Assembly (aka how to sequence a genome)

Isheng Jason Tsai

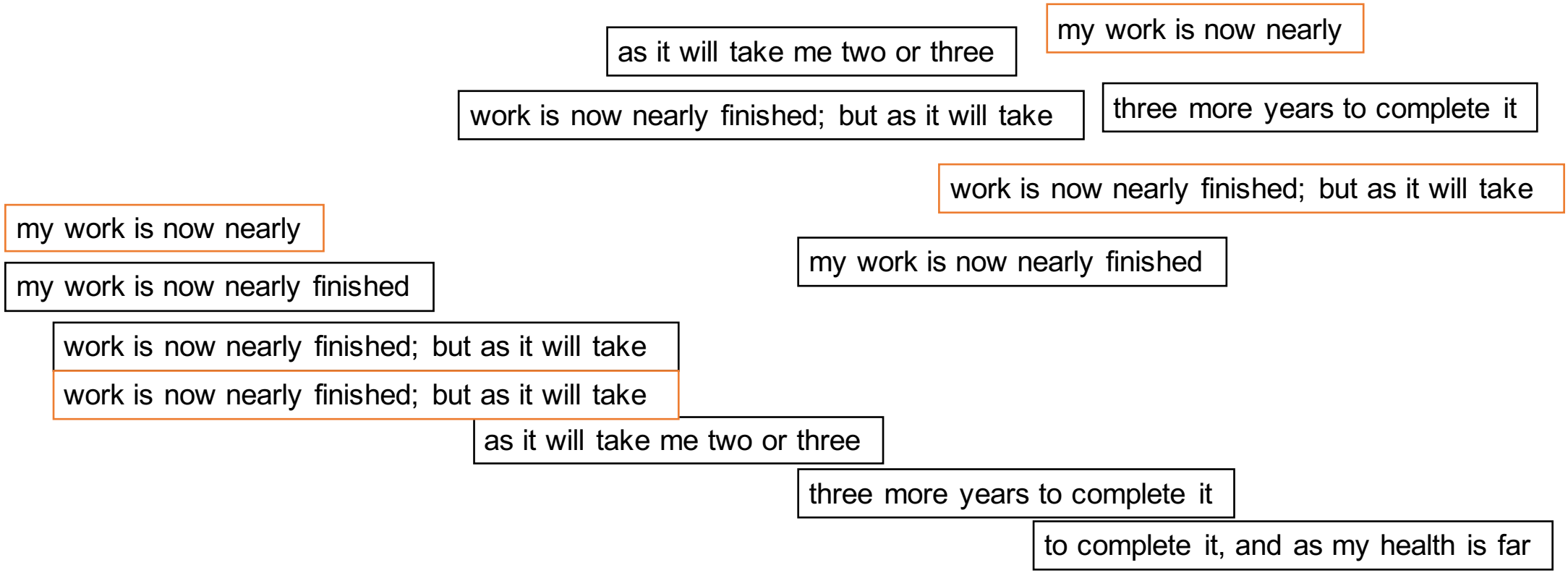
Introduction to NGS Data and Analysis
Lecture 3

Problem

My RA accidently printed five copies of "Origin of Species", and shredded into pieces after a 'lengthy lab meeting'

my work is now nearly finished; but as it will take me two or three more years to complete it, and as my health is far from strong, I have been urged to publish this Abstract. I have more especially been induced to do this, as Mr Wallace, who is now studying the natural history of the Malay archipelago, has arrived at almost exactly the same general conclusions that I have on the origin of species. Last year he sent to me a memoir on this subject, with a request that I would forward it to Sir Charles Lyell, who sent it to the Linnean Society, and it is published in the third volume of the journal of that Society. Sir C. Lyell and Dr Hooker, who both knew of my work -- the latter having read my sketch of 1844 -- honoured me by thinking it advisable to publish, with Mr Wallace's excellent memoir, some brief extracts from my manuscripts.

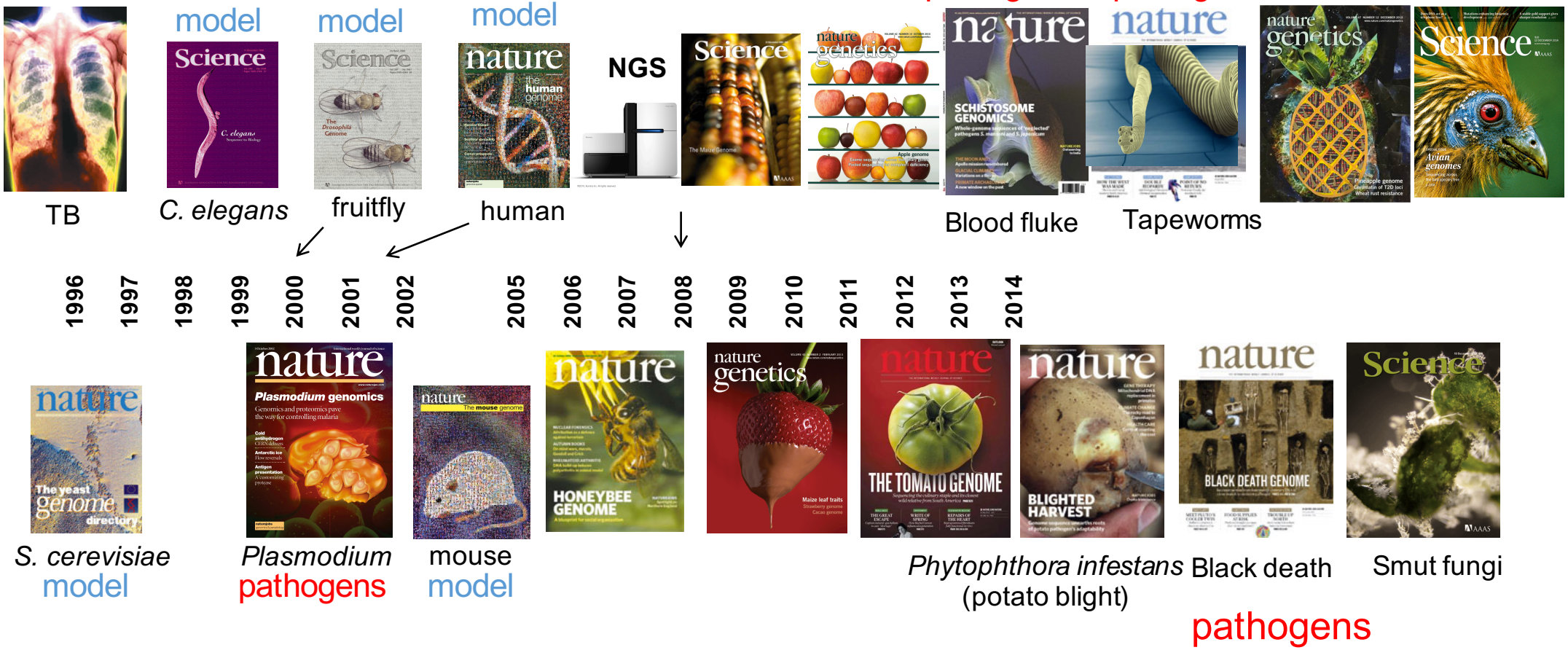
Long shredded pieces (read) = easier assembly



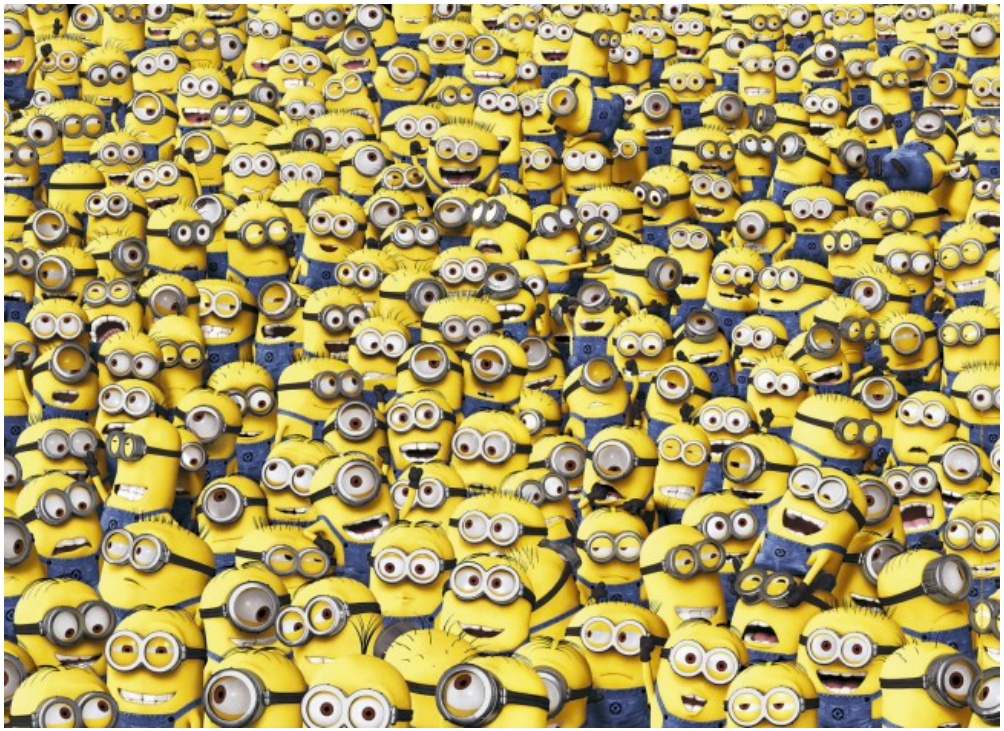
my work is now nearly finished; but as it will take me two or three more years to complete it, and as my health is far

So you want to sequence a genome

Genomics advance our understanding of organisms across tree of life



Assemble a genome is hard



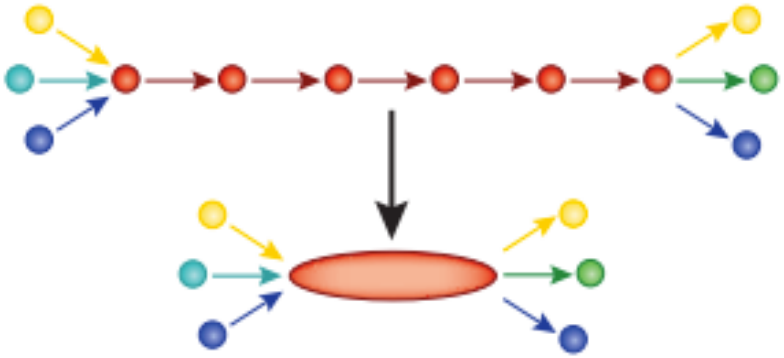
Assembly process

1. Fragment DNA and sequence

2. **Find overlaps** between reads

...AGCCTAGACCTACAGGATGCGCGACACGT
GGATGCGCGACACGTCGCATATCCGGT...

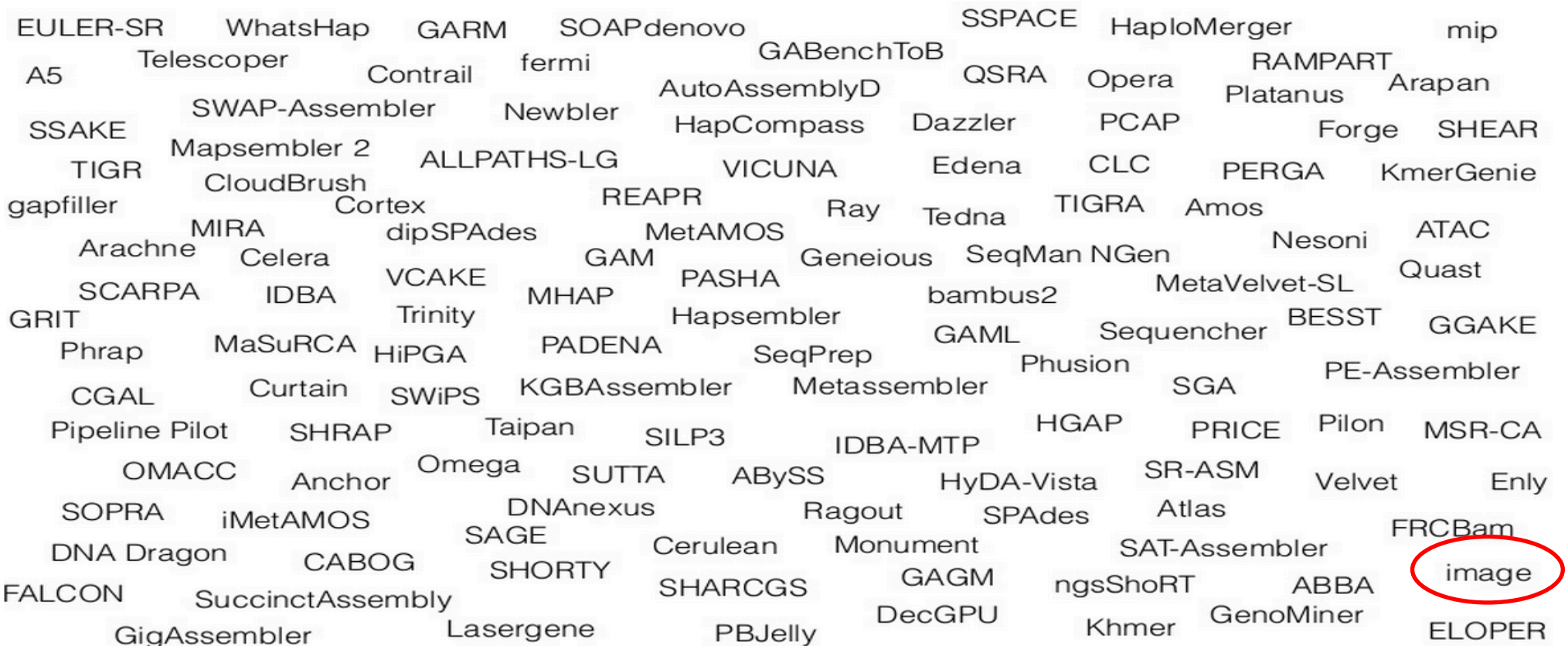
3. **Assemble overlaps** into **contigs**



4. Assemble **contigs** into **scaffolds**

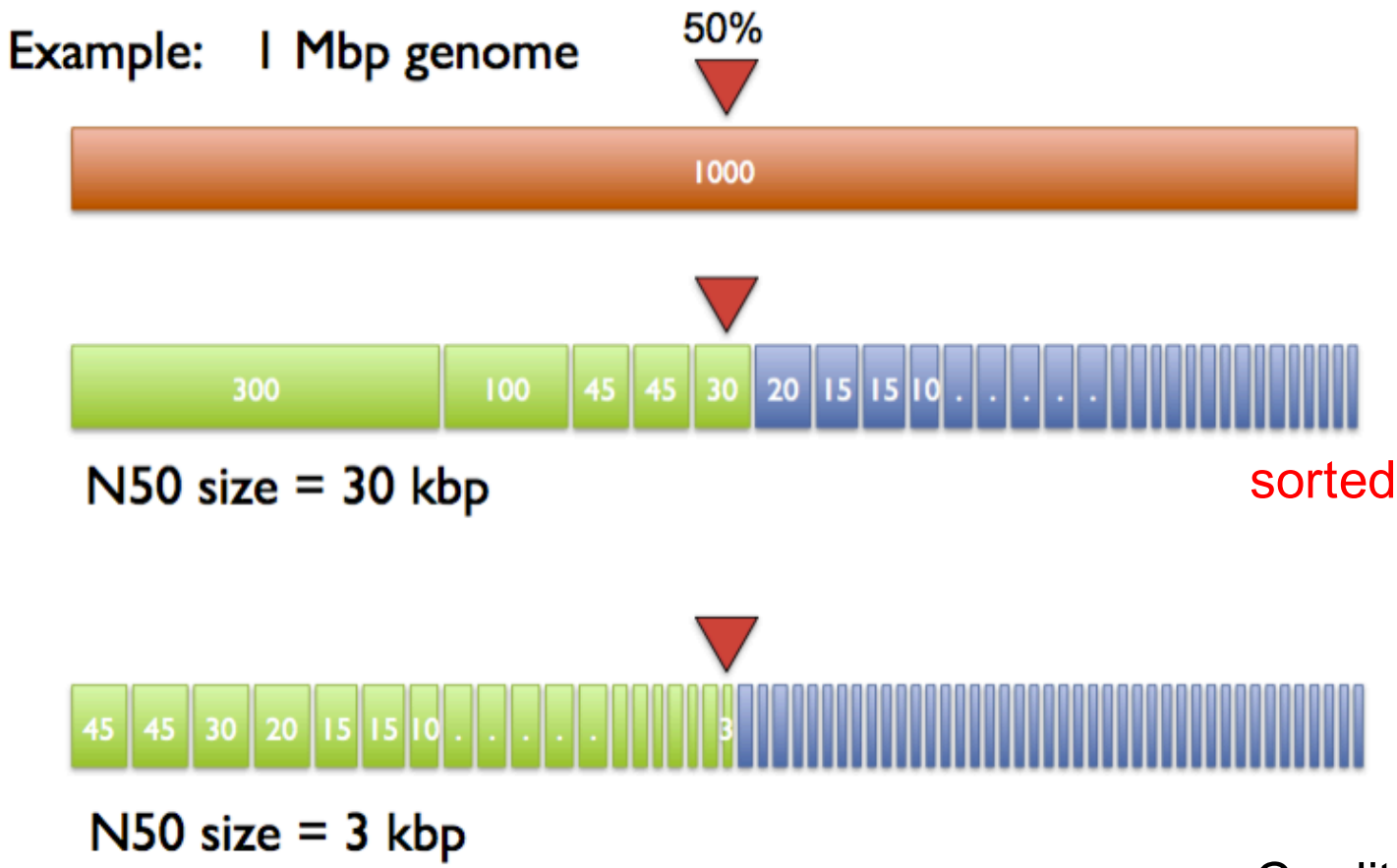
http://www.nature.com/nmeth/journal/v9/n4/pdf/nmeth.1935.pdf

Which program to choose?



Contiguity (good) is a genome? N50

Definition: 50% of genome in contigs/scaffolds of length **N50 bp** or greater



Credit: Michael Schatz

Why do we need a good assembly?

It is easier to analyse

10000 pieces vs. 23 pieces (chromosomes)

Allows more accurate representation of genes locations on genome

Is gene A close to gene B? On the same chromosome?

Transposon dynamics

Missing in most assemblies (located in NNNNNNNNNN gaps)

Responsibility to contribute to your community

Do you want others to work on the same genome / species as well?

Bottom line:

It's really no point to do one if you can't produce an accurate and useful one

Assembly qualities

	Whole Genome Representation	Sequence Status	Genes	Usability
1	Incomplete for non-repetitive regions	Small scaffolds and contigs	Incomplete genes	Markers development
2	Complete for non-repetitive regions	Medium scaffolds and contigs	Complete but 1-2 genes/contig	Gene mining
3	Complete for non-repetitive regions	Large scaffolds and contigs	Several dozens of genes/contig	Microsynteny
4	Complete for almost the whole genome	Pseudomolecules	Hundreds of genes/ contig	Any (Synteny, Candidate gene by QTLs)
5	Complete genome	Pseudomolecules	Thousands of genes/ contig	

Credit: Aureliano Bombarely

Approaches

Approaches

- Greedy extension
  - only mentioned for historical reasons

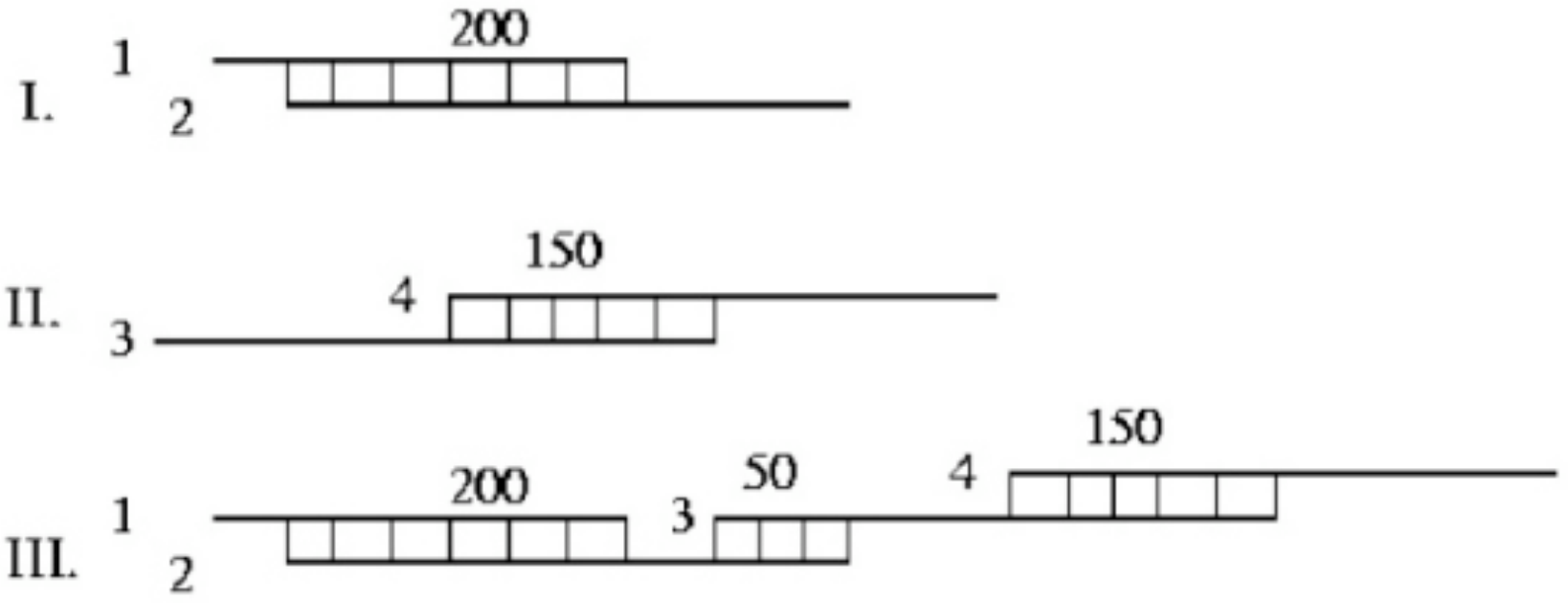
- Overlap – Layout – Consensus (**OLC**) assembly: ’traditional’ and well established method, but challenging to implement at each stage
  - Most ”old” and “newest” assemblies were produced using this approach

- de Bruijn graph (**DBG**) assembly

Greedy extension

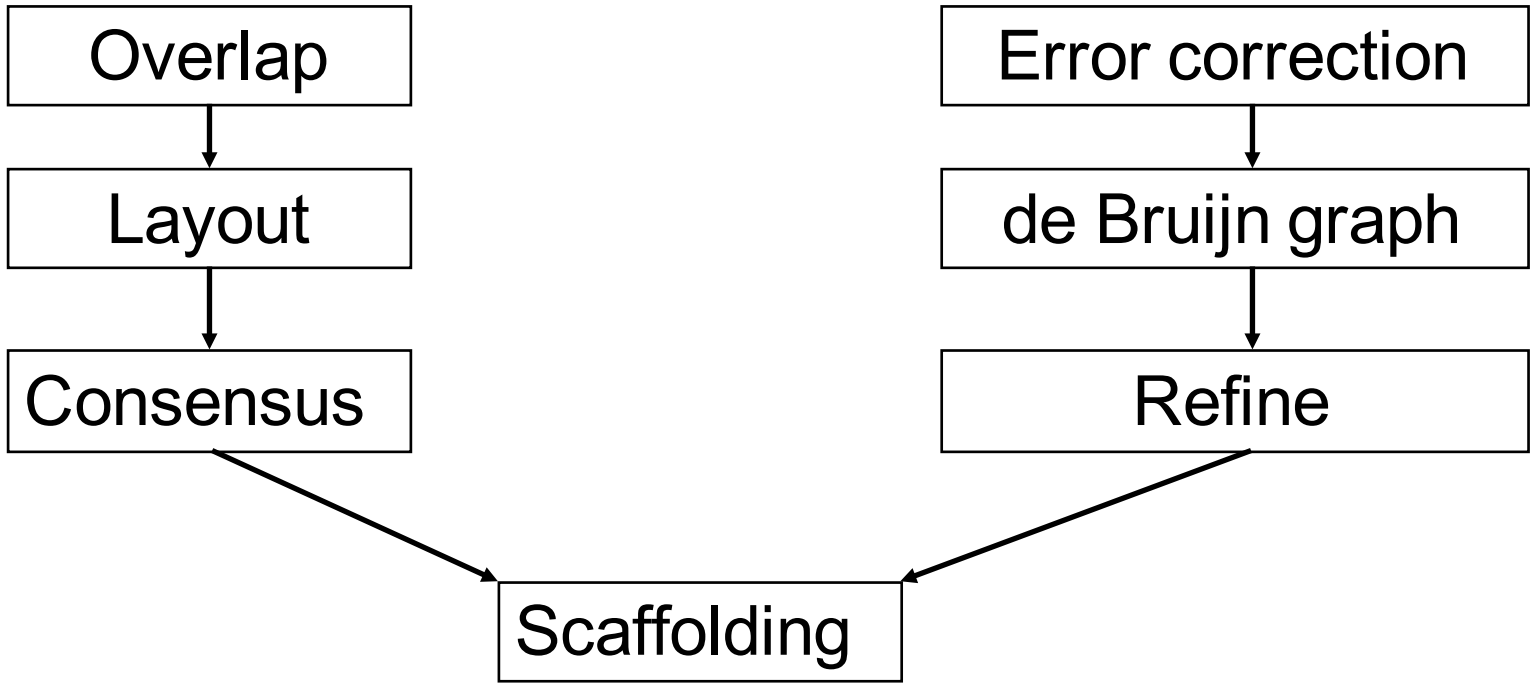
time consuming

- Oldest and not really useful in most cases



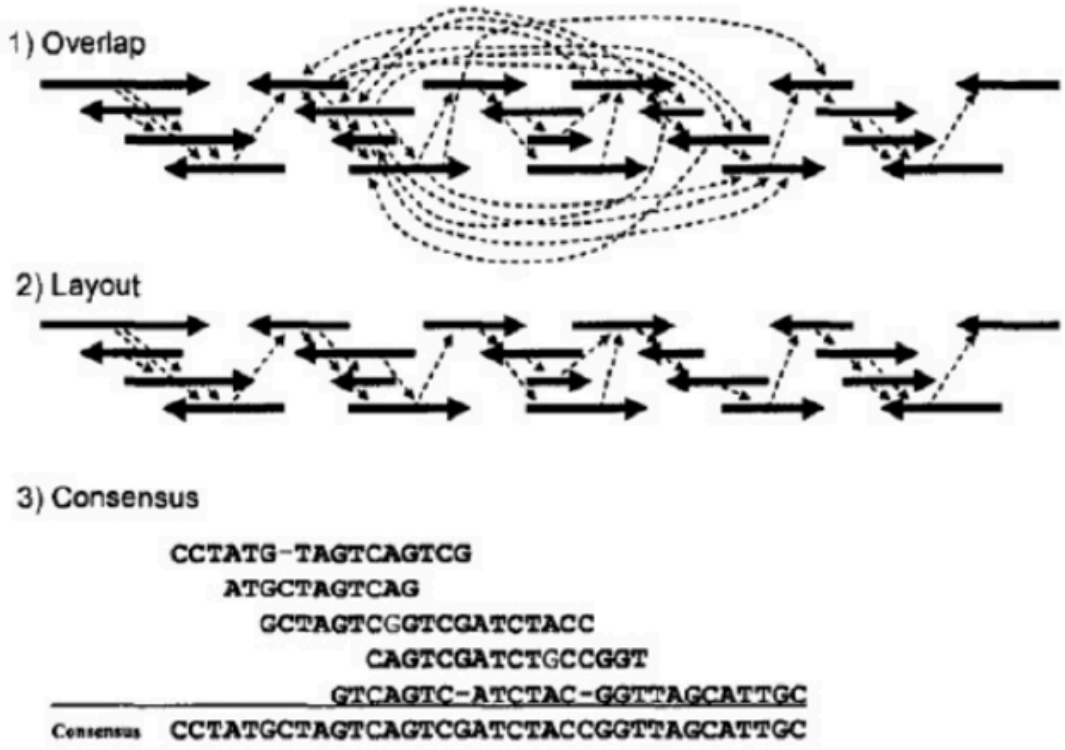
http://www.cbcb.umd.edu/research/assembly_primer.shtml

OLC and DBG assemblers



OLC approach

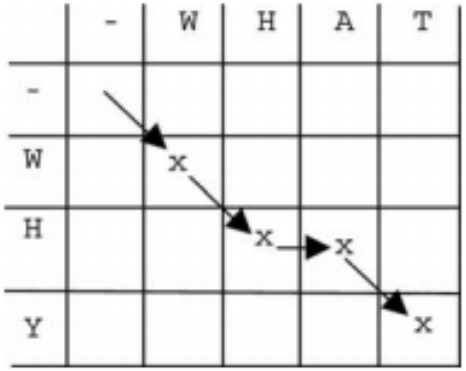
- Pairwise alignment of all reads to find **overlaps**
- **Layout** the reads to decide which read align to which
- Get **consensus** by joining join the read sequences, merging overlaps
- All three are challenging

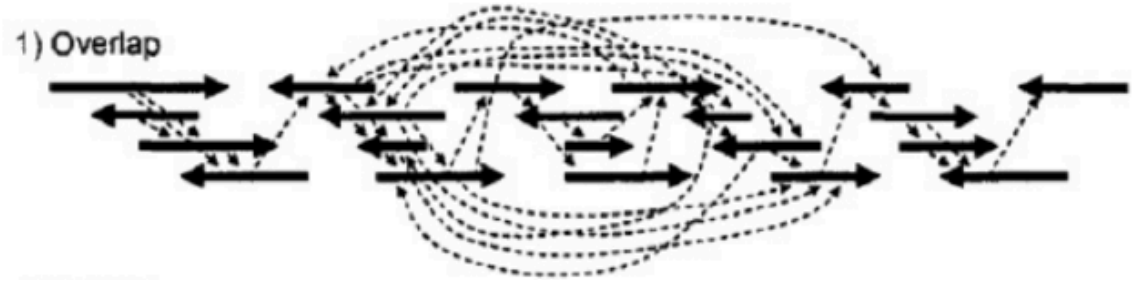


Overlap

- All vs. all pairwise alignment
  - Smith-Waterman? Blast? Kmer based? Suffix tree? Dynamic programming?

