Melbourne Bioinformatics

BIOINFORMATICS + DATA SERVICES + INFRASTRUCTURE, FOR LIFE SCIENCES TODAY

COMP90014

Algorithms for Bioinformatics

Week 8B: Assembly in Practice

Assembly In Practice

De Bruijn Graph Simplification

Kmers, coverage, depth

Assessing Assemblies

Software

De Bruijn Graph Simplification

Read Errors

Mainly affects De Bruijn Graphs

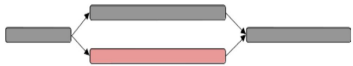
OLC: alignment allows mismatches, gaps
De Bruijn: use of kmers & exact matching

Middle of read: causes bubbles

End of read: cause spurs

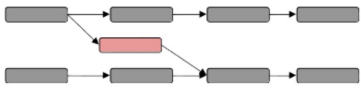
Use of kmers: causes erroneous edges

Bubbles



Spurs
(tips / dead ends)

Erroneous edges



De Bruijn Graph Simplification

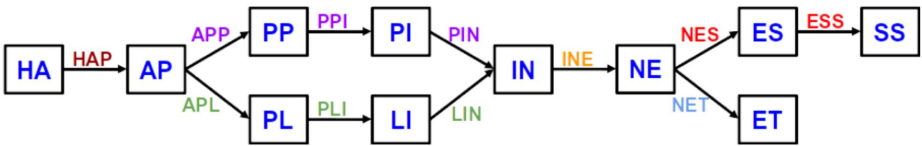
Read Errors

Example

```
HAPPI INESS APLIN PINET
               *       *
```

Unique 3-mers (k = 3):

```
HAP APP PPI
 INE NES ESS
  APL PLI LIN
    PIN NET
```

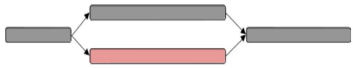


De Bruijn Graph Simplification

Graphs are very complex before simplification.

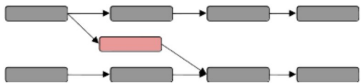
- Heterozygosity, read errors, kmer length
- Result in bubbles, spurs, erroneous edges

Bubbles



Spurs
(tips / dead ends)

Erroneous edges



De Bruijn Graph Simplification

Graphs are very complex before simplification.

- Heterozygosity, read errors, kmer length
- Result in bubbles, spurs, erroneous edges

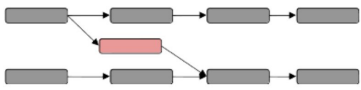
Naively

- Remove all nodes with low coverage.
- Eg. Expected = 30x
- Remove nodes <= 5x coverage

Bubbles

Spurs
(tips / dead ends)

Erroneous edges



De Bruijn Graph Simplification

杂合性（Heterozygosity）：一个位置有不同的碱基（即存在变异）。
读错误（Read errors）：测序过程中产生的错误。
kmer长度：选择的kmer长度可能不适合测序数据的特点

Graphs are very complex before simplification.

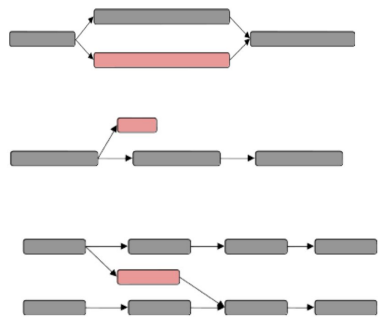
- Heterozygosity, read errors, kmer length
- Result in bubbles, spurs, erroneous edges

Naively 解决方法 移除覆盖度低的节点

- Remove all nodes with low coverage.
- Eg. Expected = 30x
- Remove nodes <= 5x coverage

Issues

- May remove genuine regions (just had low sequencing depth)
- Doesn't address heterozygosity



Will explore how Velvet handles graph simplification.

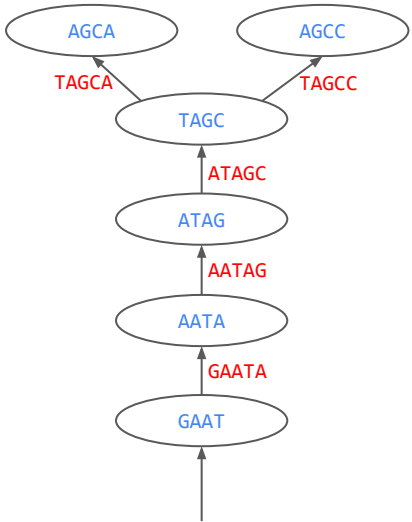
例如，如果一个基因组位置在测序过程中被10个不同的reads覆盖，那么这个位置的覆盖度就是10x。

De Bruijn Graph Simplification

Step 1: Coalesce non-branching paths

How we built the De Bruijn Graph:

- Nodes: prefix / suffix
- Edges: kmers



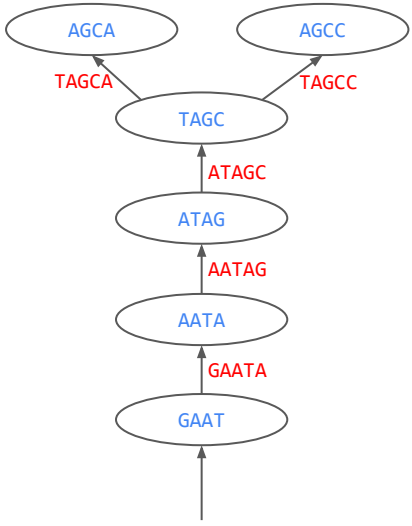
De Bruijn Graph Simplification

Step 1: Coalesce non-branching paths

How we built the De Bruijn Graph:

- Nodes: prefix / suffix
- Edges: kmers

Now we're simplifying. No reason to keep structure as-is.



De Bruijn Graph Simplification

Step 1: Coalesce non-branching paths

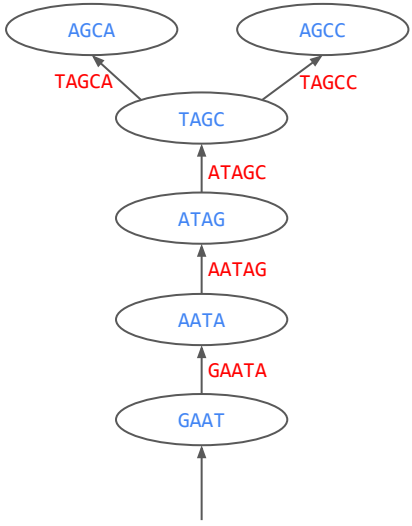
How we built the De Bruijn Graph:

- Nodes: prefix / suffix
- Edges: kmers

Now we're simplifying. No reason to keep structure as-is.

First task: collapse linear chains into single node.

- Improves space performance (less nodes / edges)
- Improves time performance (traverse less edges)



De Bruijn Graph Simplification

Step 1: Coalesce non-branching paths

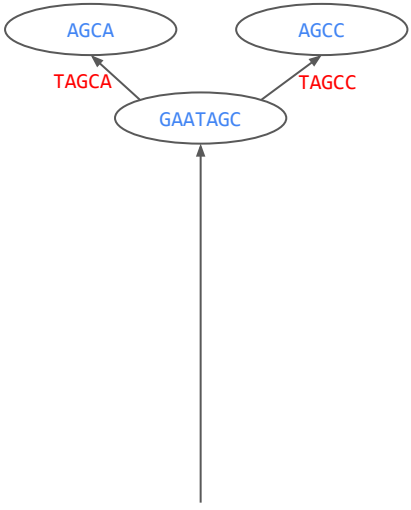
How we built the De Bruijn Graph:

- Nodes: prefix / suffix
- Edges: kmers

Now we're simplifying. No reason to keep structure as-is.

First task: collapse linear chains into single node.

