
DTU

DTU Health Technology
Bioinformatics

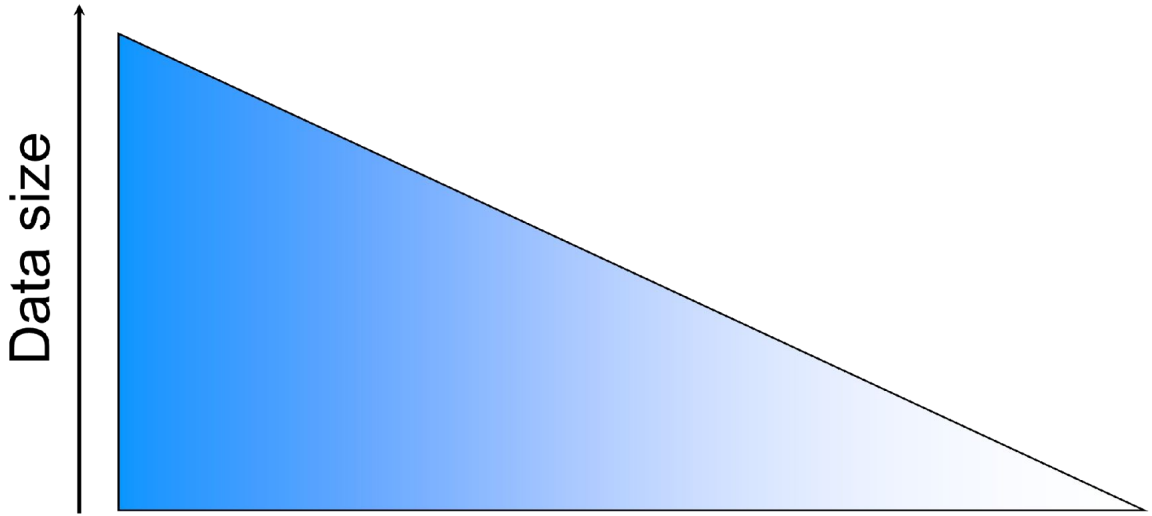
de novo assembly

Gabriel Renaud
Associate Professor
Section of Bioinformatics
Technical University of Denmark
gisves@dtu.dk