- Computationally very intensive but can be parallelised
  - Need lots of CPUs!
  - Batch1 align Batch 1 in 1st CPU
  - Batch1 align Batch 2 in 2nd CPU





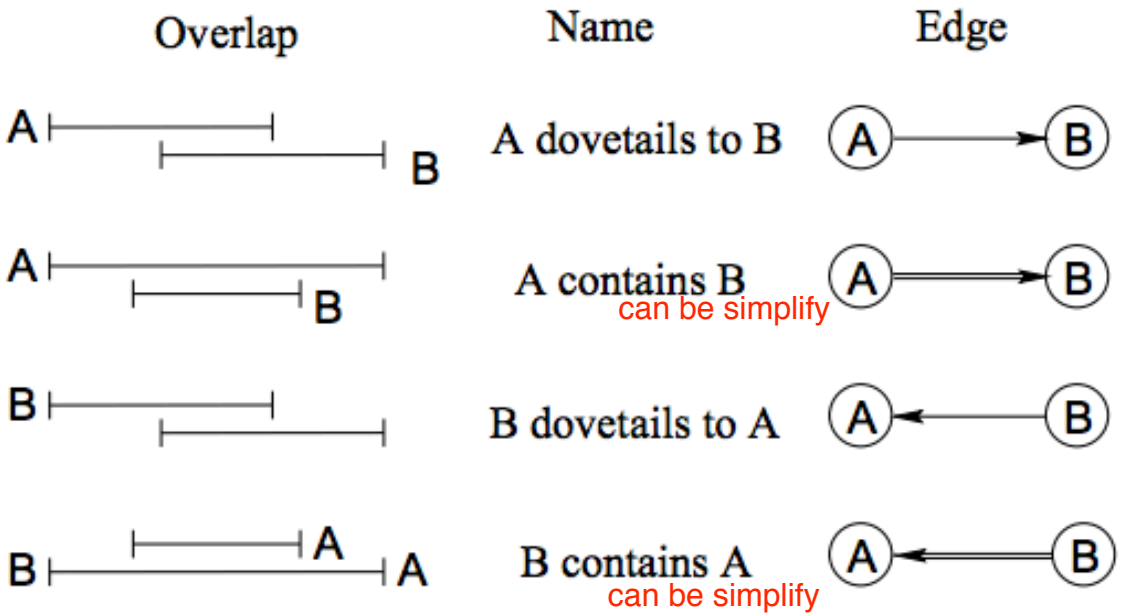
Computational Example 8.1: Pseudocode for overlap alignment

Input sequences A, B
Set $O_{i,0} = O_{0,j} = 0$ for all i, j
for $i = 1$ to n
  for $j = 1$ to m
    $O_{i,j} = \max\{O_{i-1,j} - \delta, O_{i-1,j-1} + s(a_i, b_j), O_{i,j-1} - \delta\}$
  end
end
Best overlap $= \max\{O_{i,m}, O_{n,j}; 1 \le i \le n, 1 \le j \le m\}$

Deonier *et al.,* Computational Genome Analysis

Build overlap graph

- It's common practice to represent them in **graphs**
- The actual overlaps are the edges
- Now we create the genome assembly graph



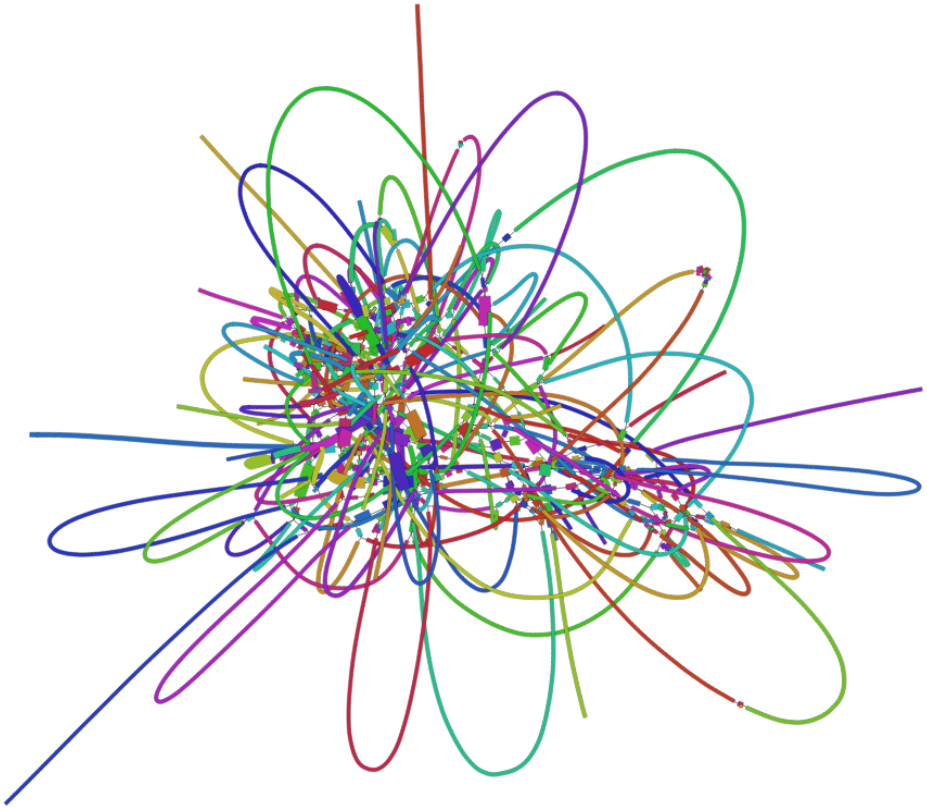
my work is now nearly finished

work is now nearly finished; but as it will take

my work is now nearly finished → work is now nearly finished; but as it will take

Kececioglu *et al.*, 1993

Some assembly graph can be complicated…



http://rrwick.github.io/Bandage/images/screenshots/screenshot02.png

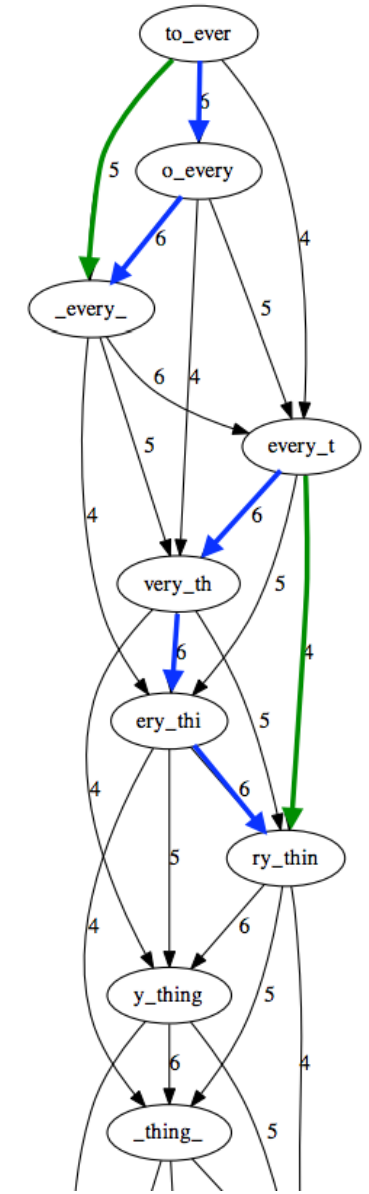
Layout

Order the reads into a consistent manner

Anything redundant about this part of overlap graph?

Some edges can be inferred from other edges

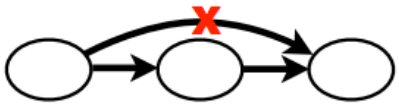
E.g., green edge can be inferred from blue



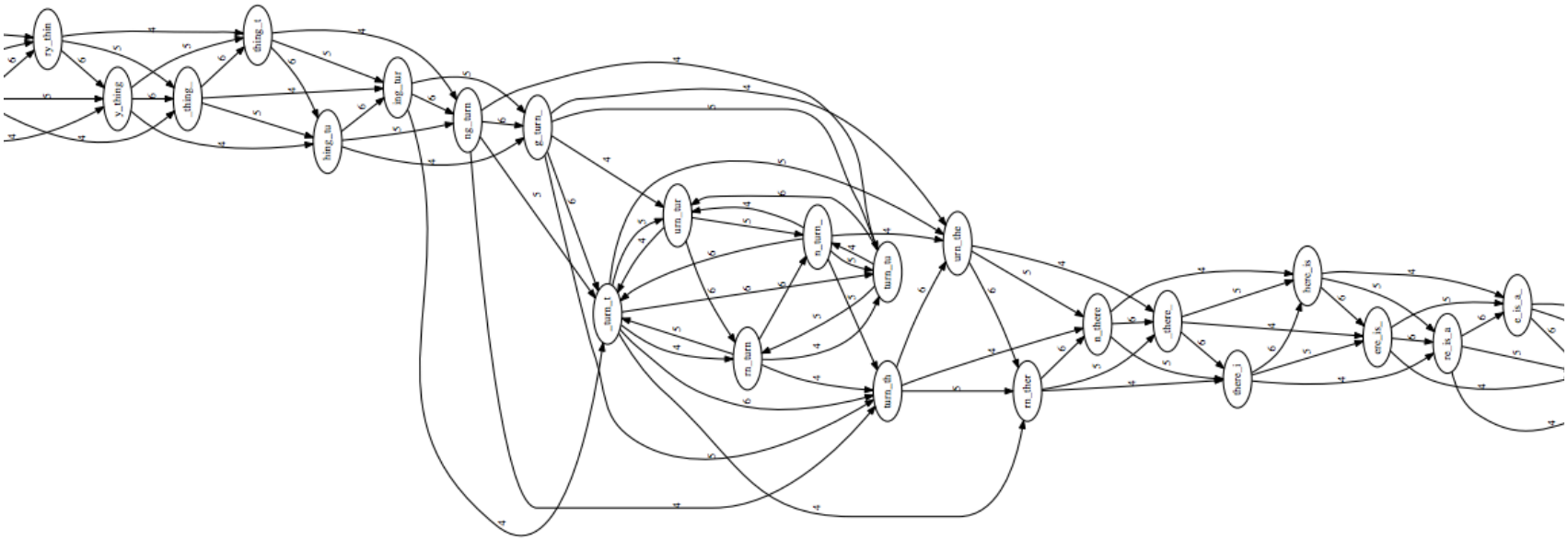
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_olc.pdf

Layout

Remove transitively-inferrible edges, starting with edges that skip one node:



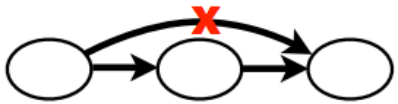
Before:



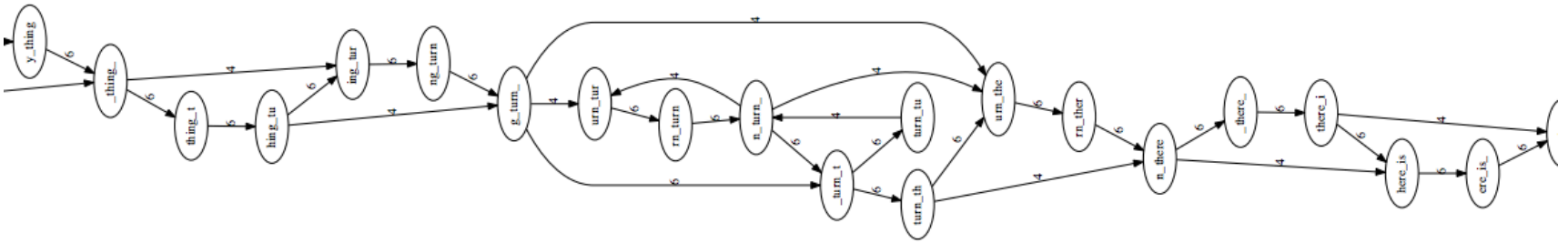
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_olc.pdf

Layout

Remove transitively-inferrible edges, starting with edges that skip one node:



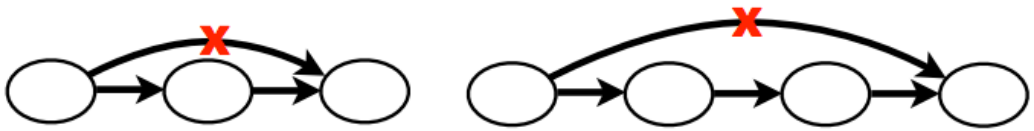
After:



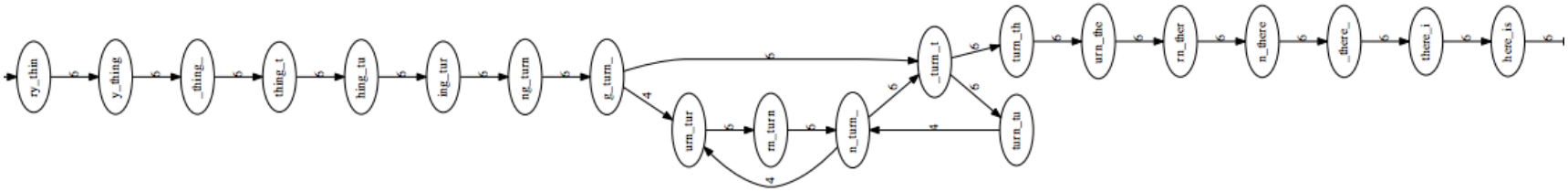
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_olc.pdf

Layout

Remove transitively-inferrible edges, starting with edges that skip one *or two* nodes:



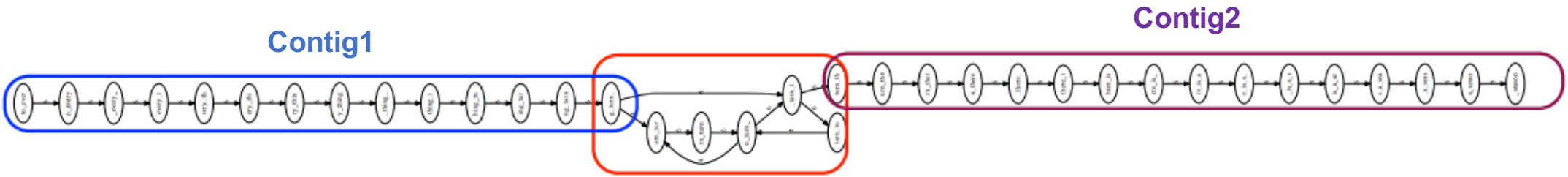
After:



Even simpler

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_olc.pdf

Layout



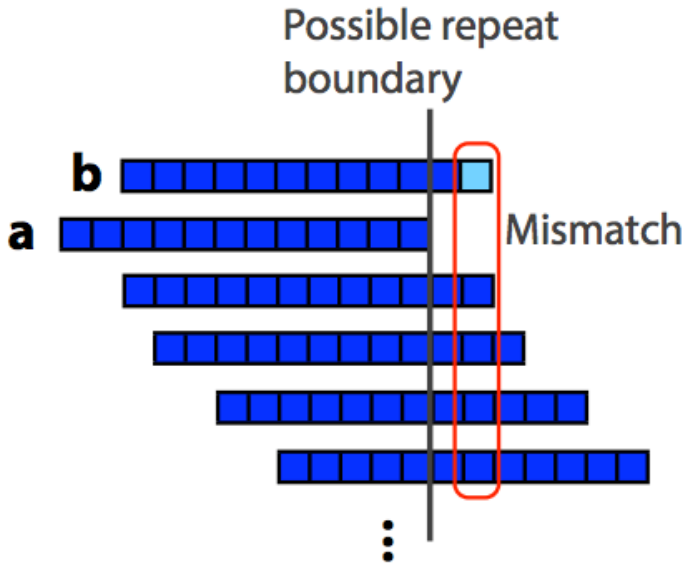
Unresolvable repeat

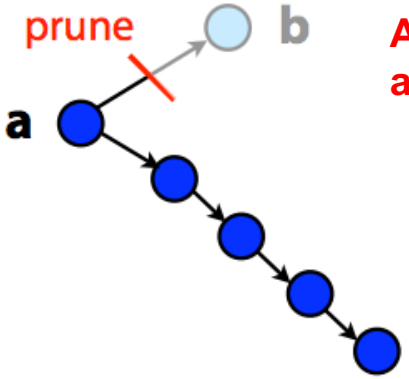
Depending on assemblers, they may result in 1 or 2 contigs

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_olc.pdf

Layout – can we do more?

In practice, layout step also has to deal with spurious subgraphs, e.g. because of sequencing error



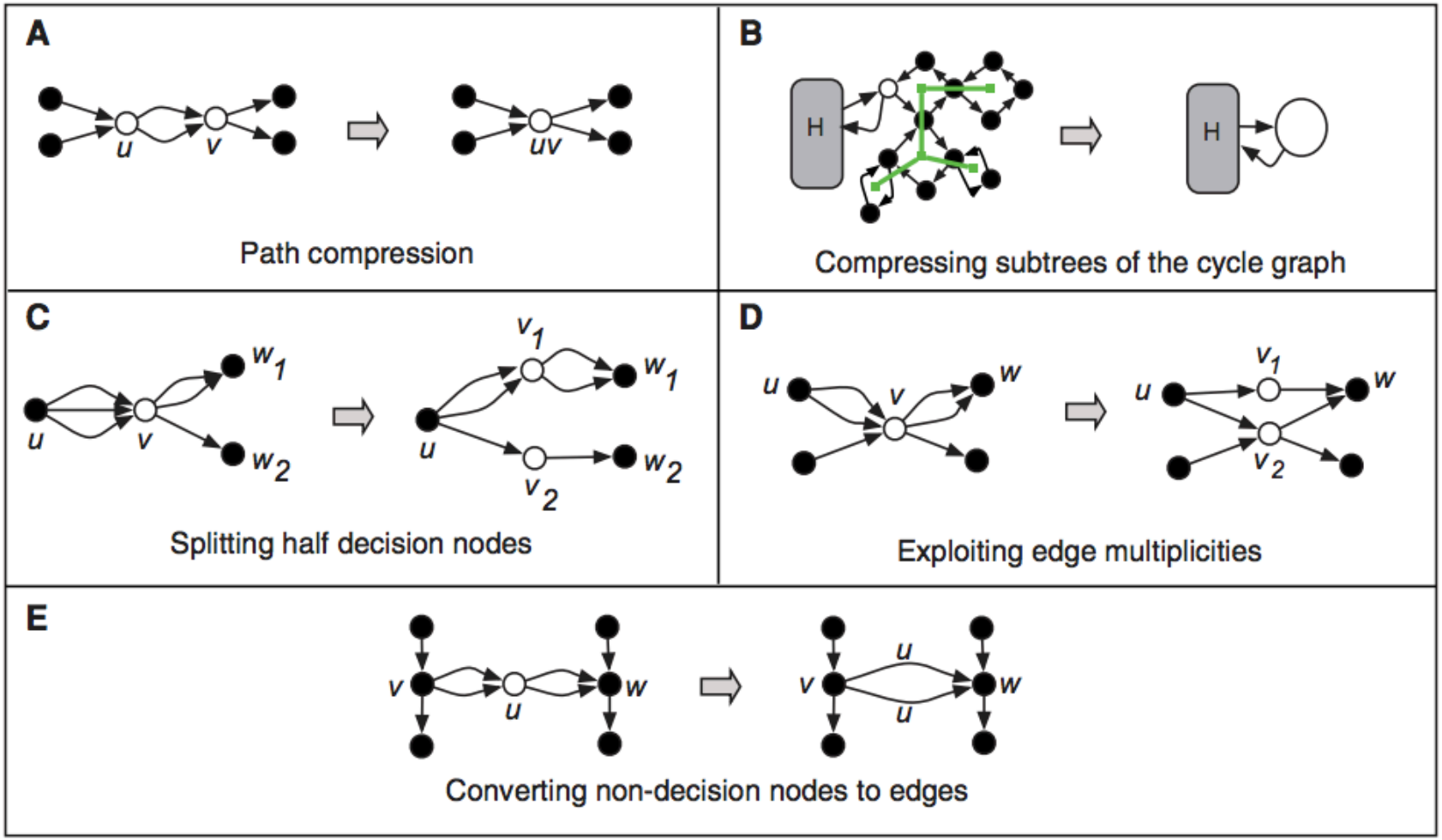


Also often referred as trim the edges

Mismatch could be due to sequencing error or repeat. Since the path through **b** ends abruptly we might conclude it's an error and prune **b**.

