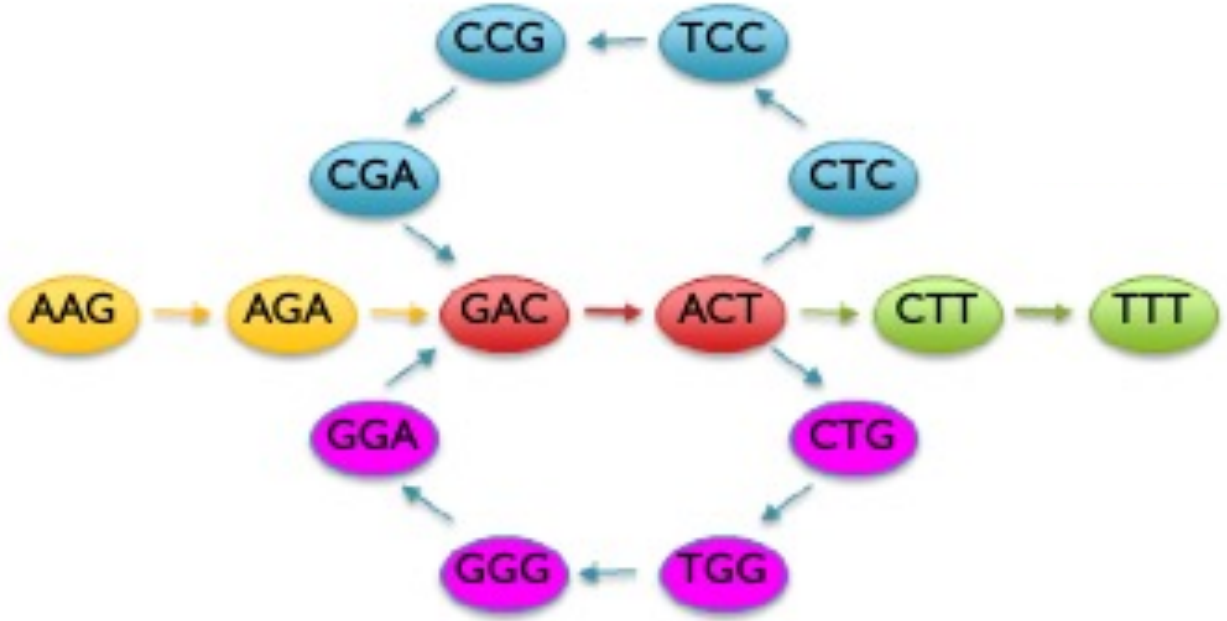
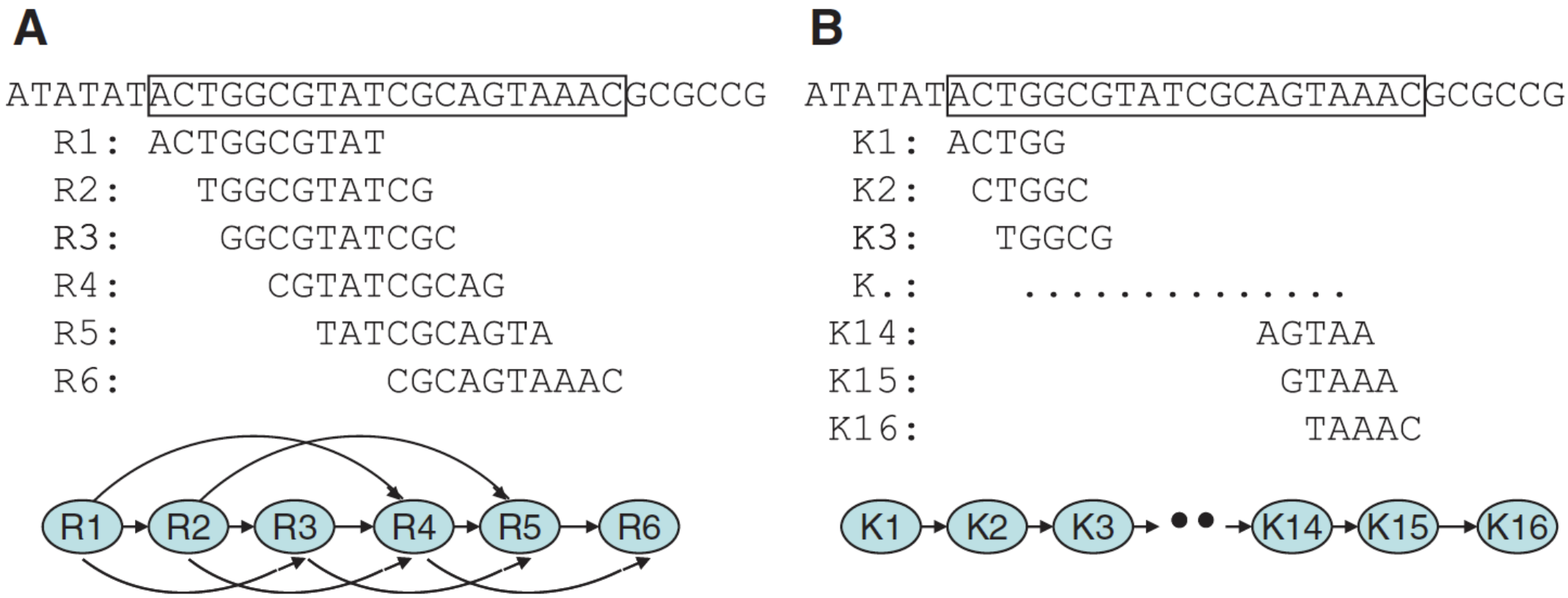
Genome and transcriptome assembly



