

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

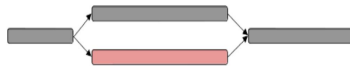
De Bruijn: use of kmers & exact matching

Middle of read: causes **bubbles**

End of read: cause **spurs**

Use of kmers: causes **erroneous edges**

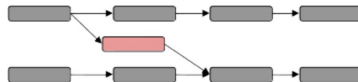
Bubbles



Spurs
(tips / dead ends)



Erroneous
edges



De Bruijn Graph Simplification

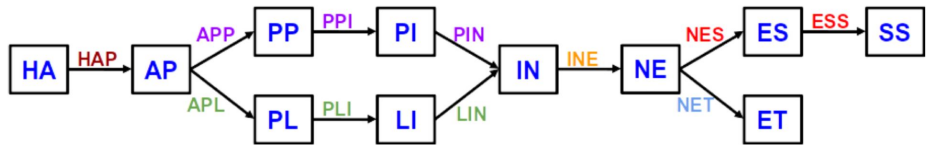
Read Errors

Example

Unique 3-mers ($k = 3$):

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

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

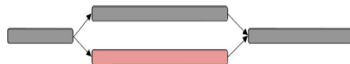


De Bruijn Graph Simplification

Graphs are very complex before simplification.

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

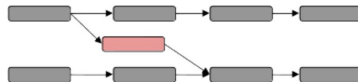
Bubbles



Spurs
(tips / dead ends)



Erroneous
edges



De Bruijn Graph Simplification

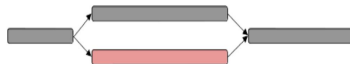
Graphs are very complex before simplification.

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

Naively

- Remove all nodes with low coverage.
- Eg. Expected = 30x
- Remove nodes $\leq 5x$ coverage

Bubbles

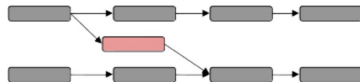


Spurs

(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 $\leq 5x$ coverage

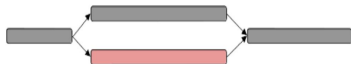
Issues

- 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。

Bubbles

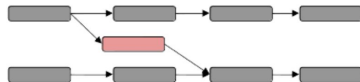


Spurs

(tips / dead ends)



Erroneous edges

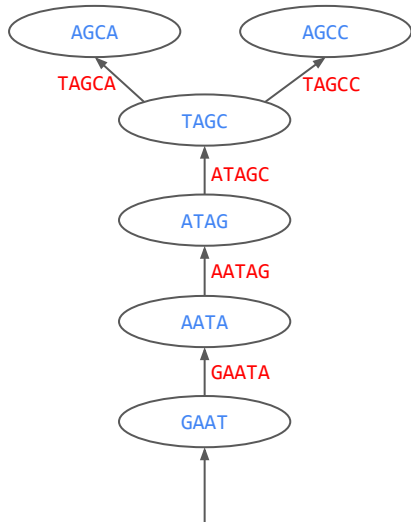


De Bruijn Graph Simplification

Step 1: Coalesce non-branching paths

How we built the De Bruijn Graph:

- Nodes: prefix / suffix
- Edges: kmers



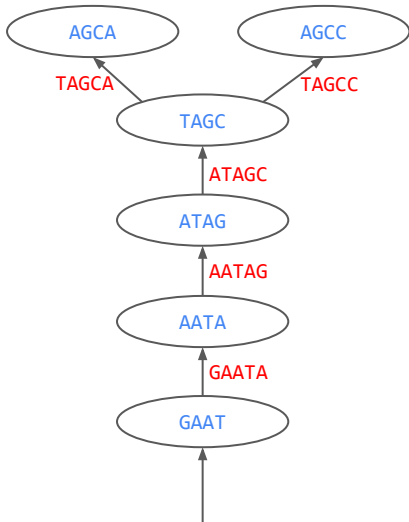
De Bruijn Graph Simplification

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

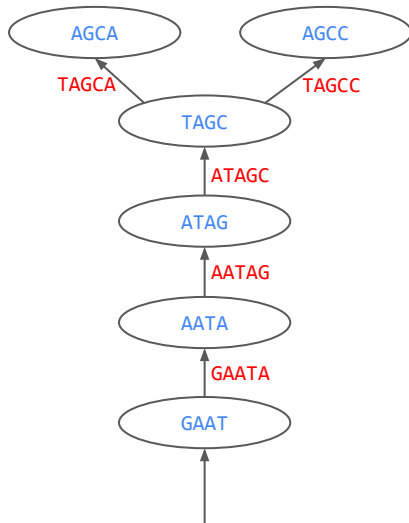
How we built the De Bruijn Graph:

- 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

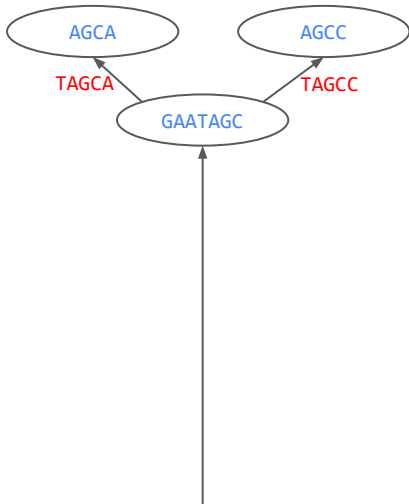
How we built the De Bruijn Graph:

- Nodes: prefix / suffix
- Edges: kmers

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

First task: collapse linear chains into single node.

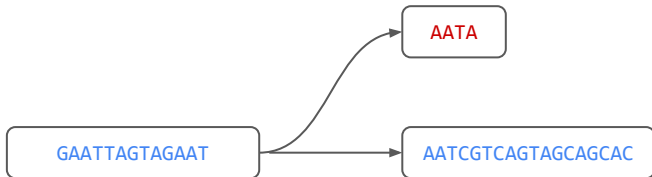
- Improves space performance (less nodes / edges)
- Improves time performance (traverse less edges)



De Bruijn Graph Simplification

Step 1: Remove Spurs (tips)

Spur: chain of nodes that is disconnected on one end



De Bruijn Graph Simplification

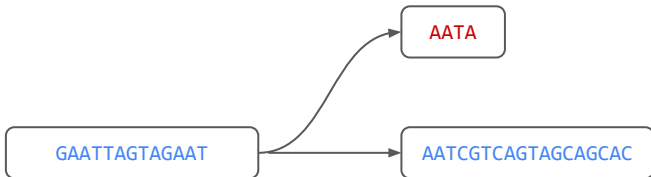
Step 1: Remove Spurs (tips)

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

Step 1: Remove Spurs (tips)

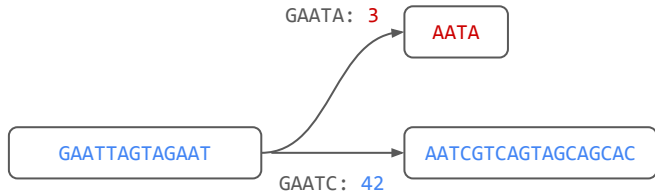
不能过于
粗暴 (不
能heavy-
handed)

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.