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_olc.pdf

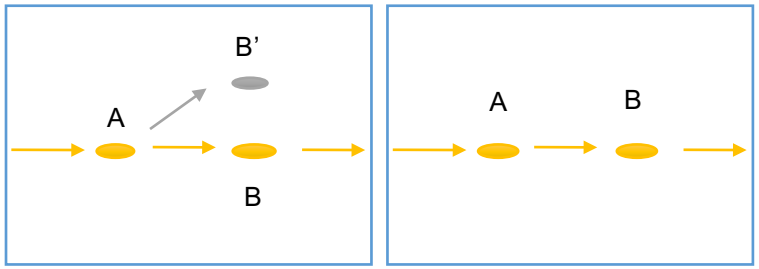
Usefulness of graph transformation



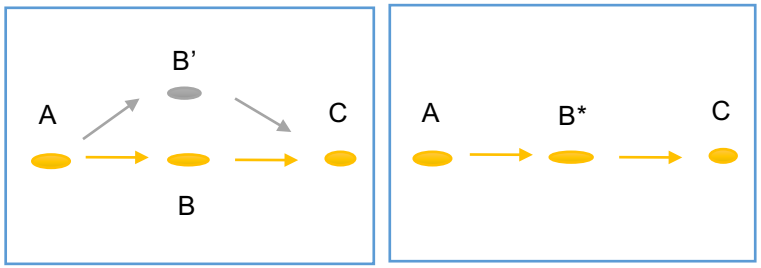
Kingsford *et al.,* (2010)

Error correction in graph

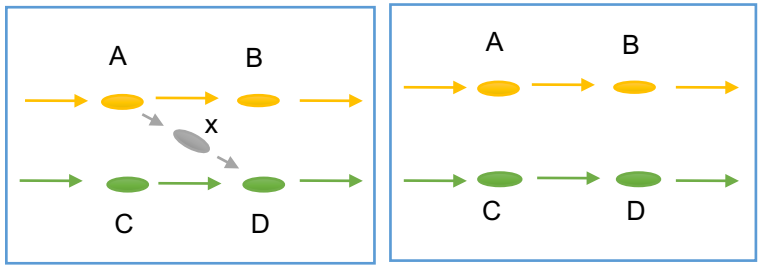
- Errors at end of read
  - Trim off ‘dead-end’ tips



- Errors in middle of read
  - Pop Bubbles



- Chimeric Edges
  - Clip short, low coverage nodes

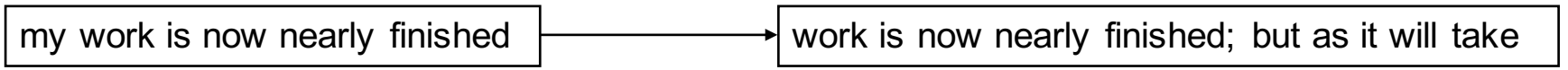


Credit: Michael Schatz

OLC Assemblers

- Mostly used in the Sanger sequencing era
  - Celera, Phusion, PCAP, Arachne

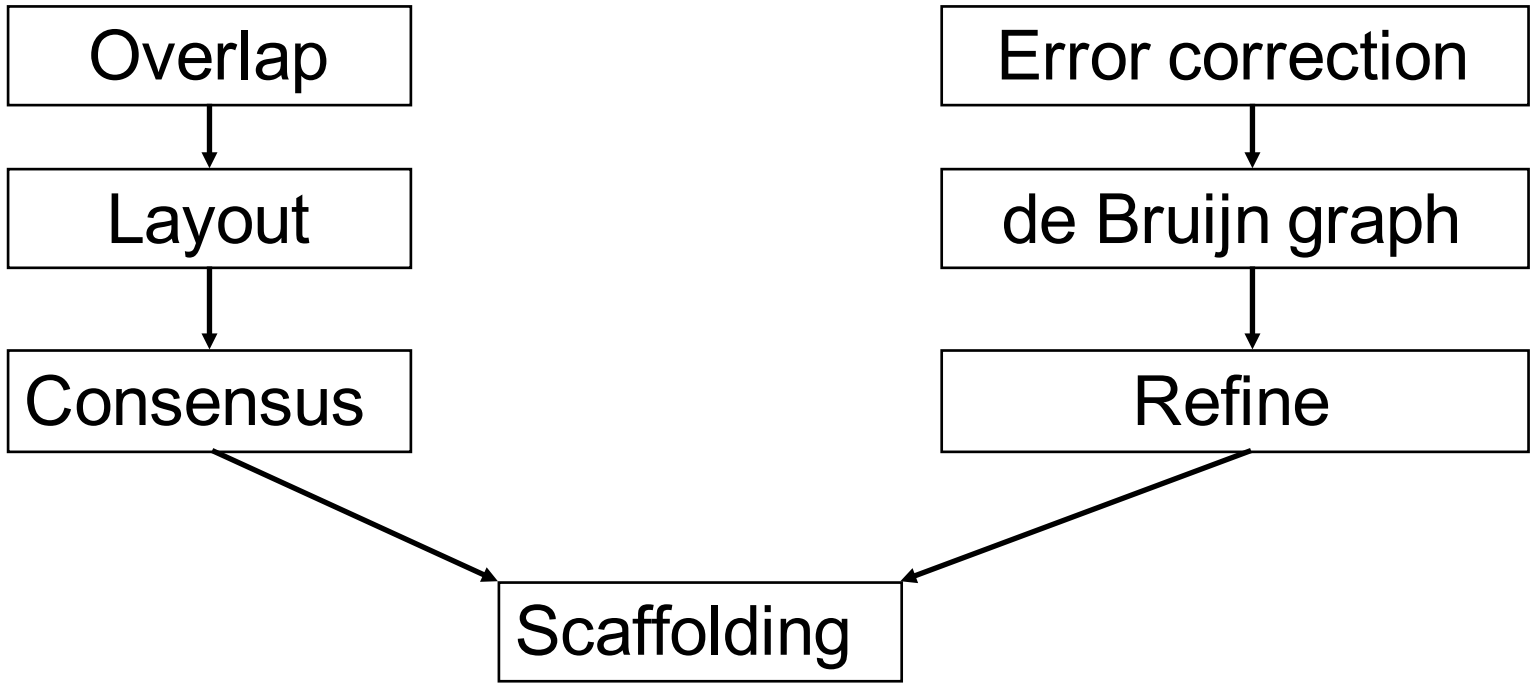
- Disadvantages of OLC
  - Computing overlaps is slow
  - 5 billion reads -> takes **400 years** to compute overlaps if 1 million overlap per second
  - Overlap graph is big and complicated
    - One node per read
    - Number of edges grows superlinearly with number of reads



- When 2nd generation dataset first arrived
  - Millions and millions of reads
  - Short read length – difficult to build sufficient overlap

Break

OLC and DBG assemblers



k-mer

"*k*-mer" is a substring of length *k*

S: GGCGATTCATCG

A 4-mer of *S*: ATTC

All 3-mers of *S*:

GGC
GCG
CGA
GAT
ATT
TTC
TCA
CAT
ATC
TCG

mer: from Greek meaning "part"

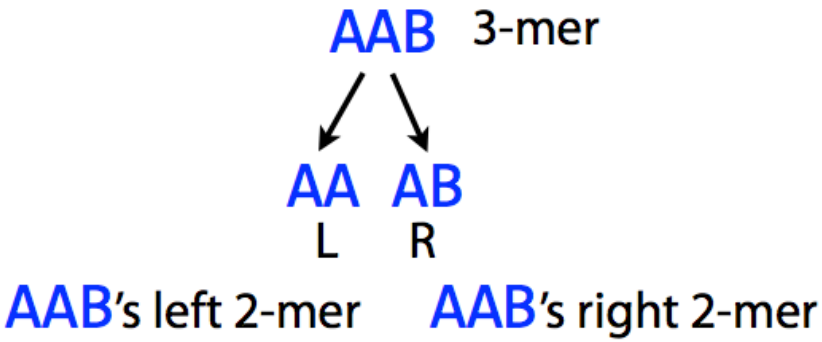
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph

We start with a collection of reads of **3bp** from the reference genome **AAABBBA**

AAA, AAB, ABB, BBB, BBA

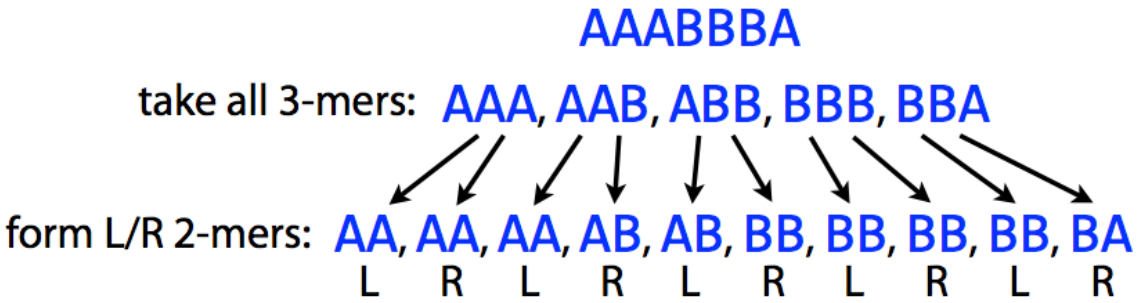
AAB is a *k*-mer (*k* = 3). AA is its *left* *k*-1-mer, and AB is its right *k*-1-mer.



http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph

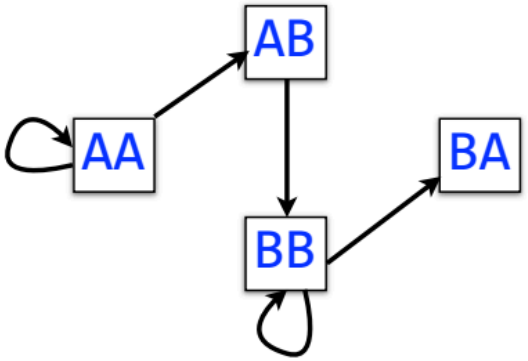
Take each length-3 input string and split it into two overlapping substrings of length 2. Call these the *left* and *right 2-mers*.



From these 2-mers, only AA, AB, BA, BA are present **(they will be nodes)**

Let 2-mers be nodes in a new graph. Draw a directed edge from each left 2-mer to corresponding right 2-mer:

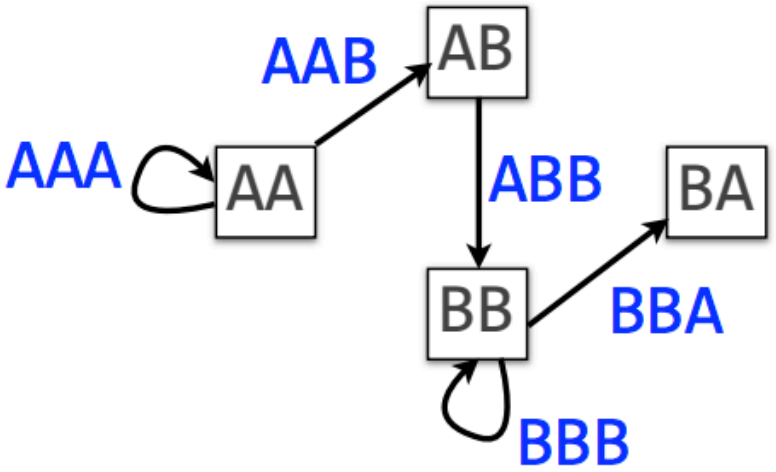
So AAB will be AA → AB



Each *edge* in this graph corresponds to a length-3 input string

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph



How do we get contigs from the graph?

Intuitively we walk and visited all edges and node of the graph, but how?

An edge corresponds to an overlap (of length *k*-2) between two *k*-1 mers.
More precisely, it corresponds to a *k*-mer from the input.

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph is a directed multigraph

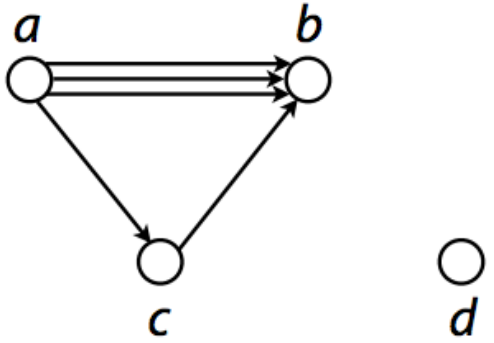
Directed **multigraph** $G(V, E)$ consists of set of *vertices*, V and **multiset** of *directed edges*, E

Otherwise, like a directed graph

Node's *indegree* = # incoming edges

Node's *outdegree* = # outgoing edges

De Bruijn graph is a directed multigraph



$V = \{ a, b, c, d \}$

$E = \{ (a, b), (a, b), (a, b), (a, c), (c, b) \}$

Repeated

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

Eulerian walk definitions and statement

1.A **directed graph** is a *graph* in which each edge has a direction, usually represented as an arrow from a node *v* to a node *w*.

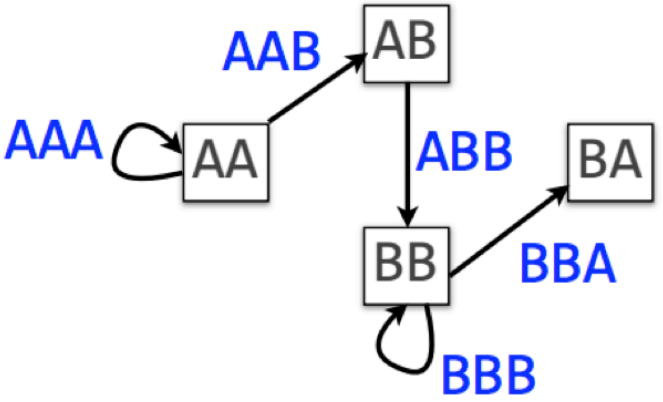
Yes

2.Graph is connected if each node can be reached by some other node.

Yes

3.Node is **balanced** if indegree equals outdegree Node is **semi-balanced** if indegree differs from outdegree by 1

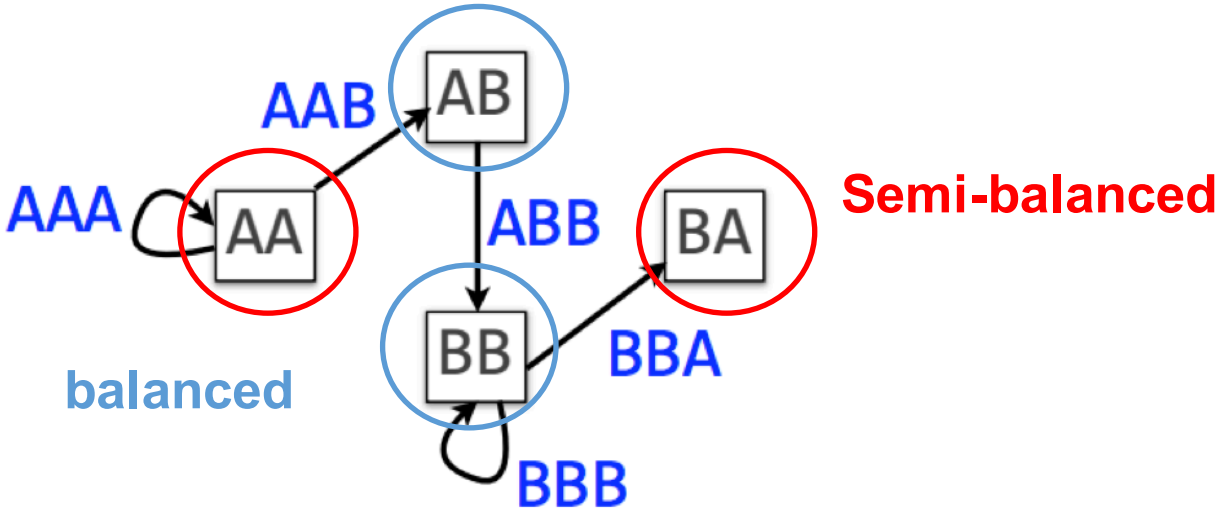
4.A directed, connected graph is Eulerian if and only if it has **at most 2 semi-balanced nodes** and all other nodes are balanced



https://en.wikipedia.org/wiki/Eulerian_path

Eulerian walk definitions and statement

A **directed, connected graph** is Eulerian if and only if **it has at most 2 semi-balanced nodes and all other nodes are balanced**



Is it Eulerian? Yes

Argument 1: AA → AA → AB → BB → BB → BA

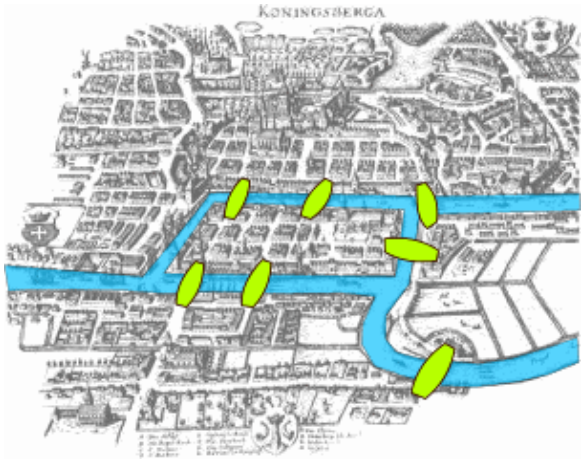
Argument 2: AA and BA are semi-balanced, AB and BB are balanced

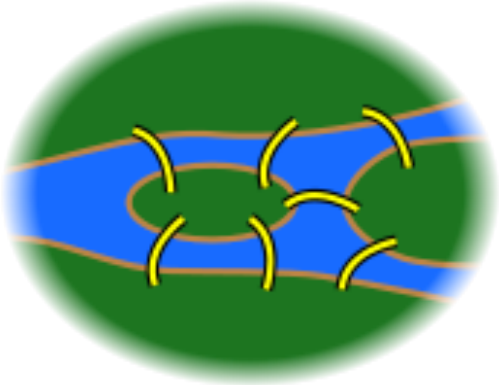
Then we can search for Eulerian path from the graph:
Eulerian walk visits each edge exactly once ****

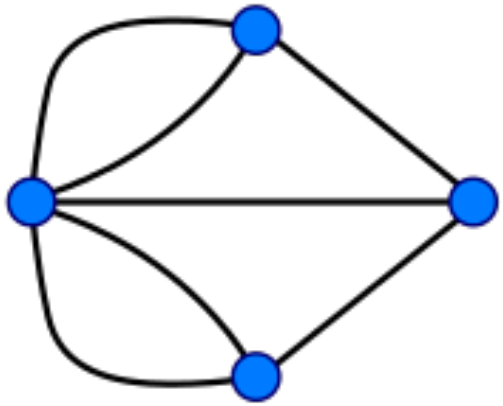
Eulerian walk

In graph theory, an **Eulerian trail** (or **Eulerian path**) is a trail in a graph which visits every edge exactly once. Similarly, an **Eulerian circuit** or **Eulerian cycle** is an Eulerian trail which starts and ends on the same vertex. They were first discussed by Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736. Mathematically the problem can be stated like this:

Given the graph in the image, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) which visits each edge exactly once?







PS. This graph is Not Eulerian

https://en.wikipedia.org/wiki/Eulerian_path

De Bruijn graph

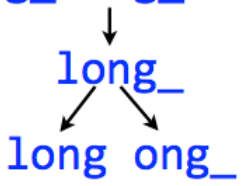
A procedure for making a De Bruijn graph
for a genome

Assume *perfect sequencing* where each length-*k*
substring is sequenced exactly once with no errors

Pick a substring length *k*: 5

Start with an input string: a_long_long_long_time

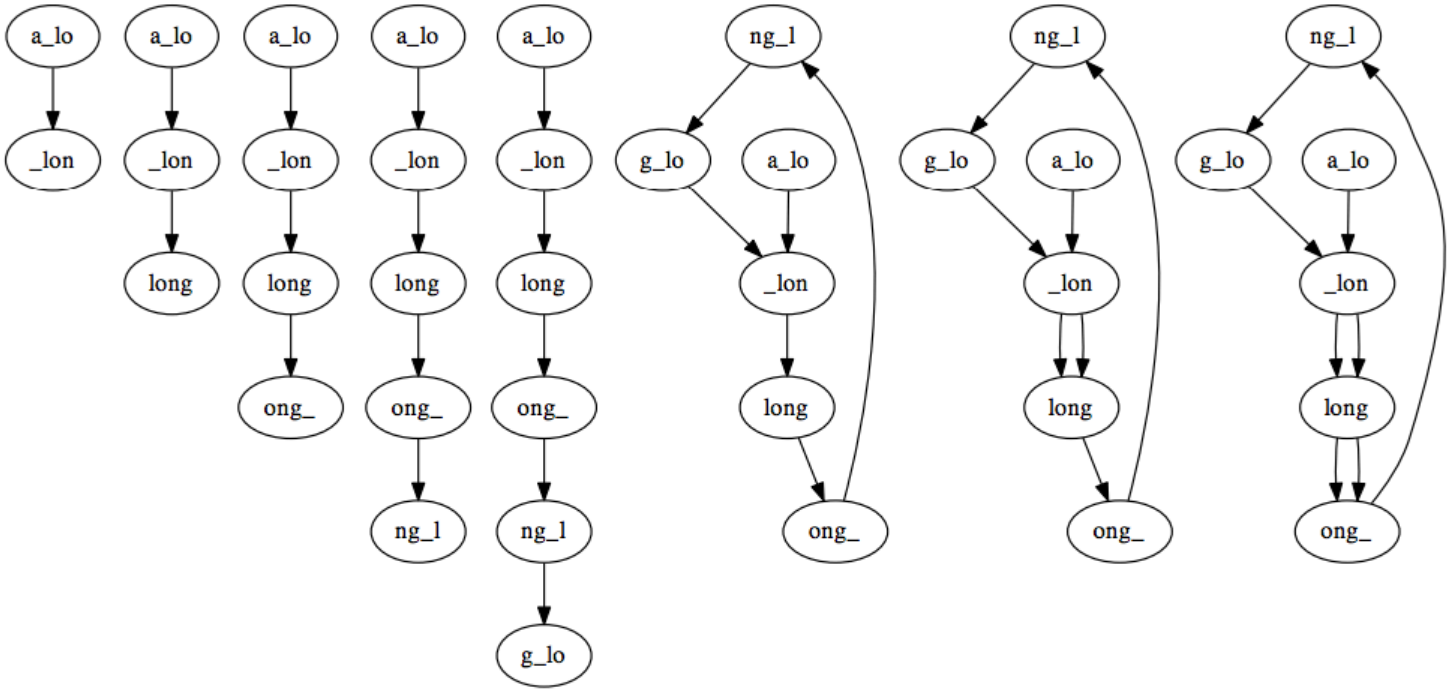
Take each *k* mer and split
into left and right *k*-1 mers



Add k-1 mers as nodes to De Bruijn graph
(if not already there), add edge from left *k*-1
mer to right *k*-1 mer

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph

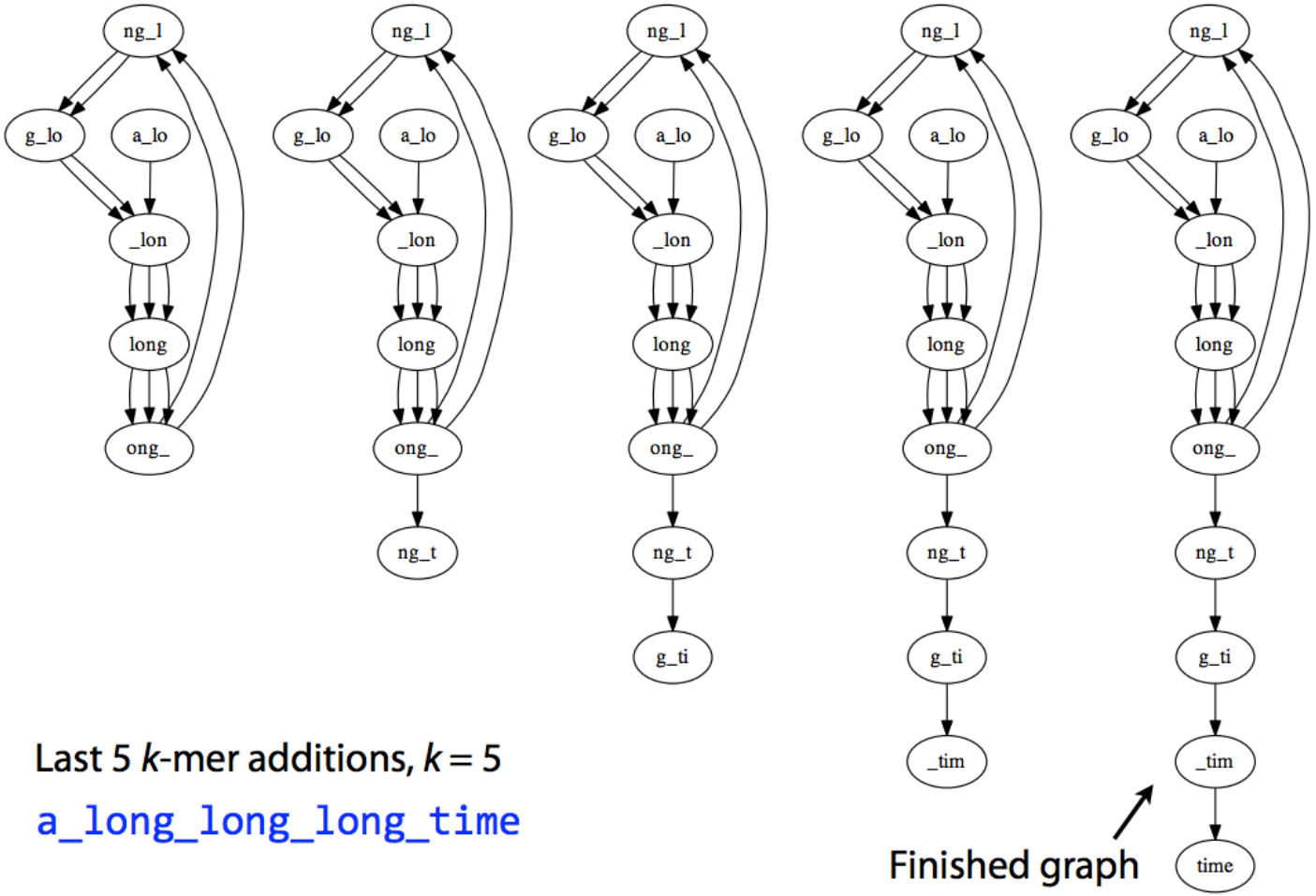


First 8 *k*-mer additions, $k = 5$

```
a_long_long_long_time
```

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph



http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph

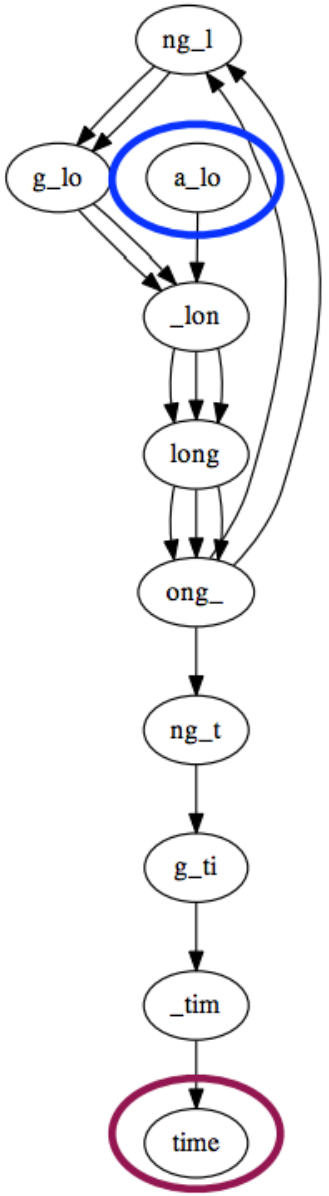
With perfect sequencing, this procedure always yields an Eulerian graph. Why?

Node for *k*-1-mer from left end is semi-balanced with one more outgoing edge than incoming *

Node for *k*-1-mer at right end is semi-balanced with one more incoming than outgoing *

Other nodes are balanced since # times *k*-1-mer occurs as a left *k*-1-mer = # times it occurs as a right *k*-1-mer

* Unless genome is circular

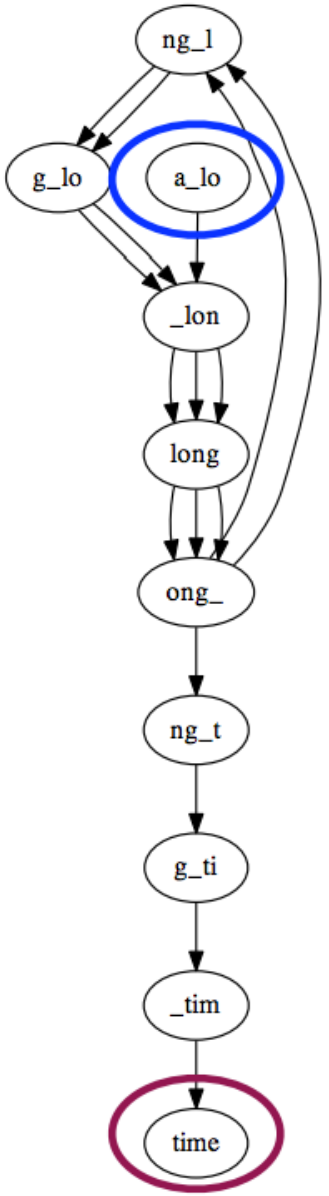


http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph with actual data

Assuming perfect sequencing, procedure yields graph with Eulerian walk that can be found efficiently.