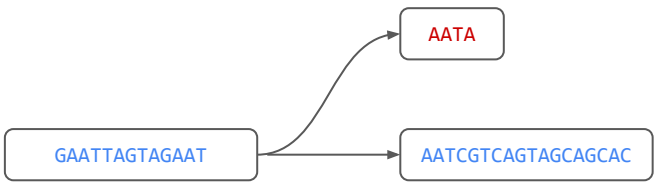
- Improves space performance (less nodes / edges)
- Improves time performance (traverse less edges)



De Bruijn Graph Simplification

Step 1: Remove Spurs (tips)

Spur: chain of nodes that is disconnected on one end



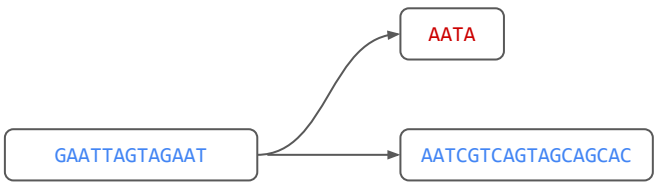
De Bruijn Graph Simplification

Step 1: Remove Spurs (tips)

Spur: chain of nodes that is disconnected on one end

Similar to OLC spur removal. Straightforward.

Can't be heavy-handed!
Don't want to remove genuine sequence.



De Bruijn Graph Simplification

不能过于粗暴（不能heavy-handed

筛选tip的时候看length和mc

Step 1: Remove Spurs (tips)

Identify tips via length and minority count

Spur: chain of nodes that is disconnected on one end

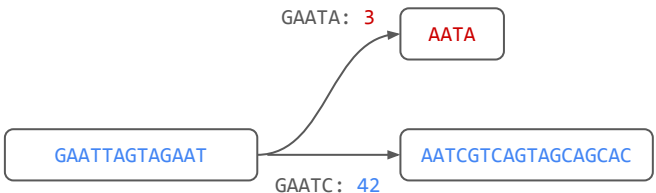
Length: Path length along tip < 2k.

Similar to OLC spur removal. Straightforward.

Minority count:

Can't be heavy-handed!
Don't want to remove genuine sequence.

- The branch leading to the tip is an inferior route.
- Edge (kmer) to tip branch has lower occurrences than other branches



长度：沿着尖刺的路径长度小于2k，这里的k是k-mer的长度。例如，如果k=20，那么尖刺的长度会小于40个核苷酸。
少数计数：通向尖刺的分支是次优路径，意味着这条路径比其他路径覆盖度低或出现次数少。

De Bruijn Graph Simplification

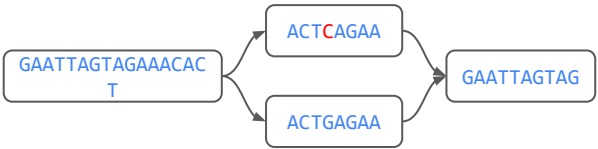
Step 2: Pop bubbles

Conditions for merging (popping bubbles)

De Bruijn Graph Simplification

Step 2: Pop bubbles

Scenario 1: sequencing error / heterozygosity

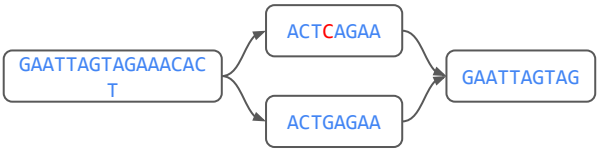


Conditions for merging (popping bubbles)

De Bruijn Graph Simplification

Step 2: Pop bubbles

Scenario 1: sequencing error / heterozygosity



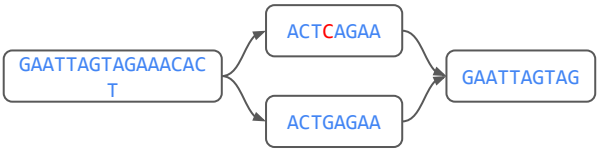
Conditions for merging (popping bubbles)

- Must have same start and end node

De Bruijn Graph Simplification

Step 2: Pop bubbles

Scenario 1: sequencing error / heterozygosity



Scenario 2: both paths valid - unique sequences

Conditions for merging (popping bubbles)

- Must have same start and end node

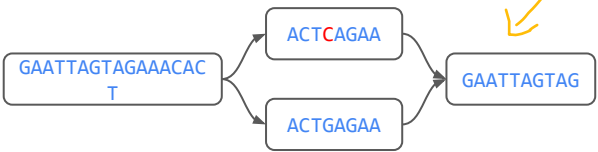
De Bruijn Graph Simplification

Step 2: Pop bubbles

Conditions for merging (popping bubbles)

- Must have same start and end node
- Must have similar sequence

Scenario 1: sequencing error / heterozygosity

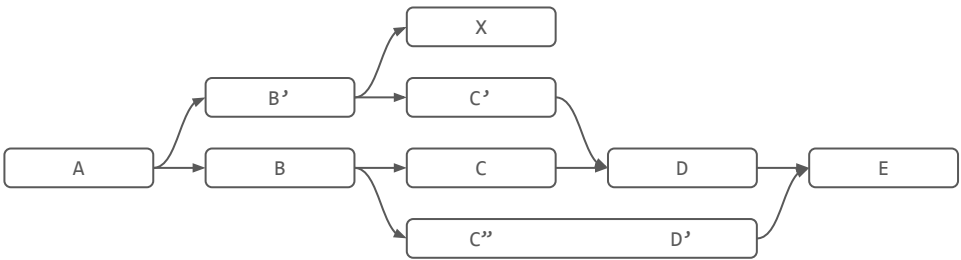


Scenario 2: both paths valid - unique sequences

De Bruijn Graph Simplification

Step 2: Pop bubbles

`Scenario 3: complex`
(in this figure - nodes labelled, not showing seq)



Conditions for merging (popping bubbles)

- Must have same start and end node
- Must have similar sequence

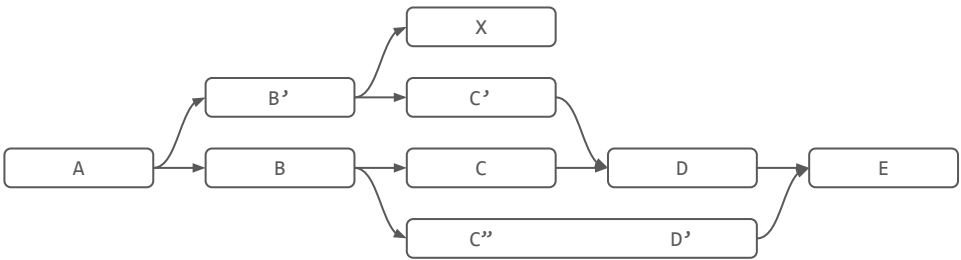
De Bruijn Graph Simplification

Step 2: Pop bubbles

`Scenario 3: complex`
(in this figure - nodes labelled, not showing seq)

Conditions for merging (popping bubbles)

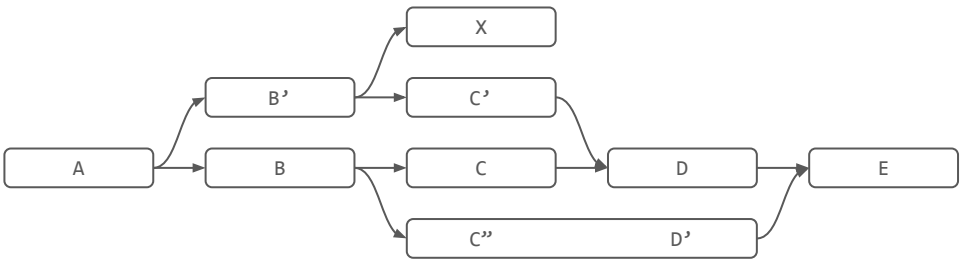
- Must have same start and end node
- Must have similar sequence
- Only merge two paths at a time



De Bruijn Graph Simplification

Step 2: Pop bubbles

`Scenario 3: complex`
(in this figure - nodes labelled, not showing seq)



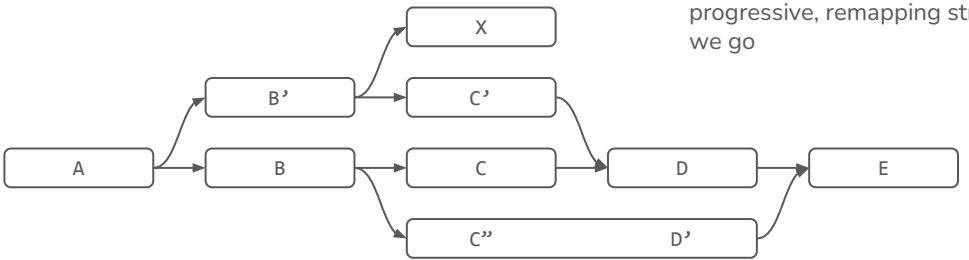
Conditions for merging (popping bubbles)

