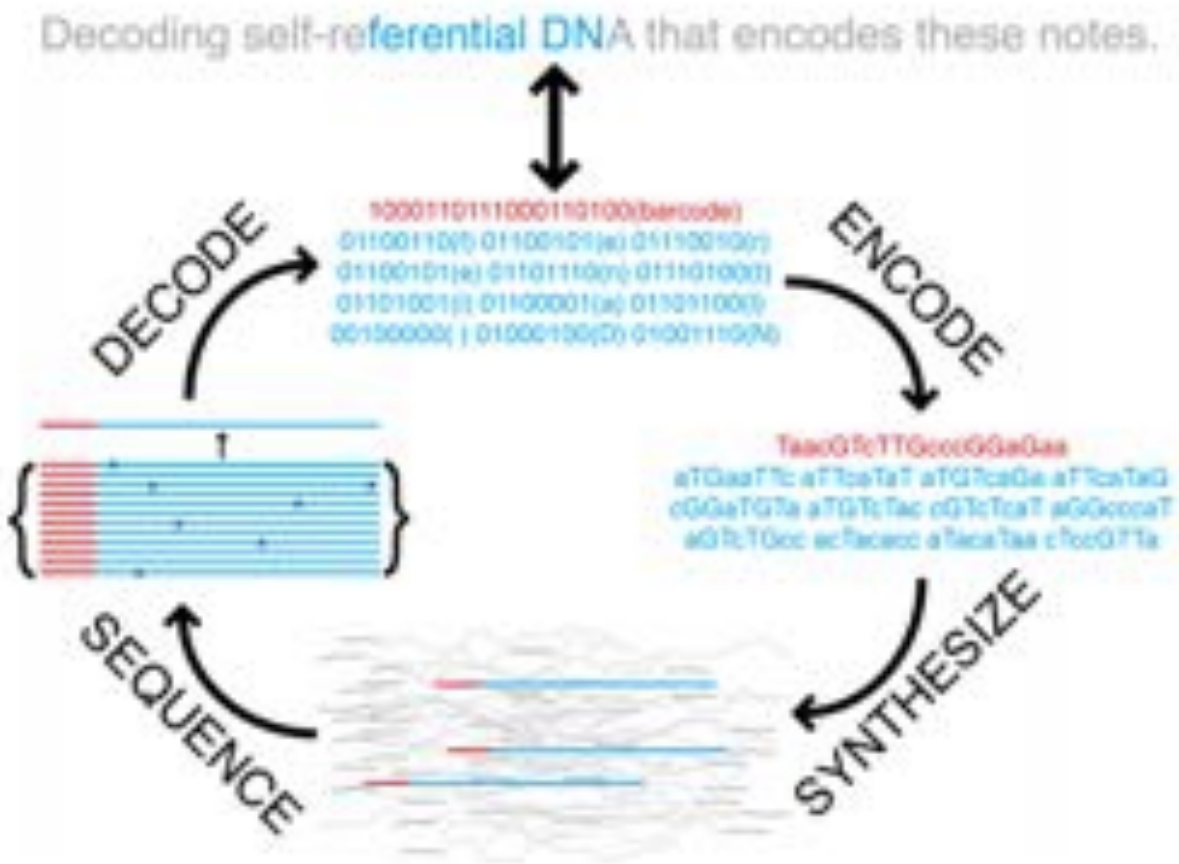


Fig. S1. Schematic of DNA information storage.

Encoding/decoding algorithm implemented in dna-encode.pl from David Dooling.

Next-generation Digital Information Storage in DNA
Church et al (2010) *Science*. 337(6102)1628

Assignment 2: Genome Assembly
Due Wednesday Feb 13 @ 11:59pm

1. ***Setup Docker/VirtualBox/Ubuntu***
2. ***Initialize Tools***
3. ***Download Reference Genome & Reads***
4. ***Decode the secret message***
    1. *Estimate coverage, check read quality*
    2. *Check kmer distribution*
    3. *Assemble the reads with spades*
    4. *Align to reference with MUMmer*
    5. *Extract foreign sequence*
    6. *dna-encode.pl -d*

https://github.com/schatzlab/appliedgenomics2019/blob/master/assignments/assignment2/README.md