Menu

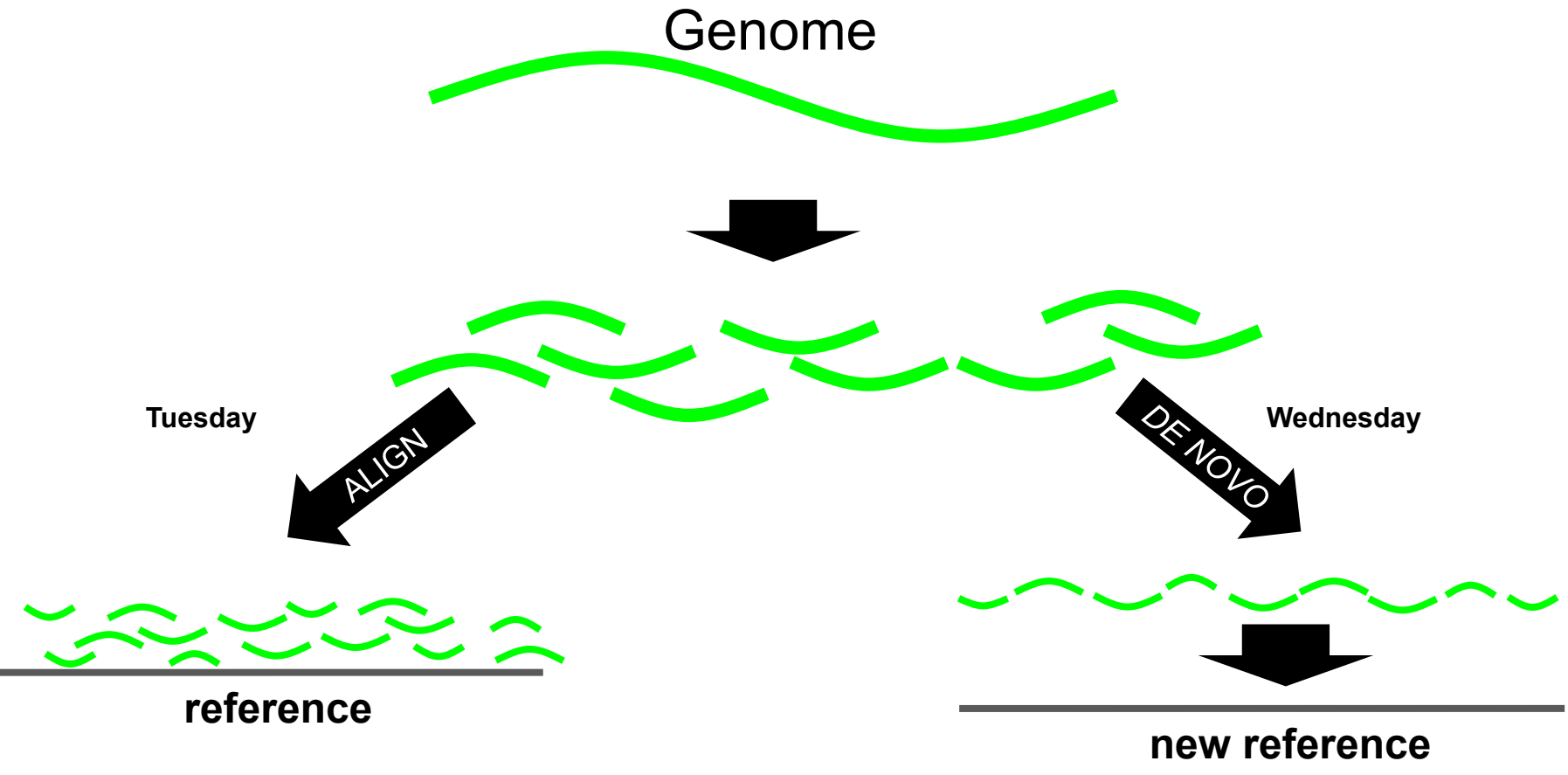
- Assembly approaches
- Assembly graphs
- Graph postprocessing filtering
- The woes of repetition
- Benchmarking your assembly

Generalized NGS analysis



Question | Raw reads | Pre-processing | Assembly: Alignment / *de novo* | Application specific: Variant calling, count matrix, ... | Compare samples / methods | Answer?

Whole genome sequencing



Input

Output

Input

```
@MISEQ423_0:+:7218:7278:60-2
GTTACTCGGACTACCCCGATGCATACACCACATGAAACA
T
+
]V]P]]\]]]]]]\]]]]]]]][]]]\]][]]]]]]]\
]
@MISEQ423_0:-:15245:15305:60-2
AGGGCAAGATGAAGTGAAAGGTAAAGAATCGTGTGAGGG
T
+
]]]][Z]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
]
@MISEQ423_0:-:242:302:60-2
TTTGGTGGAAATTTTTTGTTATGATGTCTGTGTGGAAAG
T
+
]]]]]]]]Z]]]]]]]]]]]]Z]]]\]]]]Z]]]]]]]]
]
@MISEQ423_0:-:1729:1789:60-2
TGCGGTACTATATCTATTGCGCCAGGTTTCAATTTCTAT
C
+
]]]]]]]]X]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
```