- Must have same start and end node
- Must have similar sequence
- Only merge two paths at a time
- Cannot just delete nodes because this separates part of graph

De Bruijn Graph Simplification

Step 2: Pop bubbles

`Scenario 3: complex`
(in this figure - nodes labelled, not showing seq)

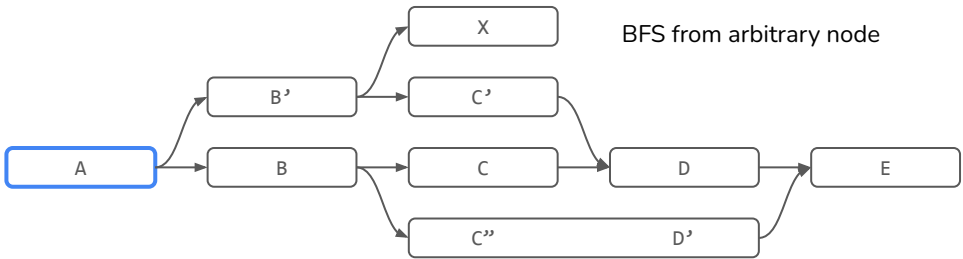


Conditions for merging (popping bubbles)

- Must have same start and end node
- Must have similar sequence
- Only merge two paths at a time
- Cannot just delete nodes because this separates part of graph. Must be progressive, remapping structures as we go

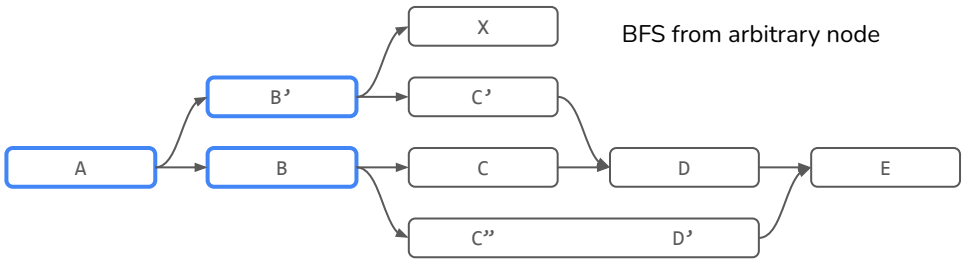
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



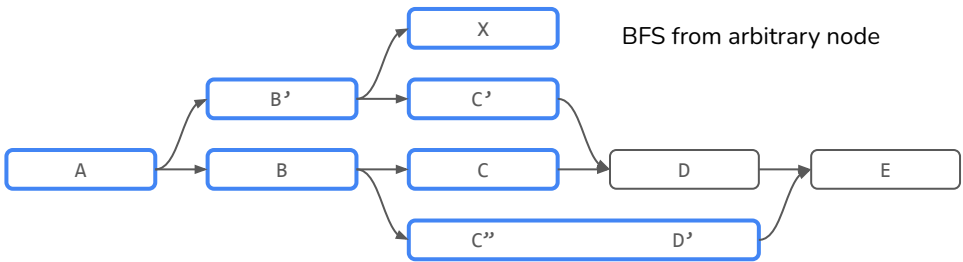
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



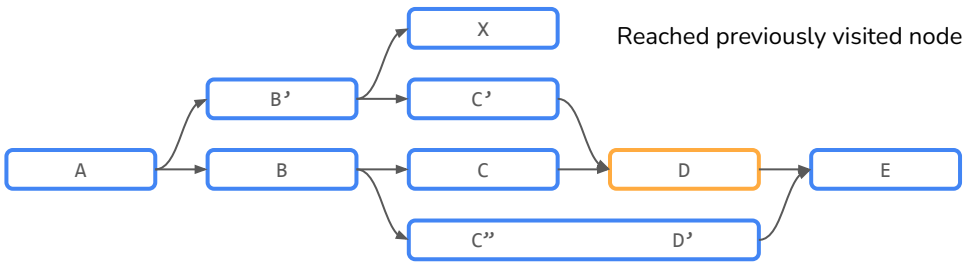
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



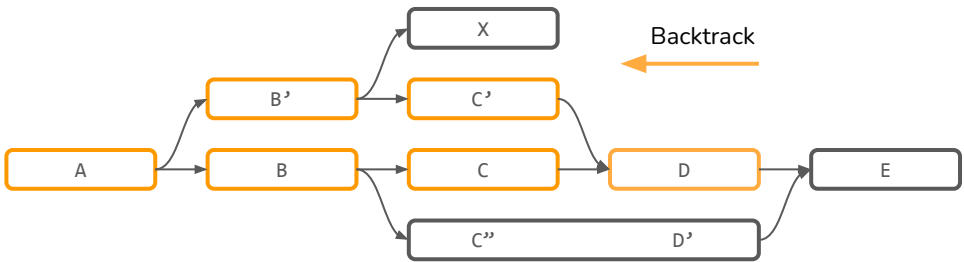
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



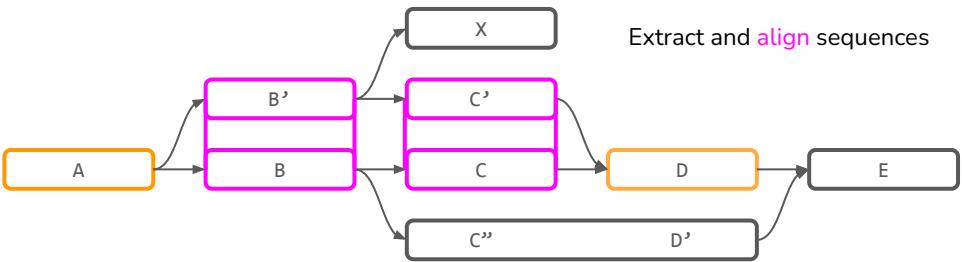
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

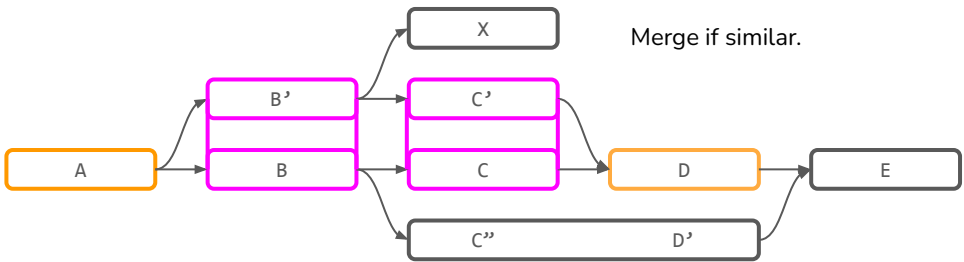
Step 2: Pop bubbles - Tour Bus



Extract and align sequences

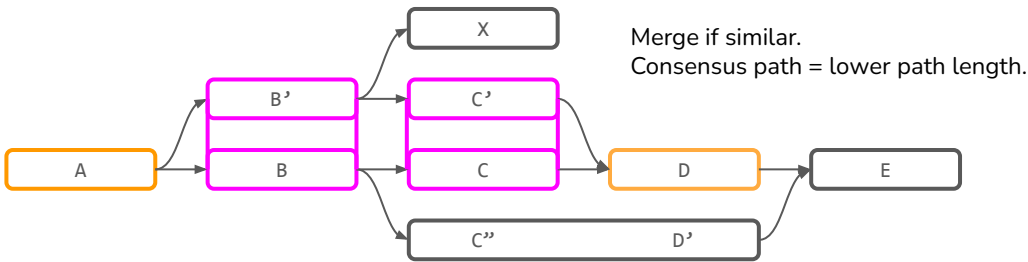
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

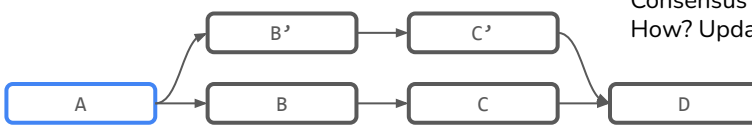
Step 2: Pop bubbles - Tour Bus



Merge if similar.
Consensus path = lower path length.