BIOL435/535: Bioinformatics

March 24, 2022

Two primary assembly methods:
Overlap-Layout-Consensus vs. de Bruijn Graphs



Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*, determined by read length)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

Overlap

```
R1: ACTGGCGTAT
R2: --TGGCGTATCG

Percent overlap 8/10 = 0.8
```

Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*, determined by read length)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

Overlap (*T*=6)

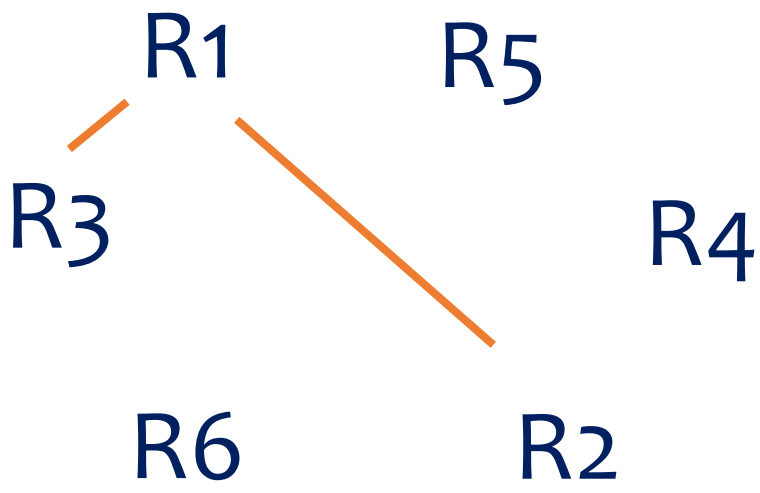
	R1	R2	R3	R4	R5	R6
R1	–					
R2	**8**	–				
R3	**7**	**9**	–			
R4	5	**7**	**8**	–		
R5	3	5	**6**	**8**	–	
R6	0	2	3	5	**8**	–

Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*)
2. Layout reads by degree of overlap (Greater overlap = closer together)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