筛选tip的时候
看length和mc

Identify tips via length and minority count

Length: Path length along tip $< 2k$.

Minority count:

- 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

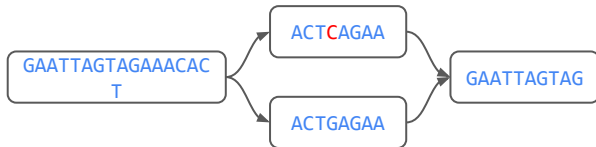
Conditions for merging (popping bubbles)

De Bruijn Graph Simplification

Step 2: Pop bubbles

Conditions for merging (popping bubbles)

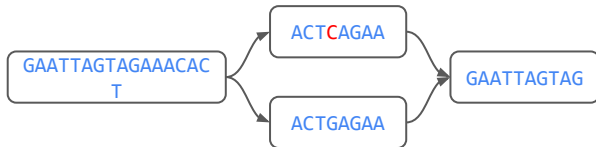
Scenario 1: sequencing error / heterozygosity



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

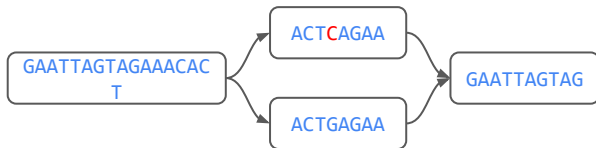
De Bruijn Graph Simplification

Step 2: Pop bubbles

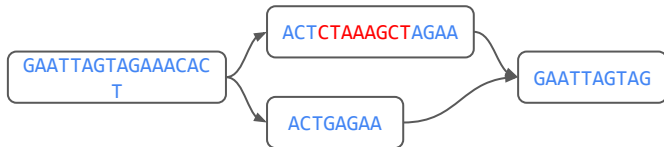
Conditions for merging (popping bubbles)

- Must have same start and end node

Scenario 1: sequencing error / heterozygosity



Scenario 2: both paths valid - unique sequences



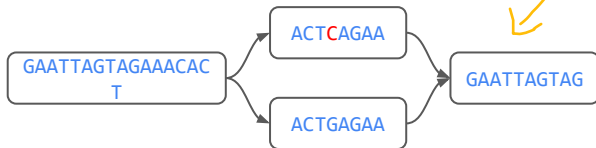
De Bruijn Graph Simplification

Step 2: Pop bubbles

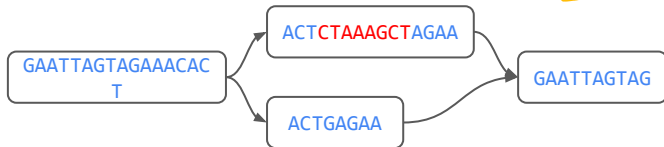
Conditions for merging (popping bubbles)

- 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

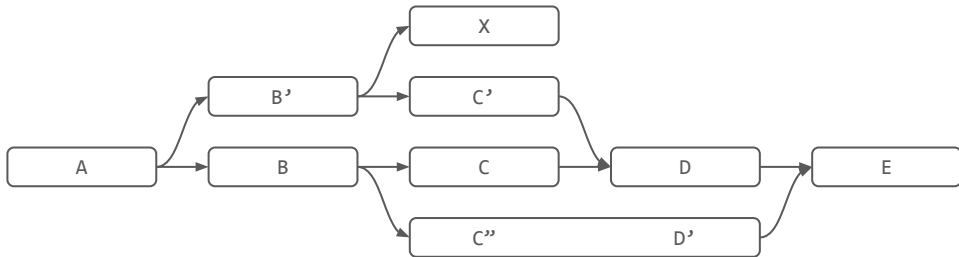
Step 2: Pop bubbles

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

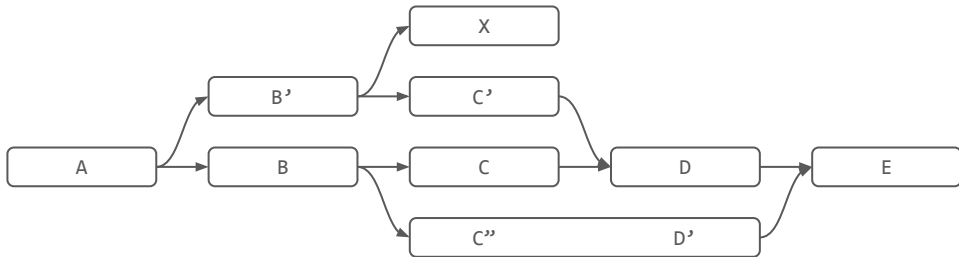
Step 2: Pop bubbles

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

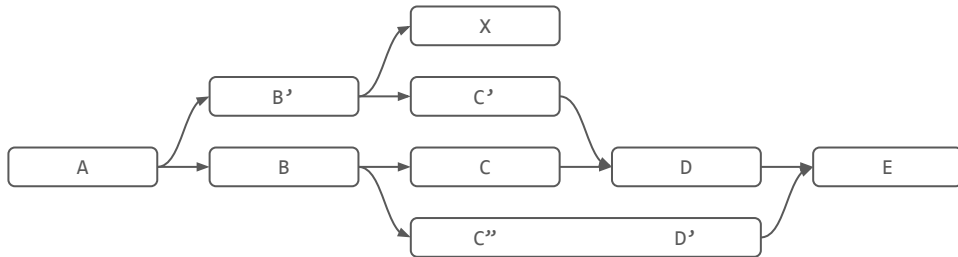


De Bruijn Graph Simplification

Step 2: Pop bubbles

Scenario 3: complex

(in this figure - nodes labelled, not showing seq)



Conditions for merging (popping bubbles)

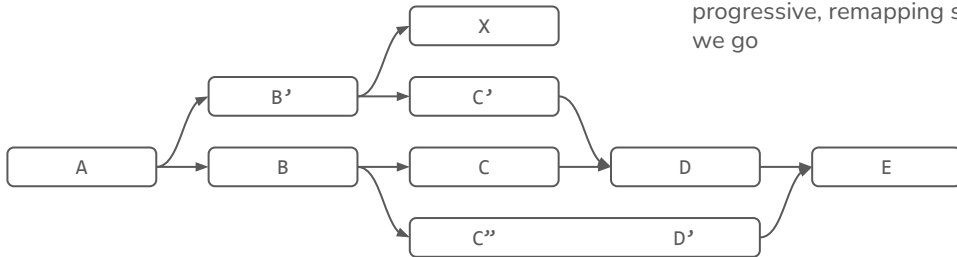
- Must have same start and end node
- Must have similar sequence
- Only merge two paths at a time
- Cannot just delete nodes because this separates part of graph

De Bruijn Graph Simplification

Step 2: Pop bubbles

Scenario 3: complex

(in this figure - nodes labelled, not showing seq)