Output

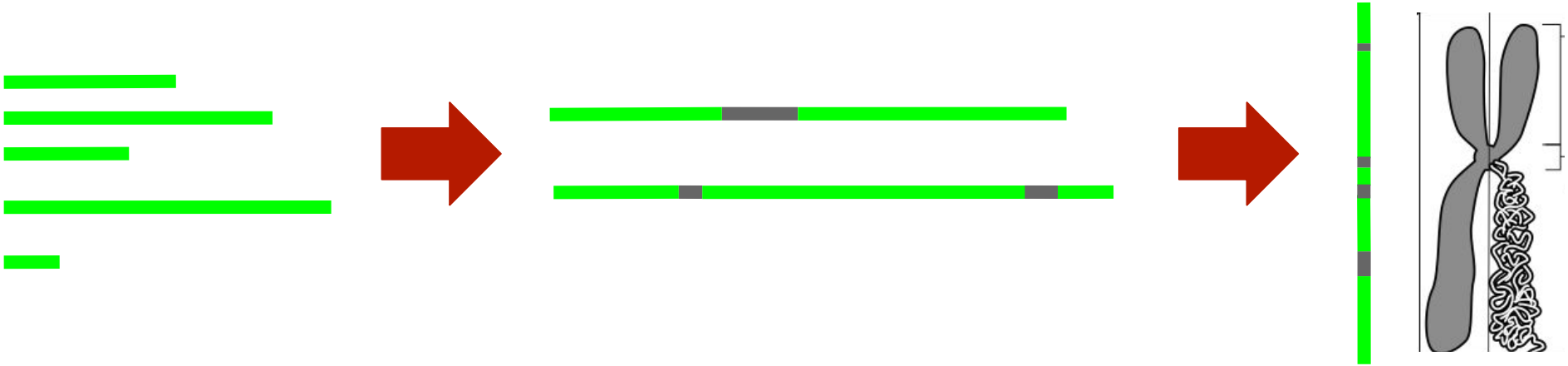
```
>contig#25_0
GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCA
GTATGCACGCGATAGCATTGCGAGACGCTGGAGCCGGAGCACCCT
CTGCCTCATCCTATTATTTATCGCACCTACGTTCAATATTACAGG
ATTAATTAATGCTTGTAGGACATAATAATAACAATTGAATGTCTG
ATAACAAAAAAATTTCCACCAAACCCCCCCTCCCCCGCTTCTGGCC
AACCCCAAAAACAAAGAACCCTAACACCAGCCTAACCAGATTTCA
TTTTAACAGTCACCCCCCAACTAACACATTATTTTCCCCTCCCAC
CAACCCCCGCCCATCCTACCCAGCACACACACACCGCTGCTAACC
AAAGACACCCCCCACAGTTTATGTAGCTTACCTCCTCAAAGCAAT
ACATCACCCCATAAACAAATAGGTTTGGTCCTAGCCTTTCTATTA
GCATCCCCGTTCCAGTGAGTTCACCCTCTAAATCACCACGATCAA
AATGCAGCTCAAAACGCTTAGCCTAGCCACACCCCCACGGGAAAC
ACGAAAGTTTAACTAAGCTATACTAACCCCAGGGT
```

Important definitions

Contigs

Scaffolds

Chromosome



Important definitions

Contigs

```
>contig#1
GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCA
GTATGCACGCGATAGCATTGCGAGACGCTGGAGCCGGAGCACCCT
CTGCCTCATCCTATTATTTATCGCACCTACGTTCAATATTACAGG
>contig#2
ATTAATTAATGCTTGTAGGACATAATAATAACAATTGAATGTCTG
ATAACAAAAAATTTCCACCAAACCCCCCCTCCCCCGCTTCTGGCC
>contig#3
AACCCCAAAAACAAAGAACCCTAACACCAGCCTAACCAGATTTCA
TTTTAACAGTCACCCCCCAACTAACACATTATTTTCCCCTCCCAC
CAACCCCCGCCCATCCTACCCAGCACACACACACCGCTGCTAACC
AAAGACACCCCCCACAGTTTATGTAGCTTACCTCCTCAAAGCAAT
>contig#4
ACATCACCCCATAAACAAATAGGTTTGGTCCTAGCCTTTCTATTA
GCATCCCCGTTCCAGTGAGTTCACCCTCTAAATCACCACGATCAA
AATGCAGCTCAAAACGCTTAGCCTAGCCACACCCCCACGGGAAAC
ACGAAAGTTTAACTAAGCTATACTAACCCCAGGGT
```