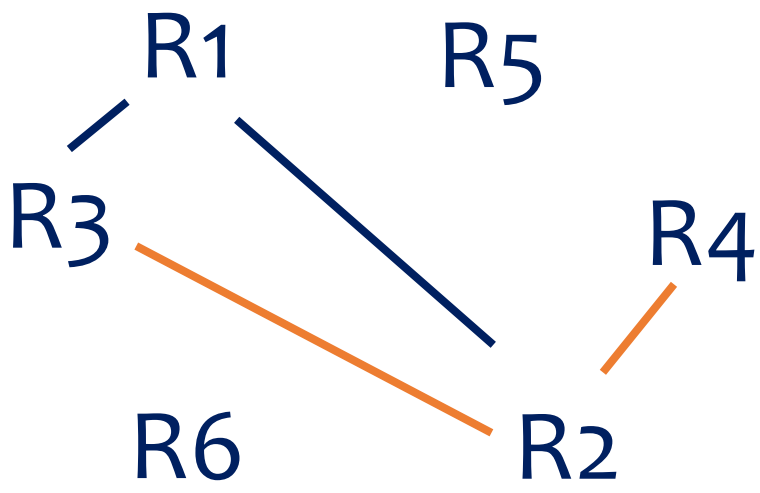


Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*)
2. Layout reads by degree of overlap (Greater overlap = closer together)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

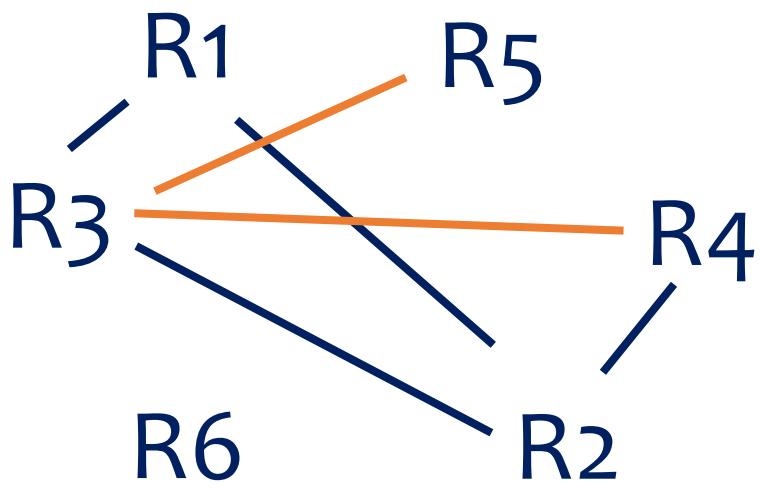


Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*)
2. Layout reads by degree of overlap (Greater overlap = closer together)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

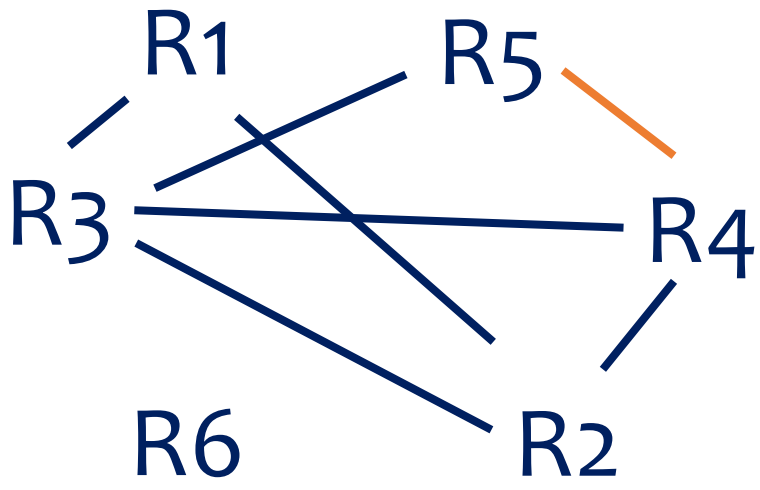


Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*)
2. Layout reads by degree of overlap (Greater overlap = closer together)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