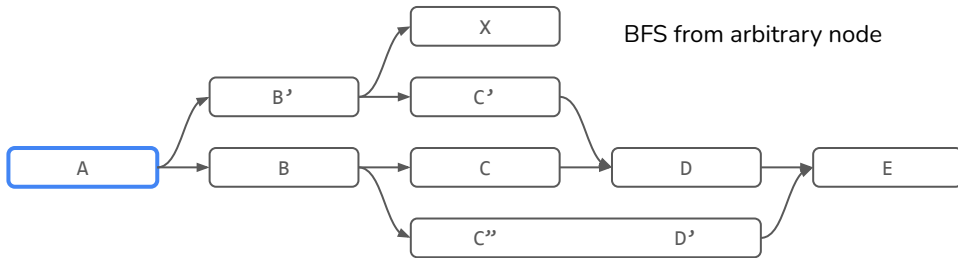


Conditions for merging (popping bubbles)

- 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

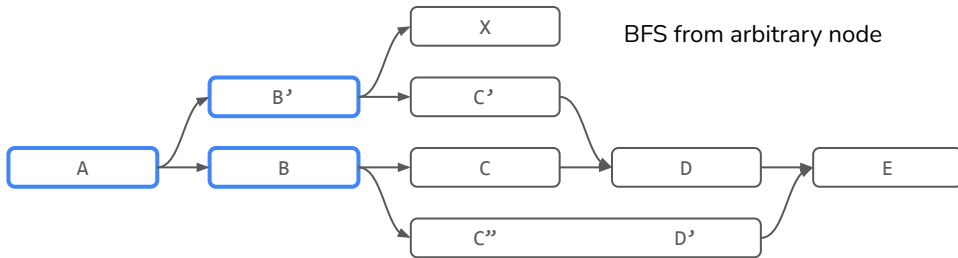
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



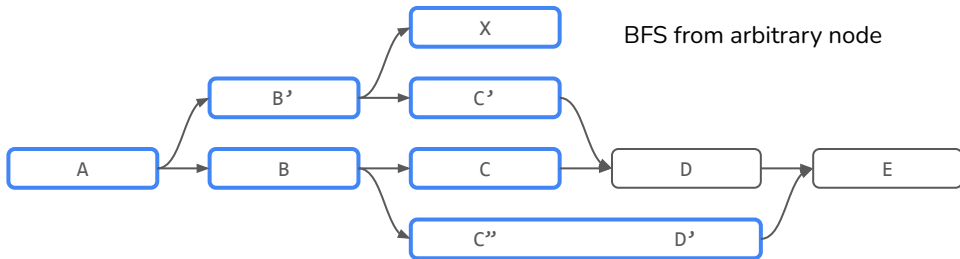
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



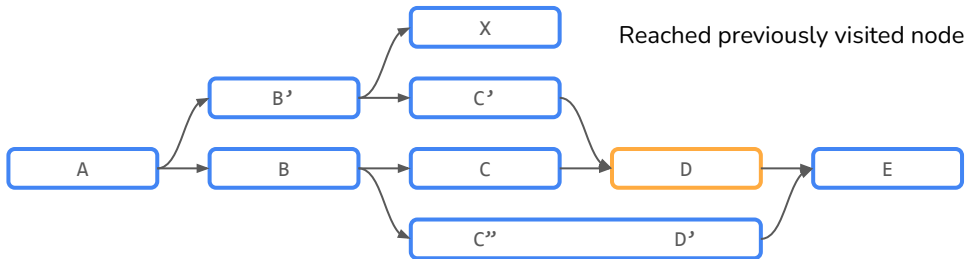
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



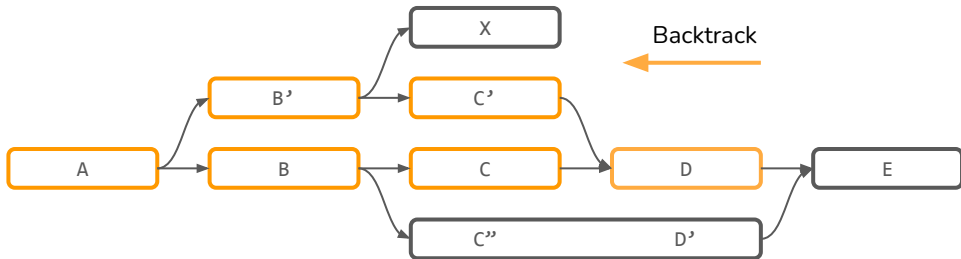
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



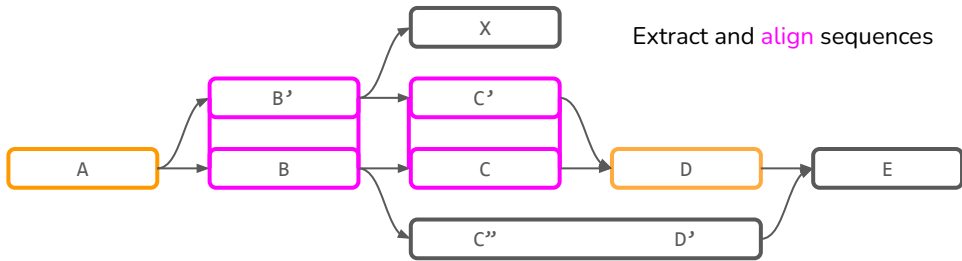
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



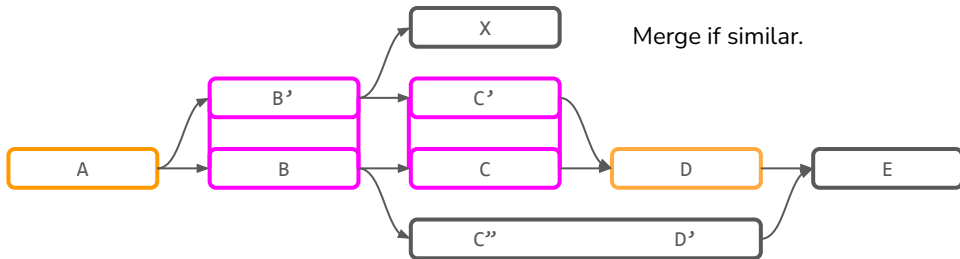
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



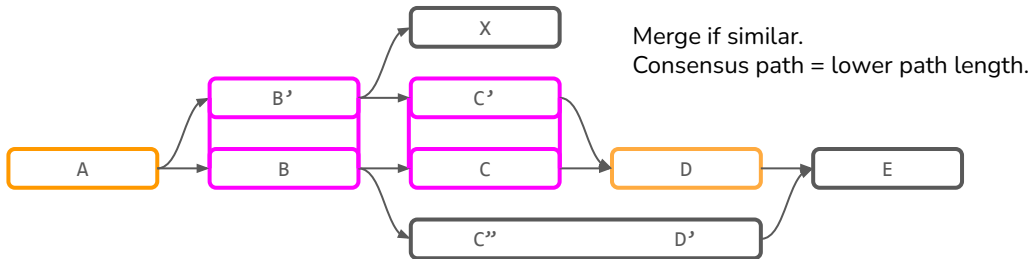
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



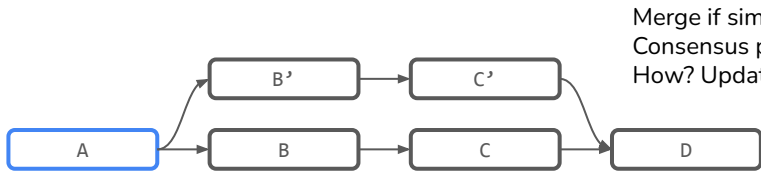
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



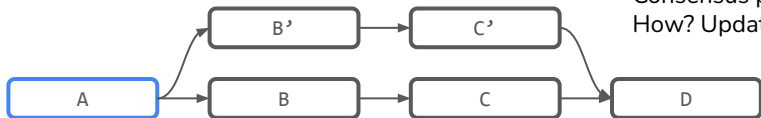
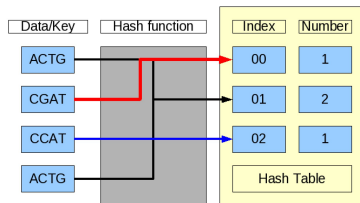
Merge if similar.