Important definitions

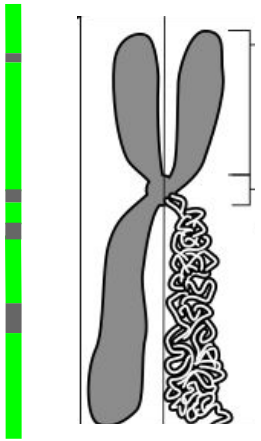
```
>scaffold#1
AACCCCAAAAACAAAGAACCCTAACACCAGCCTAACCAGATTTCA
TTTTAACAGTCACCCCCCAACTAACACATTATTTTCCCCTCCCAC
CAACCCCCGCCCATCCTACCCAGCACACACACACCGCTGCTAACC
AAAGACACCCCCCACAGTTTATGTAGCTTACCTCCTCAAAGCAAT
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNGATCACAGGTCTATC
ACCCTATTAACCACTCACGGGAGCTCTCCA
>scaffold#2
GTATGCACGCGATAGCATTGCGAGACGCTGGAGCCGGAGCACCCT
CTGCCTCATCCTATTATTTATCGCACCTACGTTCAATATTACAGG
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNATTAATTAATGCT
GTAGGACATAATAATAACAATTGAATGTCTGATAACAAAAAATTC
CACCAAACCCCCCCTCCCCCGCTTCTGGCCNNNNNNNACATCACC
CATAAACAAATAGGTTTGGTCCTAGCCTTTCTATTAGCATCCCCT
TCCAGTGAGTTCACCCTCTAAATCACCACGATCAAAATGCAGCTA
AAACGCTTAGCCTAGCCACACCCCCACGGGAAACACGAAAGTTTA
ACTAAGCTATACTAACCCCAGGGT
```

Scaffolds

Important definitions

```
>chr22
GTATGCACGCGATAGCATTGCGAGACGCTGGAGCCGGAGCACCCT
T
CTGCCTCATCCTATTATTTATCGCACCTACGTTCAATATTACAGG
G
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNATTAATTAATGCT
TGTAGGACATAATAATAACAATTGAATGTCTGATAACAAAAAATT
TCCACCAAACCCCCCCTCCCCCGCTTCTGGCCNNNNNNNACATCA
CCCCATAAACAAATAGGTTTGGTCCTAGCCTTTCTATTAGCATCC
CCGTTCCAGTGAGTTCACCCTCTAAATCACCACGATCAAAATGCA
GCTCAAAACGCTTAGCCTAGCCACACCCCCACGGGAAACACGAAA
GTTTAACTAAGCTATACTAACCCCAGGGTNNNNNNNAACCCCAAA
AACAAAGAACCCTAACACCAGCCTAACCAGATTTCATTTTAACAG
TCACCCCCCAACTAACACATTATTTTCCCCTCCCACCAACCCCCG
CCCATCCTACCCAGCACACACACACCGCTGCTAACCAAAGACACC
CCCCACAGTTTATGTAGCTTACCTCCTCAAAGCAATNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNGATCACAGGTCTATCACCCTATTA
ACCACTCACGGGAGCTCTCCA
```