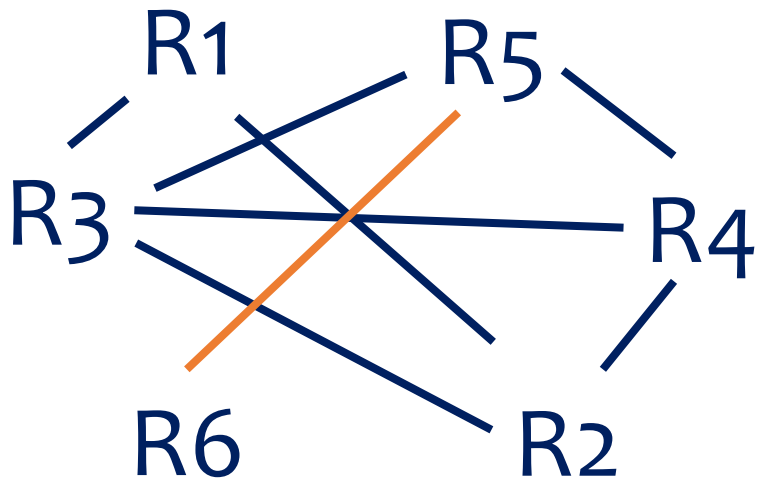


Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*)
2. Layout reads by degree of overlap (Greater overlap = closer together)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

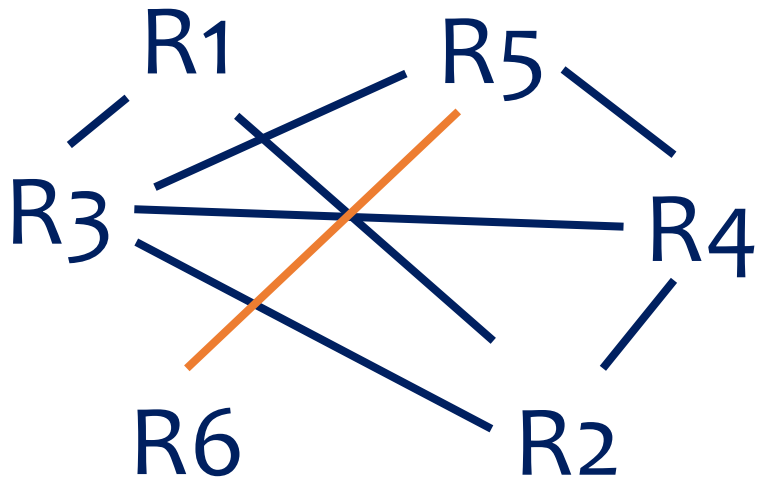


Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*)
2. Layout reads by degree of overlap (Greater overlap = closer together)
3. Collapse aligned reads base-by-base using majority rules (consensus) into a **contig**

Reads

```
R1: ACTGGCGTAT
R2: --TGGCGTATCG
R3: ---GGCGTATCGC
R4: -----CGTATCGCAG
R5: -------TATCGCAGTA
R6: ----------CGCAGTAAAC
```



Overlap-Layout-Consensus

1. Identify reads with significant overlap (i.e., number of bases overlapping exceeds some threshold *T*)
2. Layout reads by degree of overlap (Greater overlap = closer together)
3. Collapse aligned reads base-by-base using majority rules (consensus) into a **contig**

```
R1: ACTGGCGTAT
R2: --TGGCGTATCG
R3: ---GGCGTATCGC
R4: -----CGTATCGCAG
R5: -------TATCGCAGTA
R6: ----------CGCAGTAAAC
C:  ACTGGCGTATCGCAGTAAAC
```