We saw cases where Eulerian walk corresponds to the original superstring. Is this always the case?



http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

When k-mer is repeat

No: graph can have multiple Eulerian walks, only one of which corresponds to original superstring

Right: graph for ZABCDABEFABY, $k = 3$

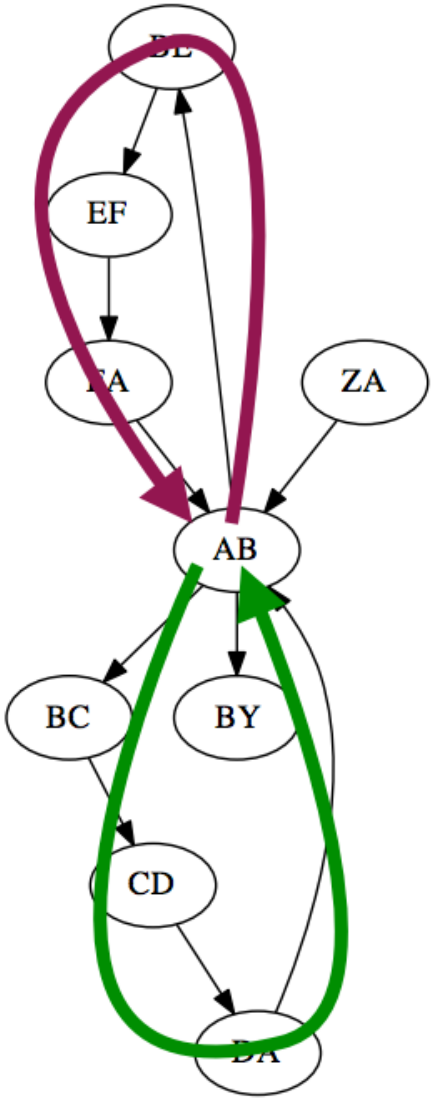
Alternative Eulerian walks:

ZA → AB → BE → EF → FA → AB → BC → CD → DA → AB → BY

ZA → AB → BC → CD → DA → AB → BE → EF → FA → AB → BY

These correspond to two edge-disjoint directed cycles joined by node AB

AB is a repeat: ZABCDABEFABY



http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

When k-mer is repeat (in practice)

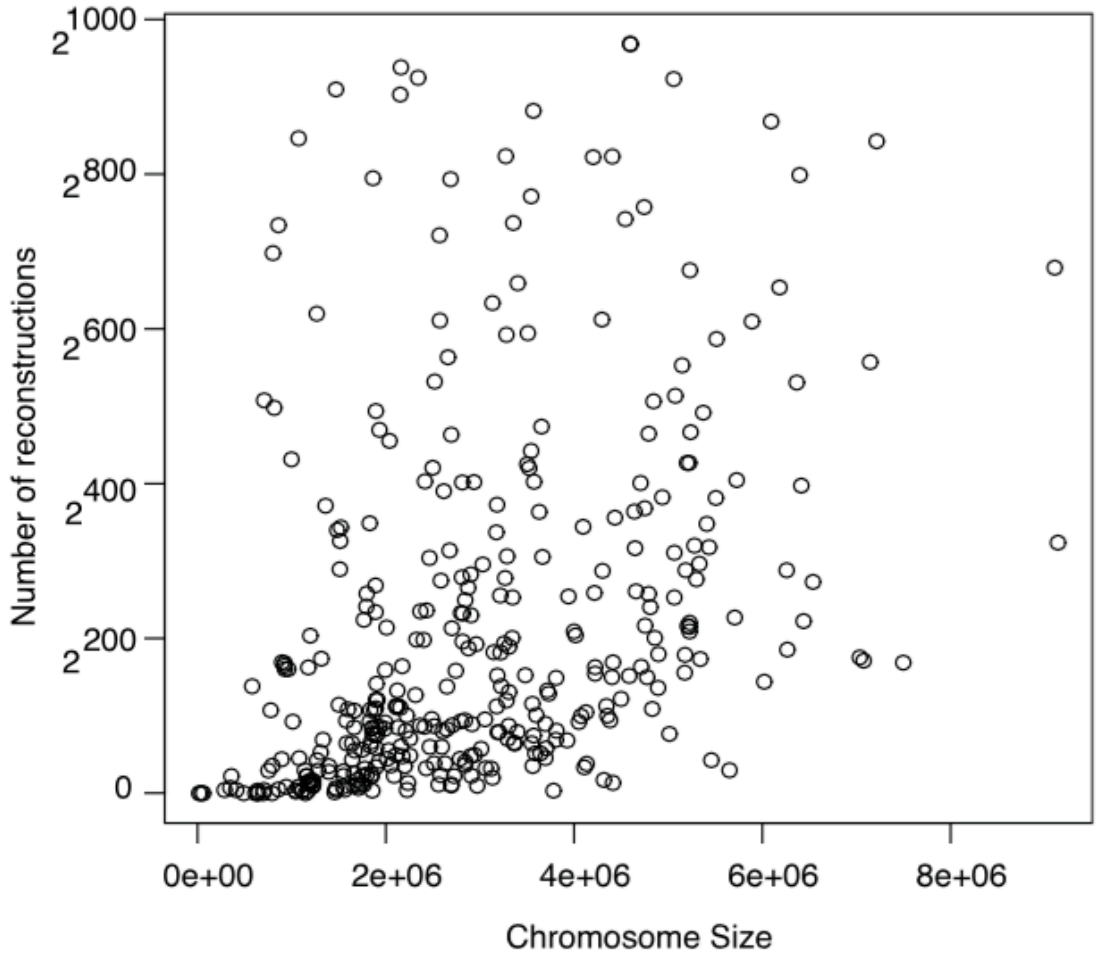


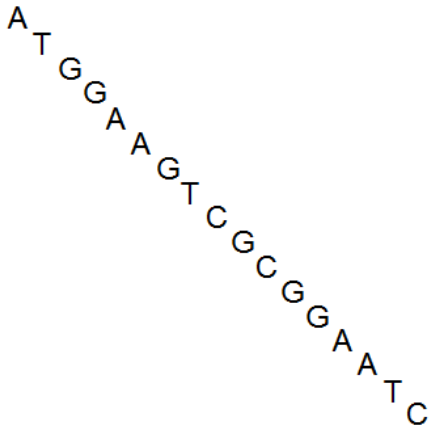
Figure 2 Number of words consistent with genome graphs. The

Kingsford *et al.,* 2010

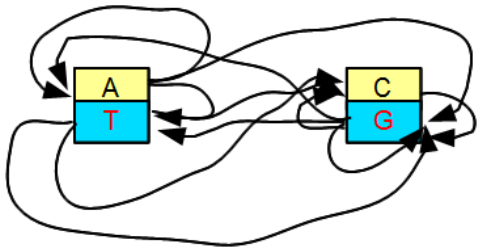
Impact of changing kmer size

sequence ATGGAAGTCGCGGAATC

1mers



de Bruijn graph



Example of a kmer of 1 basically means A,C,T,G are all repeats..

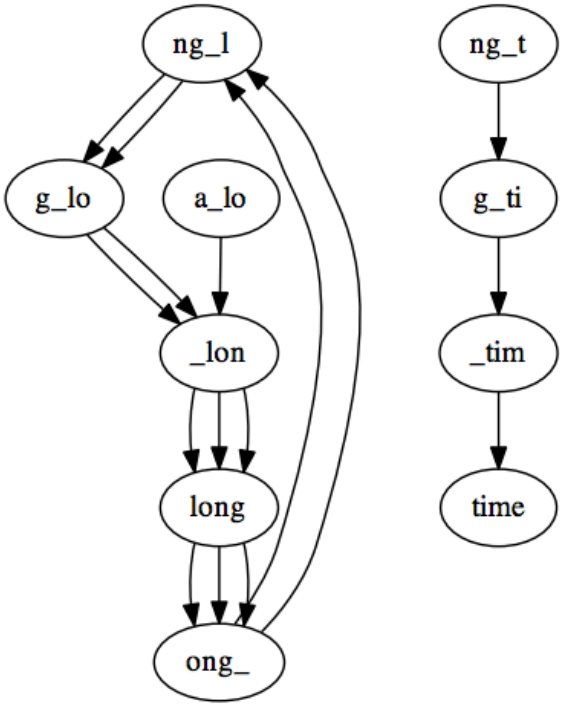
Larger kmer will span more small repeat less than kmer size, but likely to have less overlap

http://www.homolog.us/Tutorials/index.php?p=2.4&s=1

Low coverage = disconnected graph

Gaps in coverage can lead to *disconnected* graph

Graph for `a_long_long_long_time`, $k = 5$ but *omitting* `ong_t` :



Connected components are individually Eulerian, overall graph is not

Illumina sequence can't seq the AT rich region

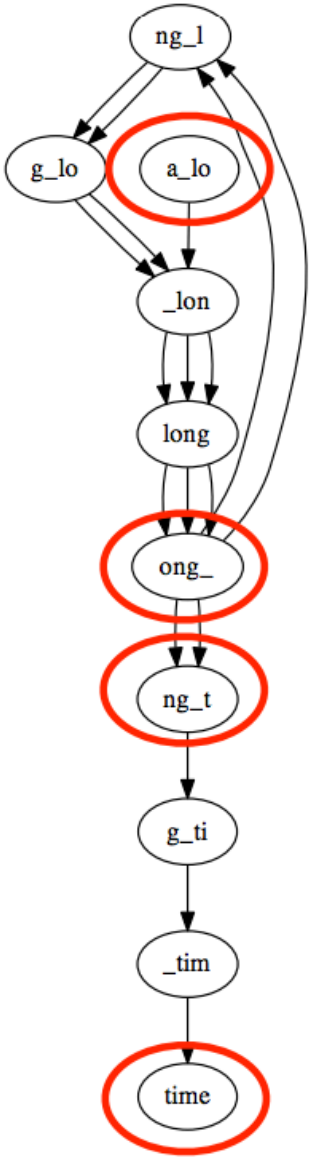
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

Coverage difference = not Eulerian

Differences in coverage also lead to non-Eulerian graph

Graph for `a_long_long_long_time`, $k = 5$ but with *extra copy* of `ong_t` :

Graph has 4 semi-balanced nodes, isn't Eulerian



http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

De Bruijn graph

Gaining assembly as Eulerian walk is appealing, not many practical cases impede this:

- Uneven coverage, sequencing errors, make graph non-Eulerian
- Repeats produces many possible walks

But there is one major advantage of De Bruijn graph over OLC

- Computationally efficient

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

Efficiency

Assume you have 5 billion reads to assemble

- Not uncommon nowadays in some plant species

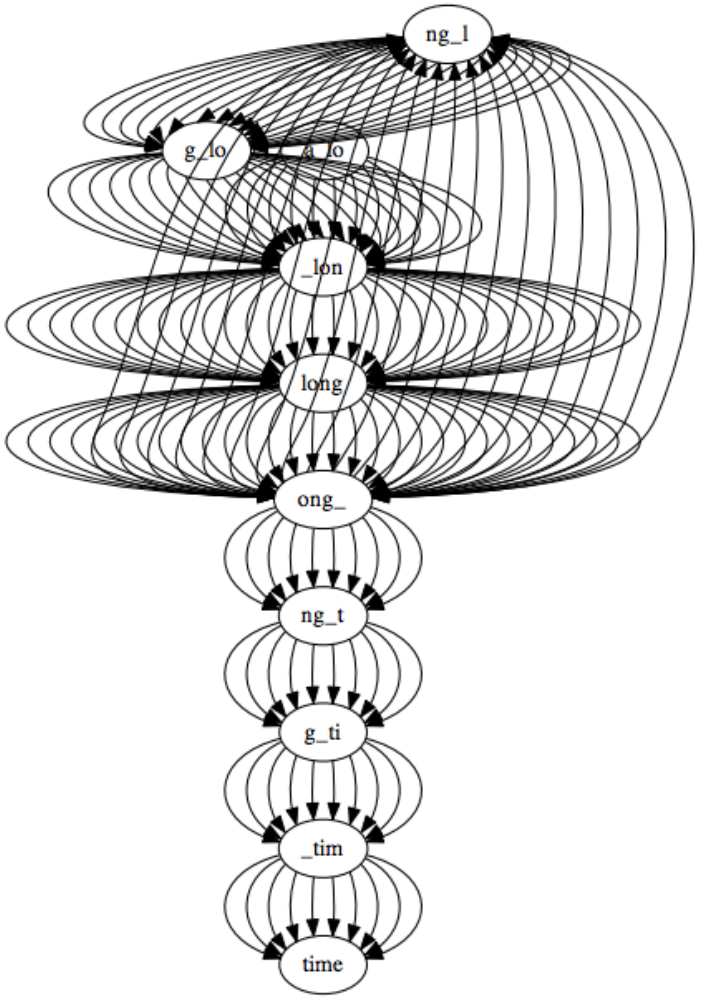
OLC: 12.5 quadrillion overlaps to compute first
Even 1 million overlap per sec will equate to **400 years**

DBG: depends on genome size

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

Size of De Bruijn graph depends on genome size

Usually ~50X paired end reads in coverage



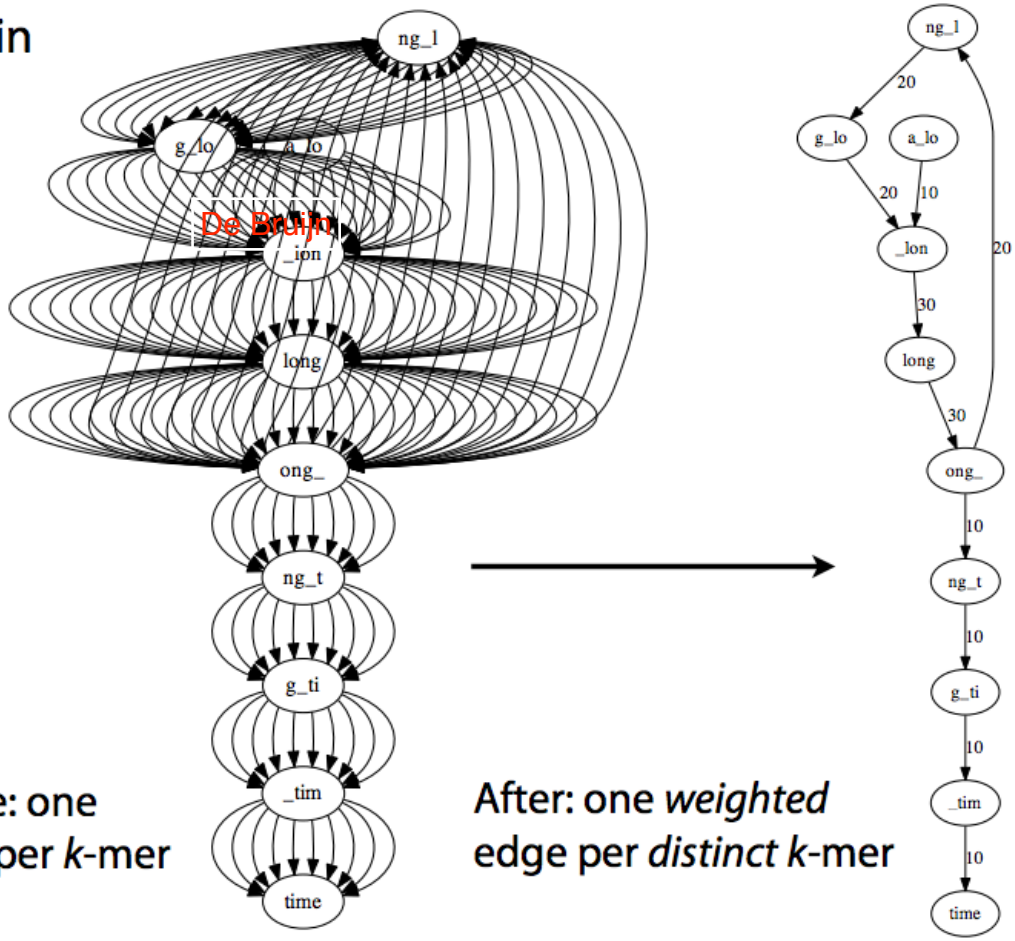
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

Size of De Bruijn graph depends on genome size

Same edge might appear in dozens of copies; let's use edge *weights* instead

Weight = # times *k*-mer occurs

Using weights, there's one *weighted* edge for each *distinct k*-mer

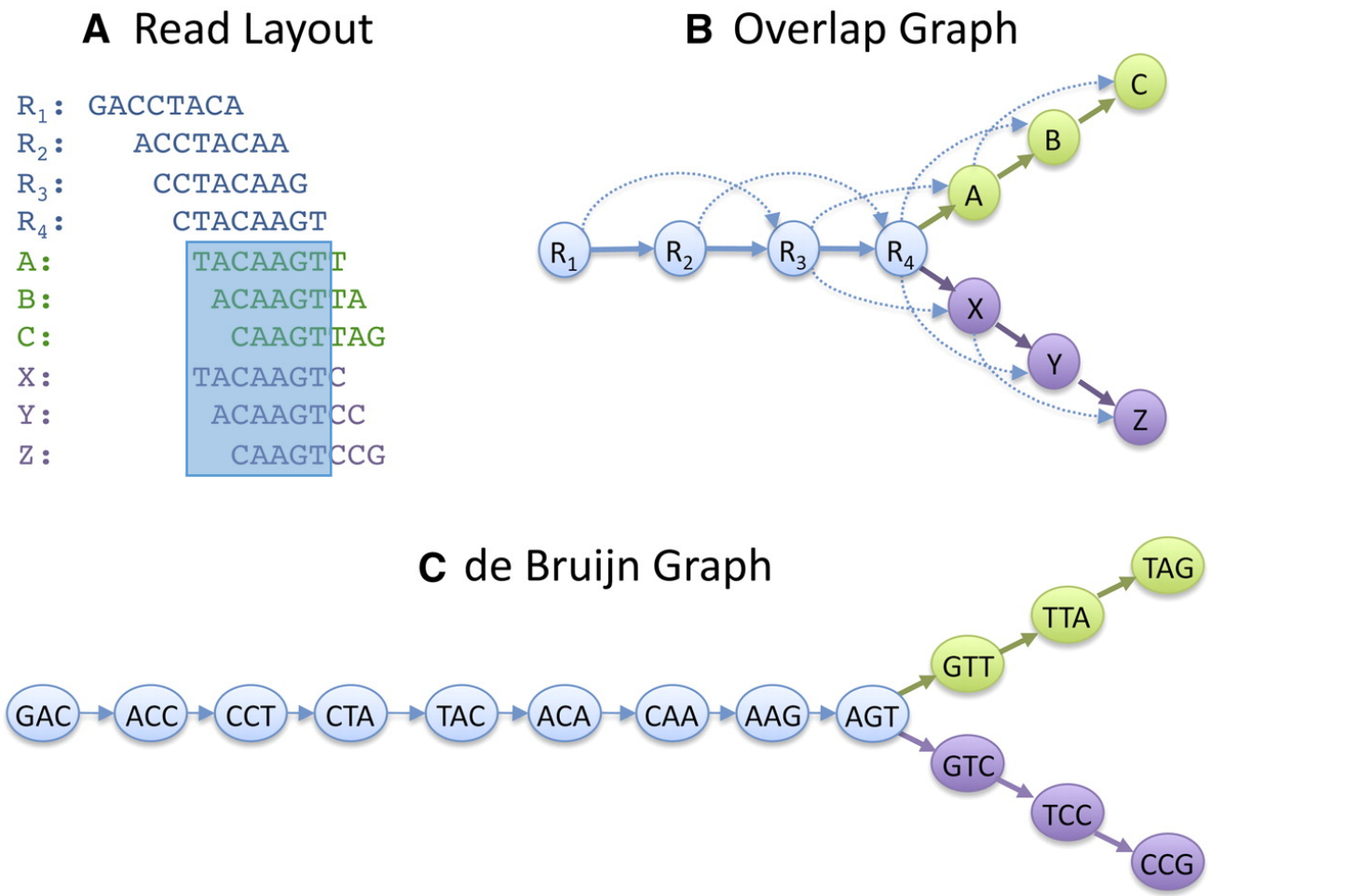


Before: one edge per *k*-mer

After: one *weighted* edge per *distinct k*-mer

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

Summary



Schatz M C et al. Genome Res. 2010;20:1165-1173

Summary

Advantage of DBG:

Time to build based on Genome size (G) or total length of reads (N)

For OLC: time to build overlap graph is based on number of reads

But:

DBG not flexible: only overlap of **fixed length** (=kmer)

can't solve repeat with repeat > kmer

Read information is lost: All reads are split into kmers.

(A lot of work on later DBG assemblers are put in this)

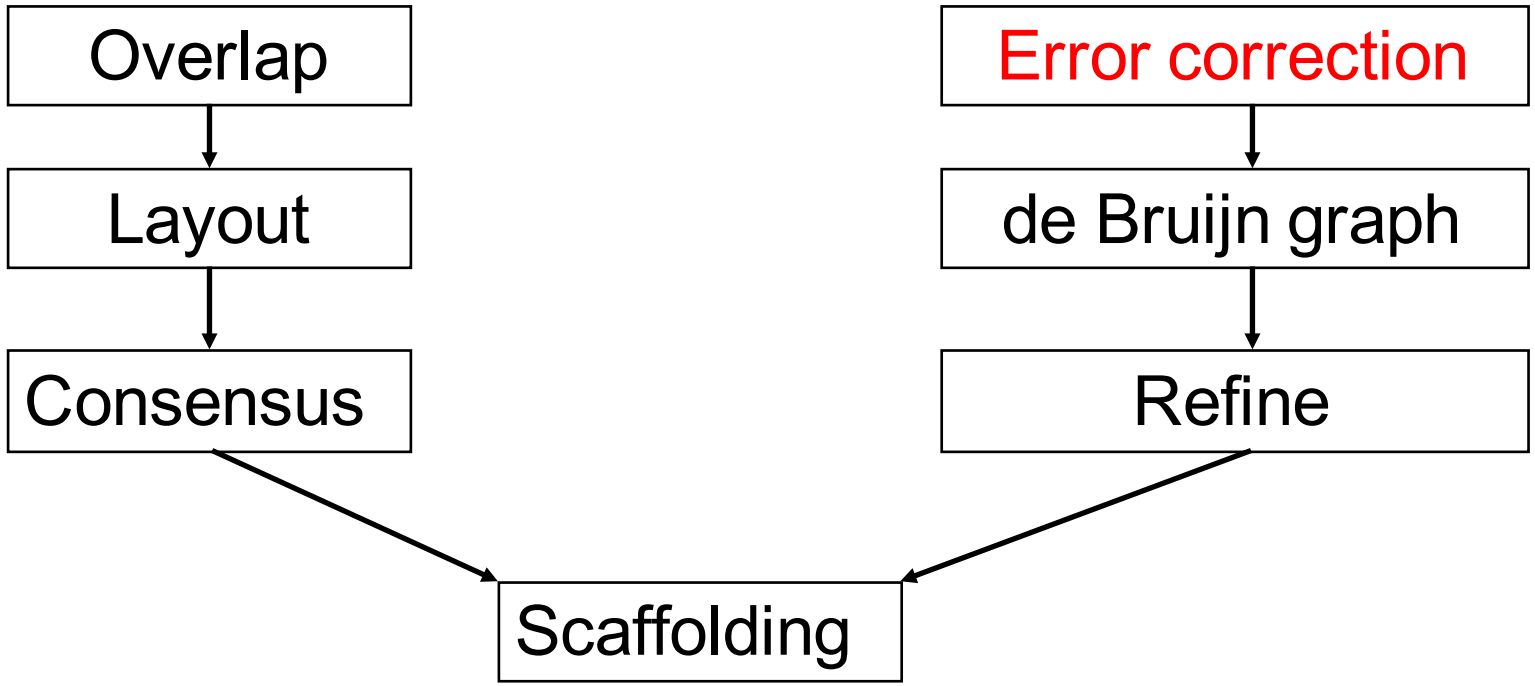
Tradeoff between DBG and OLC needed

Some improve over existing approach: Spades

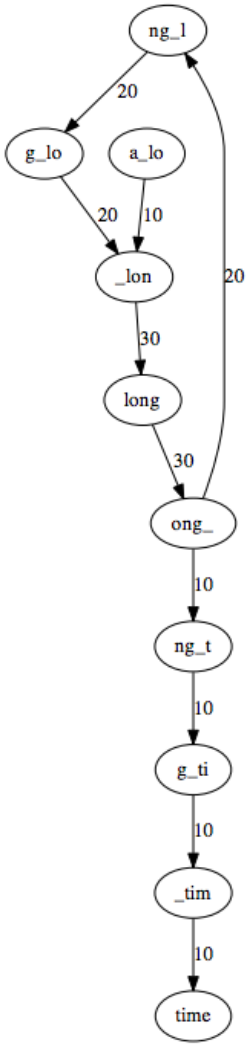
Some combine both: Masurca

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

OLC and DBG assemblers



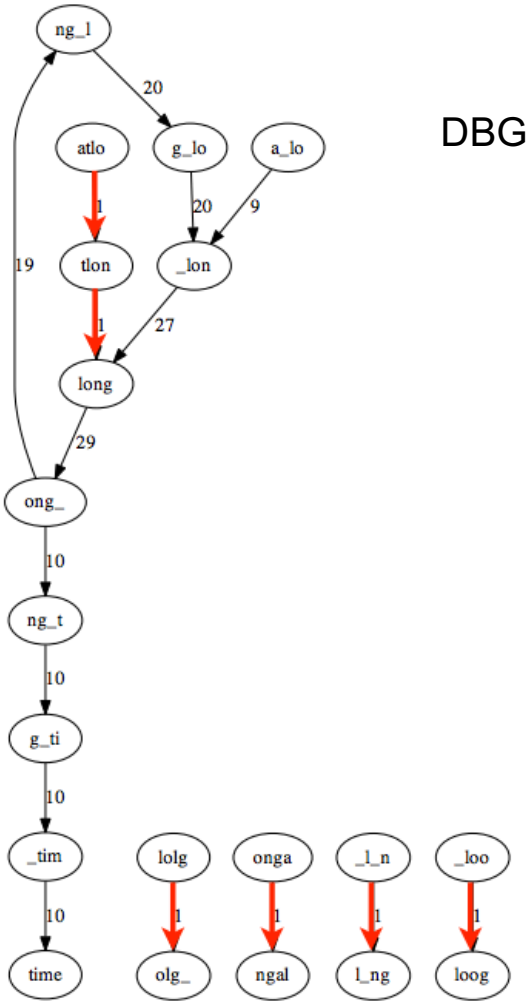
Error in graphs



DBG from perfect reads (10X)