Find and decode

```
nucmer -maxmatch ref.fasta \
   default/ASSEMBLIES/test/final.contigs.fasta
   -maxmatch     Find maximal exact matches (MEMs) without repeat filtering
   -p refctg     Set the output prefix for delta file

mummerplot --layout --png out.delta
   --layout      Sort the alignments along the diagonal
   --png         Create a png of the results

show-coords -rclo out.delta
   -r            Sort alignments by reference position
   -c            Show percent coverage
   -l            Show sequence lengths
   -o            Annotate each alignment with BEGIN/END/CONTAINS

samtools faidx default/ASSEMBLIES/test/final.contigs.fasta
   Index the fasta file

samtools faidx default/ASSEMBLIES/test/final.contigs.fasta \
    contig_XXX:YYY-ZZZ | ./dna-encode -d
```

See manual at http://mummer.sourceforge.net/manual

Outline

1. ***Assembly theory***
   - Assembly by analogy

2. ***Practical Issues***
   - Coverage, read length, errors, and repeats

3. ***Next-next-gen Assembly***
   - Canu: recommended for PacBio/ONT project

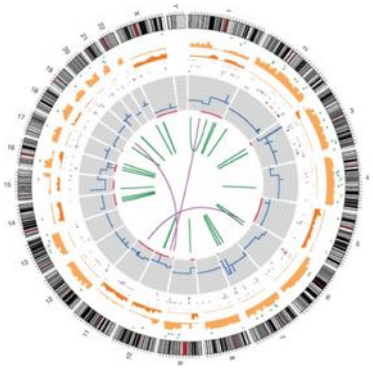
4. Whole Genome Alignment
   - MUMmer recommended

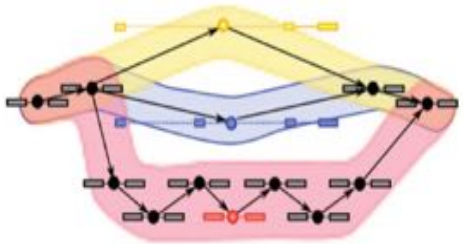
Assembly Applications

- Novel genomes

- Metagenomes

- Sequencing assays
  - Structural variations
  - Transcript assembly
  - …





Why are genomes hard to assemble?

1. ***Biological***:
   - (Very) High ploidy, heterozygosity, repeat content

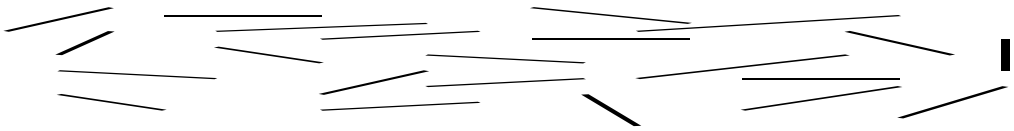
2. ***Sequencing***:
   - (Very) large genomes, imperfect sequencing

3. ***Computational***:
   - (Very) Large genomes, complex structure

4. ***Accuracy***:
   - (Very) Hard to assess correctness

Assembling a Genome

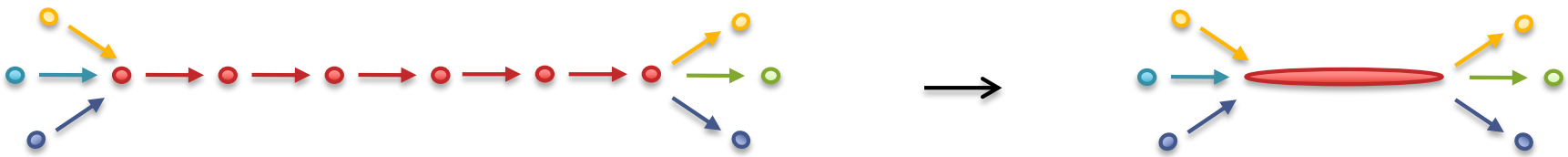
1. Shear & Sequence DNA



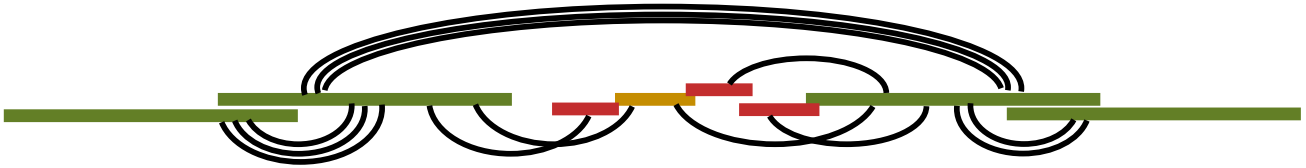
2. Construct assembly graph from reads (de Bruijn / overlap graph)