de Bruijn Graph

1. Identify all **unique** subsequences of length *k* (i.e., a k-mer)

Reads

```
R1: ACTGGCGTAT
R2: TGGCGTATCG
R3: GGCGTATCGC
R4: CGTATCGCAG
R5: TATCGCAGTA
R6: CGCAGTAAAC
```

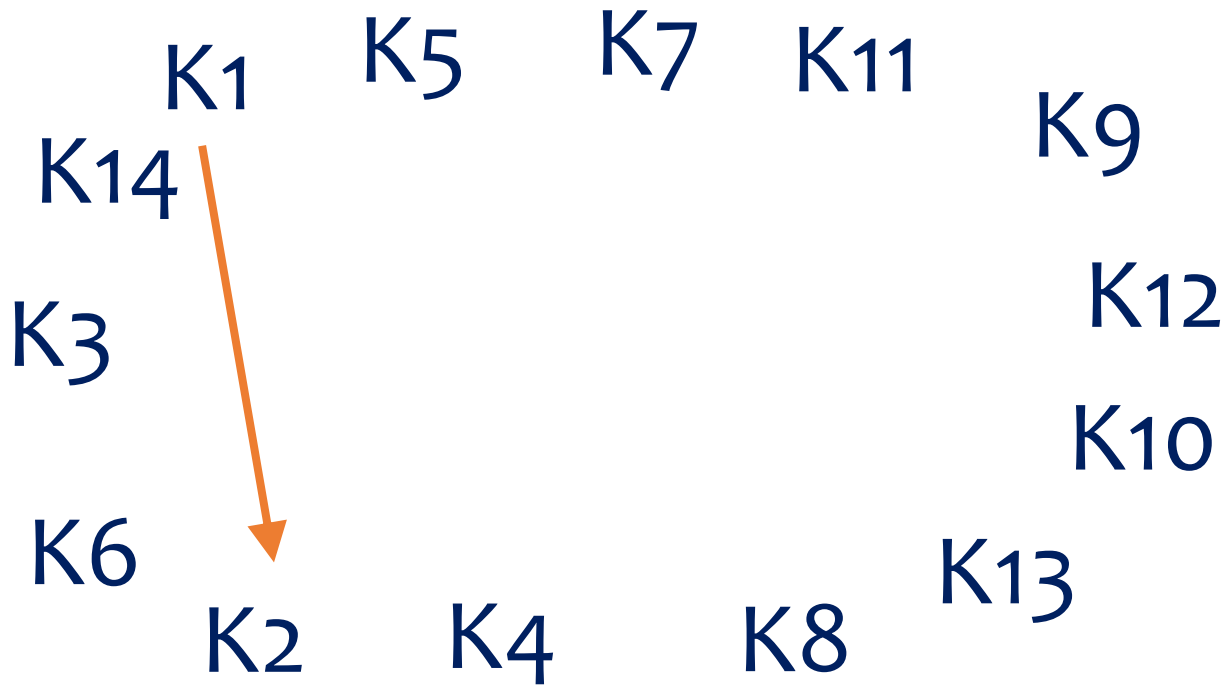
***k* = 7**

K1: ACTGGCG	**K8: TATCGCA**
K2: CTGGCGT	**K9: ATCGCAG**
K3: TGGCGTA	**K10: TCGCAGT**
K4: GGCGTAT	**K11: CGCAGTA**
K5: GCGTATC	**K12: GCAGTAA**
K6: CGTATCG	**K13: CAGTAAA**
K7: GTATCGC	**K14: AGTAAAC**

de Bruijn Graph

1. Identify all unique subsequences of length *k* (i.e., a k-mer)
2. Each k-mer becomes a **node** and k-mers are connected by **edges** to all other k-mers sharing sequence of length k-1