De Bruijn Graph Simplification

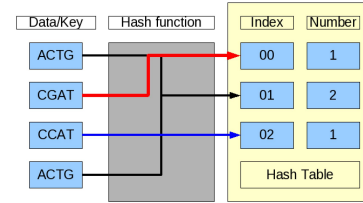
Step 2: Pop bubbles - Tour Bus

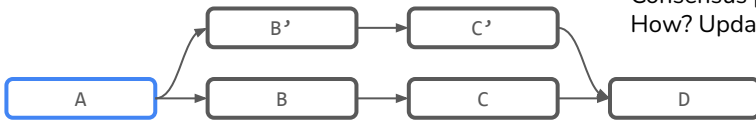


Merge if similar.
Consensus path = lower path length.
How? Update path lengths during BFS traversal.

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus





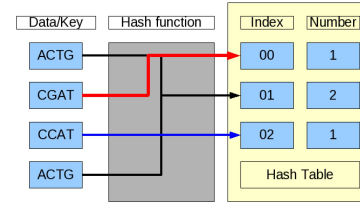
Merge if similar.
Consensus path = lower path length.
How? Update path lengths during BFS traversal.

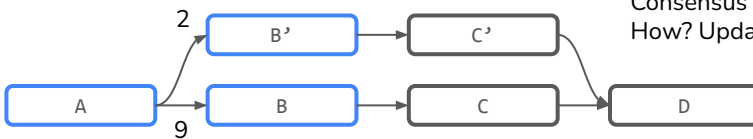
AB	9
BC	12
CD	7

AB’	2
B’C’	2
C’D	1

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus





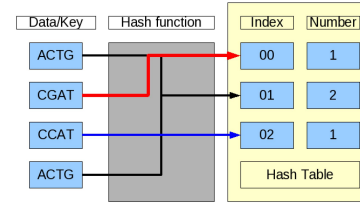
Merge if similar.
Consensus path = lower path length.
How? Update path lengths during BFS traversal.

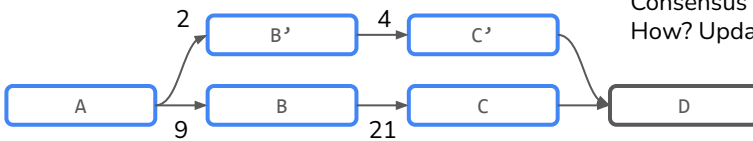
AB	9
BC	12
CD	7

AB’	2
B’C’	2
C’D	1

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus





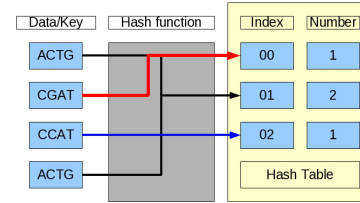
Merge if similar.
Consensus path = lower path length.
How? Update path lengths during BFS traversal.

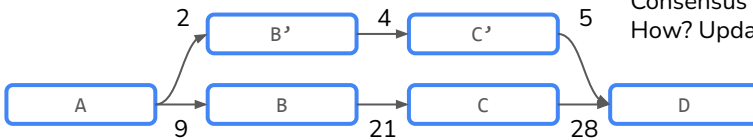
AB	9
BC	12
CD	7

AB’	2
B’C’	2
C’D	1

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus　　kmer的出现次数





Merge if similar.
Consensus path = lower path length.
How? Update path lengths during BFS traversal.

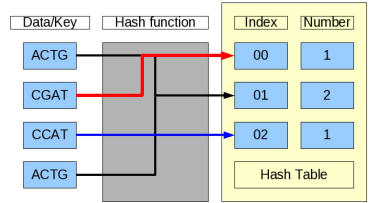
AB	9
BC	12
CD	7

AB’	2
B’C’	2
C’D	1

路径权重代表了这条路径上序列的频率或可信度。

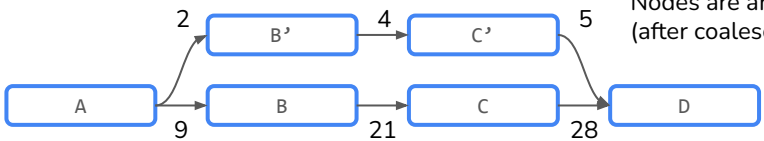
这里 权重可能代表特定序列在读序数据中出现的次数

De Bruijn Graph Simplification



Step 2: Pop bubbles - Tour Bus

图中B到C和B'到C'形成了两个泡泡，算法将识别这些泡泡，并根据一定的规则（如路径长度、权重或覆盖度）决定哪些路径保留，哪些路径移除。



A tweak!
Nodes are arbitrary length
(after coalescing linear chains)

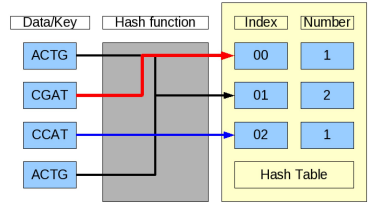
AB	9
BC	12
CD	7

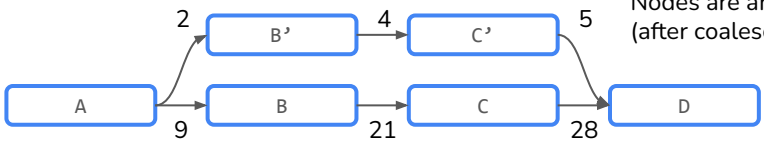
AB’	2
B’C’	2
C’D	1

意味着在这一步骤中，具有线性关系的节点（没有分叉的节点链）可以合并成一个具有任意长度的节点。这有助于进一步简化图的结构

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus





A tweak!
Nodes are arbitrary length
(after coalescing linear chains)

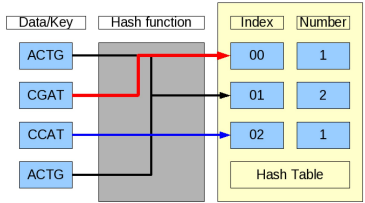
AB	9
BC	12
CD	7

AB’	2
B’C’	2
C’D	1

Multiplicity was updated during coalescing.

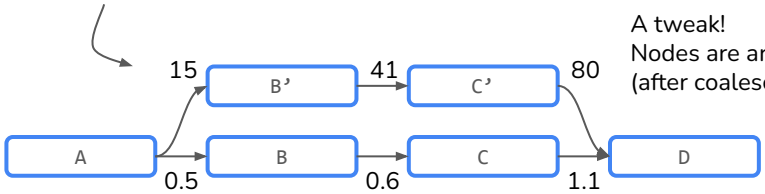
Path length (AB) = len(B) / multiplicity

De Bruijn Graph Simplification



Step 2: Pop bubbles - Tour Bus

Actual path lengths using definition



A tweak!
Nodes are arbitrary length
(after coalescing linear chains)

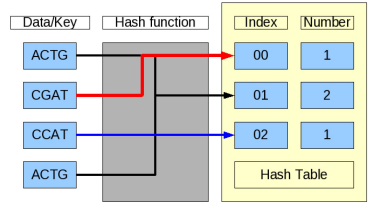
AB	9
BC	12
CD	7

AB’	2
B’C’	2
C’D	1

Multiplicity was updated during coalescing.