Chromosom
e

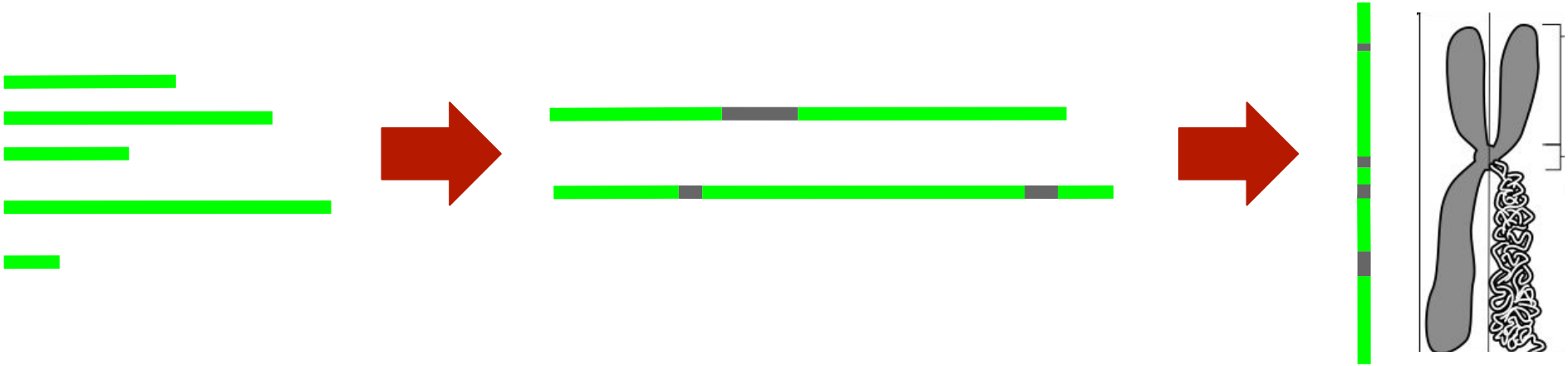


Important definitions

Contigs

Scaffolds

Chromosome



Which approaches?

- Greedy (“Simple” approach)
- Overlap-Layout-Consensus (OLC)
- de Bruijn graphs

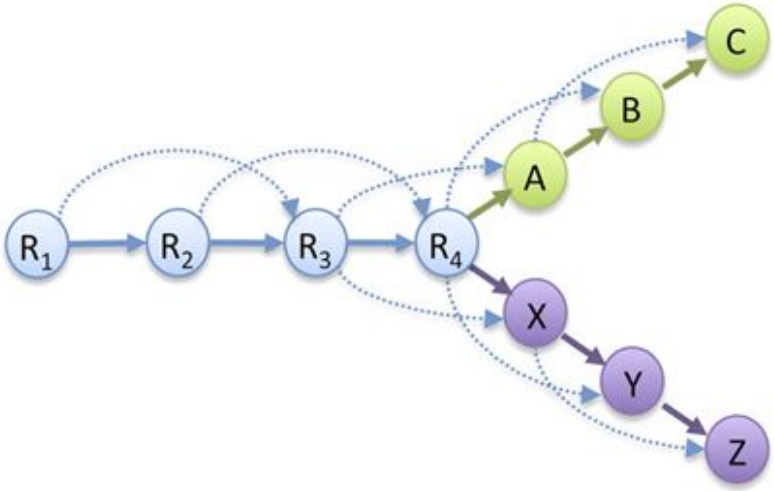
Simple approach - Greedy

- Principle:
    1. Pairwise alignment of all reads
    2. Identify fragments that have largest overlap
    3. Merge these
    4. Repeat until all overlaps are used
- Can only resolve repeats smaller than read length
- High computational cost with increasing no. reads

Overlap-Layout-Consensus

- Create overlap graph by all-vs-all alignment (Overlap)
- Build graph where each node is a read, edges are overlaps between reads (Layout)