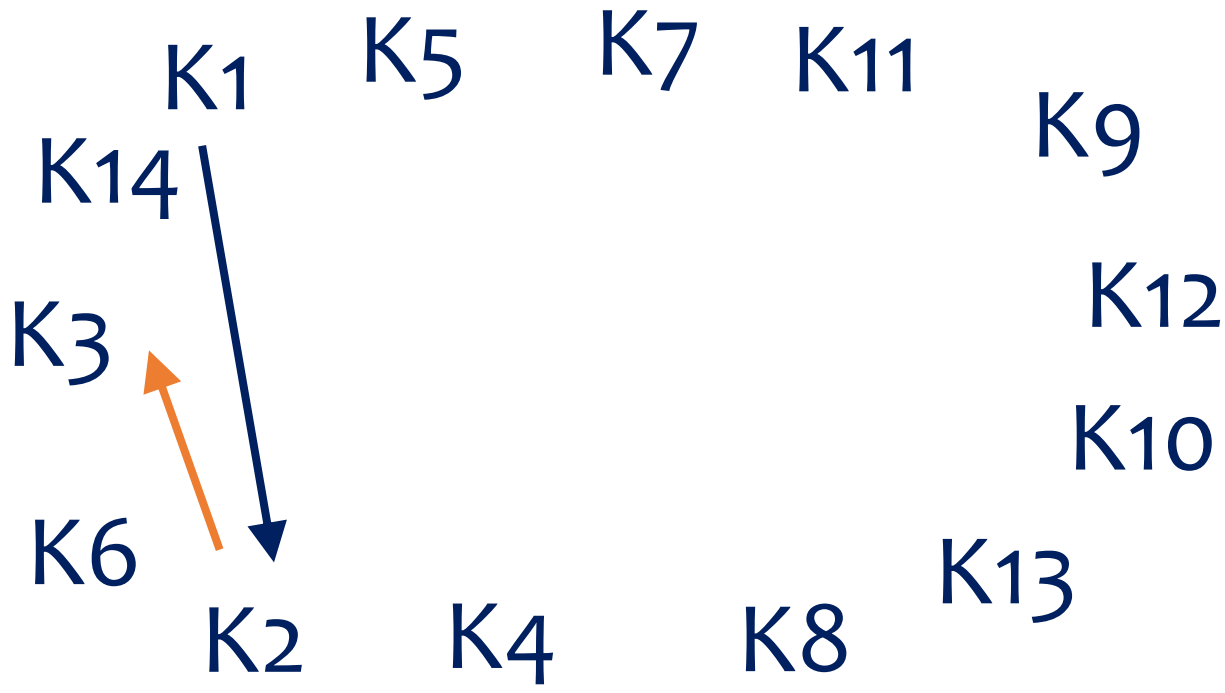
```
K1: ACTGGCG     K8: TATCGCA
K2: CTGGCGT     K9: ATCGCAG
K3: TGGCGTA    K10: TCGCAGT
K4: GGCGTAT    K11: CGCAGTA
K5: GCGTATC    K12: GCAGTAA
K6: CGTATCG    K13: CAGTAAA
K7: GTATCGC    K14: AGTAAAC
```



de Bruijn Graph

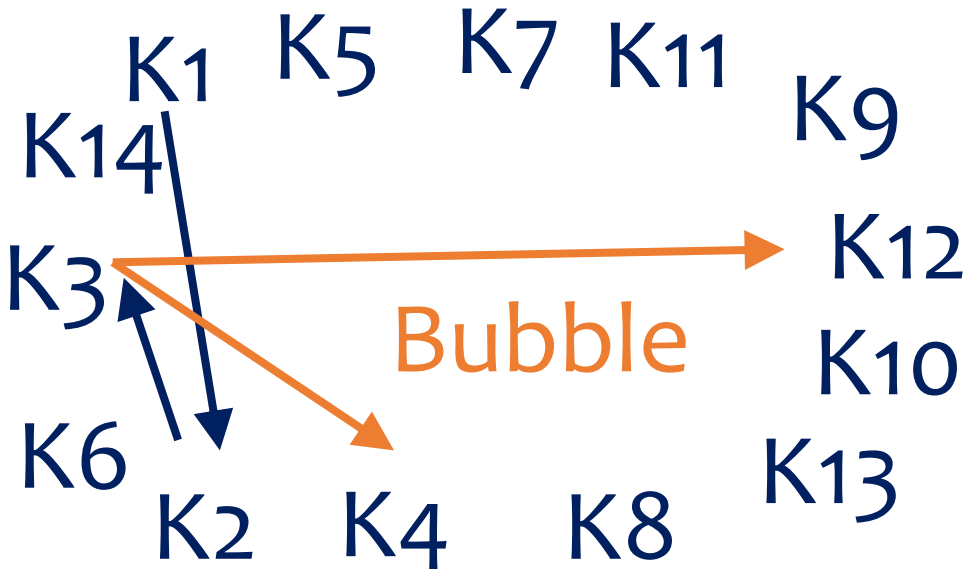
1. Identify all unique subsequences of length *k* (i.e., a k-mer)
2. Each k-mer becomes a **node** and k-mers are connected by **edges** to all other k-mers sharing sequence of length k-1

```
K1: ACTGGCG     K8: TATCGCA
K2: CTGGCGT     K9: ATCGCAG
K3: TGGCGTA    K10: TCGCAGT
K4: GGCGTAT    K11: CGCAGTA
K5: GCGTATC    K12: GCAGTAA
K6: CGTATCG    K13: CAGTAAA
K7: GTATCGC    K14: AGTAAAC
```



de Bruijn Graph

1. Identify all unique subsequences of length *k* (i.e., a k-mer)
2. Each k-mer becomes a node and k-mers are connected by edges to all other k-mers sharing sequence of length k-1
3. Collapse unambiguous paths into contigs, break at **bubbles** (i.e., pathways with multiple pathways)