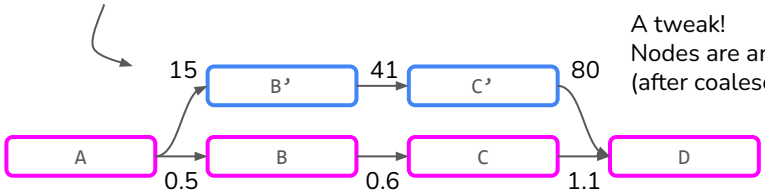
Path length (AB) = len(B) / multiplicity

De Bruijn Graph Simplification



Step 2: Pop bubbles - Tour Bus

Actual path lengths using definition



A tweak!
Nodes are arbitrary length
(after coalescing linear chains)

AB	9
BC	12
CD	7

AB’	2
B’C’	2
C’D	1

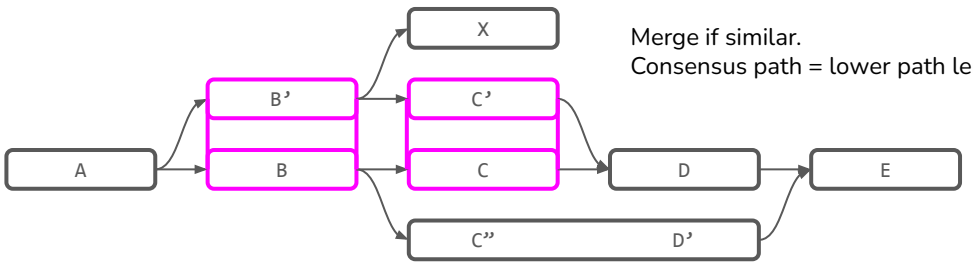
Consensus path has shortest path length
Essentially majority voting

Multiplicity was updated during coalescing.

Path length (AB) = len(B) / multiplicity

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus

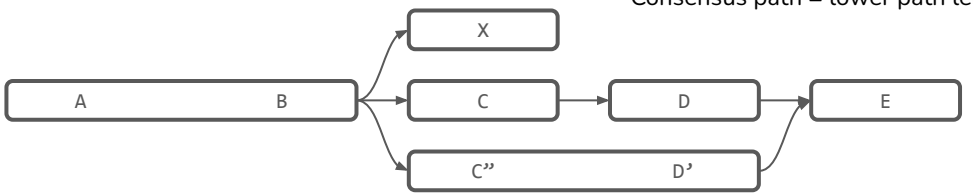


Merge if similar.
Consensus path = lower path length.

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus

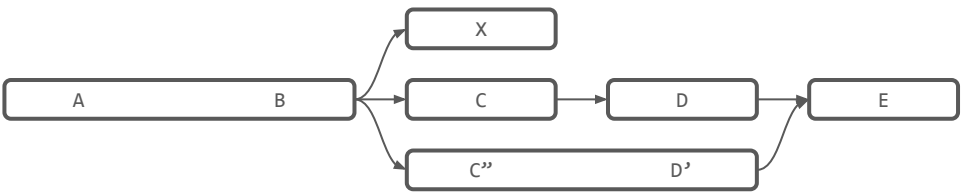
Merge if similar.
Consensus path = lower path length.



De Bruijn Graph Simplification

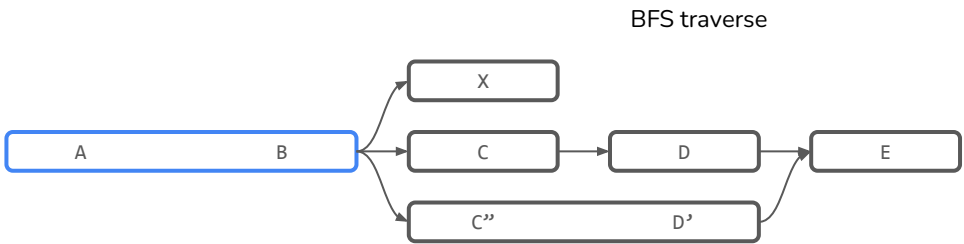
Step 2: Pop bubbles - Tour Bus

Repeat the process!



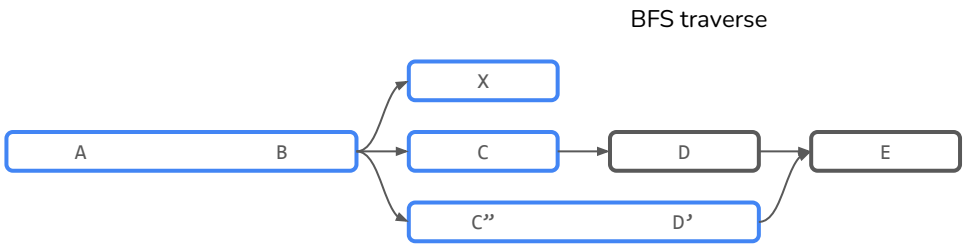
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



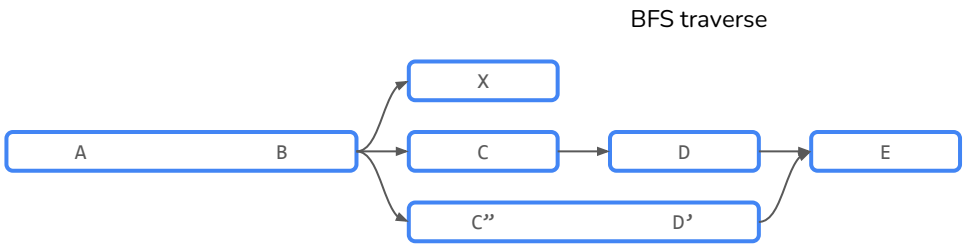
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



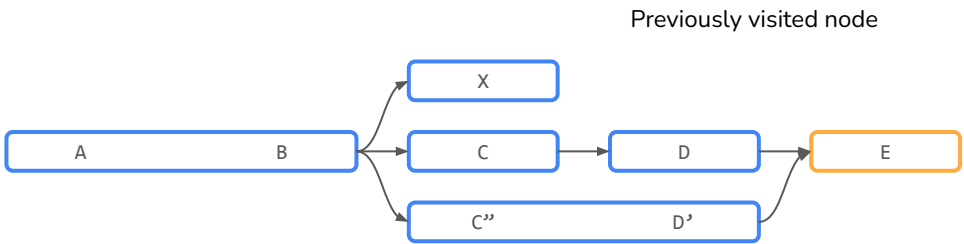
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



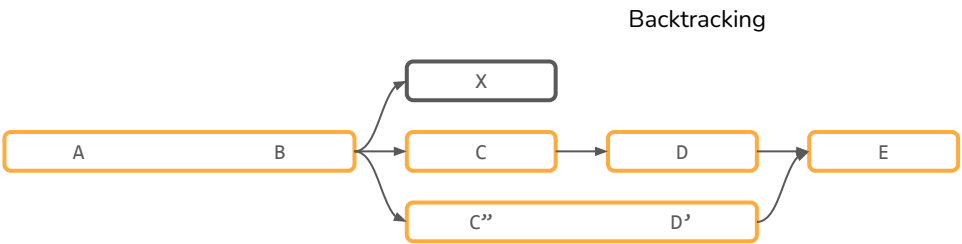
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

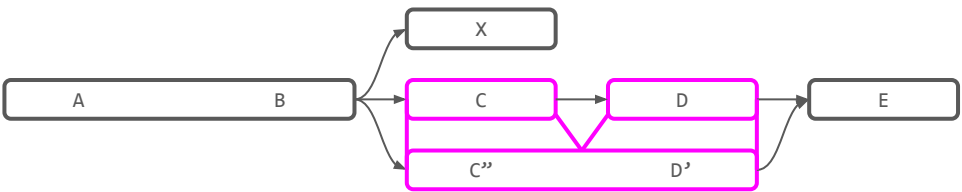
Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

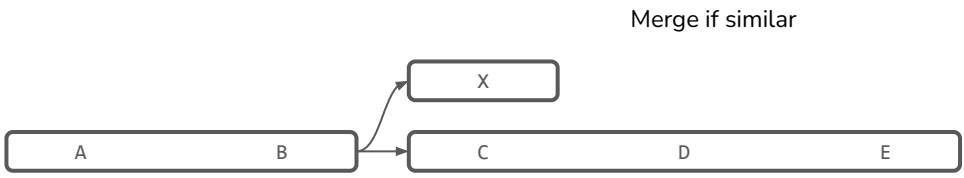
Step 2: Pop bubbles - Tour Bus

Extracting and aligning sequences



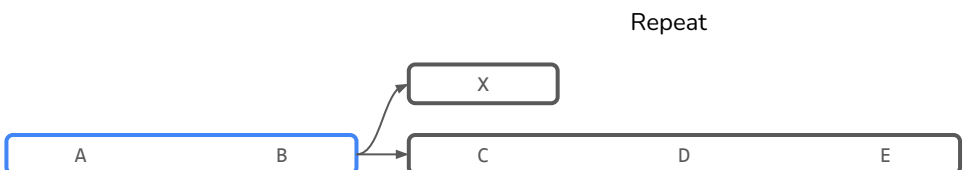
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

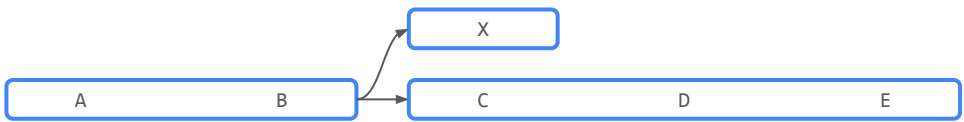
Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus

No more moves.