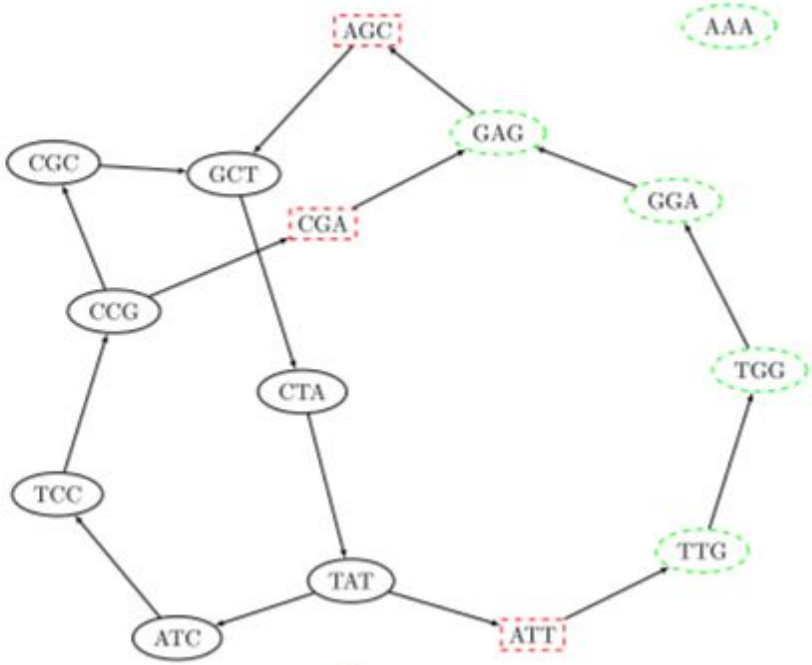
...AGCCTAGGGATGCGCGACACGT
GGATGCGCGACACGTCGCATATCCGGTTTGGTCAACCTCGGACGGAC
CAACCTCGGACGGACCTCAGCGAA...

3. Simplify assembly graph



4. Detangle graph with long reads, mates, and other links





(a)

Table 2 de novo human genome (NA18507) assemblies

Method	Minia	C. & B.	ABySS	SOAPdenovo
Value of *k* chosen	27	27	27	25
Number of contigs (M)	3.49	7.69	4.35	-
Longest contig (kbp)	18.6	22.0	15.9	-
Contig N50 (bp)	1156	250	870	886
Sum (Gbp)	2.09	1.72	2.10	2.08
Nb of nodes/cores	1/1	1/8	21/168	1/16
Time (wall-clock, h)	23	50	15	33
Memory (sum of nodes, GB)	**5.7**	**32**	**336**	**140**

de novo human genome (NA18507) assemblies reported by our assembler (Minia), Conway and Bromage assembler [9], ABySS [8], and SOAPdenovo [7]. Contigs shorter than 100 bp were discarded. Assemblies were made without any pairing information.

$a_1 \ldots a_k$	$\sum_{i=1}^{k} a_i^i \bmod 10$
ATC	0
CCG	0
TCC	5
CGC	6
...	...

(b)

Bloom filter
1
0
0
0
0
1
1
0
0
0

(c)

Nodes self-information:

$$\lceil log_2 \binom{4^3}{7} \rceil = 30 \text{ bits}$$

Structure size:

$$\underbrace{10}_{\text{Bloom}} + \underbrace{3 \cdot 6}_{\text{False positives}} = 28 \text{ bits}$$

(d)

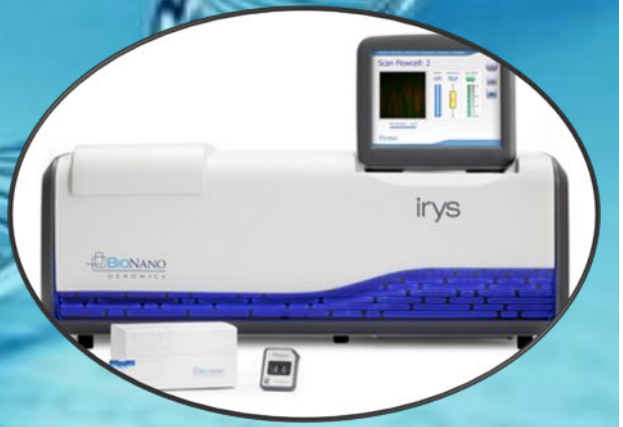
Space-efficient and exact de Bruijn graph representation based on a Bloom filter
Chikhi and Rizk (2013) *Algorithms for Molecular Biology*. 8:22