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.

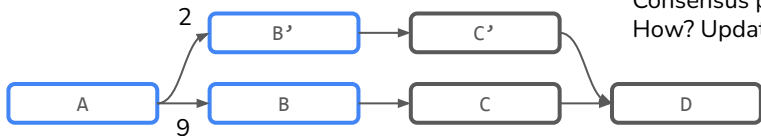
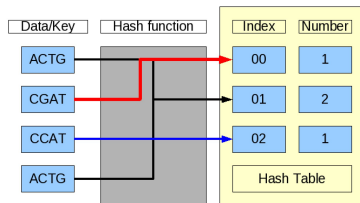
How? Update path lengths during BFS traversal.

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.

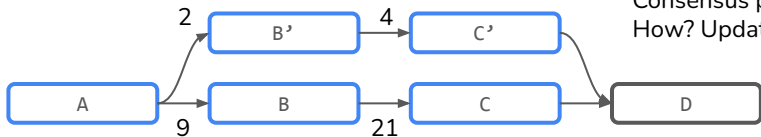
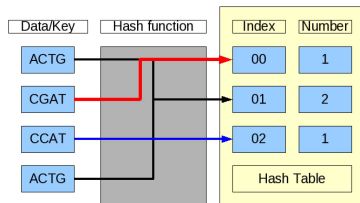
Consensus path = lower path length.

How? Update path lengths during BFS traversal.

AB	9	AB'	2
BC	12	B'C'	2
CD	7	C'D	1

De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



Merge if similar.

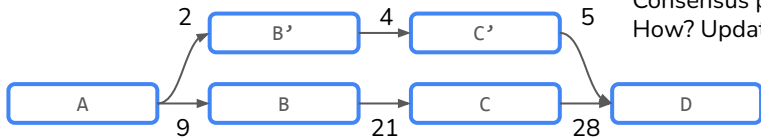
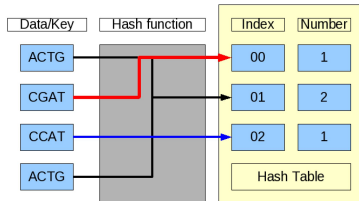
Consensus path = lower path length.

How? Update path lengths during BFS traversal.

AB	9	AB'	2
BC	12	B'C'	2
CD	7	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	AB'	2
BC	12	B'C'	2
CD	7	C'D	1

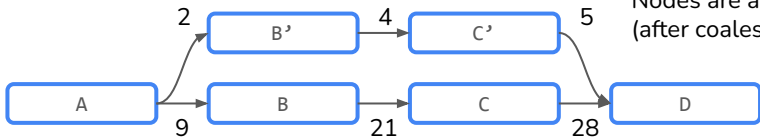
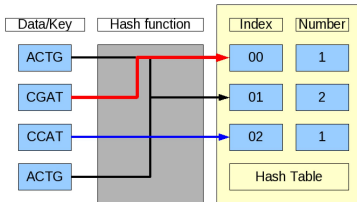
路径权重代表了这条路径上序列的频率或可信度。

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!

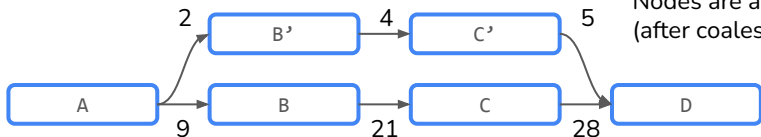
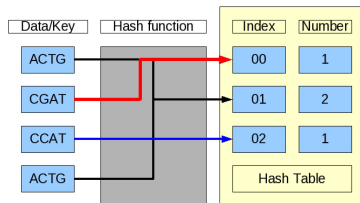
Nodes are arbitrary length
(after coalescing linear chains)

AB	9	AB'	2
BC	12	B'C'	2
CD	7	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)

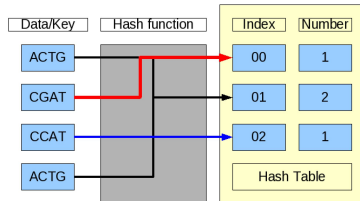
AB	9	AB'	2
BC	12	B'C'	2
CD	7	C'D	1

Multiplicity was updated during coalescing.

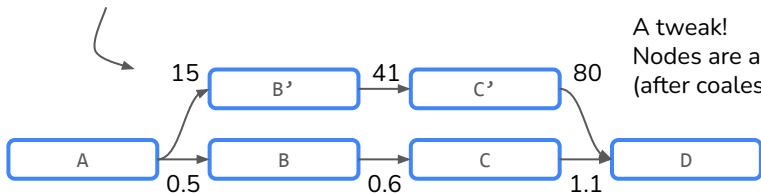
Path length (AB) = $\text{len}(B) / \text{multiplicity}$

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)

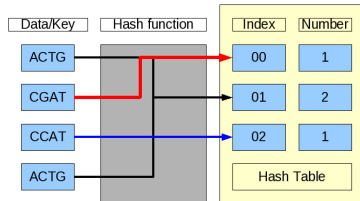
AB	9	AB'	2
BC	12	B'C'	2
CD	7	C'D	1

Multiplicity was updated during coalescing.

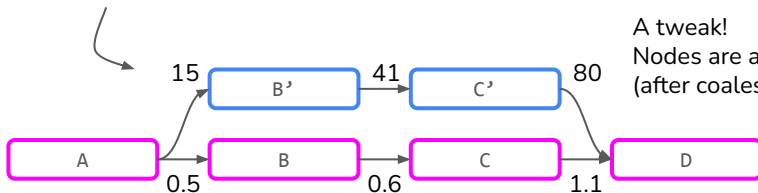
Path length (AB) = $\text{len}(B) / \text{multiplicity}$

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)

Consensus path has shortest path length
Essentially majority voting

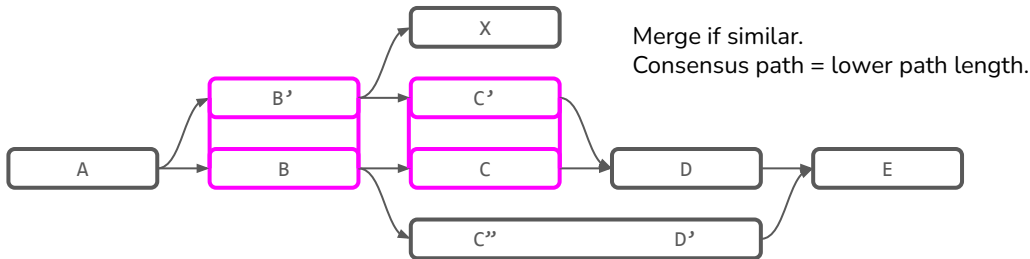
AB	9	AB'	2
BC	12	B'C'	2
CD	7	C'D	1

Multiplicity was updated during coalescing.