As you can see all weighted **edges** have high coverage

Some higher than the other; this is obviously repeat



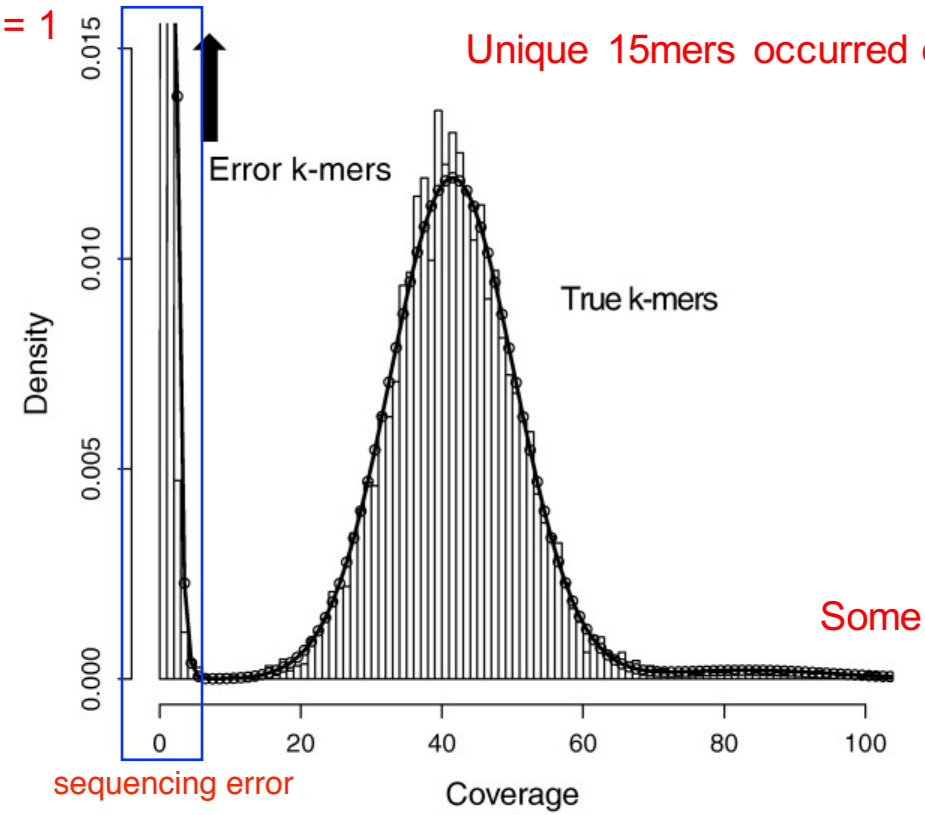
DBG from reads with some errors

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/error_correction.pdf

Kmer coverage

Indeed most erroneous kmer has kmer = 1

Unique 15mers occurred exactly ~41X coverage



Some 15mer occurred at >80X ; repeat

Variance = 77X

Almost twice of mean suggest it's Poisson distribution

Kelley *et al.,* (2010)

Choosing the right kmer

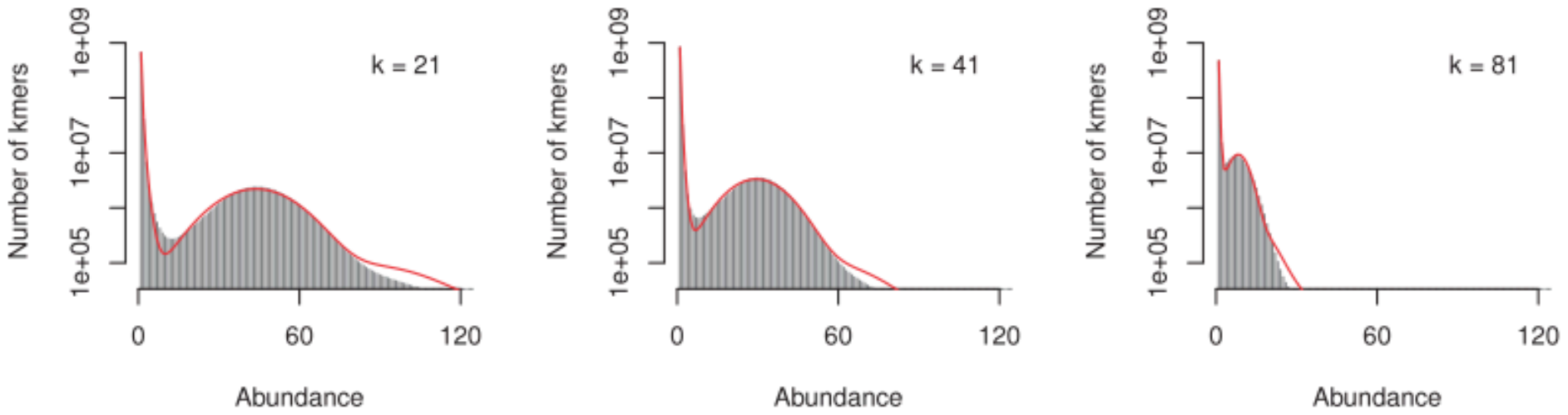


Fig. 2. The abundance histograms for *chr14* with *k* values of 21, 41 and 81 (on a y log scale). Each plot also shows a curve corresponding to the optimized statistical model (haploid)

Chikhi and Medvedev *et al.*, (2014)

Error correction: rationale

Idea: errors tend to turn frequent *k*-mers to infrequent *k*-mers, so corrections should do the reverse

Say we have a collection of reads where each distinct 8-mer occurs an average of ~10 times, and we have the following read:

Read: GCGTATTACGCGTCTGGCCT (20 nt)

8-mers:

```
GCGTATTA: 8
 CGTATTAC: 8
  GTATTACG: 9
   TATTACGC: 9
    ATTACGCG: 9
     TTACGCGT: 12
      TACGCGTC: 9
       ACGCGTCT: 8
        CGCGTCTG: 10
         GCGTCTGG: 10
          CGTCTGGC: 11
           GTCTGGCC: 9
            TCTGGCCT: 8
```

times each 8-mer occurs in the dataset. "*k*-mer count profile"

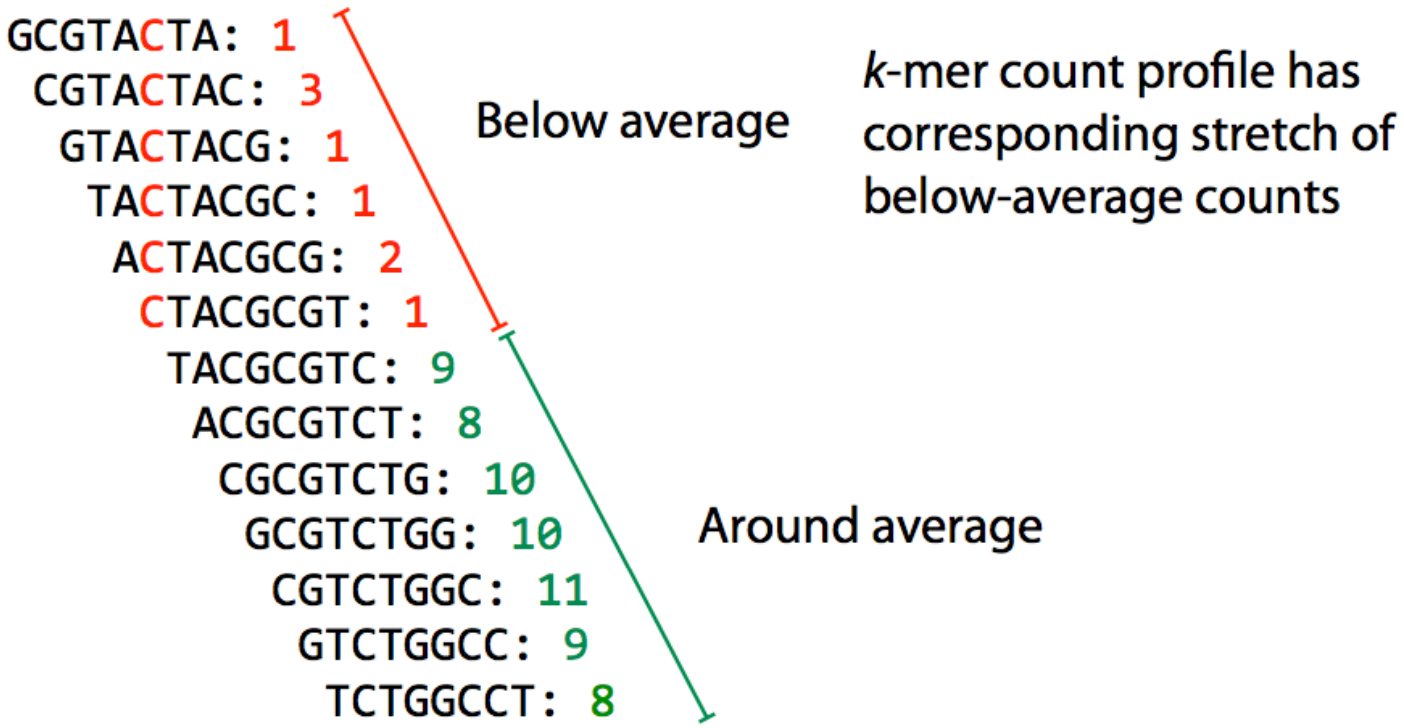
All 8-mer counts are around the average, suggesting read is error-free

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/error_correction.pdf

Error correction: rationale

Suppose there's an error

Read: GCGTACTACGCGTCTGGCCT



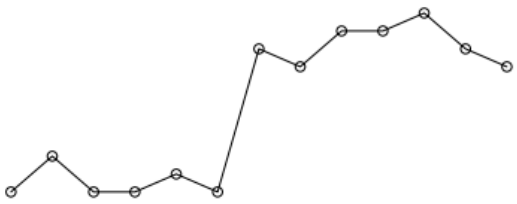
k-mer count profile has corresponding stretch of below-average counts

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/error_correction.pdf

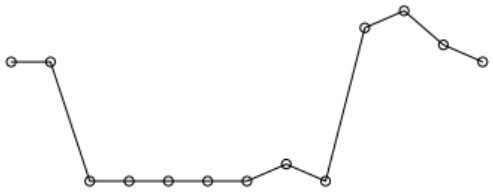
Error correction: rationale

k-mer count profiles when errors are in different parts of the read:

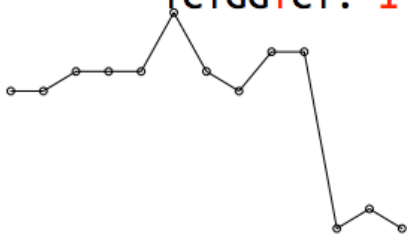
```
GCGTACTACGCGTCTGGCCT
GCGTACTA: 1
 CGTACTAC: 3
  GTACTACG: 1
   TACTACGC: 1
    ACTACGCG: 2
     CTACGCGT: 1
      TACGCGTC: 9
       ACGCGTCT: 8
        CGCGTCTG: 10
         GCGTCTGG: 10
          CGTCTGGC: 11
           GTCTGGCC: 9
            TCTGGCCT: 8
```



```
GCGTATTACACGTCTGGCCT
GCGTATTA: 8
 CGTATTAC: 8
  GTATTACA: 1
   TATTACAC: 1
    ATTACACG: 1
     TTACACGT: 1
      TACACGTC: 1
       ACACGTCT: 2
        CACGTCTG: 1
         GCGTCTGG: 10
          CGTCTGGC: 11
           GTCTGGCC: 9
            TCTGGCCT: 8
```



```
GCGTATTACGCGTCTGGTCT
GCGTATTA: 8
 CGTATTAC: 8
  GTATTACG: 9
   TATTACGC: 9
    ATTACGCG: 9
     TTACGCGT: 12
      TACGCGTC: 9
       ACGCGTCT: 8
        CGCGTCTG: 10
         GCGTCTGG: 10
          CGTCTGGT: 1
           GTCTGGTC: 2
            TCTGGTCT: 1
```



http://www.cs.jhu.edu/~langmea/resources/lecture_notes/error_correction.pdf

Localize error and correct

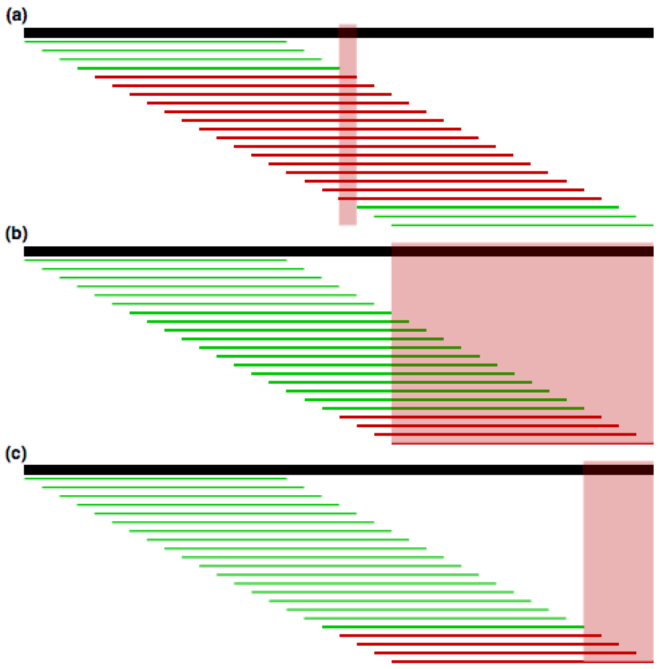
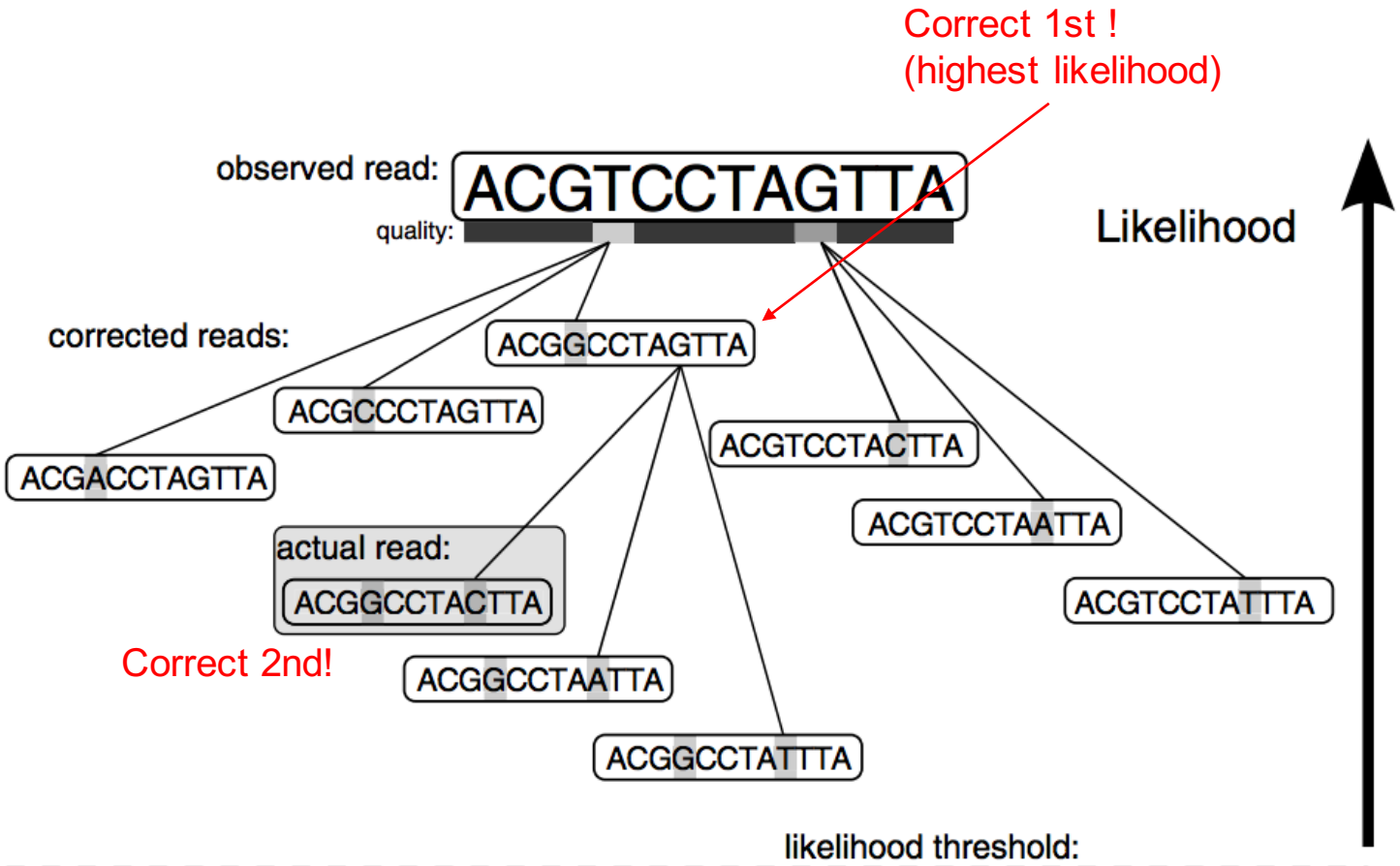


Figure 4 Localize errors. Trusted (green) and untrusted (red) 15-mers are drawn against a 36 bp read. In (a), the intersection of the untrusted *k*-mers localizes the sequencing error to the highlighted column. In (b), the untrusted *k*-mers reach the edge of the read, so we must consider the bases at the edge in addition to the intersection of the untrusted *k*-mers. However, in most cases, we can further localize the error by considering all bases covered by the right-most trusted *k*-mer to be correct and removing them from the error region as shown in (c).



Kelley *et al.*, (2010)

Summary

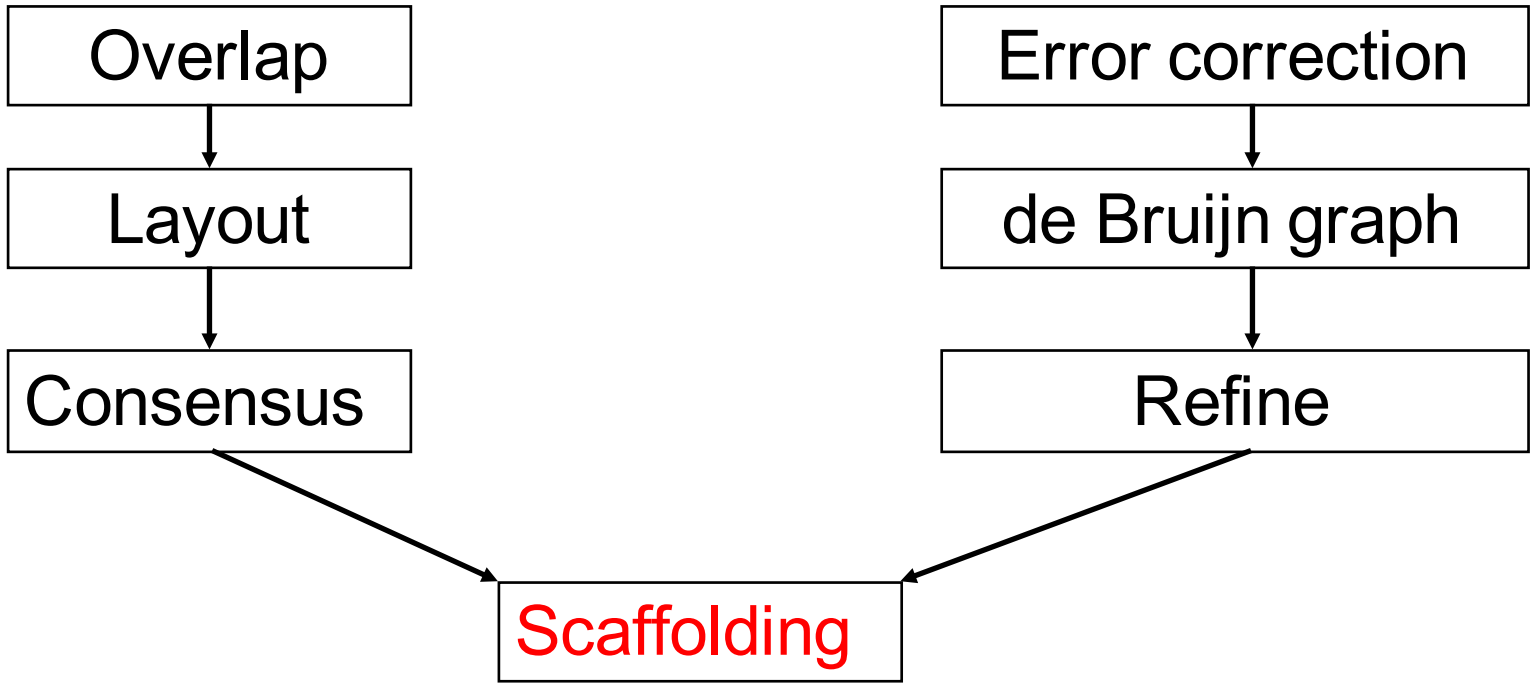
Error correction will **definitely** improve assembly

But, for it to work well:

- Sequenced coverage should be high enough
- Choose kmer wisely otherwise we can't distinguish erroneous kmer from frequent kmers

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

OLC and DBG assemblers



Scaffolding

OLC and DBG attempt to construct longest and most accurate **contigs** (**contig**uous stretch of assembled bases)

Scaffolding is to order and orient contigs with respect to each other

Various data types:

- Paired ends / Mate pairs
- Genetic map
- Additional long range information

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/scaffolding.pdf

Scaffolding: Paired end sequencing

Because of technology limitation (usually ~150bp at each end), whole fragment is not sequenced. But the distance between two mates equals to length of fragment **(insert size)**

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/scaffolding.pdf

Scaffolding: Paired end sequencing

- DNA fragment (200-800 bp)

  ==========

- Single end

  ---------->==========

- Paired end (up to 800 bp span)

  ----------==========<----------

- Mate pair (up to 40 kbp span)

  ---->========/+/=========<----

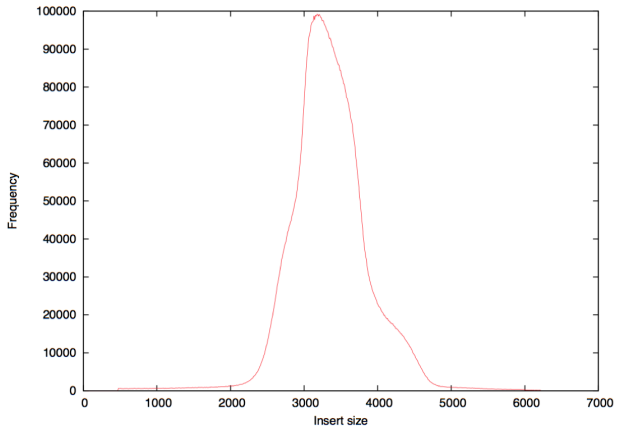
Examples

How to check insert size?

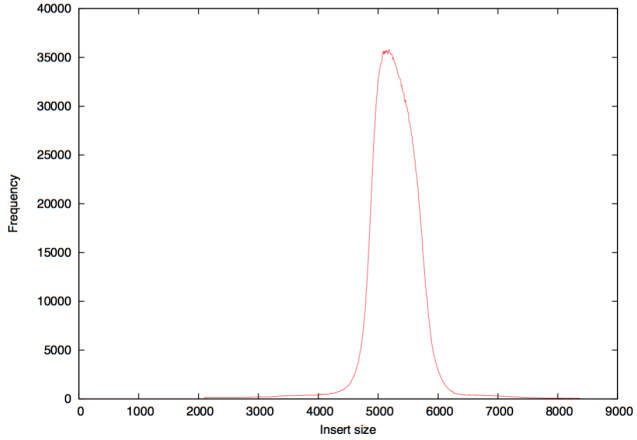
Remap the data back to the assembly

Problem can arise in larger insert sizes

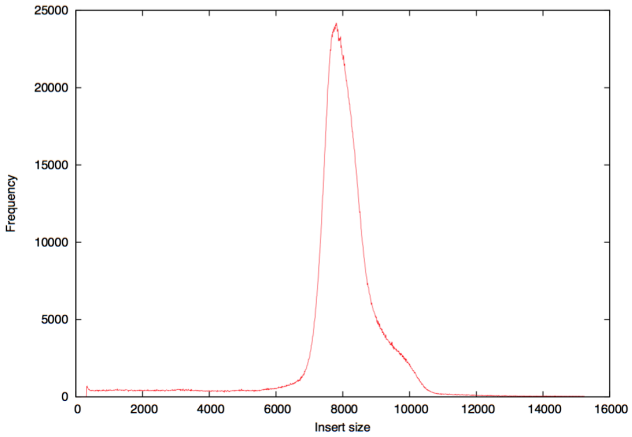
3kb



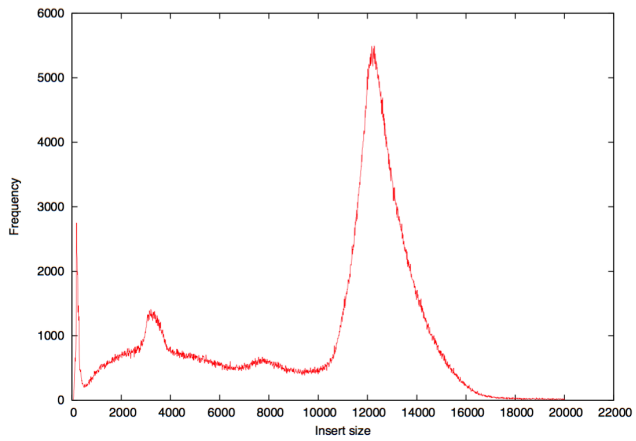
5kb



8kb



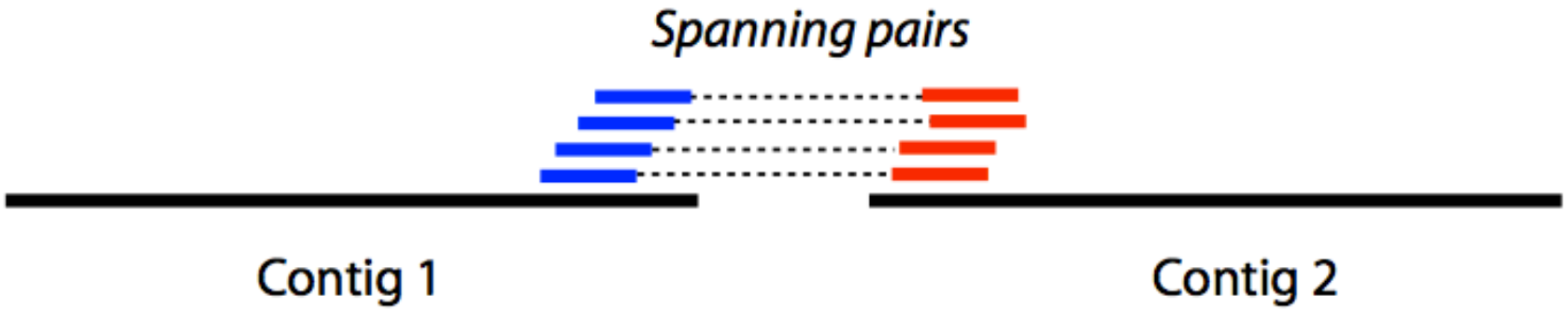
12kb



Scaffolding

Say we have a collection of pairs and we assemble them as usual

Assembly yields two contigs:

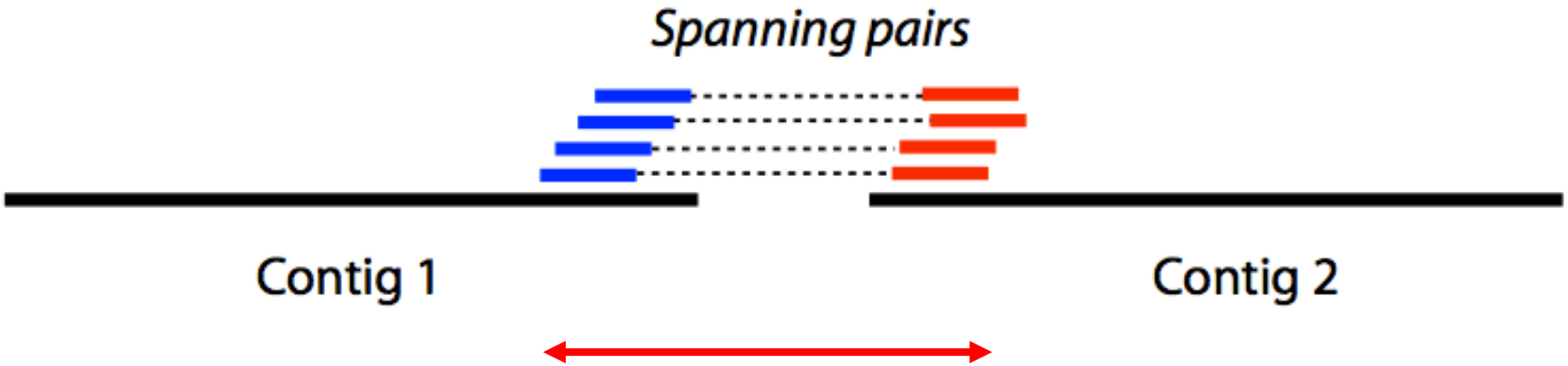


...and we discover that some of the mates at one edge of contig 1 are paired with mates in contig 2

Call these *spanning pairs*

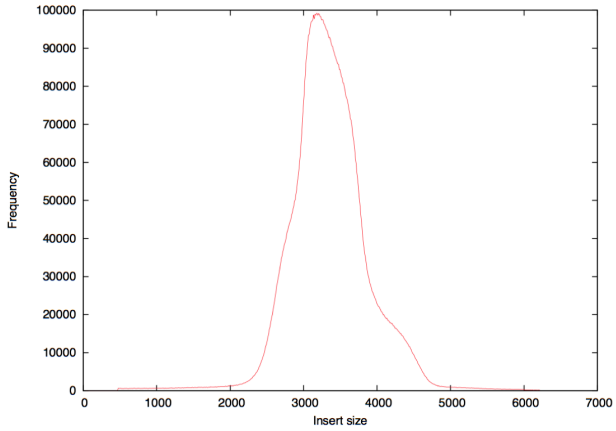
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/scaffolding.pdf

Scaffolding



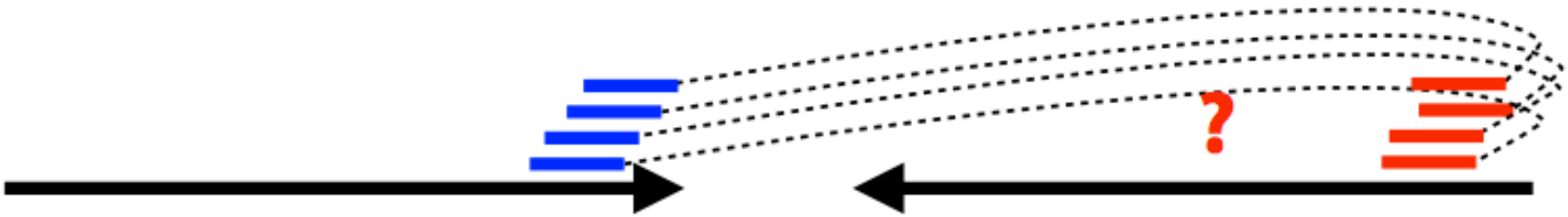
Does the distance equal to the insert size distribution mapped to the rest the genome?