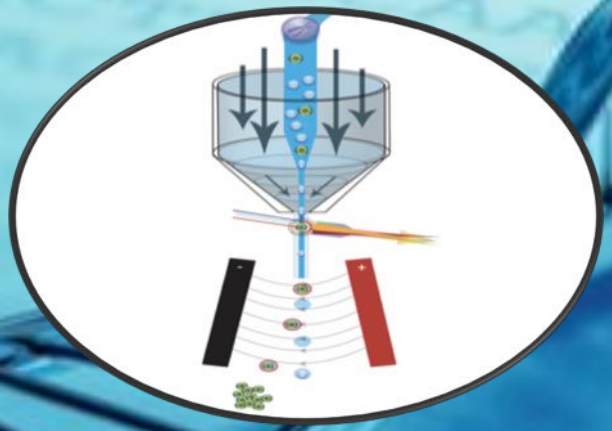
Genomics Arsenal in the year 2019

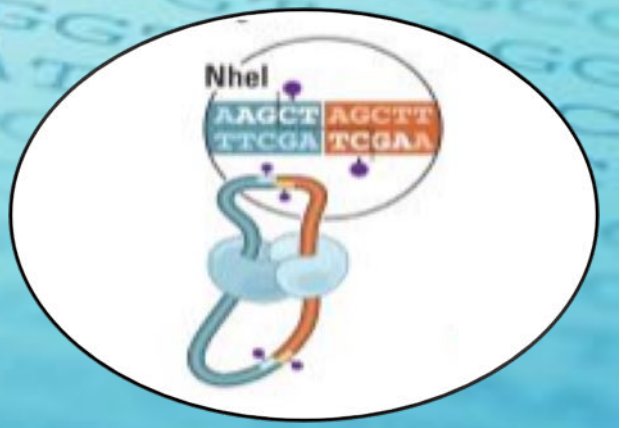
Sample Preparation

Sequencing

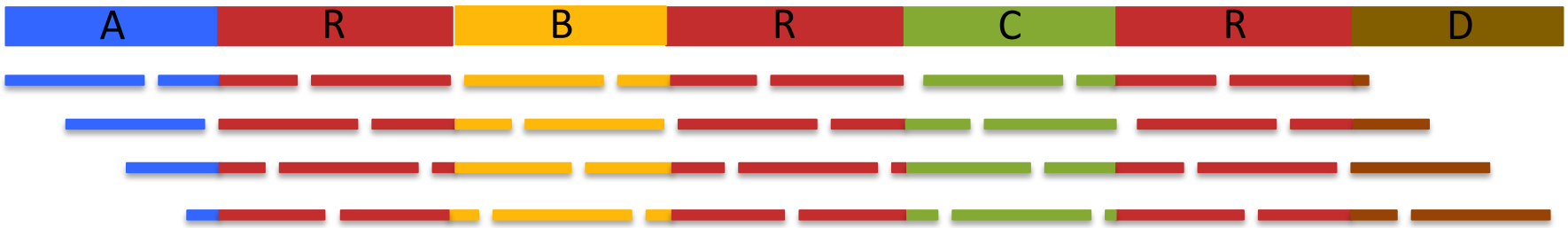
Chromosome Mapping

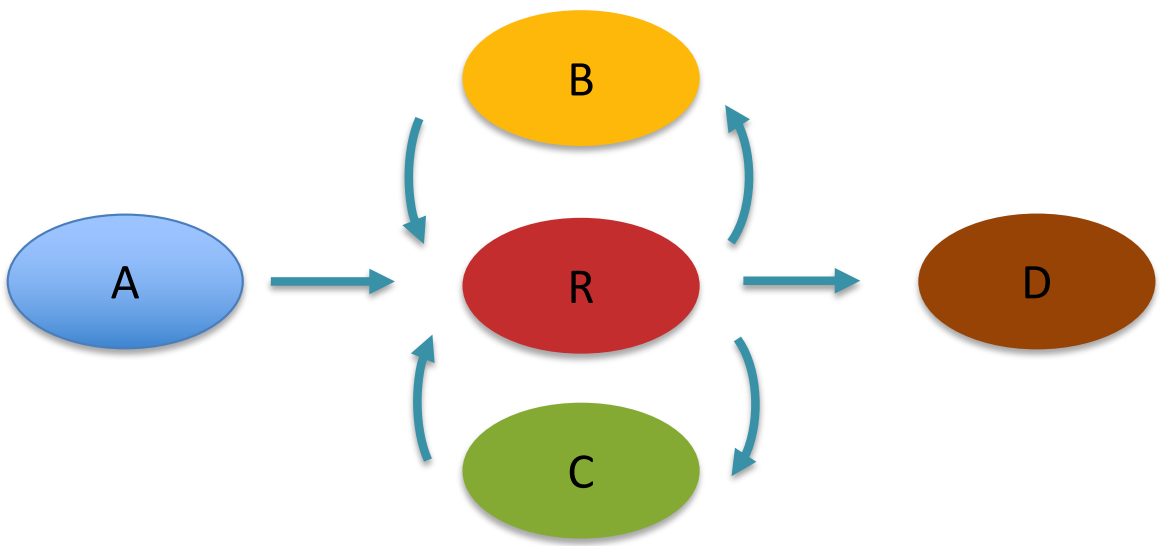




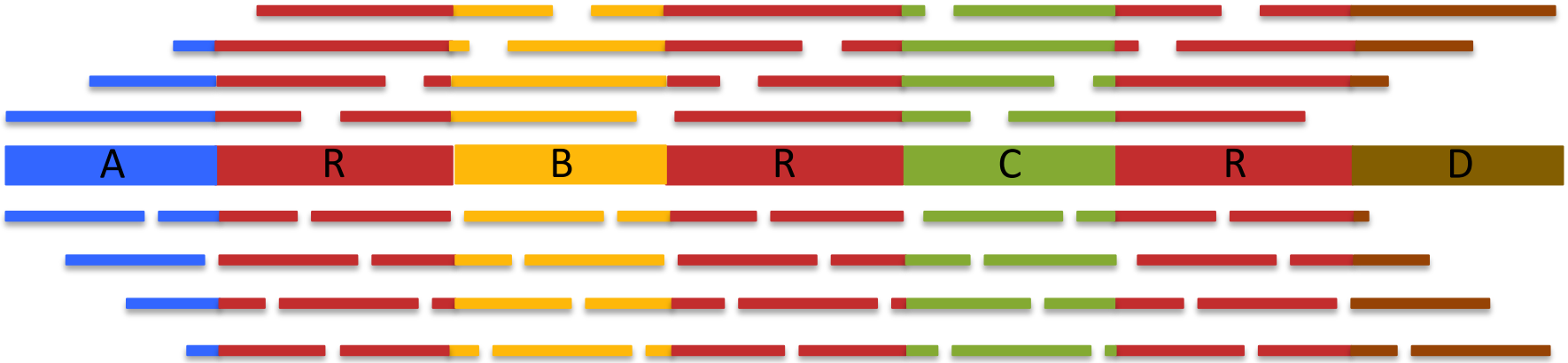


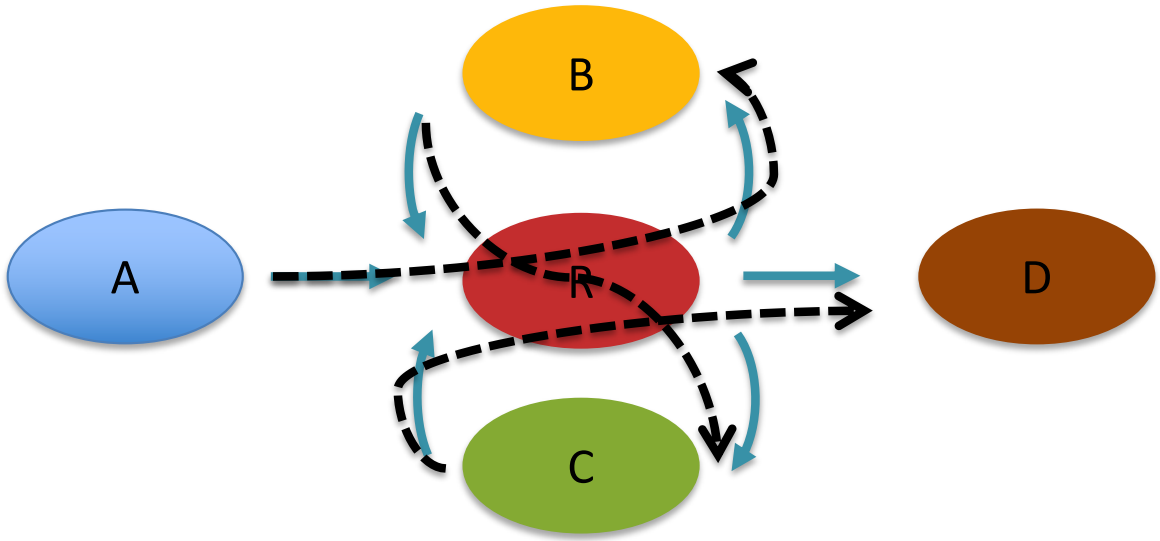
Assembly Complexity



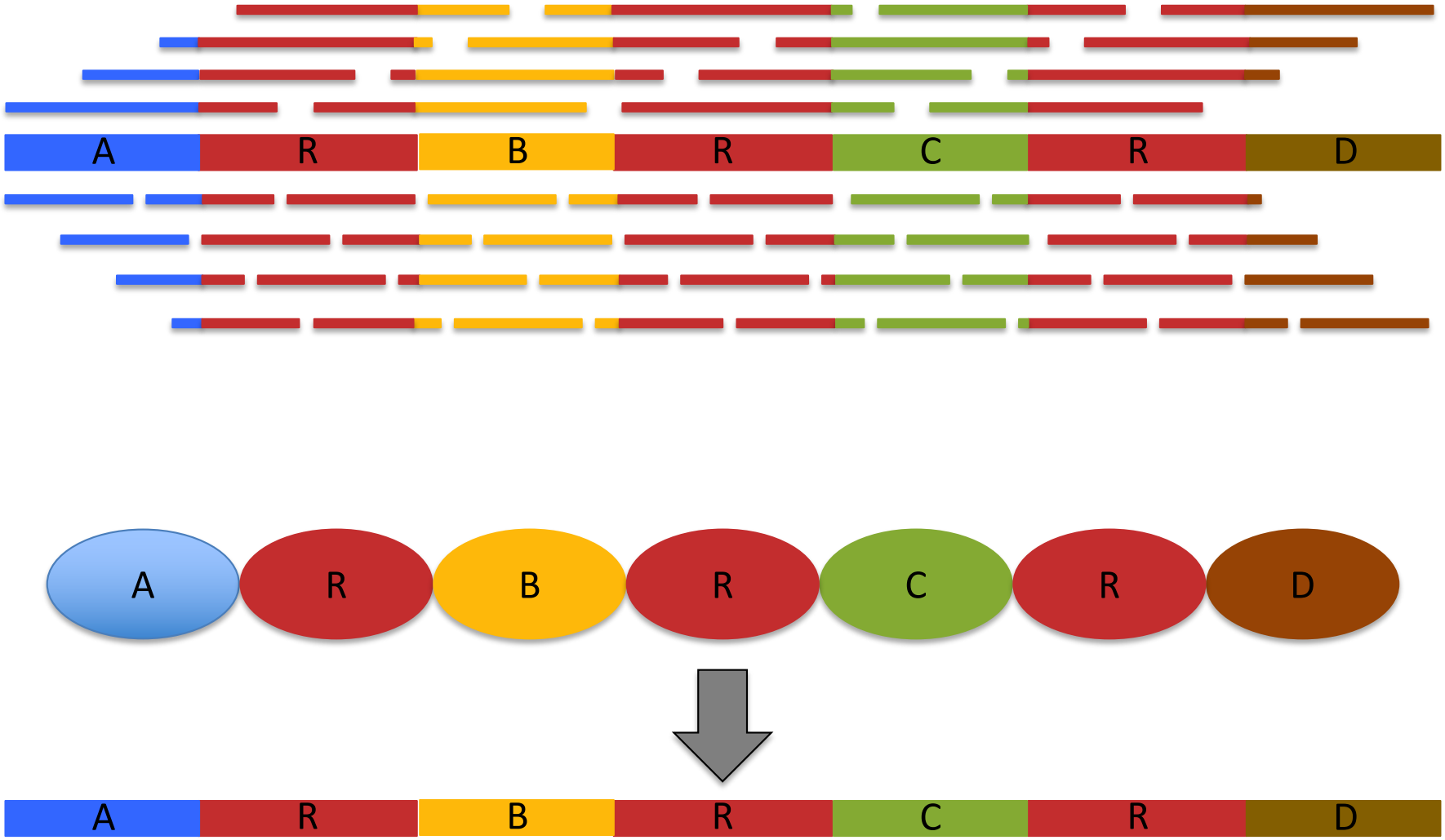