Overlap-layout-consensus

Pros:

- Simple, intuitive
- Easy traversal across het sites, repeats
- Flexible based on datatype

Cons:

- Computationally expensive
- Does not scale well with short reads

de Bruijn Graph

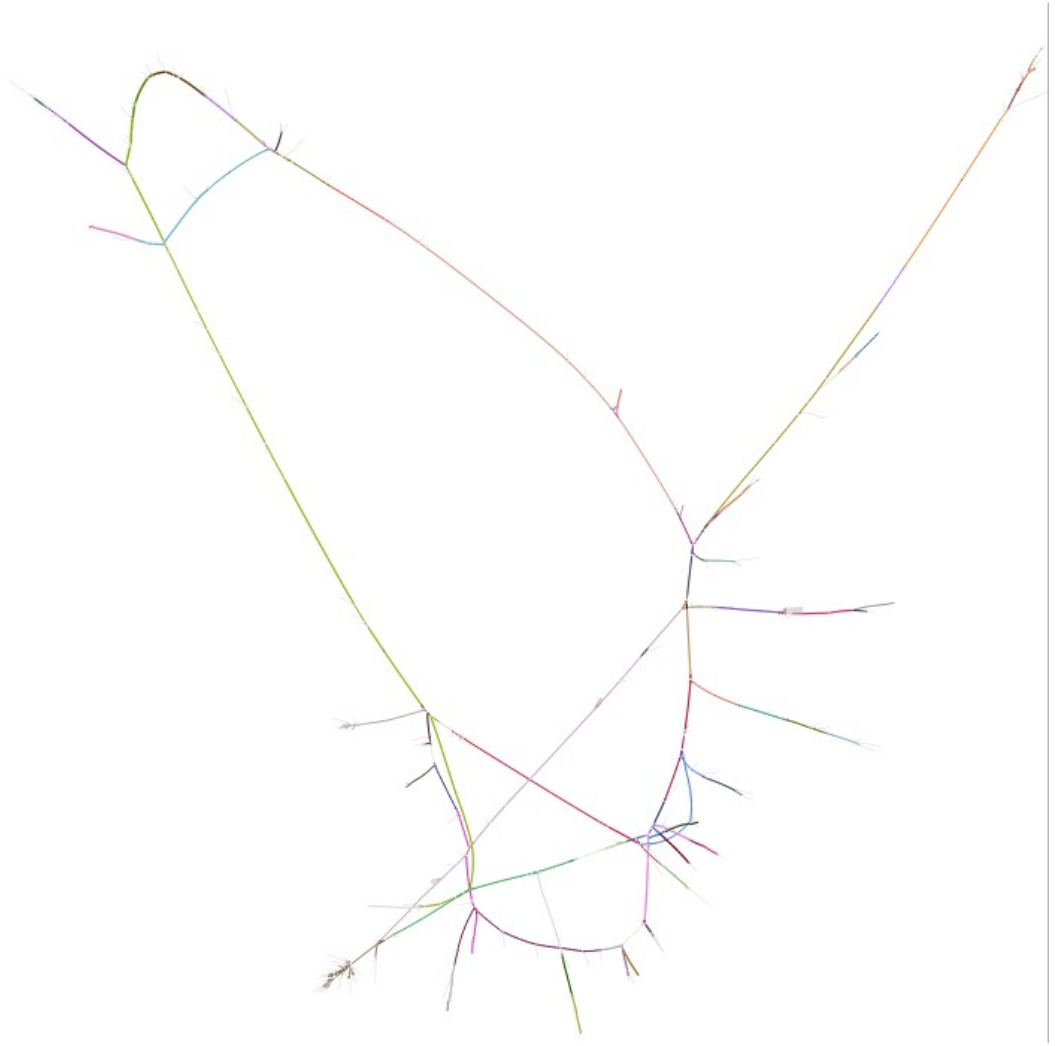
Pros:

- Fast, useful for short reads
- Can assemble a whole genome

Cons:

- Bubbles, het sites
- Dealing with sequencing errors a challenge with long reads

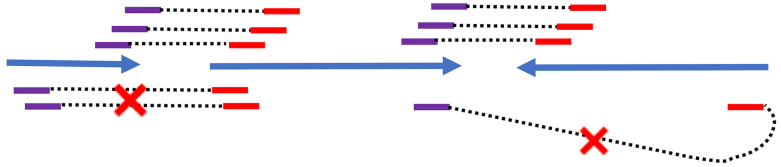
Visualizing assemblies w/ Bandage



Scaffolding genomic contigs

- Map reads to contigs (e.g., using BWA)
- Paired reads that connect contigs?
- Mate Pairs, **chromatin-level barcoding**, long reads

Alignment of reads to contigs



Orienting and Ordering contigs based on alignments

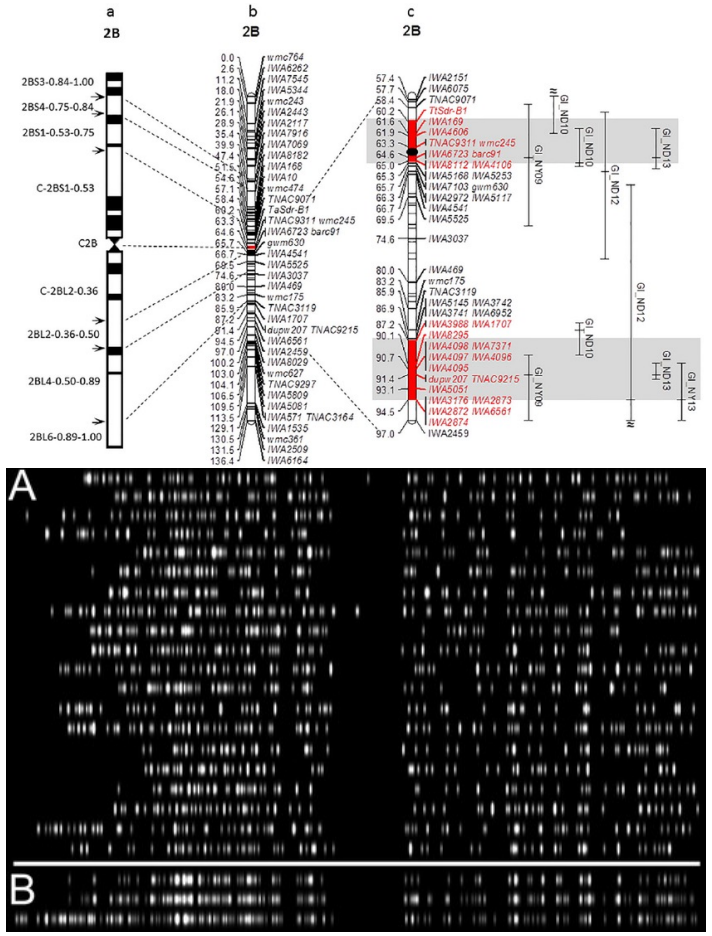
10x GENOMICS®

Dovetail GENOMICS

Chromosome-scale assemblies

Generally will require at least one of:

- **linkage map**
  - Need F2 recombinants
- **Long reads**
  - Pacbio/Nanopore
- **optical mapping** (e.g., BioNano)
  - Expensive



Useful assemblers

SPAdes

MaSuRCA

SOAPdenovo

Velvet

Transcriptome assembly – Trinity

- No scaffolding post contig assembly
- In transcriptome, bubbles represent different transcripts
- How to distinguish between alleles vs. isoforms vs. paralogs?

Contig Assembly

Contig Clustering

Allele/ isoform/ paralog disambiguation

Additional reading:

Li et al., 2011 (OLC vs. de Bruijn graphs)

Haas et al., 2013 (Trinity Assembler)