Yes = Accept and Contig1NNNNContig2
No = Reject

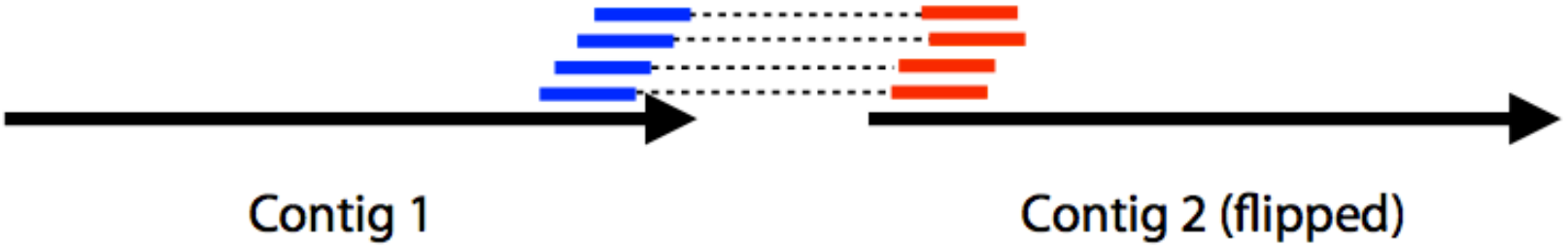


http://www.cs.jhu.edu/~langmea/resources/lecture_notes/scaffolding.pdf

Scaffolding



What does the picture look like if contigs 1 and 2 are close, but we assembled contig 2 "backwards" (i.e. reverse complemented)

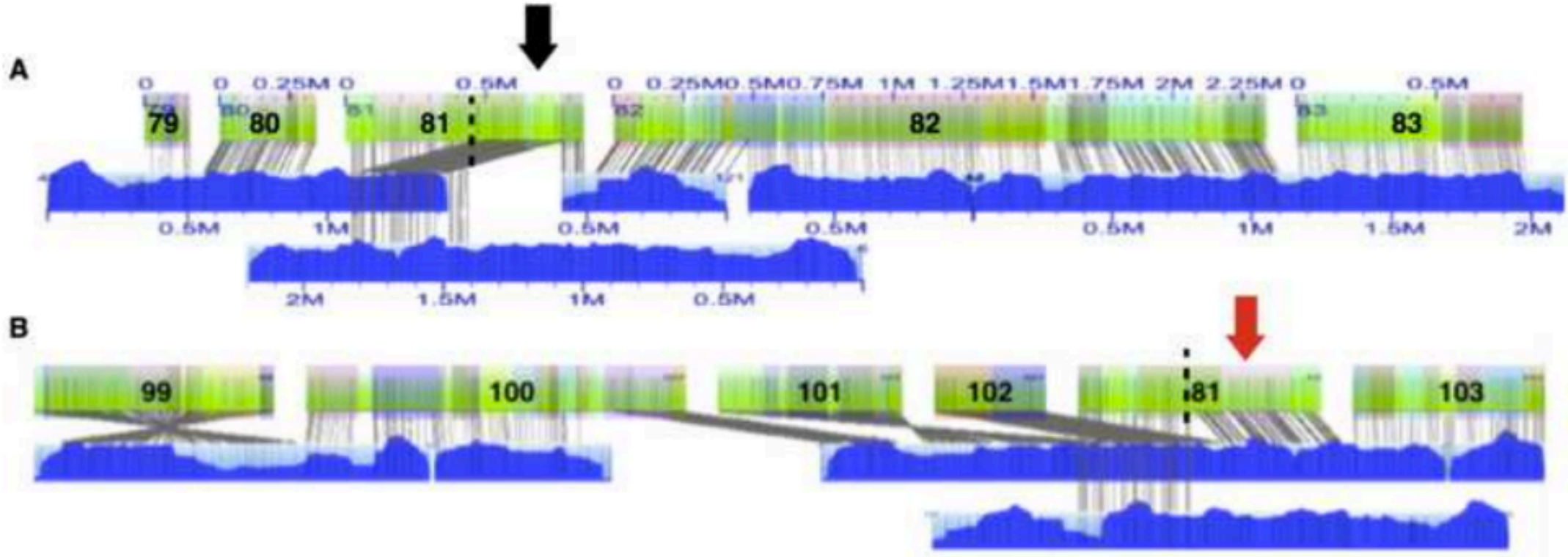


Pairs also tell us about contigs' relative *orientation*

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/scaffolding.pdf

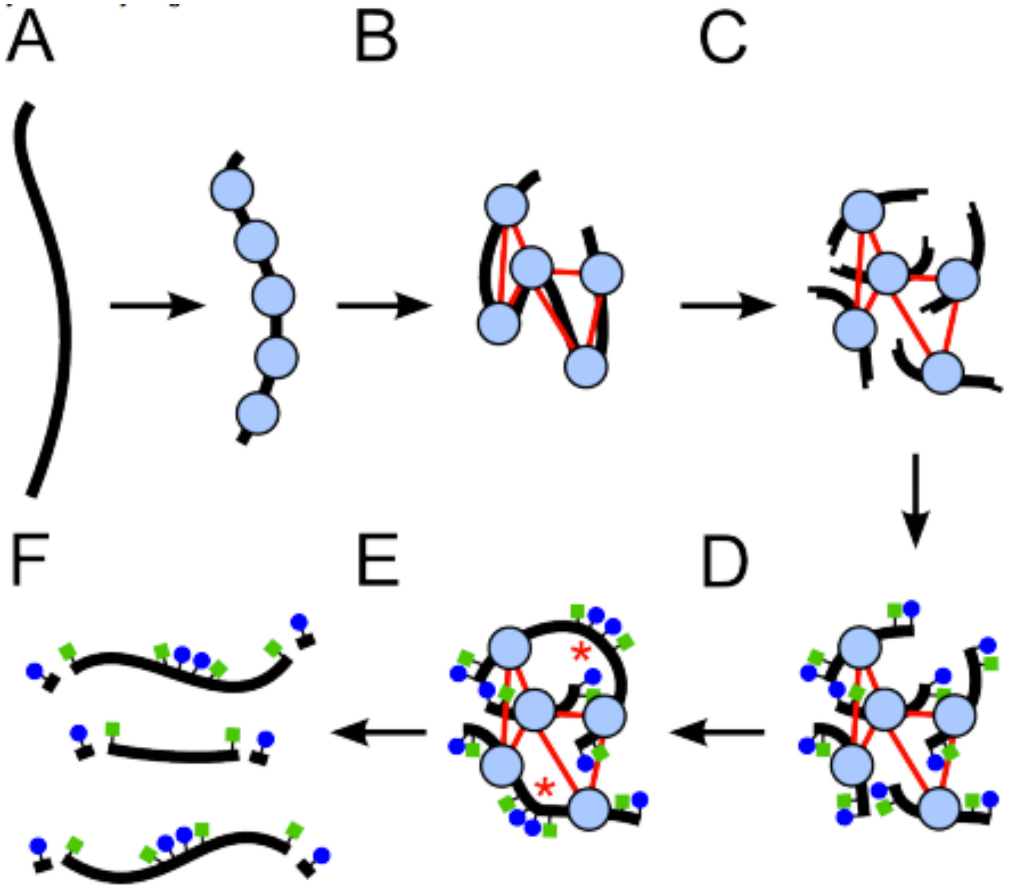
Extreme long range information I

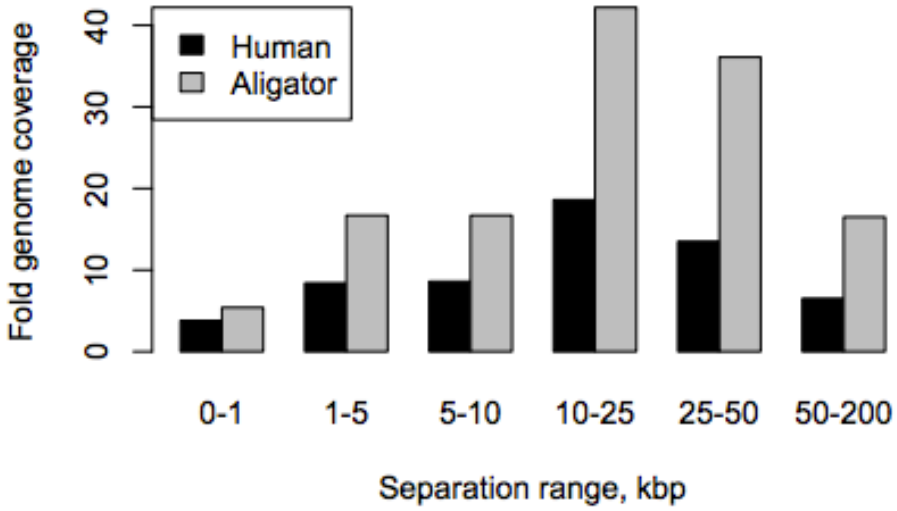
BIONANO GENOMICS



http://biorxiv.org/content/biorxiv/early/2015/06/15/020966.full.pdf

Extreme long range information II





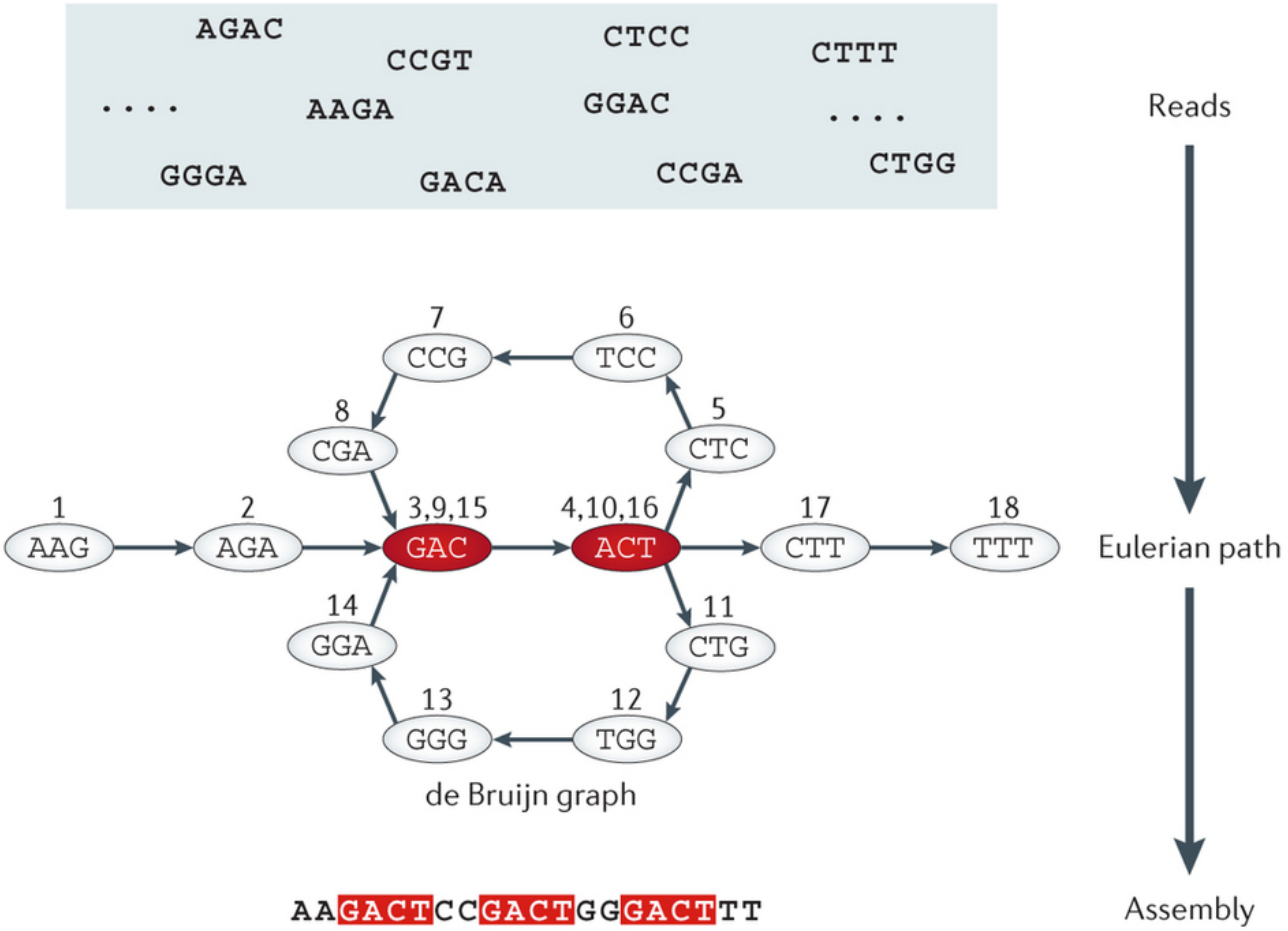
http://arxiv.org/pdf/1502.05331.pdf

read pairs generated by proximity ligation of DNA in chromatin of living tissue

Summary

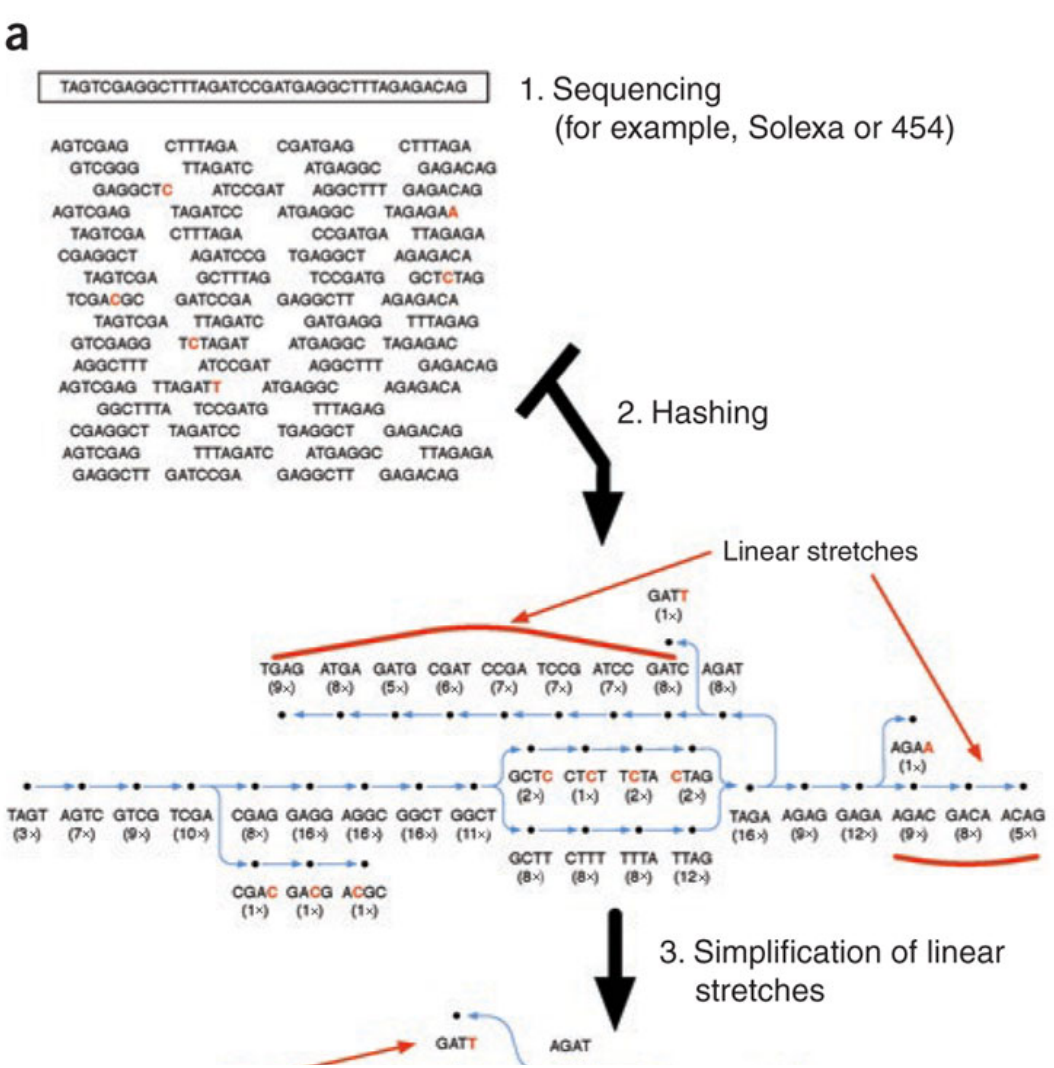
Break

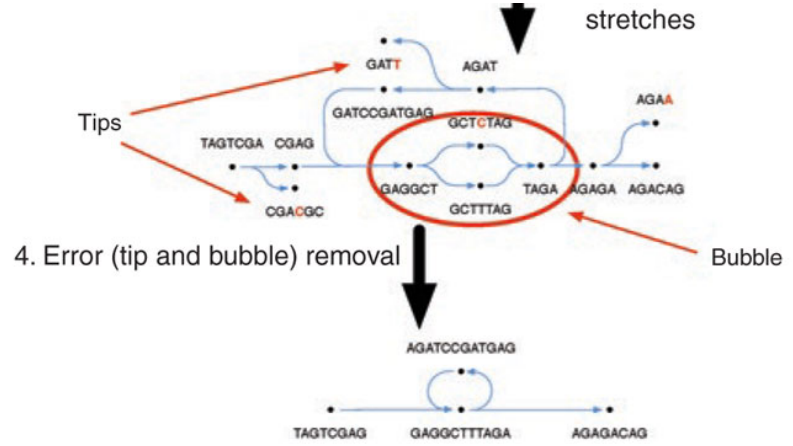
Review I



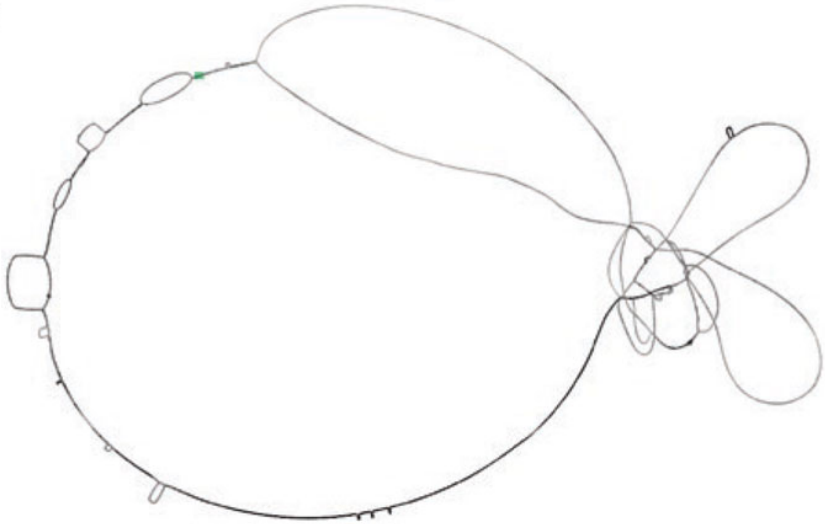
http://www.nature.com/nrg/journal/v14/n5/full/nrg3433.html

Review II

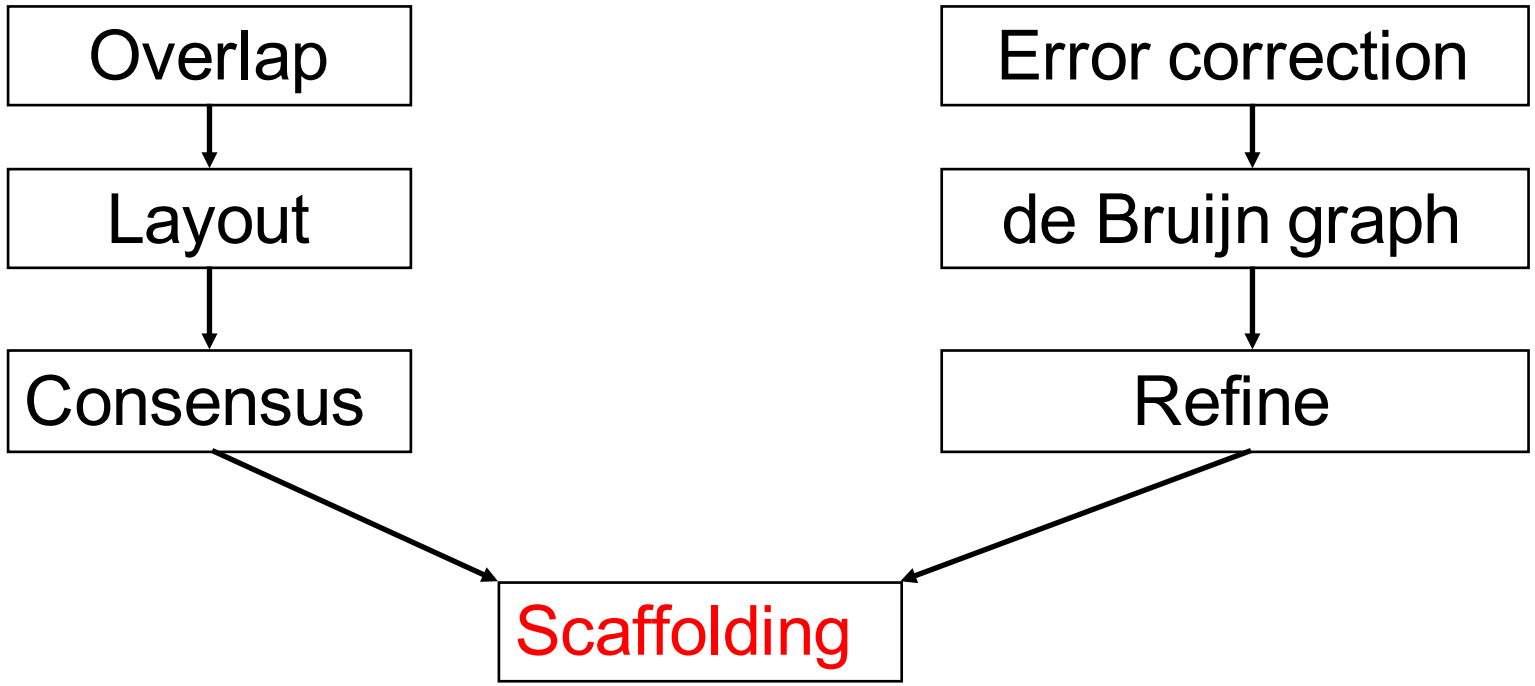

a
TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG
1. Sequencing
(for example, Solexa or 454)
2. Hashing
Linear stretches
3. Simplification of linear
stretches


stretches
Tips
Bubble
4. Error (tip and bubble) removal

b



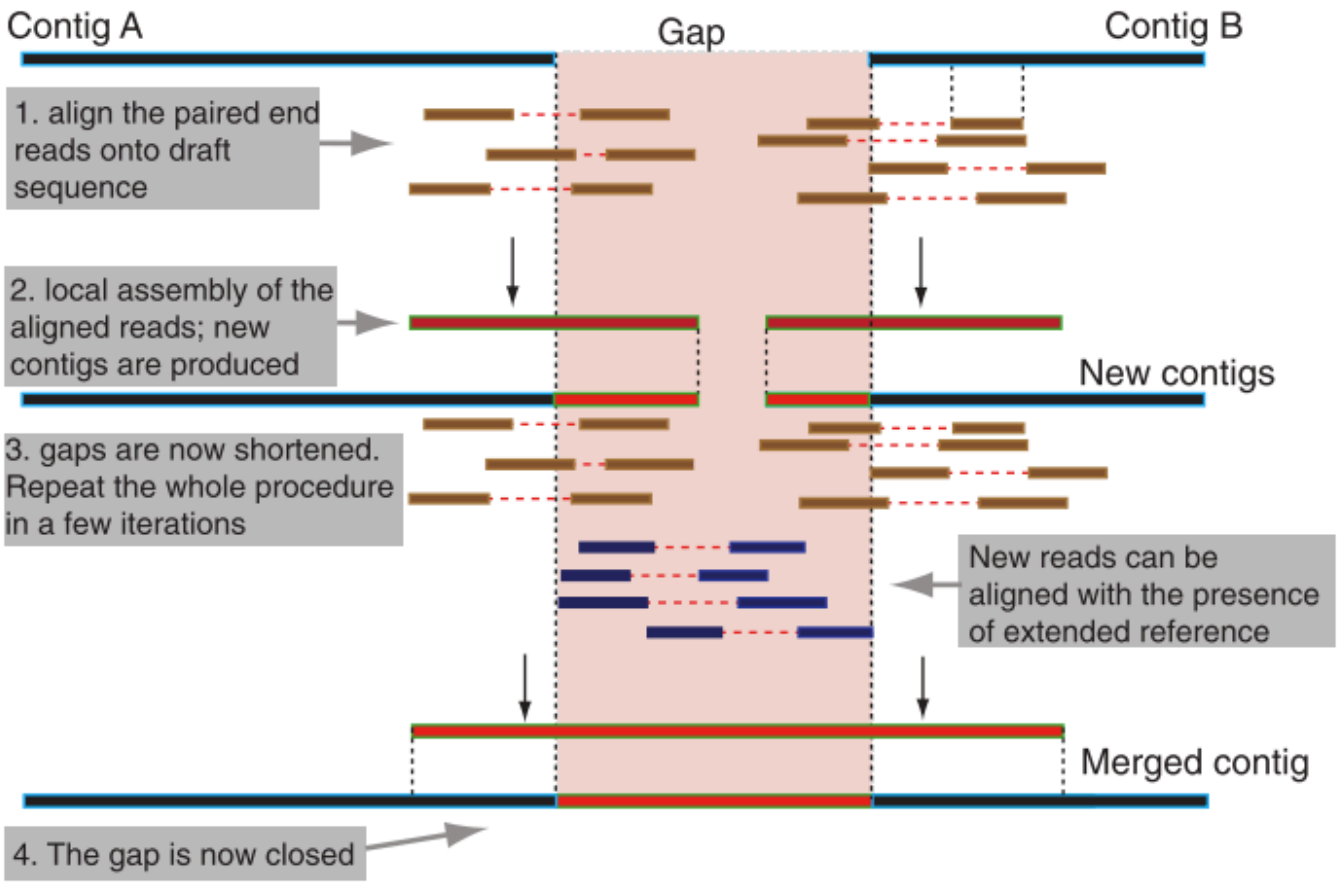
Anything else?



