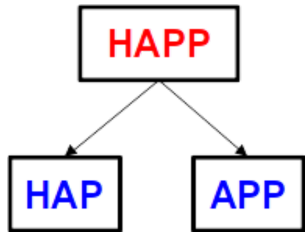
k-mer de Bruijn graph — build the graph

- Take the first k-mer
- **Split it into a $k-1$ prefix and $k-1$ suffix**
- Add both to graph as nodes (if required)
- Add a directed edge to the graph connecting them
- Label this edge as the k-mer



k-mers for $k = 4$:

```
HAPP APPI PINE PPIN INES NESS
^^^^
```

k-mer de Bruijn graph — build the graph

- Take the first k-mer
- Split it into a $k-1$ prefix and $k-1$ suffix
- **Add both to graph as nodes (if required)**
- Add a directed edge to the graph connecting them
- Label this edge as the k-mer

k-mers for $k = 4$:

```
HAPP APPI PINE PPIN INES NESS
^^^^
```

k-mer de Bruijn graph — build the graph

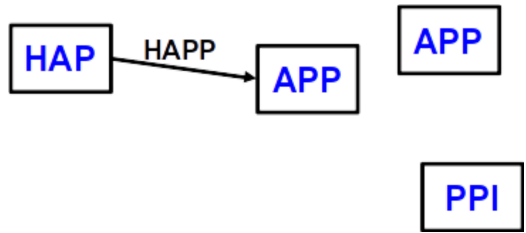
- Take the first k-mer
- Split it into a $k-1$ prefix and $k-1$ suffix
- Add both to graph as nodes (if required)
- **Add a directed edge to the graph connecting them**
- **Label this edge as the k-mer**

k-mers for $k = 4$:

```
HAPP APPI PINE PPIN INES NESS
^^^^
```

k-mer de Bruijn graph — build the graph

- **Take the next k-mer**
- **Split it into a $k-1$ prefix and $k-1$ suffix**
- **Add both to graph as nodes (if required)**
- Add a directed edge to the graph connecting them
- Label this edge as the k-mer

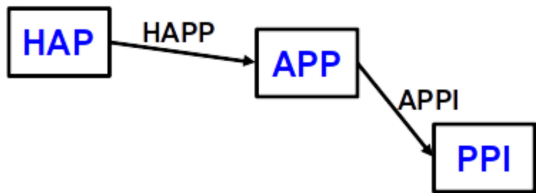


k-mers for $k = 4$:

```
HAPP APPI PINE PPIN INES NESS
     ^^^^
```

k-mer de Bruijn graph — build the graph

- Take the next k-mer
- Split it into a $k-1$ prefix and $k-1$ suffix
- Add both to graph as nodes (if required)
- **Add a directed edge to the graph connecting them**
- **Label this edge as the k-mer**