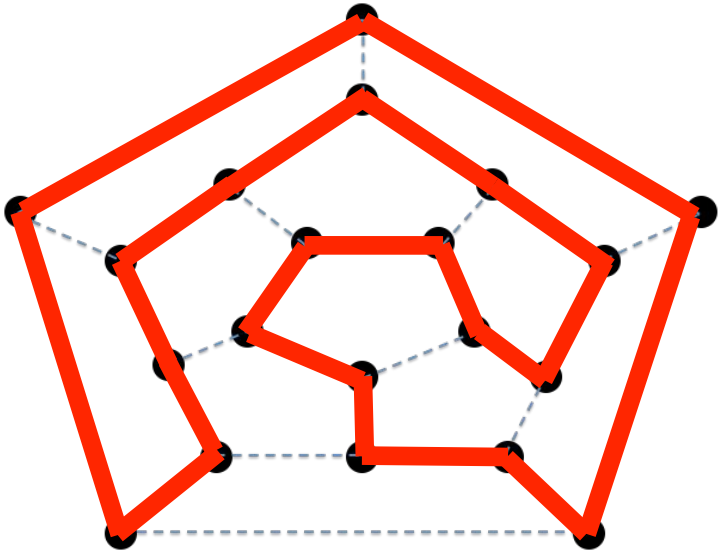
```
R1: GACCTACA
R2:   ACCTACAA
R3:    CCTACAAG
R4:     CTACAAGT
A:       TACAAGTT
B:        ACAAGTTA
C:         CAAGTTAG
X:       TACAAGTC
Y:        ACAAGTCC
Z:         CAAGTCCG
```



Schatz et al., Genome Res, 2010

Overlap-Layout-Consensus

- Create consensus sequence
- We need to use **graph theory** to solve the graph
- Find the *Hamiltonian path*
- i.e. visit each node *exactly once*



Imagine trying to solve this for a graph of hundred of thousands of nodes (=reads)

Overlap-Layout-Consensus

- Not good with many short reads -> lots of alignment!
- With short read lengths, hard to resolve repeats

- Good for large read lengths:
  - PacBio, Oxford Nanopore, 10X Genomics, 454, Ion Torrent, Sanger
- Example assemblers: Canu, Celera, Newbler

de Bruijn graph

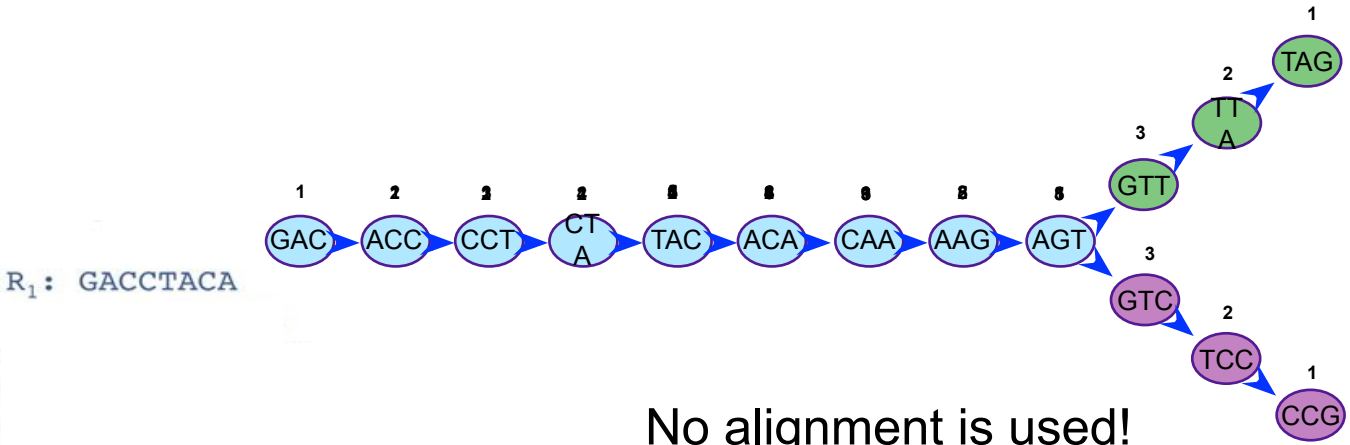
- Directed graph of overlapping items (here DNA sequences)
- Instead of comparing reads, decompose reads into *k*-mers
  - Graph is created by mapping the *k*-mers to the graph
  - Each *k*-mer only exists once in the graph
  - Problem is reduced to walking Eulerian path (visiting each edge once) - this is a solve-able problem

Drawbacks ...