Path length (AB) = $\text{len}(B) / \text{multiplicity}$

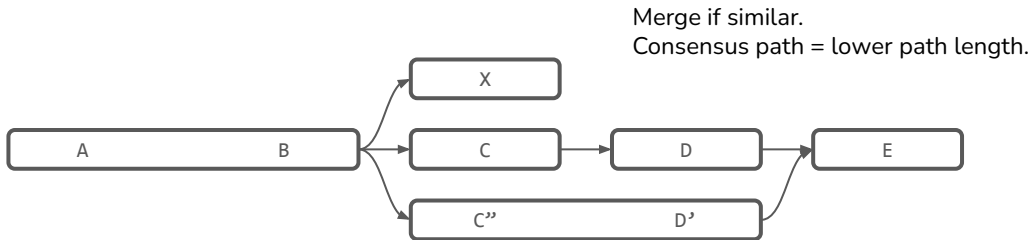
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



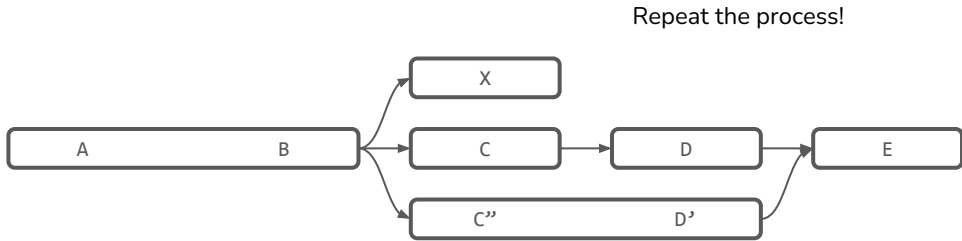
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



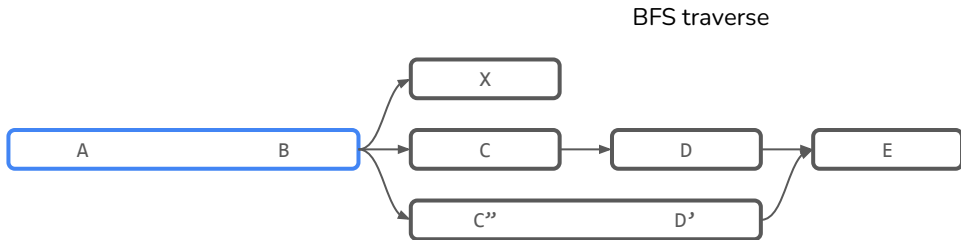
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



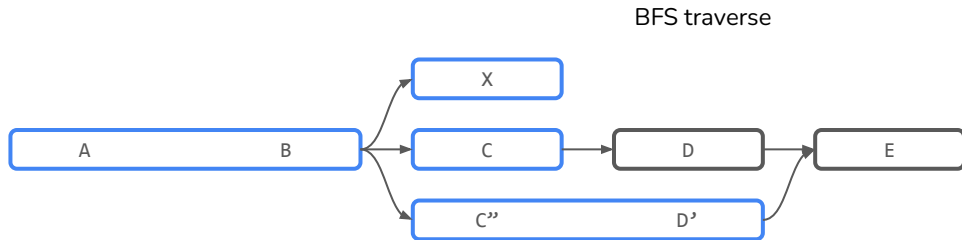
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



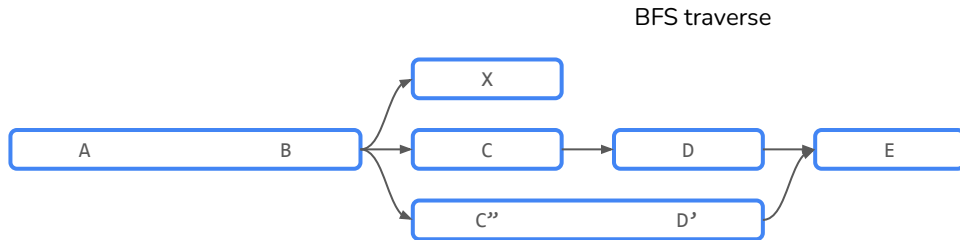
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



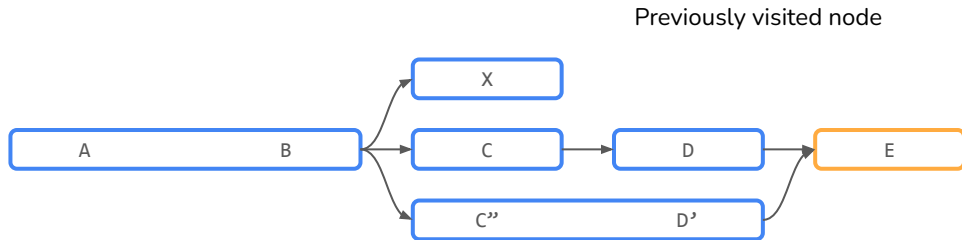
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



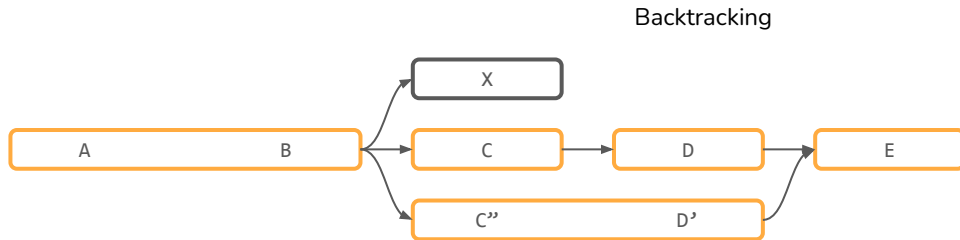
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

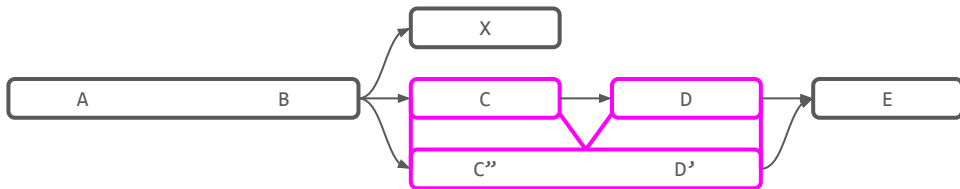
Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