k-mers for $k = 4$:

```
HAPP APPI PINE PPIN INES NESS
     ^^^^
```

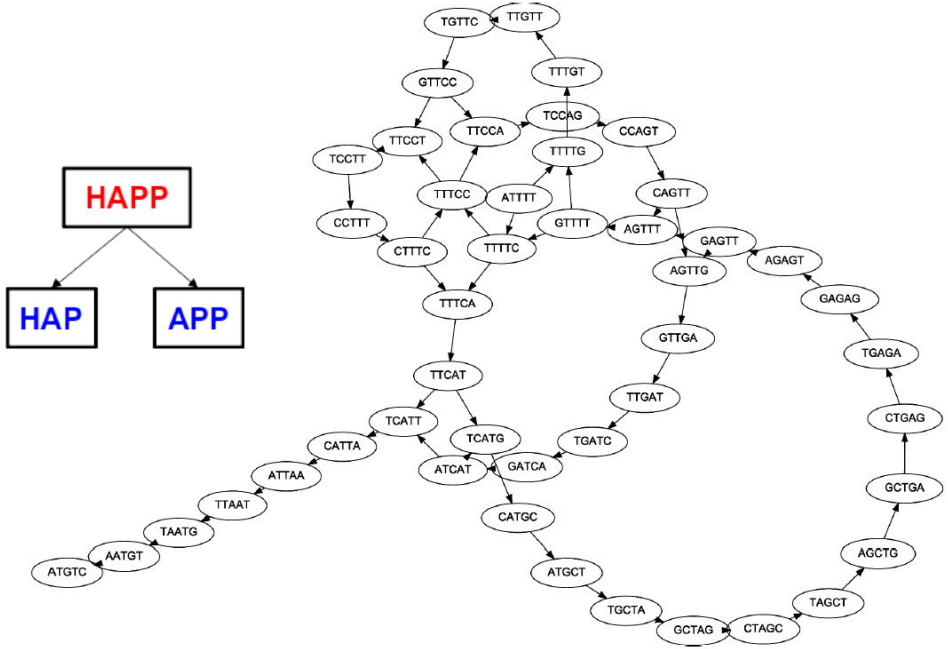
Recap

De Bruijn Graphs

Strategy of choice for short read assembly

1. Break reads into kmers
2. Separate kmers into prefix - suffix
3. Add prefix / suffix and edge to graph

Graph nodes = prefixes / suffixes
Graph edges = kmers



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

Issues

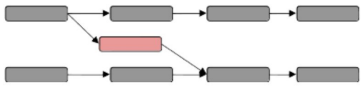
- May remove genuine regions (just had low sequencing depth)
- Doesn't address heterozygosity

Will explore how Velvet handles graph simplification.

Bubbles

Spurs
(tips / dead ends)

Erroneous edges



De Bruijn Graph Simplification

Step 1: Coalesce non-branching paths

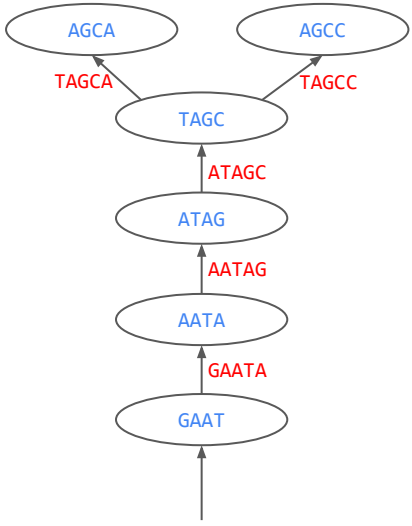
How we built the De Bruijn Graph:

- Nodes: prefix / suffix
- Edges: kmers

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)



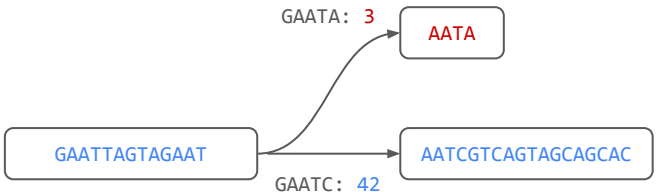
De Bruijn Graph Simplification

Step 1: Remove Spurs (tips)

Spur: chain of nodes that is disconnected on one end

Similar to OLC spur removal. Straightforward.

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



Identify tips via length and minority count

Length: Path length along tip < 2k.

Minority count:

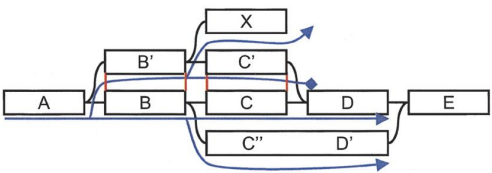
- The branch leading to the tip is an inferior route.
- Edge (kmer) to tip branch has lower occurrences than other branches

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

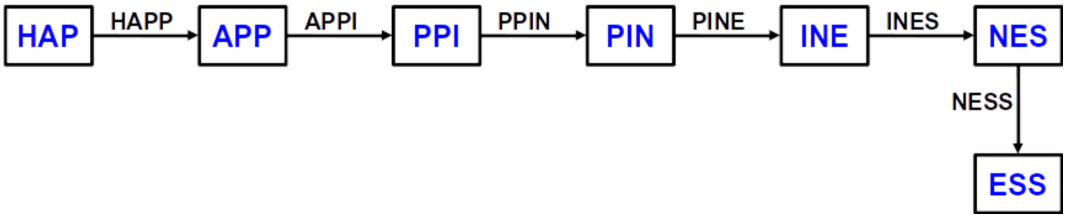
When reach visited node (D):

- 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

k-mer de Bruijn graph — walk the graph



1. Start at an unbalanced node with Degree $\neq 0$
2. Walk through the graph, adding edges
3. Stop when you hit another unbalanced node