Gap closure

Logic:

1. Local assembly is less complicated than whole assembly
2. Treat every gap as a possible case of local assembly



Method improving draft assemblies by iterative mapping and assembly of short reads to eliminate gaps
IJ Tsai, TD Otto, M Berriman
Genome Biol 11, R41

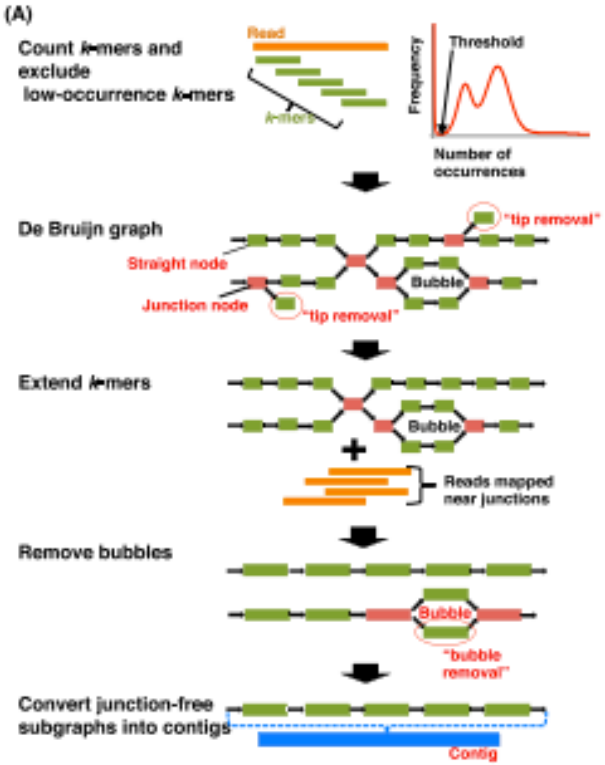
174 2010

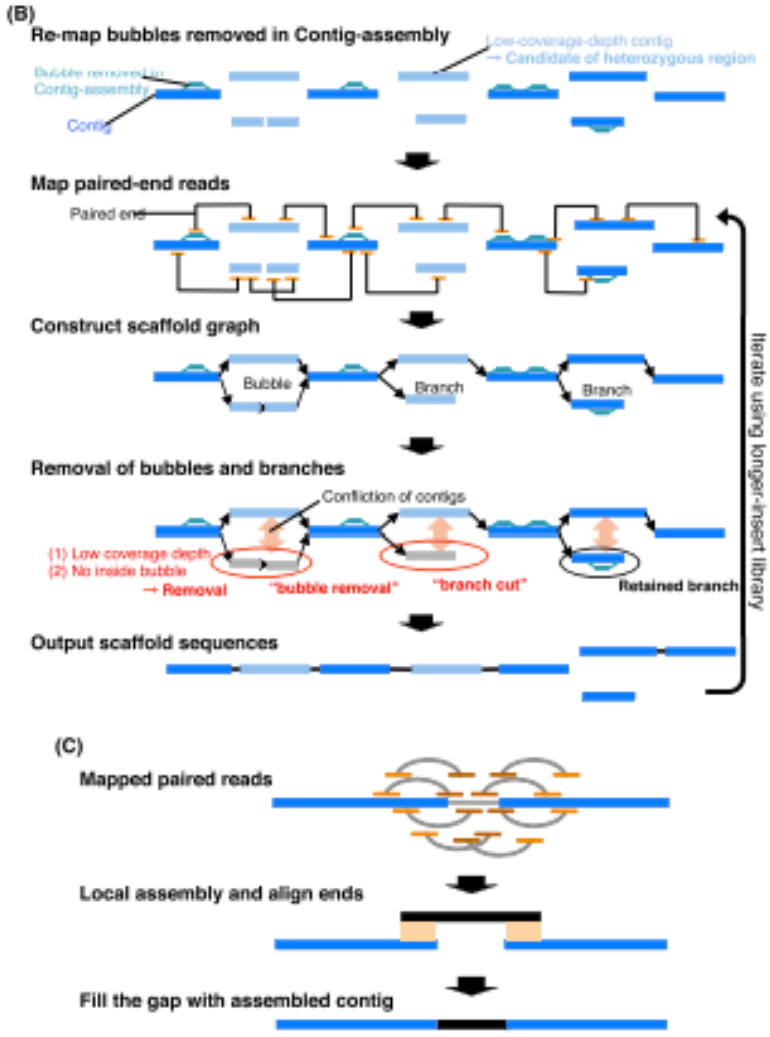
By now, you should have an idea of what assembly is

Would you understand everything in this paper?

Resource

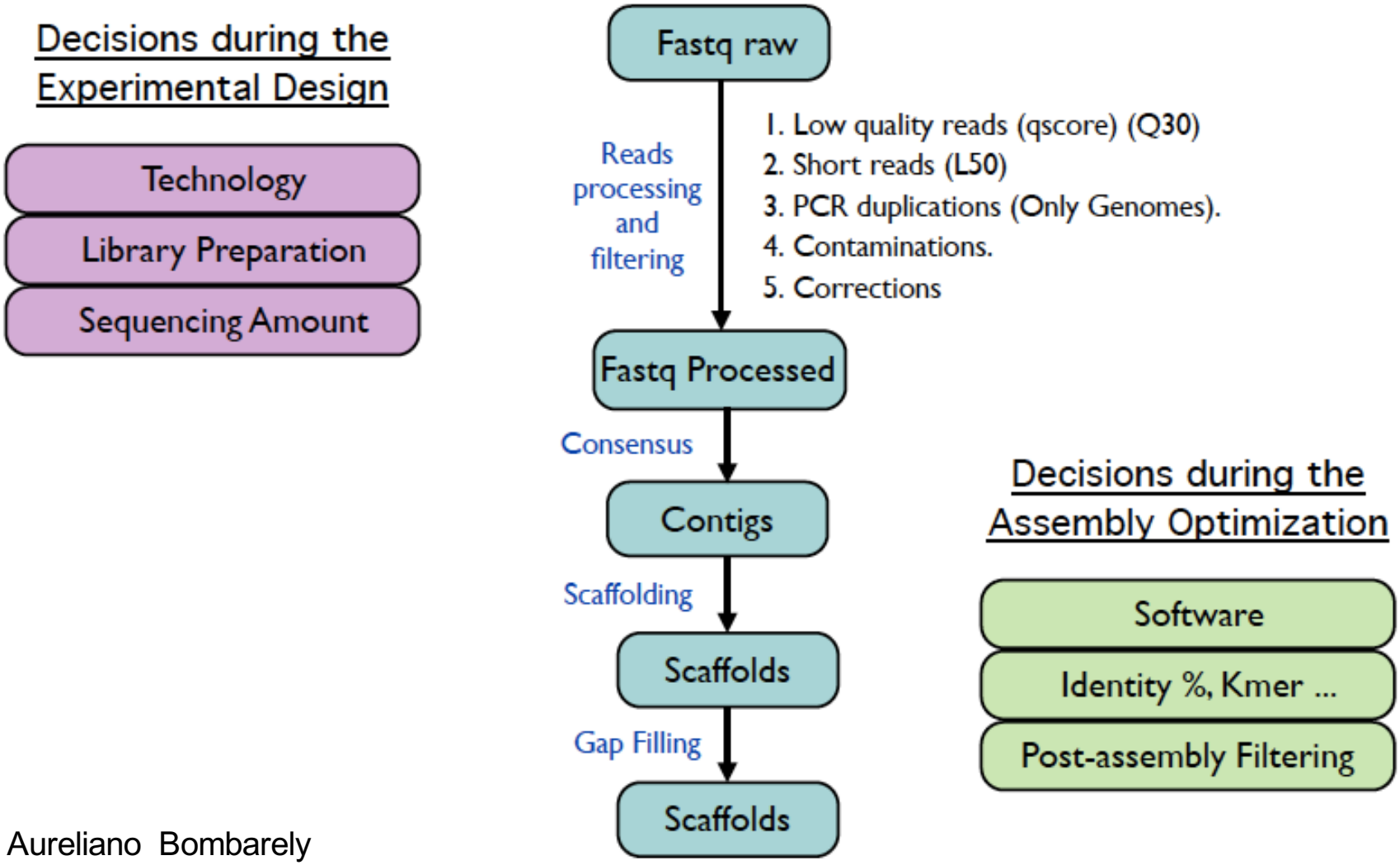
Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads





Some practical point

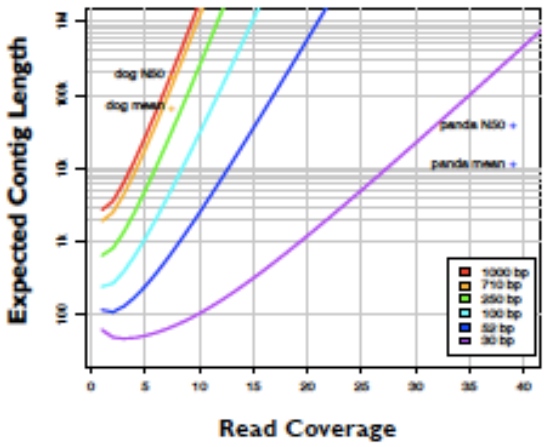
Overview of a sequencing project



Credit: Aureliano Bombarely

Ingredients for a good assembly

Coverage



High coverage is required

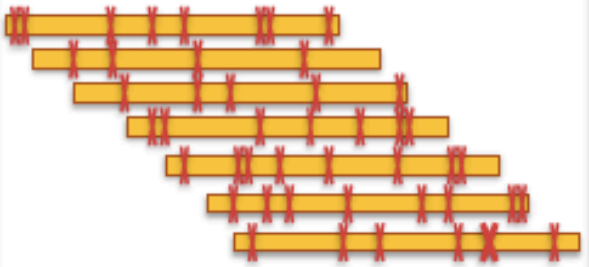
- Oversample the genome to ensure every base is sequenced with long overlaps between reads
- Biased coverage will also fragment assembly

Read Length

Reads & mates must be longer than the repeats

- Short reads will have ***false overlaps*** forming hairball assembly graphs
- With long enough reads, assemble entire chromosomes into contigs

Quality



Errors obscure overlaps

- Reads are assembled by finding kmers shared in pair of reads
- High error rate requires very short seeds, increasing complexity and forming assembly hairballs

Credit: Michael Schatz

Okay, once you have an assembly

Know: Every assembler produce different slightly output

Q:
How to compare different assemblies?
How to check for errors?
If error is found, how to best improve it?

Not a easy answer

Assembly competitions

Resource

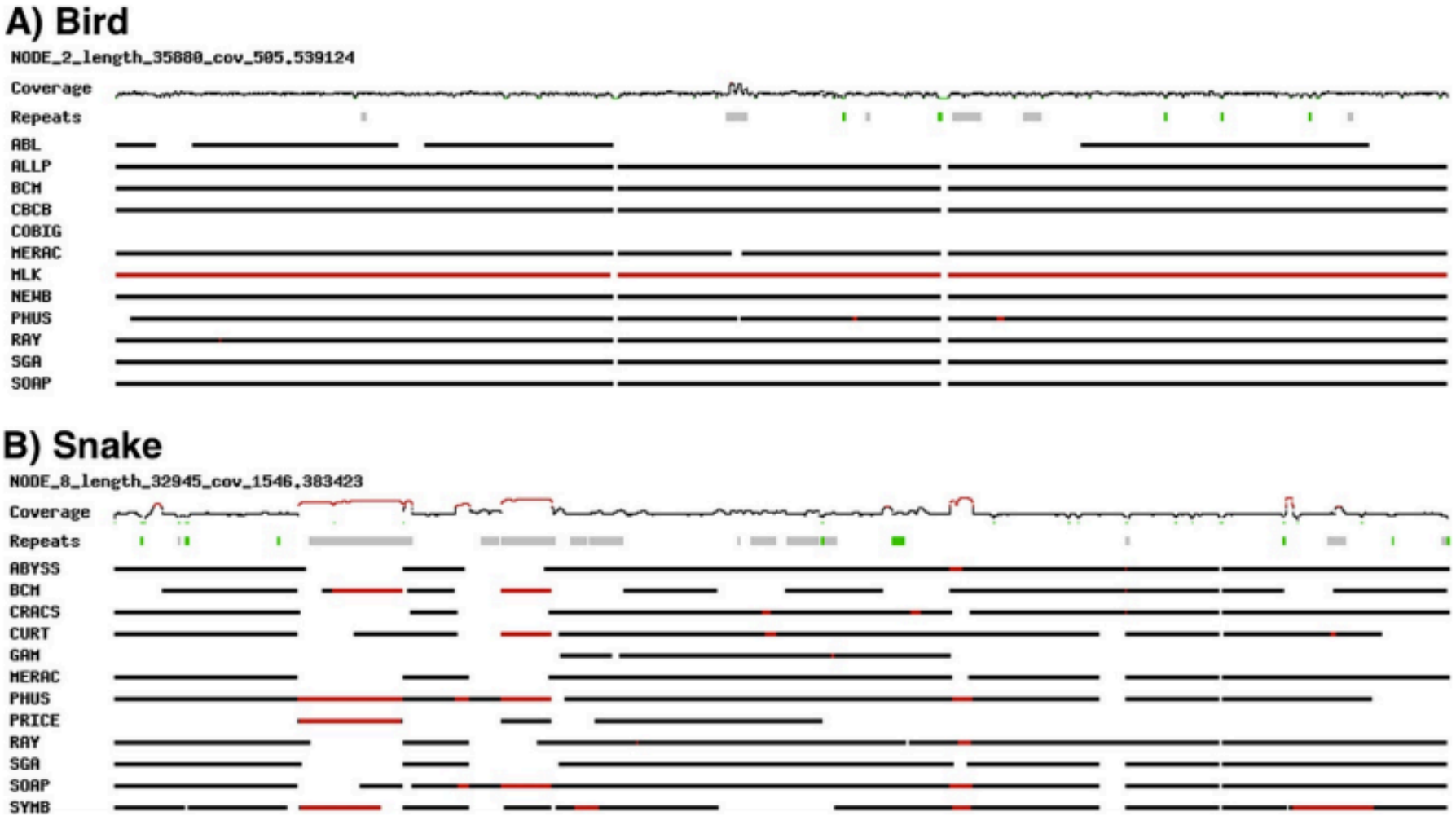
Assemblathon 1: A competitive assessment of de novo short read assembly methods

Bradnam *et al. GigaScience* 2013, **2**:10
http://www.gigasciencejournal.com/content/2/1/10

RESEARCH Open Access

Assemblathon 2: evaluating *de novo* methods of genome assembly in three vertebrate species

No assembly is best..



Bradnam et al., (2013)

Find ways to validate your assembly

- Use the same dataset that made the assembly to self check
  - Map read back to contigs
  - Check for errors or discordant pairs
  - REAPR
- Alternative data
  - Genetic map?
  - Long range?
  - Additional data?

Hunt *et al. Genome Biology* 2013, **14**:R47
http://genomebiology.com/2013/14/5/R47

SOFTWARE Open Access

REAPR: a universal tool for genome assembly evaluation

Martin Hunt[1], Taisei Kikuchi[1,2], Mandy Sanders[1], Chris Newbold[1,3], Matthew Berriman[1] and Thomas D Otto[1*]

Additional (biological) challenges

Always be careful of contamination

PNAS

Evidence for extensive horizontal gene transfer from the draft genome of a tardigrade

Thomas C. Boothby[a,1], Jennifer R. Tenlen[a,2], Frank W. Smith[a], Jeremy R. Wang[a,b], Kiera A. Patanella[a], Erin Osborne Nishimura[a], Sophia C. Tintori[a], Qing Li[c], Corbin D. Jones[a], Mark Yandell[c], David N. Messina[d], Jarret Glasscock[d], and Bob Goldstein[a]

[a]Department of Biology, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599; [b]Department of Genetics, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599; [c]Eccles Institute of Human Genetics, University of Utah, Salt Lake City, UT 84112; and [d]Cofactor Genomics, St. Louis, MO 63110

Edited by W. Ford Doolittle, Dalhousie University, Halifax, Canada, and approved September 28, 2015 (received for review May 28, 2015)

17.5 % HGT

The genome of the tardigrade *Hypsibius dujardini*

We compare our assembly to a recently published one for the same species and do not find support for massive horizontal gene transfer

Georgios Koutsovoulos[1], Sujai Kumar[1], Dominik R. Laetsch[1,2], Lewis Stevens[1], Jennifer Daub[1], Claire Conlon[1], Habib Maroon[1], Fran Thomas[1], Aziz Aboobaker[3] and Mark Blaxter[1*]

QC and understand your data before assembly

Conversation:

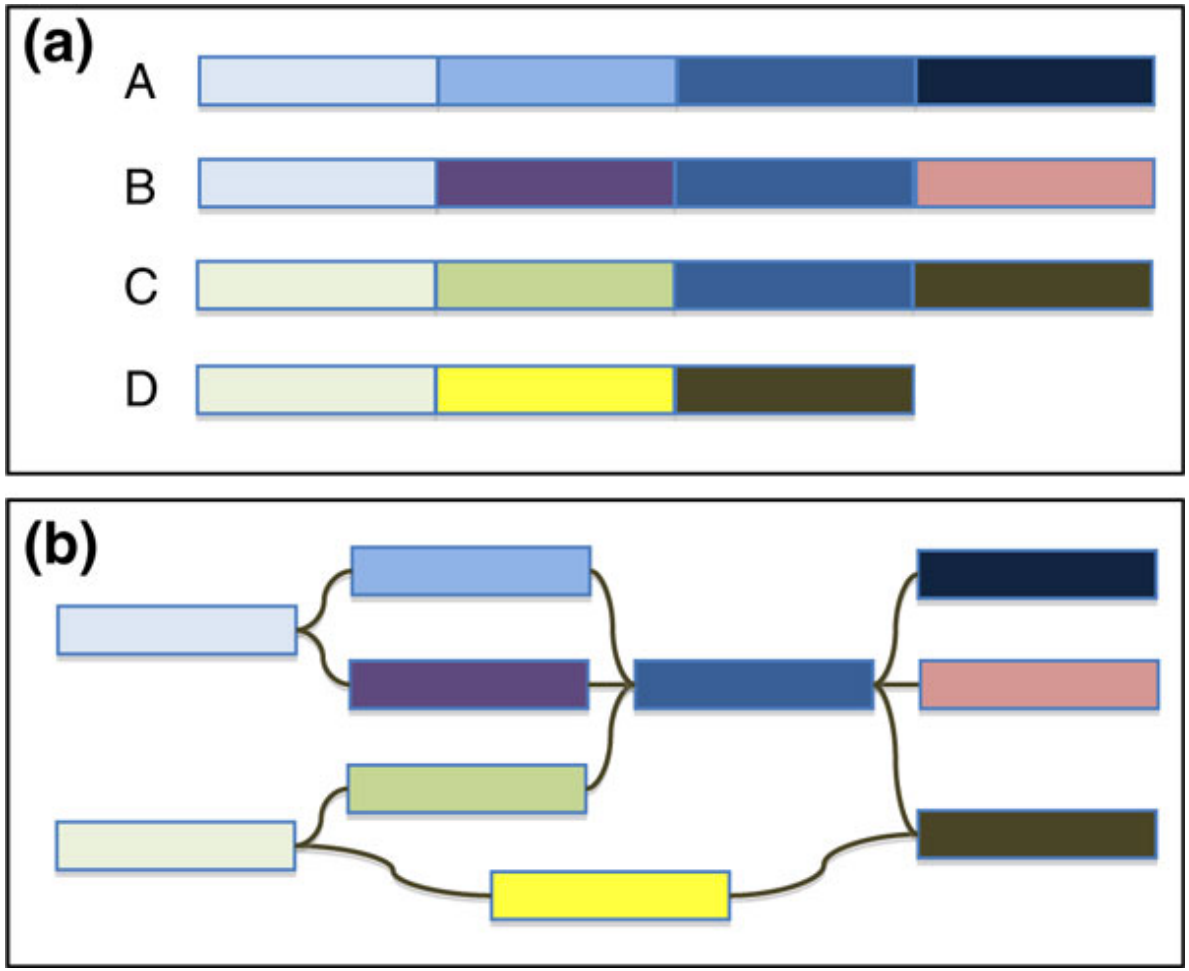
Jason, we don't know why our assembly is not as good as yours. We have about 90X of Mate pair data..

Email 2 weeks later:

Dear Jason,

I run the Trimmomatic analysis for the raw data of mate-pair libraries (10kL2_1, 10kL7_1, 15kL3_1 and 15kL8_4 as examples) with a custom adapter file containing mate-pair adapter sequences (junction and external adaptors, you may find them in the attached technote pdf file) and found **that over 80% reads of the library were dropped out.**

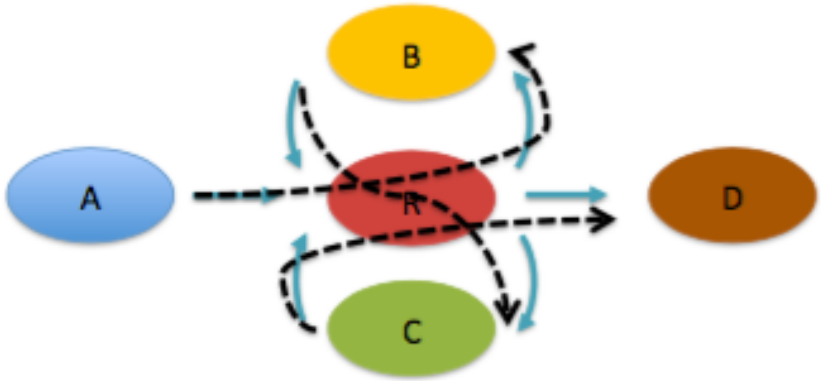
Ploidy, heterozygosity and the assembly graph



http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3446297/

Repeats

You can’t 100% resolve repeats repeats unless you have your sequence > repeat



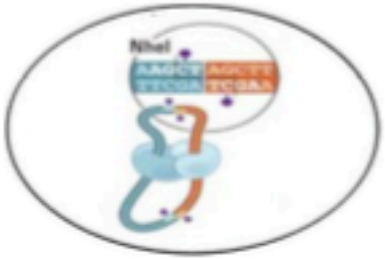
Credit: Michael Schatz

New technology to the rescue!