- Lots of RAM required (**1-1000 GB !**)
- Optimal *k* can not be identified *a priori*, must be experimentally tested for each dataset
- small *k*: very complex graph, large *k*: limited overlap in low coverage areas
- Iterative approach to find best assembly

How is the graph constructed?

• Same 10 reads, extract *k*-mers from reads and map onto graph, *k* = 3:



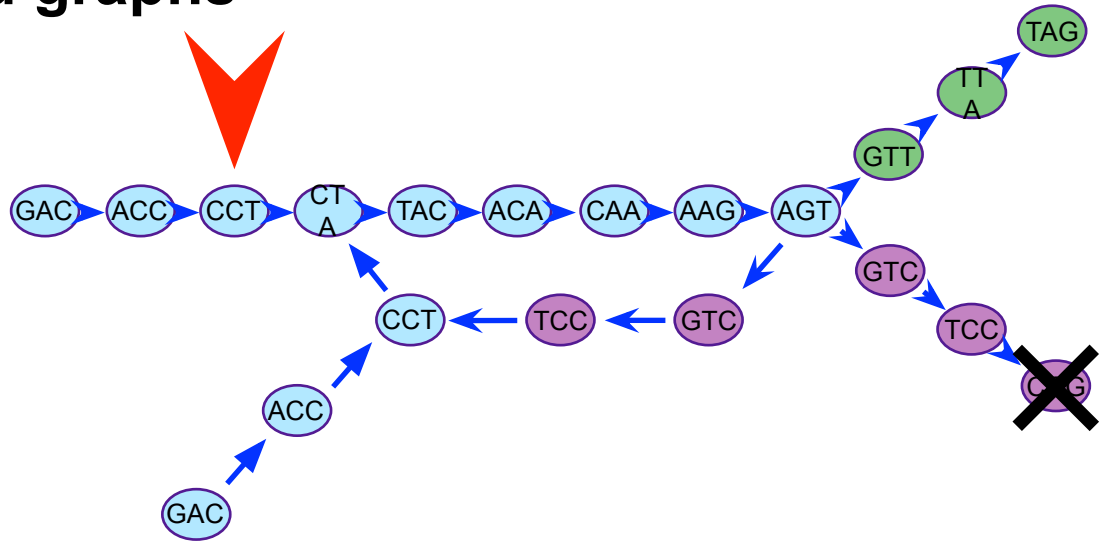
No alignment is used!

Different assemblers uses different modifications of the de Bruijn graphs

Complicated graphs

```
R1: GACCTACA
R2:   ACCTACAA
R3:    CCTACAAG
R4:     CTACAAGT
A:       TACAAGTT
B:        ACAAGTTA
C:         CAAGTTAG
X:       TACAAGTC
Y:        ACAAGTCC
Z:         CAAGTCCT
```