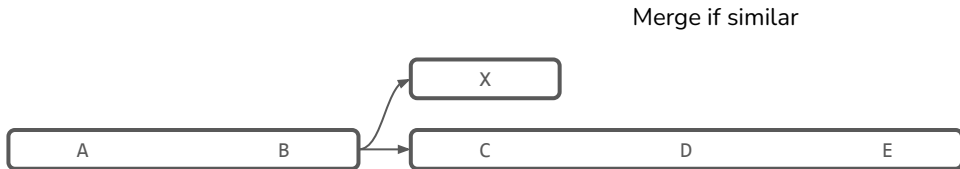
Step 2: Pop bubbles - Tour Bus

Extracting and aligning sequences



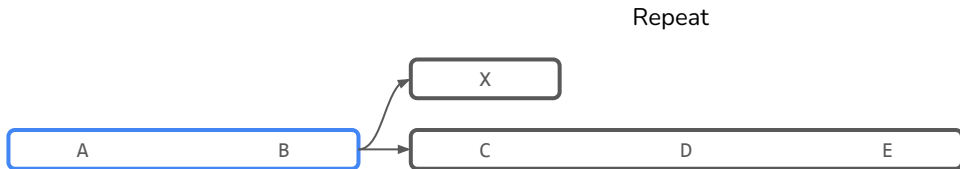
De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

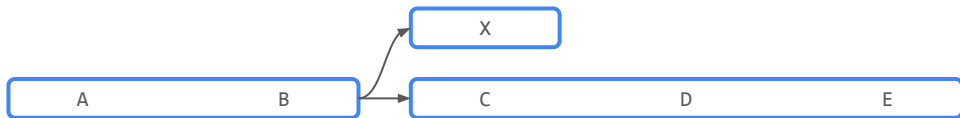
Step 2: Pop bubbles - Tour Bus



De Bruijn Graph Simplification

Step 2: Pop bubbles - Tour Bus

No more moves.



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.

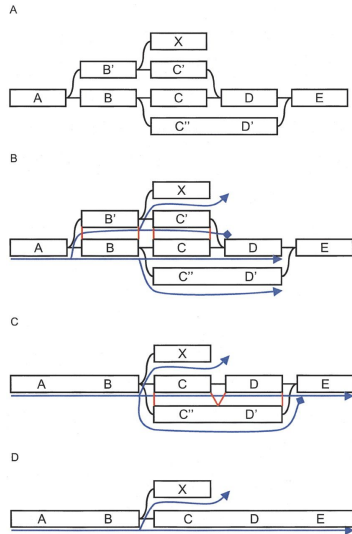
Tour Bus algorithm

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

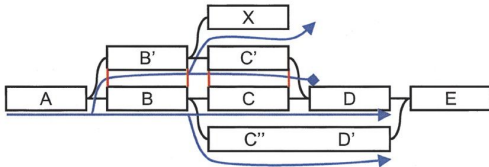
For given node, walk BFS to the right.

Update path lengths as we go.

$A \rightarrow B$: $\text{length}(B) \% \text{multiplicity of edge to } B$

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

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

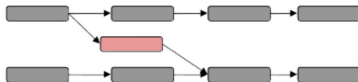
Only solution - coverage cutoff

Remove any node with less than set threshold for coverage.

Repetitive sequence will remain, as it's coverage will be high. Can't do much about this.

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。



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

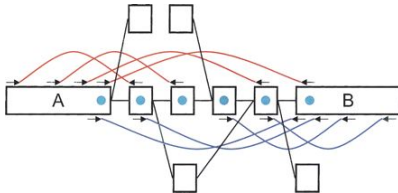
At these branch points we break the assembly into fragments (contigs)

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

基因组测序中覆盖度 (Coverage) 和深度 (Depth) 的概念。

表示至少被一个测序读段 (read) 覆盖的基因组分数。换句话说，它衡量了测序读段覆盖基因组的广度。
例如，如果覆盖度是90%，这意味着基因组中有90%的区域至少被一个测序读段覆盖到。

Coverage (or breadth)

Fraction of the genome
sequenced by at least one read

表示平均每个给定区域被多少个测序读段覆盖。
深度可以告诉我们某个特定区域的测序数据有多少重叠，是对测序重复性的一种衡量。
如果一个区域的深度是10，这意味着这个区域在平均意义上被10个测序读段所覆盖

Depth

Average number of reads that
cover any given region

Intuitively: more reads should increase coverage and depth

直观上，更多的测序读段会增加基因组的覆盖度和深度

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

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

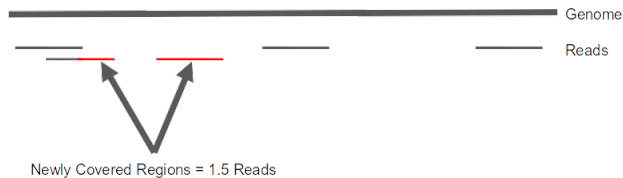
$$\text{Depth} = \frac{3}{8}$$



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

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

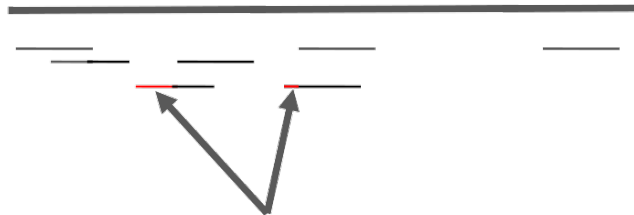
$$\text{Depth} = \frac{5}{8}$$



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

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

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

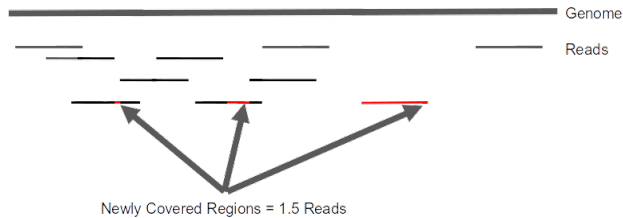


Newly Covered Regions = 0.7 Reads

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

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

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



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

k-mer出现的次数。

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

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

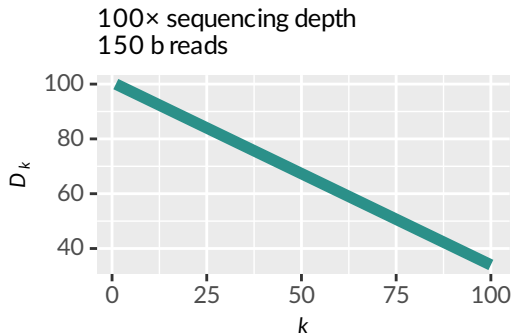
k-mer depth depends on read length:

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

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.
- 🍪 **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

- for k-mer depth,

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

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

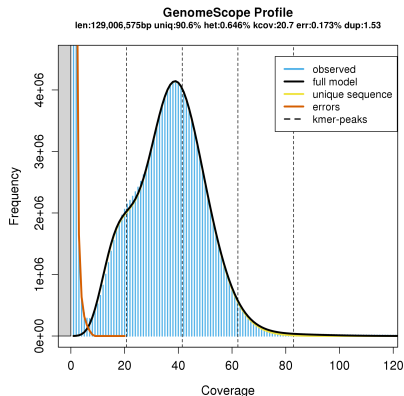
$$D = N \times \frac{L}{G}$$

- 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):

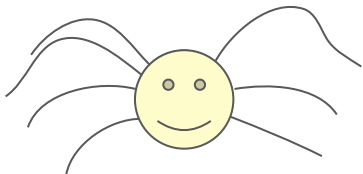
- the graph was made using 31-mers
- the peak is roughly $k = 40$
- we had 100 million 150 b reads



Here, “Coverage” means k-mer depth

Example: Coverage of the Tarantula genome

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.