Long Read Sequencing: De novo assembly, SV analysis, phasing

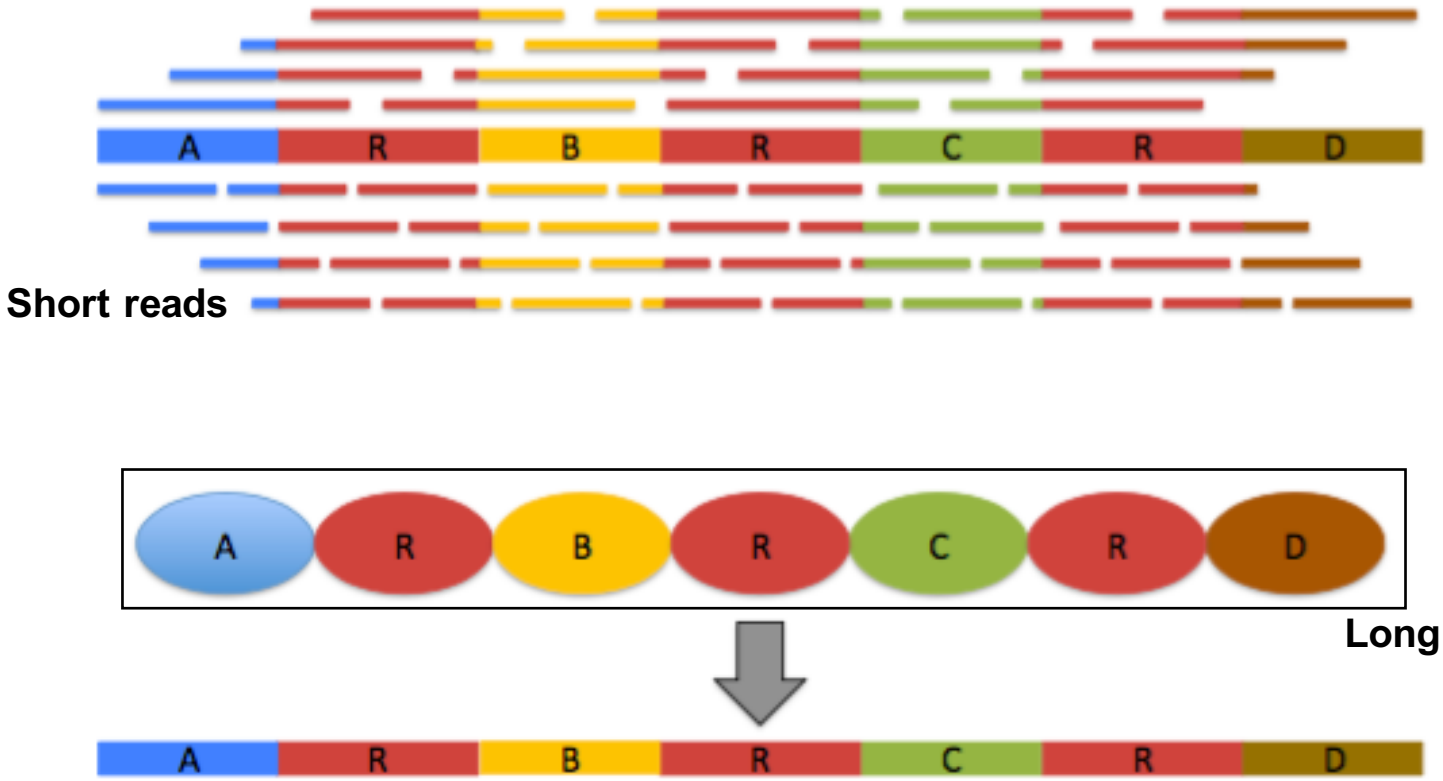
Illumina/Moleculo	*Pacific Biosciences*	*Oxford Nanopore*
		
(Kuleshov et al. 2014)	(Berlin et al, 2014)	(Quick et al, 2014)

Long Span Sequencing: Chromosome Scaffolding, SV analysis, phasing

Molecular Barcoding	*Optical Mapping*	*Chromatin Assays*
		
(10Xgenomics.com)	(Cao et al, 2014)	(Putnam et al, 2015)

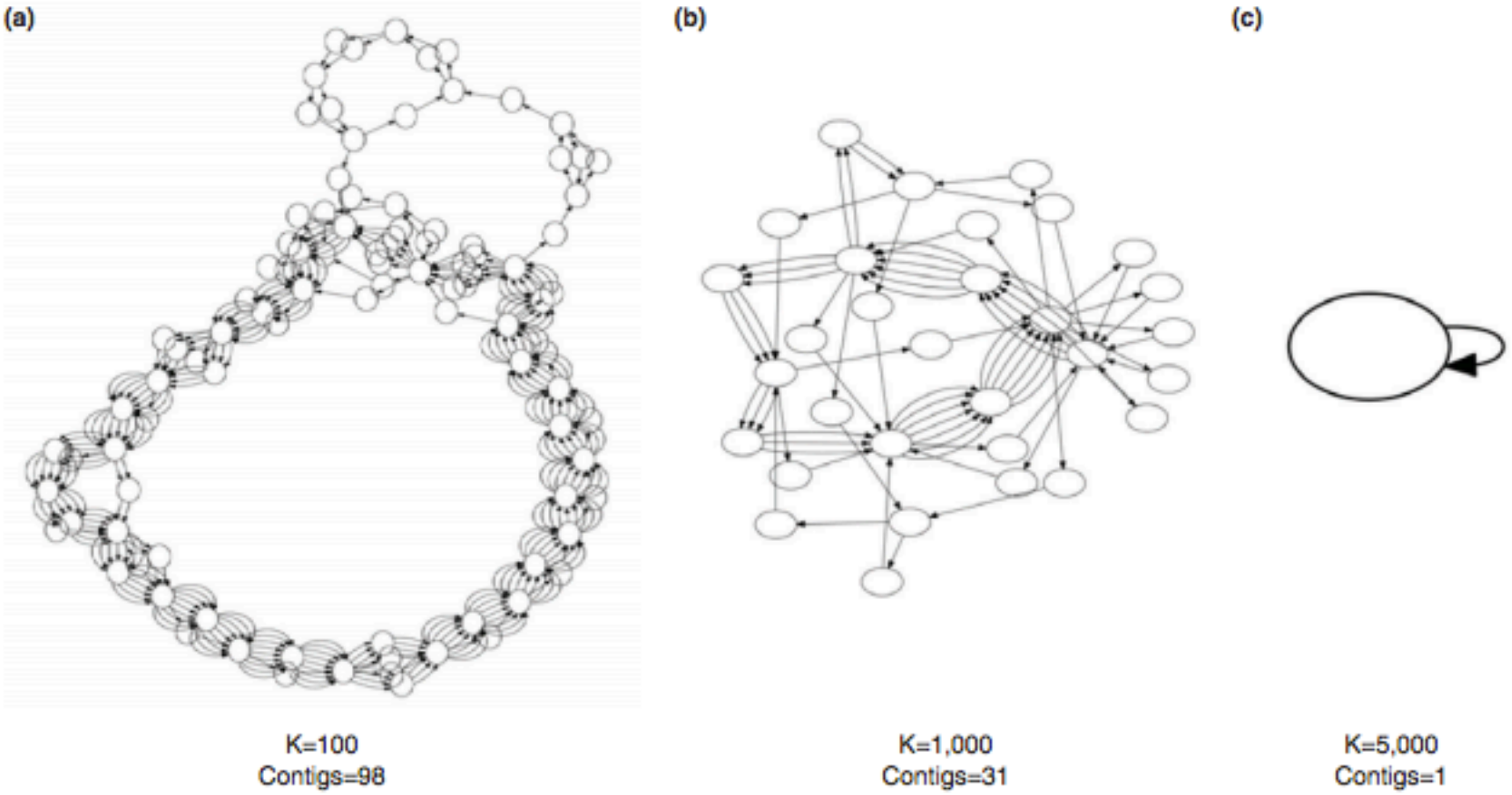
Credit: Michael Schatz

Now you can with long reads



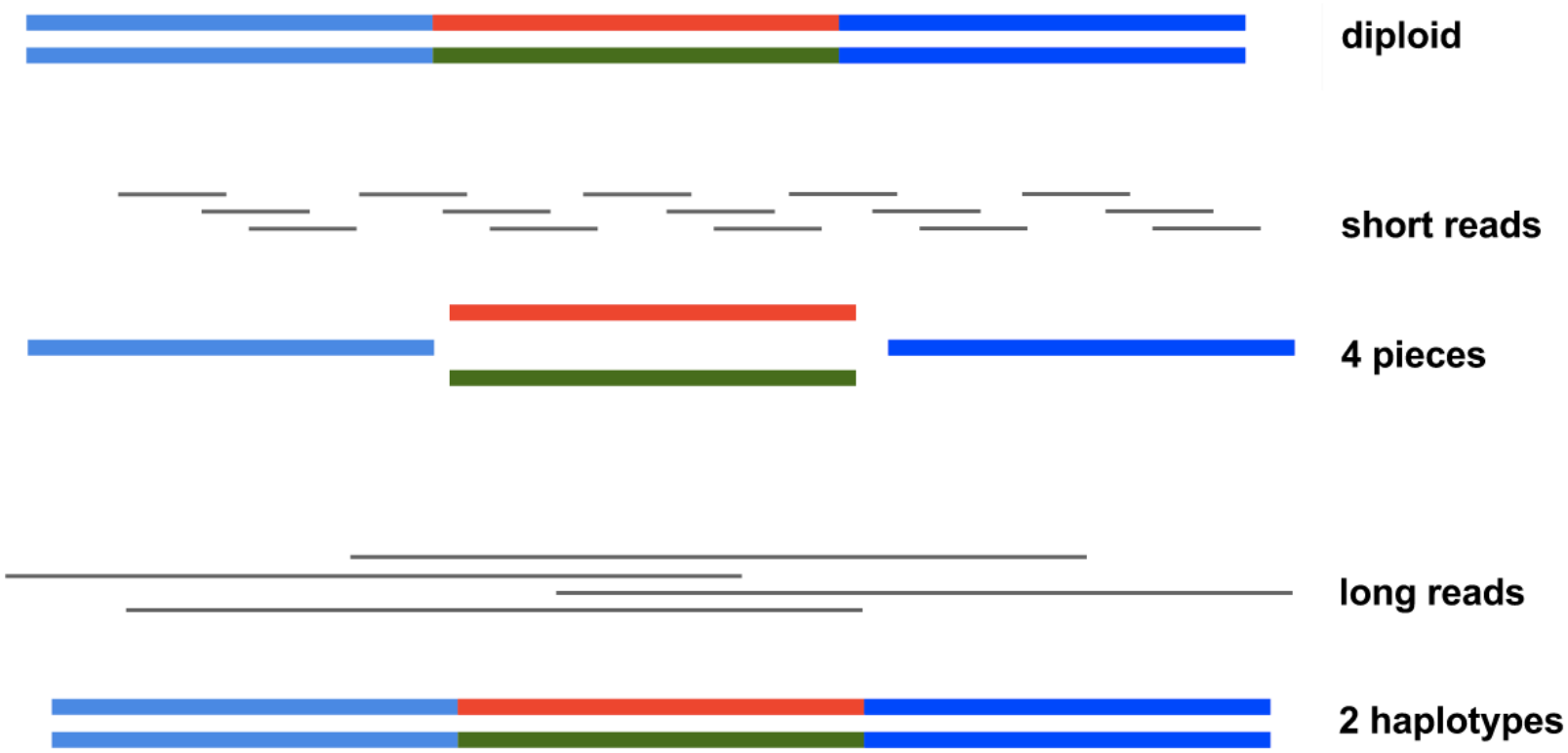
Credit: Michael Schatz

Simpler graphs



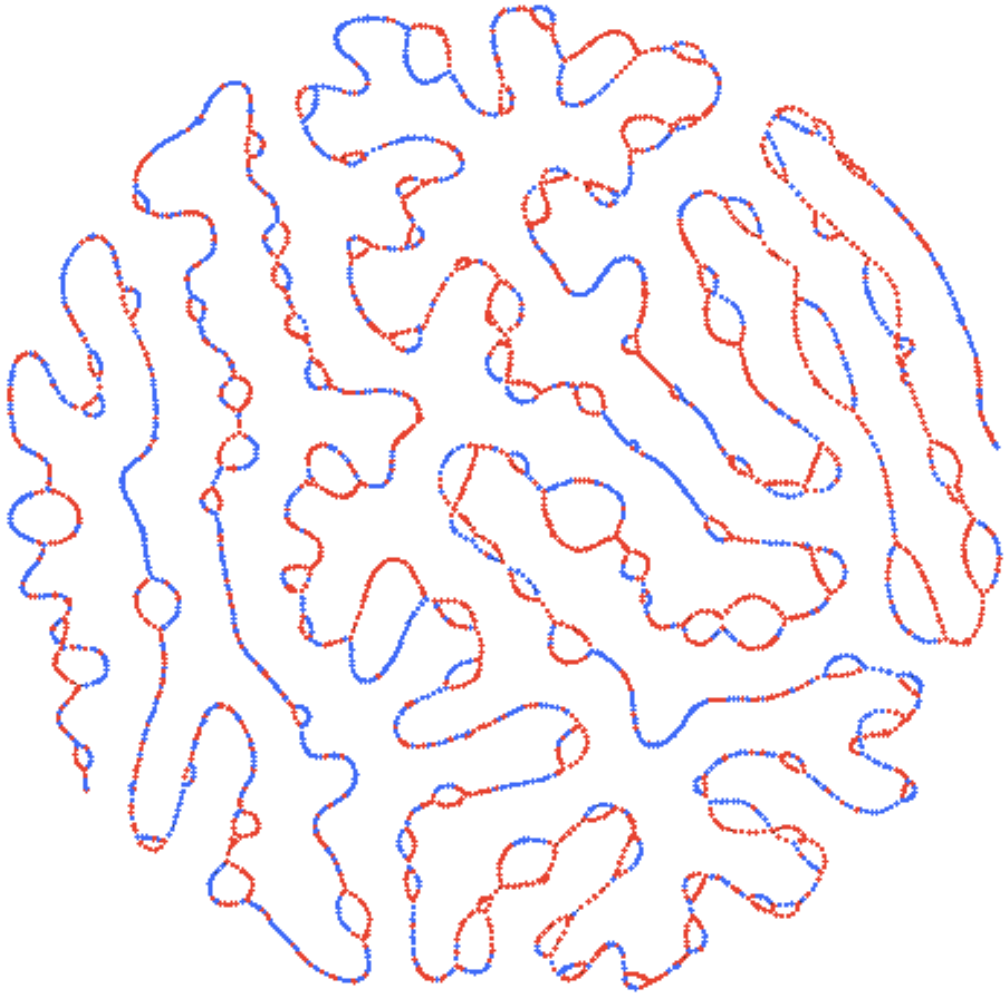
Roberts *et al.,* 2013

Long reads can also resolve haplotypes (with sufficient coverage)



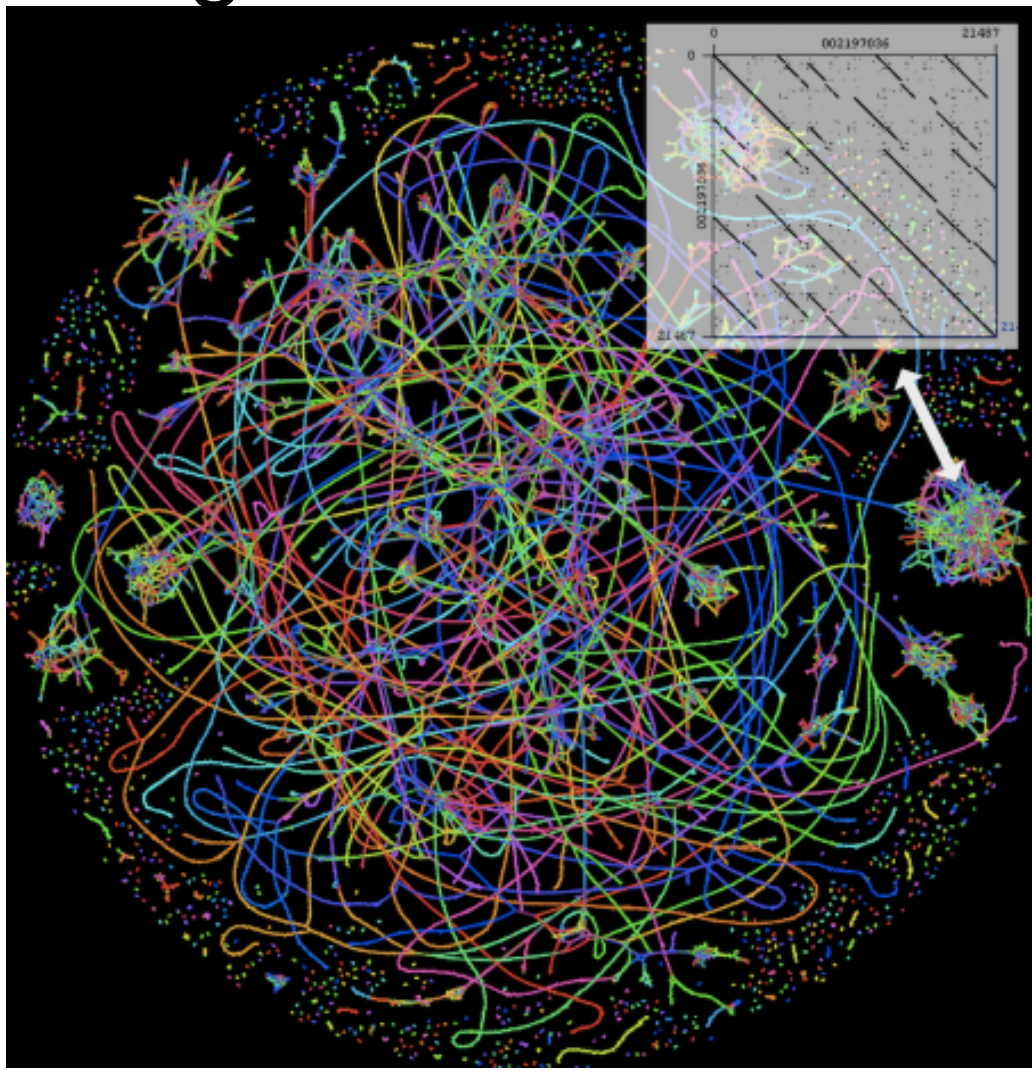
What you can do with long reads

Credit: Jason Chin
Two genomes. One assembly. Two colors. Many bubbles. Game against entropy.
http://t.co/uCPmxCRiZ6

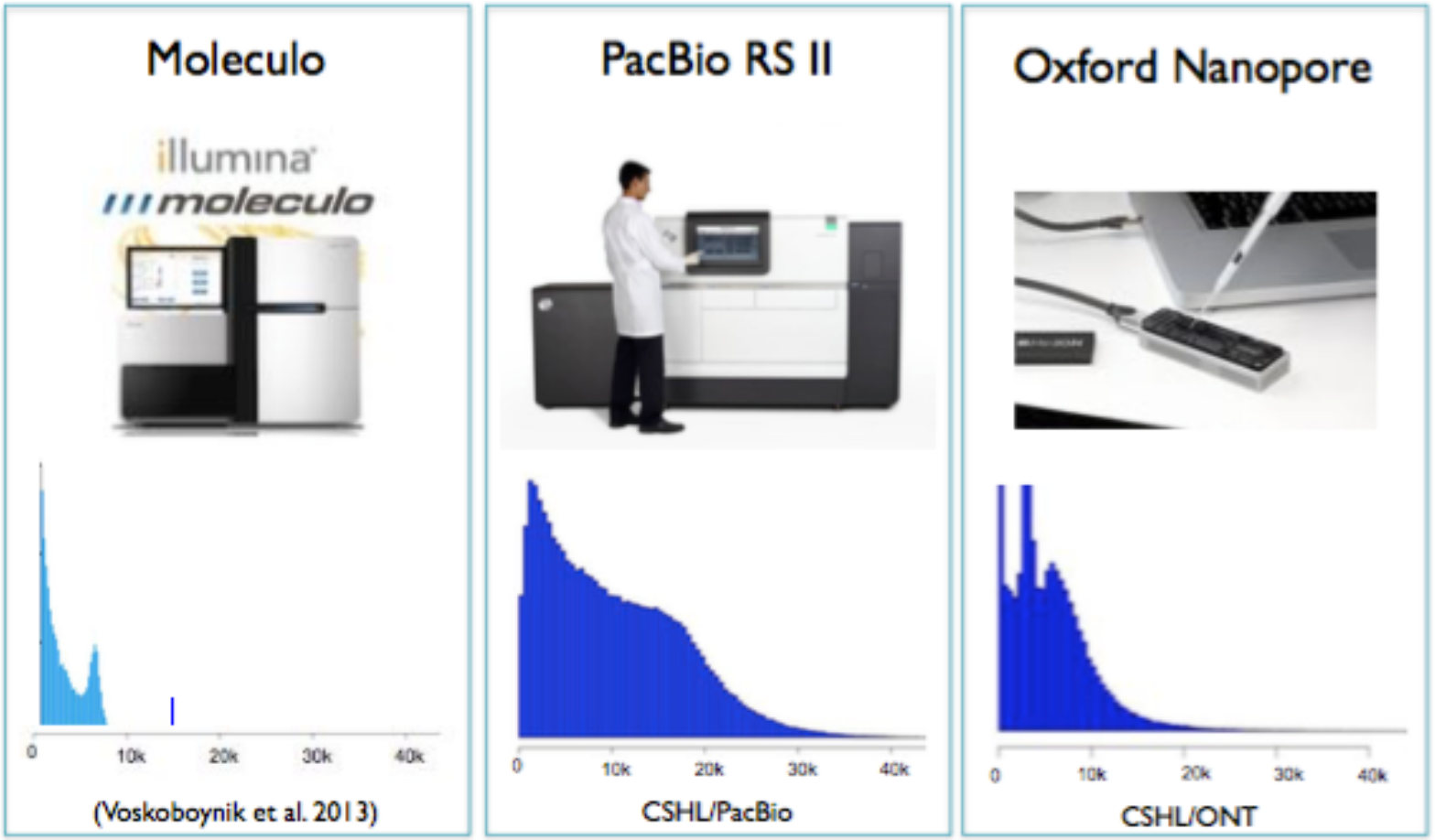


Current limitation of long reads..

Credit: Jason Chin
What are those blobs in the genome assembly graph? Intriguing repeats that have no NCBI blast hit.
http://t.co/2y7stBGs4W

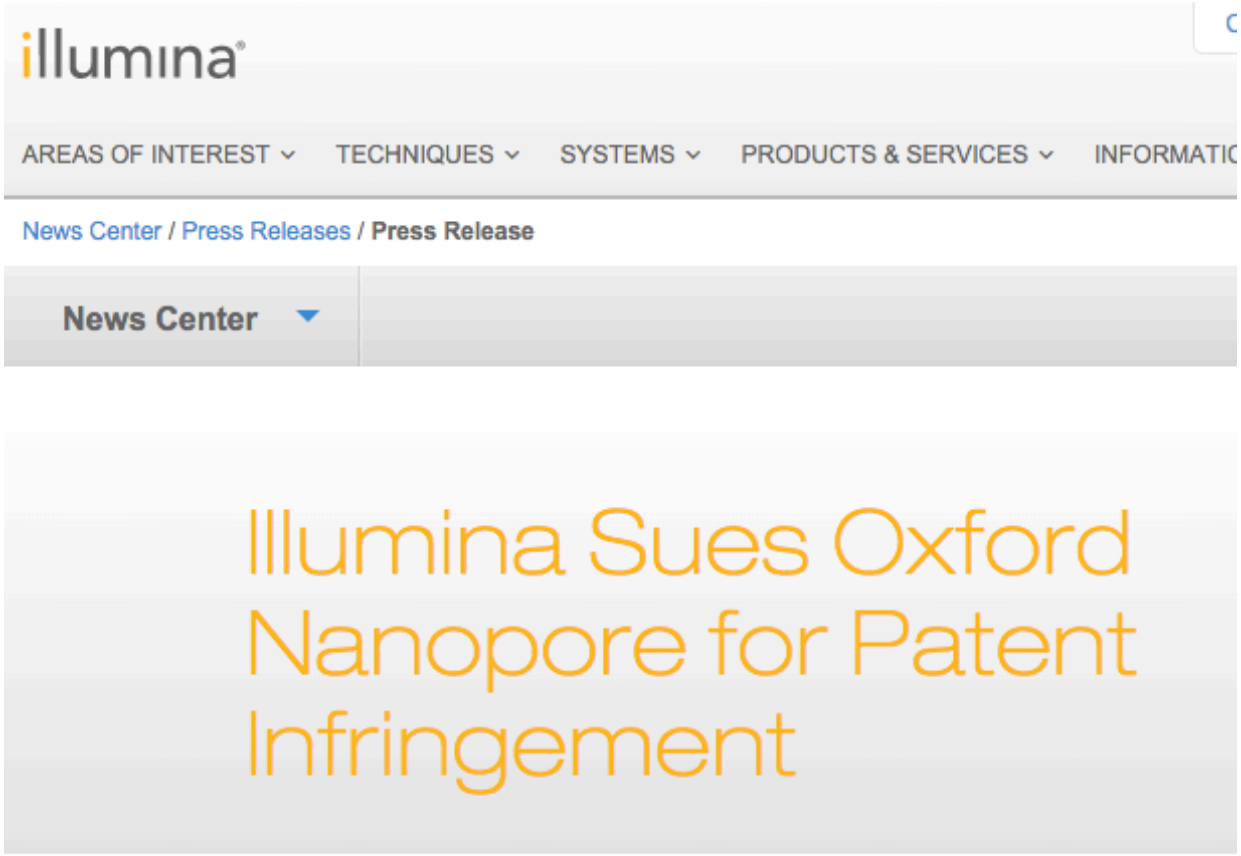


Moleculo losing out…



Credit: Michael Schatz

Then..


illumina
AREAS OF INTEREST
TECHNIQUES
SYSTEMS
PRODUCTS & SERVICES
INFORMATIC
News Center / Press Releases / Press Release
News Center
Illumina Sues Oxford Nanopore for Patent Infringement


Daniel MacArthur
@dgmacarthur
Follow
Sad @illumina employing patent litigation as a weapon against competitor @nanopore; they did same against @CompleteGenomic during their IPO
9:41 AM - 25 Feb 2016
16
20


Yaniv Erlich
@erlichya
Follow
First they laugh at you, then they sue you:@nanopore genomeweb.com/sequencing-tec…
12:11 AM - 25 Feb 2016
9
13

Things to consider

Problem	Description
Cost	In 2014 Illumina claimed the $1,000 genome barrier had been broken (if you first spend ~$10 million on hardware).
Library prep	A critical, and often overlooked, step in the process.
Sequence diversity	Illumina, 454, Ion Torrent, PacBio, Oxford Nanopore: which mix of sequence data will you be using?
Hardware	Some genome assemblers have very high CPU/RAM requirements. Might need specialized cluster.
Expertise	Not always easy to even get assembly software installed, let alone understand how to run it properly.
Software	There is a lot of choice out there.

Credit: Keith Bradnam

Things to consider

- $$$$$$$$$
- **Project type**
  virus, bacteria, eukaryote, meta-genome
- **Goals**
  Just an assembly to showcase the world?
    Sequence pandemic species = conservation? (No right or wrong answer)
  Any biological question?
  Why *de novo* sequence a species?

- **Hardware**
  - You need CPUs, but RAM is more important
    - Imagine storing all the hashes or kmers
  - This may change depending on nature of data (all long reads within n years?)

Recommendations

If Illumina only

- One Paired end library + Mate pair library of various insert sizes
- Bacteria: Spades
- Allpath or Masurca for rest
- Highly heterozygous genome / wild isolate: Pllatanus

Hybrid is okay but algorithms are still being developed

If you are serious and have $

- At least 100X of Pacbio

References

Most of the slides are inspired from:

http://www.slideshare.net/agbiotec/overview-of-genome-assembly-algorithms

Michael Schatz
http://schatzlab.cshl.edu/teaching/

Ben Langmead
http://www.langmead-lab.org/teaching-materials/

Not covered but should be

Alignment method in overlap graph
String graph (reduced overlap graph)
Shortest Superstring Problem (SSP)
Hamiltonian path

Velvet (the first DBG assembler)
Choice of kmers in DBG

Bloom filter

Interesting read

https://flxlexblog.wordpress.com/2015/04/09/on-graph-based-representations-of-a-set-of-genomes/

http://lh3.github.io/2014/07/25/on-the-graphical-representation-of-sequences/