Zerbino & Birney (2008): Velvet

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus

Paths are redundant if start and end at same nodes. Also required to have similar sequence.

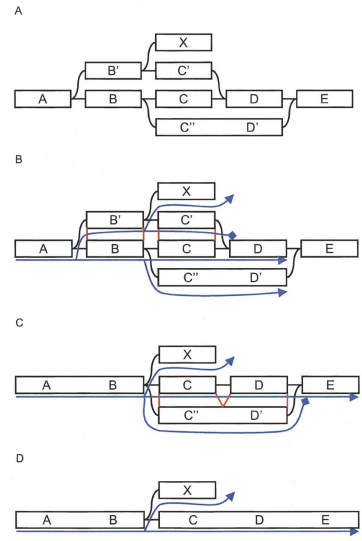
Tour Bus algorithm

Dijkstra-like breadth-first search

Since we have coalesced non-branching paths already, start at any arbitrary node. All have 2+ in- or out-edges.

Visit nodes in BFS manner

When reach visited node (join), backtrack to determine best path.



De Bruijn Graph Simplification

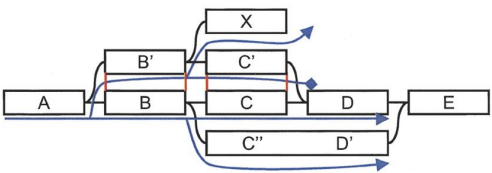
Step 2: Pop bubbles - Tour Bus

For given node, walk BFS to the right.
Update path lengths as we go.

A → B: length(B) % multiplicity of edge to B

This is essentially priority voting.

Prioritises higher confidence branches.



Zerbino & Birney (2008): Velvet

When reach visited node (D):

Traceback visited edges

Find closest common ancestor (A)

Extract sequences from traceback paths

Align each sequence & merge if similar (red)

Merging occurs consecutively, left to right

Consensus sequence determined by path length

De Bruijn Graph Simplification

Tour Bus的谨慎性：Tour Bus算法非常谨慎，它保留了大部分独特的区域。
剩余错误：剩下的错误可能是由持续的错误、嵌合体（chimeras，指的是在生物学中由两种不同物种的遗传物质混合而成的组织），或是重复序列导致的。

Step 3: Remove Erroneous Connections

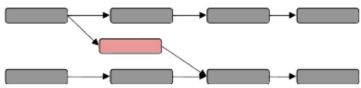
Occurs after Tour Bus.

- Tour bus is cautious, preserves most unique regions
- Remaining are erroneous due to persisting errors, chimeras, repeats

难点：错误连接通常不会形成可识别的循环或结构，使得它们难以识别和移除。

Tricky!

- Don’t form recognizable loop / structure



Only solution - coverage cutoff

- Remove any node with less then set threshold for coverage.
- Repetitive sequence will remain, as it’s coverage will be high. Can’t do much about this.

It is not possible to resolve a repeat of length **N** with reads less than length **N**

It is not possible to resolve a repeat of length **N** with reads less than length **N**

Both OLC and De Bruijn approaches handle repeats by essentially leaving them out.

After simplification, nodes with 2+ in-edges / out-edges are unresolvable

At these branch points we break the assembly into fragments (contigs)

It is not possible to resolve a repeat of length **N** with reads less than length **N**

解决长度为N的重复序列，如果读取长度小于N，是不可能的。

Both OLC and De Bruijn approaches handle repeats by essentially leaving them out.

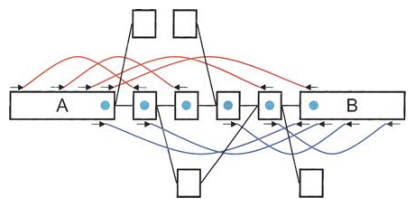
After simplification, nodes with 2+ in-edges / out-edges are unresolvable

At these branch points we break the assembly into fragments (contigs)

那些具有两个或以上进入（入边）或离开（出边）的节点被认为是无法解决的（即无法确定它们的精确连接顺序）。

*If paired-end short reads (+ De Bruijn):

- Can use mate pairs to help resolve
- See Velvet Breadcrumb algorithm https://doi.org/10.1101/gr.074492.107



Assembly In Practice

De Bruijn Graph Simplification

Kmers, coverage, depth

Assessing Assemblies

Software

Coverage *vs.* depth

Coverage (or breadth)
Fraction of the genome sequenced by at least one read

Depth
Average number of reads that cover any given region

Intuitively: more reads should increase coverage and depth

Example: Read length = $\frac{1}{8}$ Genome length

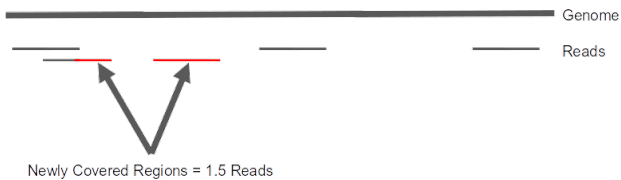
Coverage $= \frac{3}{8}$

Depth $= \frac{3}{8}$

Example: Read length = $\frac{1}{8}$ Genome length

Coverage $= \frac{4.5}{8}$

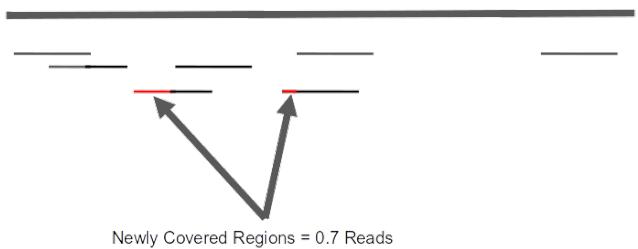
Depth $= \frac{5}{8}$



Example: Read length = $\frac{1}{8}$ Genome length

Coverage $= \frac{5.2}{8}$

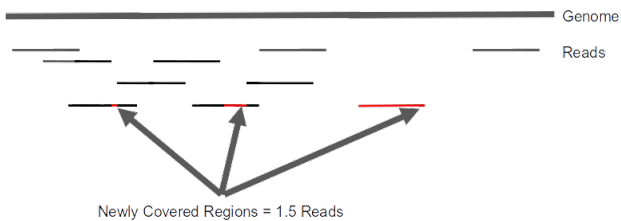
Depth $= \frac{7}{8}$



Example: Read length = $\frac{1}{8}$ Genome length

Coverage $= \frac{6.7}{8}$

Depth $= \frac{10}{8}$



Calculating depth

$$\text{Depth} = N \times \frac{L}{G}$$

$$\text{Depth} = \frac{Y}{G}$$

N = Number of reads
L = Length of a read
G = Genome length
Y = Sequence yield ($N \times L$)

k-mer depth *vs.* read depth

depth: The number of reads covering each position in the genome.

- Depends on number of reads and read length (L).