Actual image from last year was terrifying

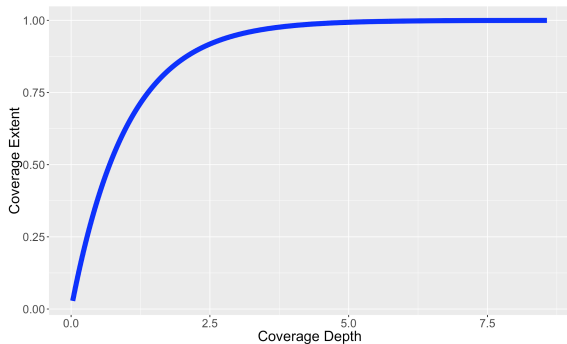
$$\text{Depth} = N \times \frac{L}{G}$$

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

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

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

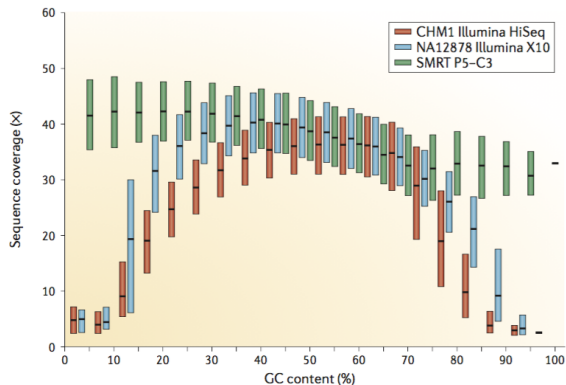
Coverage and depth are related



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

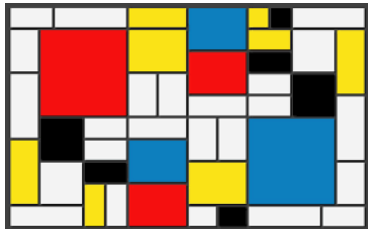
Software

Contiguity
Completeness
Correctness

Contiguity

🍪 Aim:

- Fewer contigs
- Longer contigs

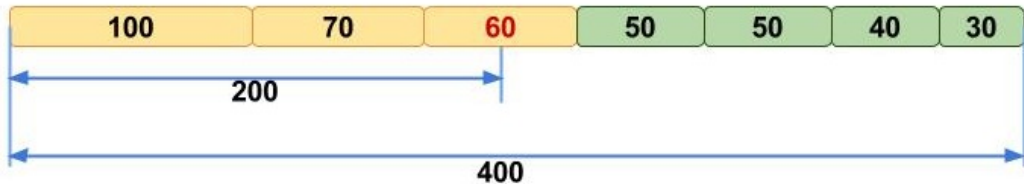


🍪 Metrics

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

contigs ” 是指在基因组组装过程中生成的连续的DNA序列

Contiguity: N_{50} length



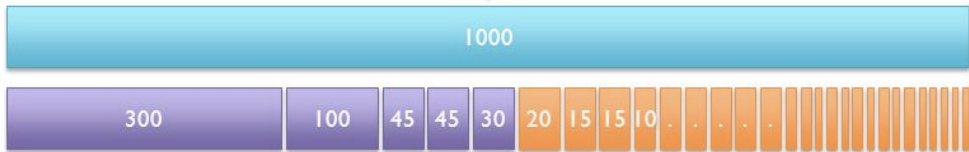
The sequence length of the shortest contig at 50% of the total genome length

N_{50} 长度是基因组组装中一个关键的质量指标。它代表的是长度在中位数以上的contigs（序列片段）的最小长度，也就是说，当你将所有contigs按长度从长到短排序，并且累加长度时， N_{50} 是那个长度，累加到这个长度的所有contigs包含了至少50%的基因组总长度。

Contiguity: N_{50} length

Example: 1 Mbp genome

50%

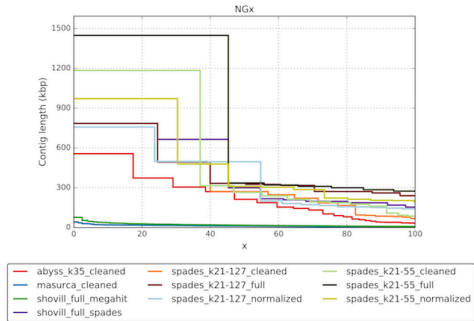


N_{50} size = 30 kbp

$$(300\text{k} + 100\text{k} + 45\text{k} + 45\text{k} + 30\text{k} = 520\text{k} \geq 500\text{kbp})$$

Area under the N_x curve?

基因组组装质量评估的另一个指标—— N_x 曲线下面积 (Area under the N_x curve)，特别是这里提到的是auN值。



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

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

图表显示的是不同基因组组装软件（如ABYSS, SPAdes, MaSuRCA等）输出的 N_x 曲线。 N_x 曲线是一个用于评估基因组组装连续性的图表，其中x轴代表的是组装得到的scaffold（基因组较大片段的组装结果）或contig所覆盖的基因组百分比，y轴代表的是长度至少为x%基因组大小的最短scaffold或contig的长度。

完整性等于
组装基因组的大小
除以
估计的基因组大小

Completeness

完整性也可以通过计算组装基因组中的核心基因数量与数据库中的核心基因数量的比例来评估。

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

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

这张幻灯片讲述了基因组组装的“正确性”（Correctness），指的是组装基因组中没有错误的比例。这里提到的错误包括：

Mis-joins - 错误连接，指的是在组装过程中，本不该连接在一起的序列被错误地拼接起来。

Repeat compressions - 重复压缩，发生在组装算法错误地将重复序列缩短。

Unnecessary duplications - 不必要的重复，指的是在基因组中错误地复制了某些片段。

Indels / SNPs caused by assembler - 组装器引起的插入/缺失（Indels）或单核苷酸多态性（SNPs），即组装过程中产生的错误插入或删除一个或几个核苷酸，或者错误地改变了单个核苷酸。

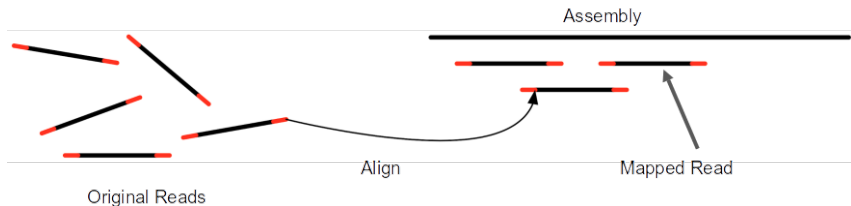
Correctness: check for self consistency

将所有的读段 (reads) 重新对齐到拼接片段 (contigs) 上 - 这一步是为了验证组装拼接的准确性。读段是原始的基因组序列数据，而拼接片段是将这些读段通过计算方法拼接起来的较长的序列片段。

- 🍪 Align all the reads back to the contigs

- 🍪 Look for inconsistencies

寻找不一致之处 - 对齐后，需要检查读段与拼接片段之间的匹配度。任何不匹配或者对齐不当的地方都可能表示组装过程中产生了错误。



幻灯片下方的插图显示了读段与组装后的基因组之间的对齐。黑色线表示组装后的基因组序列 (Assembly)，黑色带红色端点的线段表示原始读段 (Original Reads)，灰色箭头显示了如何将这些读段对齐到组装序列上 (Align)，对齐后的读段称为映射读段 (Mapped Read)。

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

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.