Assembly Complexity





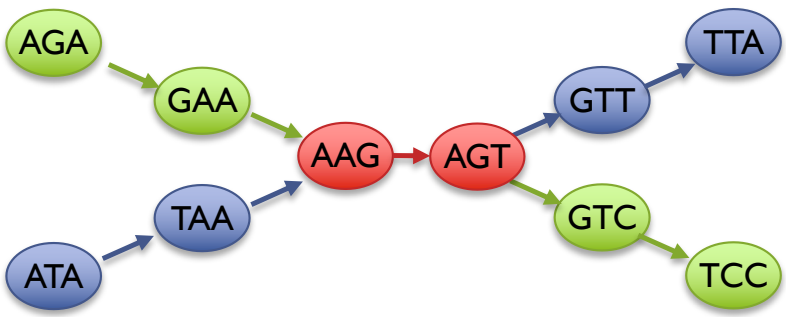
Assembly Complexity



The advantages of SMRT sequencing
Roberts, RJ, Carneiro, MO, Schatz, MC (2013) *Genome Biology*. 14:405

Two Paradigms for Assembly

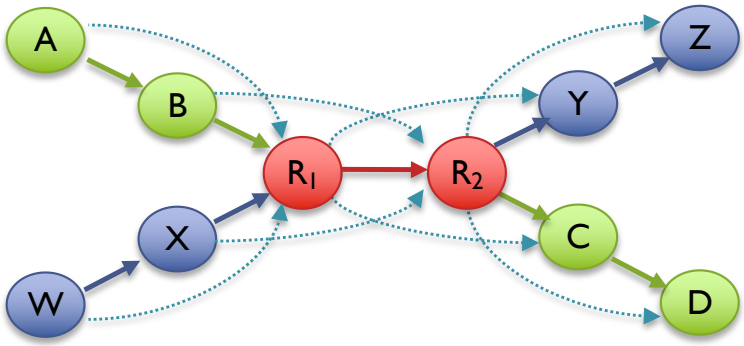
de Bruijn Graph



Short read assemblers

- Repeats depends on word length
- Read coherency, placements lost