***k*-mer depth:** The number of observed k-mers that match a given k-length segment of the genome.

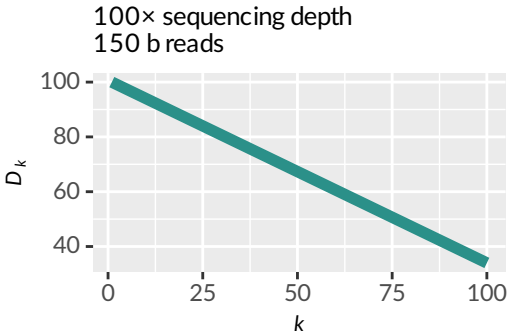
- *i.e.* the number of occurrences of an identical k-mer.

k-mer depth depends on read length:

$$D_k = D \times \frac{L - k + 1}{L}$$

- ***depth*** and ***coverage*** are related but not the same.
- number of sequences at a single position *vs.* fraction of genome sequenced
- don't worry about the terminology

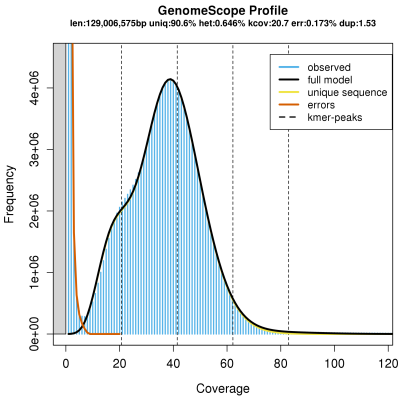
Choosing k in practice



$$D_k = D \times \frac{L - k + 1}{L}$$

- **Lower k**
  - more connections → more loops
  - lower chance of resolving small repeats
  - higher k-mer coverage
- **Higher k**
  - fewer connections → more components
  - higher chance of resolving small repeats
  - lower k-mer coverage
- Choose k (empirically) to balance these effects

k-mer based genome size estimation



Here, "Coverage" means *k*-mer depth

- for *k*-mer depth,

$$D_k = D \times \frac{L - k + 1}{L}$$

- we know read depth is the yield over the genome size,

$$D = N \times \frac{L}{G}$$

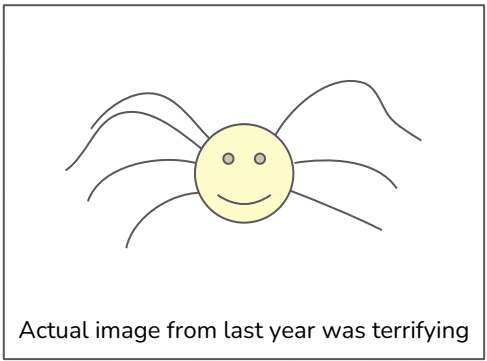
- just plug in values to get genome size,

$$G = N \times \frac{L - k + 1}{D_k}$$

- try to work out the genome size here (pretend you can't see the answer):
  - the graph was made using 31-mers
  - the peak is roughly $k = 40$
  - we had 100 million 150 b reads

Vurture *et al.*, 2017, *Bioinformatics*

Example: Coverage of the Tarantula genome



Actual image from last year was terrifying

The size estimate of the tarantula genome based on k-mer analysis is 6 Gb and we sequenced at 40× depth from a single female A. geniculata.

$$\text{Depth} = N \times \frac{L}{G}$$

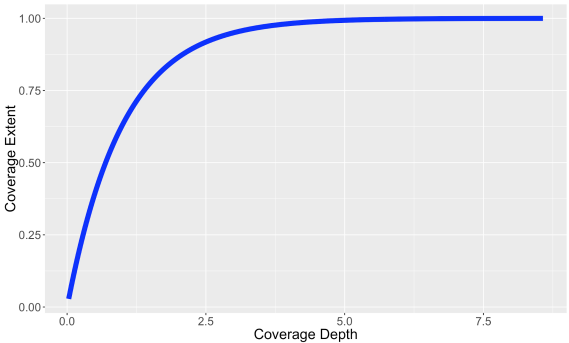
$$40 = N \times \frac{100}{6 \times 10^9}$$

$$N = 40 \times \frac{6 \times 10^9}{100}$$

$$N = 2.4 \times 10^9$$

Sanggaard *et al.* 2014, https://doi.org/10.1038/ncomms4765

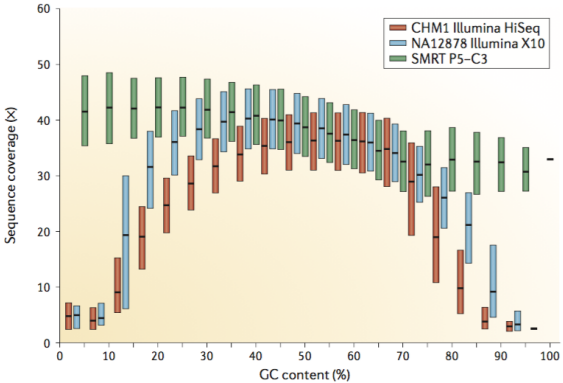
Coverage and depth are related



Approximately, assuming random reads:

$$\text{Coverage} = 1 - e^{-\text{Depth}}$$

These days: just buy more sequence



- Sequencing is not random
  - GC and AT rich regions are under represented
  - Other chemistry quirks
- More depth needed for:
  - sequencing errors
  - polymorphisms

Assembly In Practice

De Bruijn Graph Simplification

Kmers, coverage, depth

Assessing Assemblies

Software

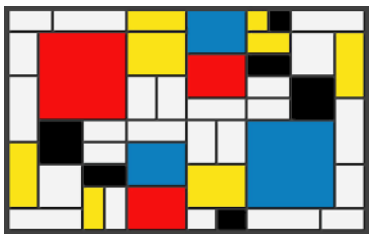
Contiguity

Completeness

Correctness

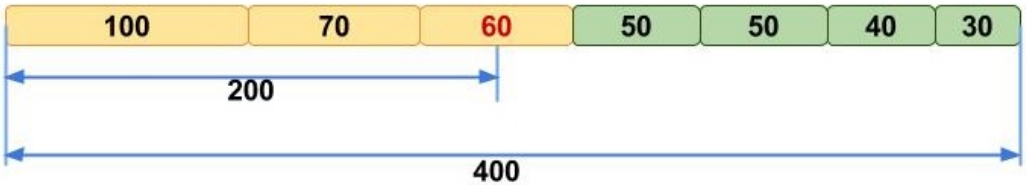
Contiguity

- Aim:
  - Fewer contigs
  - Longer contigs



- Metrics
  - Number of contigs
  - Average contig length
  - Median contig length
  - Maximum contig length
  - N_{50}

Contiguity: N_{50} length



The sequence length of the shortest contig at 50% of the total genome length

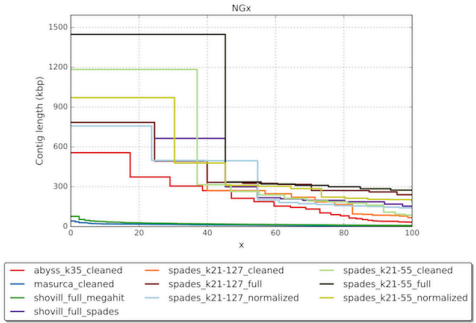
Contiguity: N_{50} length

Example: 1 Mbp genome

N50 size = 30 kbp

(300k+100k+45k+45k+30k = 520k >= 500kbp)

Area under the N_x curve?



$$\text{auN} = \frac{\sum_i L_i^2}{\sum_j L_j}$$

- sum of the scaffold lengths squared, divided by genome size
- takes the whole distribution into account
- quicker to calculate
- nobody uses it

Heng Li's blog post

Completeness

Total size:

$$\text{Completeness} = \frac{\text{Assembled genome size}}{\text{Estimated genome size}}$$

- Proportion of the original genome represented by the assembly
  - between 0 and 1
  - estimates are not perfect

Core genes:

$$\text{Completeness} = \frac{\text{Number of core genes in assembly}}{\text{Number of core genes in database}}$$

- Proportion of expected ***core genes*** found in other organisms that are present in the assembly
- Assumes that core genes and other genes are assembled at the same rate