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](#)

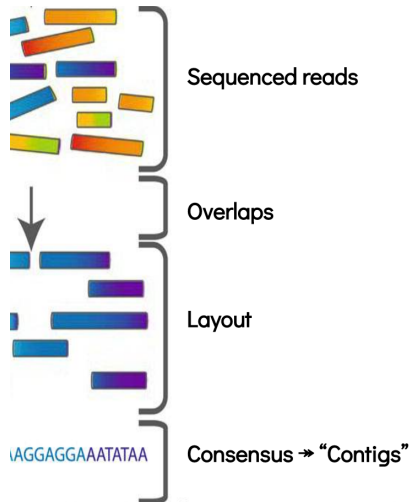
Hybrid

- 🍪 [MaSuRCA](#)
- 🍪 [Unicycler](#)

Special cases

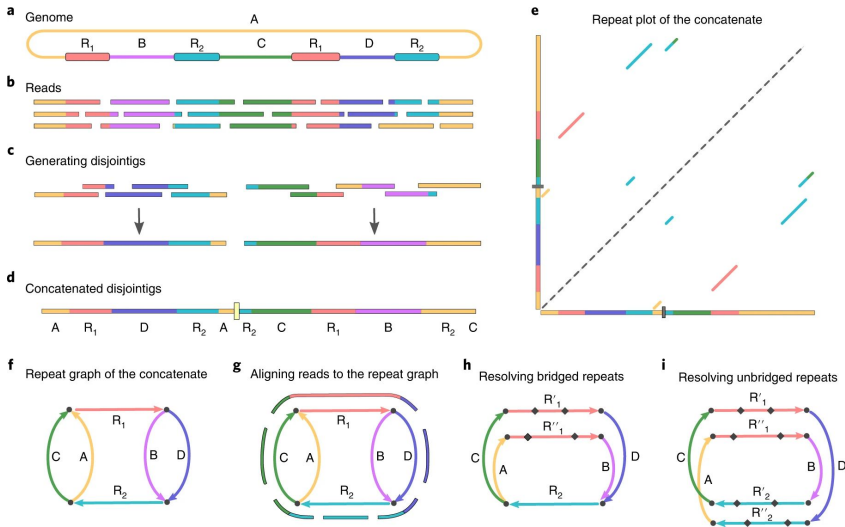
- 🍪 metagenomes, transcriptomes ...

Canu assembler

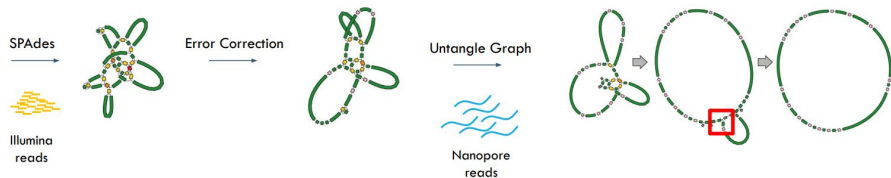


- 🍪 fork of Celera assembler
- 🍪 corrects the reads
 - takes longest reads to make $\approx 20\times$ coverage depth
 - uses left over shorter reads to "correct" long reads
 - trims off the low quality ends
- 🍪 OLC assembly

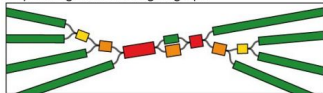
Flye: repeat graph assembler



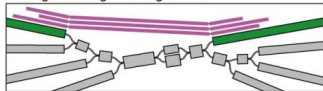
Unicycler



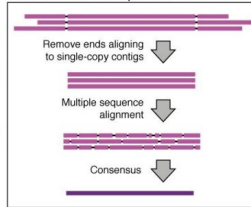
Repeat region in unbridged graph



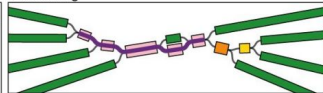
Semi-global long read alignment



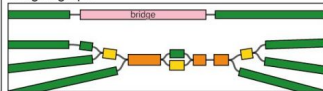
Consensus read sequence



Path finding



Bridged graph



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

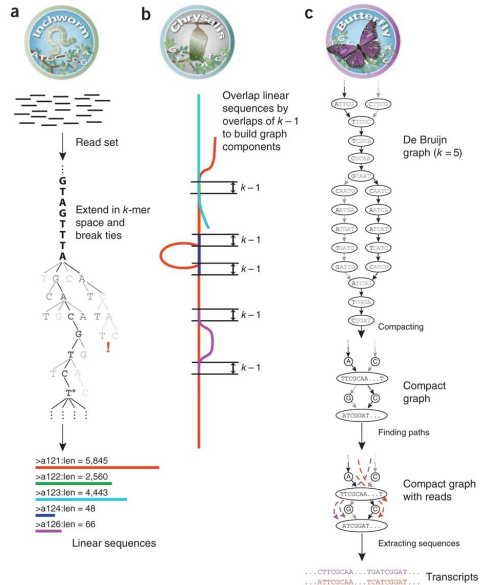
Examples

Trinity

Velvet

SOAPdenovo-Trans

Trans-ABYSS



Trinity - de novo transcriptome assembly

[Grabherr et al. \(2011\)](#)

Breakpoint Assembly

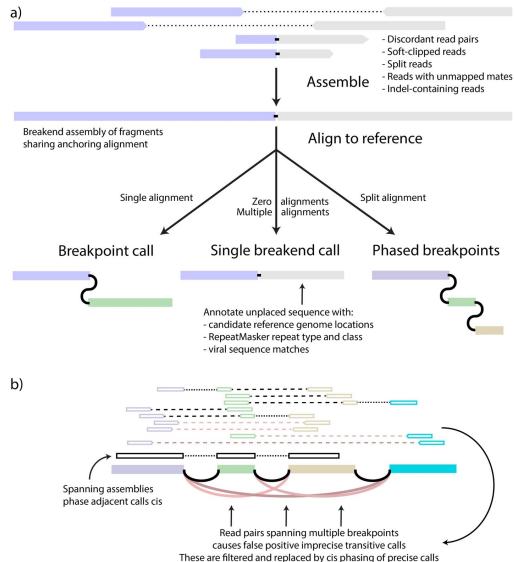
Assembly even pops up in surprising places!

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