- Robust to high coverage

Overlap Graph



Long read assemblers

- Repeats depends on read length
- Read coherency, placements kept
- Tangled by high coverage

Assembly of Large Genomes using Second Generation Sequencing
Schatz MC, Delcher AL, Salzberg SL (2010) *Genome Research*. 20:1165-1173.

Overlap between two sequences

overlap (19 bases)

overhang (6 bases)

…AGCCTAGACCTACAGGATGCGCGGACACGTAGCCAGGAC

CAGTACTTGGATGCGCTGACACGTAGCTTATCCGGT…

overhang

% identity = 18/19 % = 94.7%

overlap - region of similarity between regions

overhang - un-aligned ends of the sequences

The assembler screens merges based on:

- length of overlap
- % identity in overlap region
- maximum overhang size.

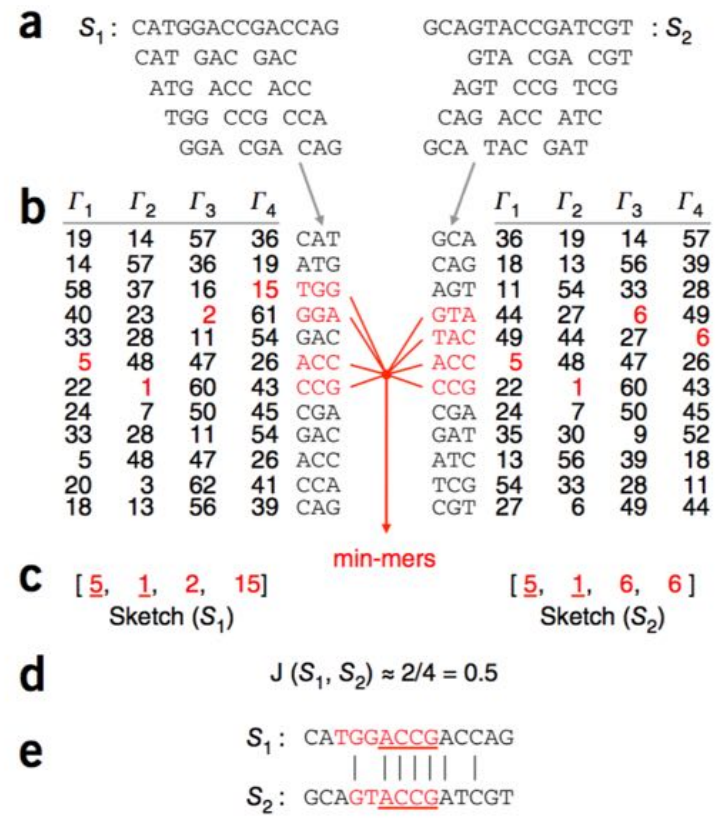
[How do we compute the overlap?]