BUSCO

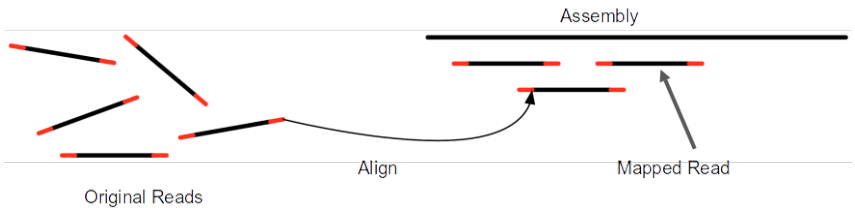
busco.ezlab.org

Correctness

- Proportion of the assembly that is free from mistakes
- Errors include:
  1. Mis-joins
  2. Repeat compressions
  3. Unnecessary duplications
  4. Indels / SNPs caused by assembler

Correctness: check for self consistency

- Align all the reads back to the contigs
- Look for inconsistencies



Assembly In Practice

De Bruijn Graph Simplification

Kmers, coverage, depth

Assessing Assemblies

Software

de Bruijn graph assemblers

`Velvet`

- `Velveth`
  - makes and counts (***hashes***) the *k*-mers in O(*N*) time
- `Velvetg`
  - Makes the graph in O(*U*) time. *U* = unique *k*-mers.
  - *k*-mer depth cutoff to simplify the graph
  - Makes contigs in O(*E*) time. *E* = edges in graph
- single *k*-mer size
- requires multiple runs to optimise parameters

Daniel Zerbino and Ewan Birney, EBI, 2008.

Spades

- uses multiple *k*-mer sizes
  - low, medium and high *k*-mer size
  - graph has connectivity AND specificity
- performs error correction on the reads first
- maps reads back to contigs to check consistency
- development still active
- slower than `Velvet`, but don't have to run multiple times.

Pavel Pevzner's group, St. Petersburg

Some commonly-used assemblers

Short-read based

de Bruijn graph:

- SPAdes
- Velvet
- AbySS
- DISCOVAR / ALLPATHS
- Meraculous
- SOAPdenovo
- many more: see *De novo sequence assemblers* on Wikipedia

OLC algorithm:

- wgs-assembler (celera)

Long read (mostly OLC)

- Canu
- FALCON
- Flye
- Shasta

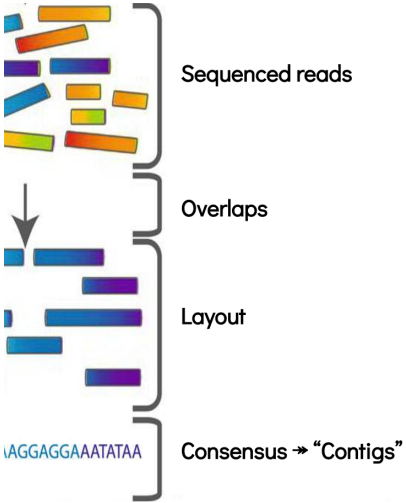
Hybrid

- MaSuRCA
- Unicycler

Special cases

- metagenomes, transcriptomes ...

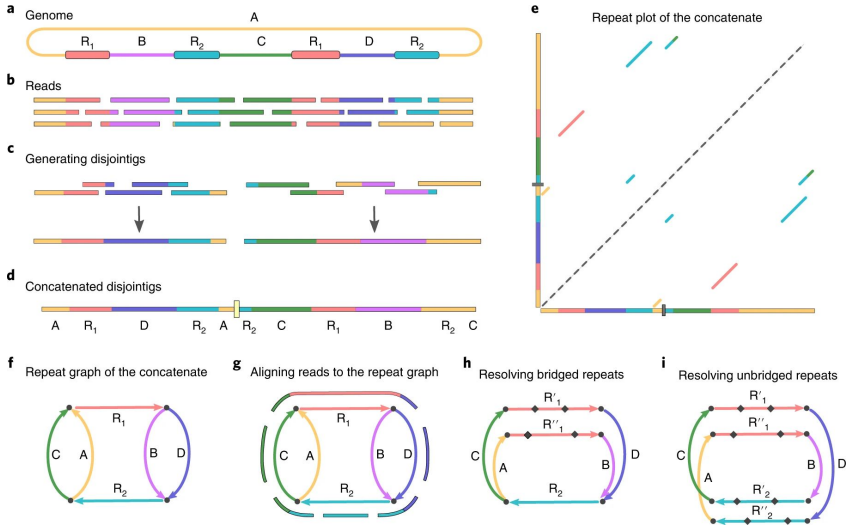
Canu assembler



- fork of `Celera` assembler
- corrects the reads
  - takes longest reads to make $\approx 20\times$ coverage depth
  - uses left over shorter reads to “correct” long reads
  - trims off the low quality ends
- OLC assembly

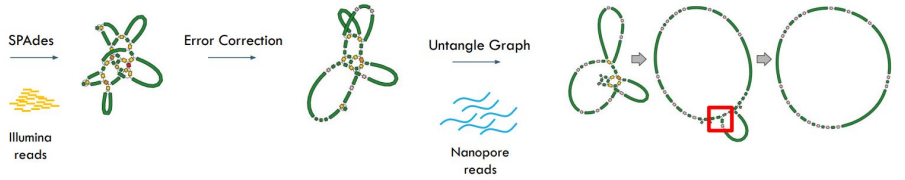
Koren, Walenz, Berlin, Miller and Phillipy, Genome Research 2017.

Flye: repeat graph assembler



Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin and Pavel A. Pevzner, Nat. Biotech. 2019

Unicycler



Ryan Wick et al. 2017

Transcriptome Assembly

Transcriptome

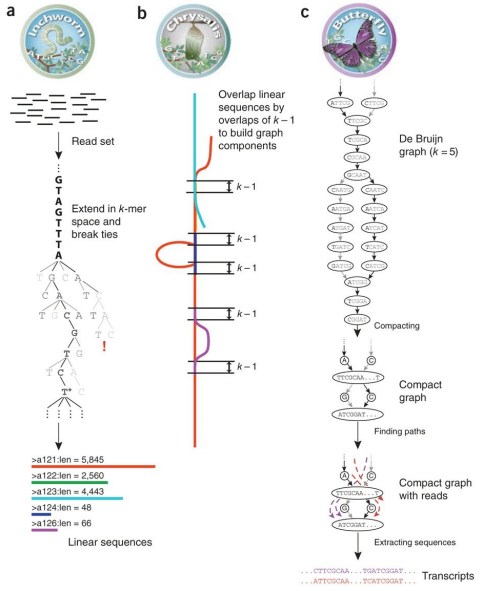
- Set of all mRNA transcripts
- Complex in Eukaryotes due to splice forms

De novo assembly can reconstruct transcriptome

- RNAseq reads are input data
- Output contigs are the set of transcripts

Examples

- Trinity
- Velvet
- SOAPdenovo-Trans
- Trans-ABySS

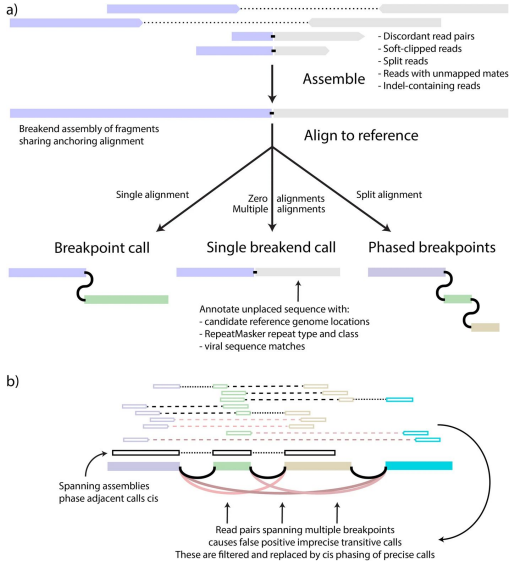


Trinity - de novo transcriptome assembly
Grabherr et al. (2011)

Breakpoint Assembly

Assembly even pops up in surprising places!

- GRIDSS2
- Short-read structural variant (SV) caller
- Gathers reads possibly involved with a SV
- Performs assembly to reconstruct variant



GRIDSS2 - Assembly to identify breakpoints
Cameron et al. (2021)

Thank you!

Today: Assembly in Practice

Next time: Dimensionality Reduction