[Do we really want to do all-vs-all?]

Very fast approximate overlapping

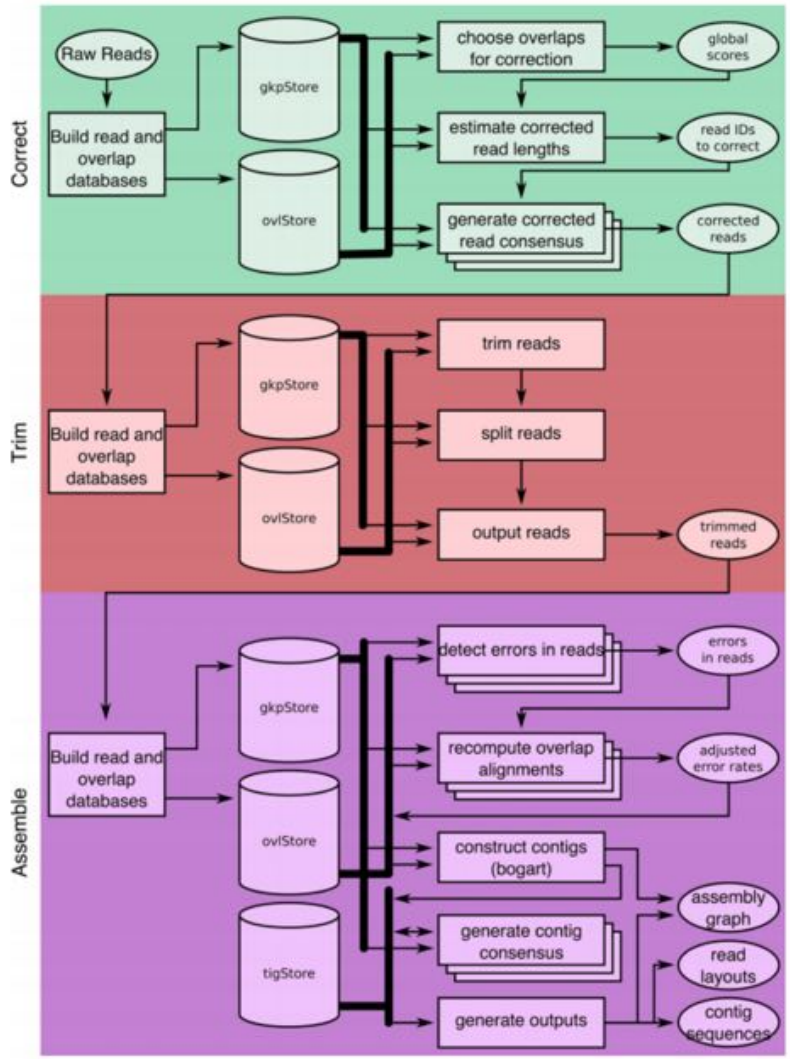
Maybe we don't need to compute the exact identity of the overlap region, just approximate it

- If two reads overlap, they should share many of the same kmers: Their Jaccard coefficient should be high: |intersection| / |union|
- But tracking all of the kmers for a read is a lot of overhead
- Instead, compare the "sketch" of the reads: a small fraction of kmers carefully chosen
- LSH: Find the sketch by applying N hash functions to the kmers, and keeping the minimum hash values reported from each (N=4 in example)
- This forms a nice "random" sample of the reads, and the Jaccard coefficient is a good approximation of the sequence similarity



Assembling large genomes with single-molecule sequencing and locality-sensitive hashing
Berlin et al (2015) *Nature Biotechnology*

Canu Workflow



Three rounds of analysis:

1. ***Error Correction:*** Use MHAP to overlap the reads, then compute a mini assembly centered around each read of good overlaps to error correct

2. ***Trim:*** Use MHAP to recompute overlaps to find regions that are not well supported and discard

3. ***Unitigging:*** Use Dynamic Programming to carefully overlap the error corrected reads, construct overlap graph, and then "unitig" those overlaps to build the contigs

Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation
Koren et al (2017) *Genome Research*

Next Steps

1. Reflect on the magic and power of DNA ☺
2. Check out the course webpage
3. Register on Piazza
4. Work on Assignment 1
    1. Set up Linux, set up Virtual Machine
    2. Set up Dropbox for yourself!
    3. Get comfortable on the command line