G to T

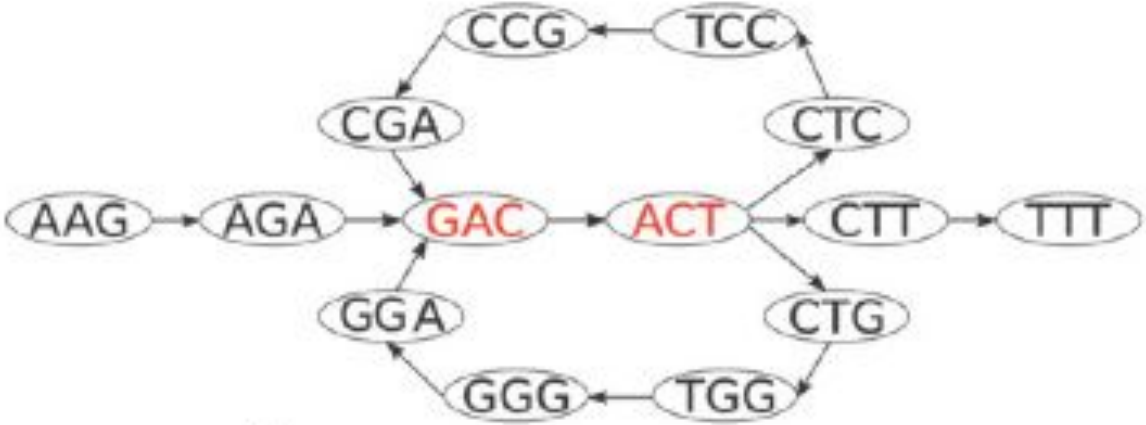


Large genomes with many repeats/errors creates very large graphs

Create the *de* Bruijn graph of this genome using *k*=3

AAGACTCCGACTGGGACTTT

AAGACTCCGACTGGGACTTT

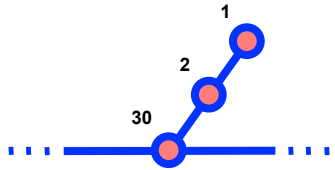


A de Bruijn graph of a sequence

After building: Simplify

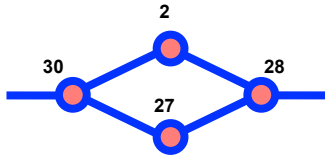
Clip tips

(seq err,end)

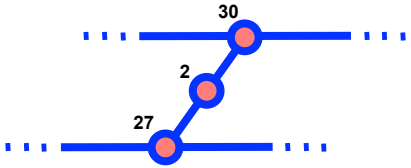


Pinch bubbles

(seq err, middle, SNP)

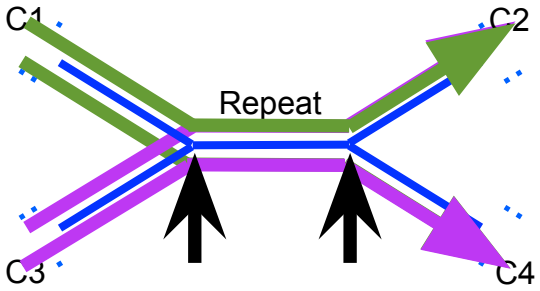


Remove low cov. links



Create contigs and scaffolds

Cut graph at repeat boundaries to create contigs



Use paired-end information to resolve repeats and combine to scaffolds

Fill potential gaps using PE reads

The assembly is done

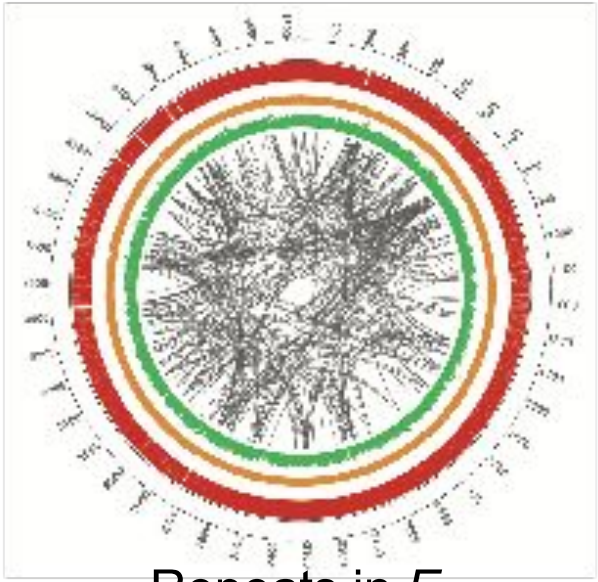
Iterate parameters

- Re-run with different *k*-sizes, find optimum
- Run with multiple k-mers at the same time! (eg. SPAdes)
- Compare assembly statistics such as, assembly length, N50, no. contigs

- Assembly refinement
  - Break contigs not supported by PE/MP reads
  - Analyze assembly using REAPR or QUAST

Successful *de novo* assembly

- Success is a factor of:
  - Genome size, **genomic repeats(!)**, ploidy
  - High coverage, long read lengths, PE/MP libraries



Repeats in *E. coli*

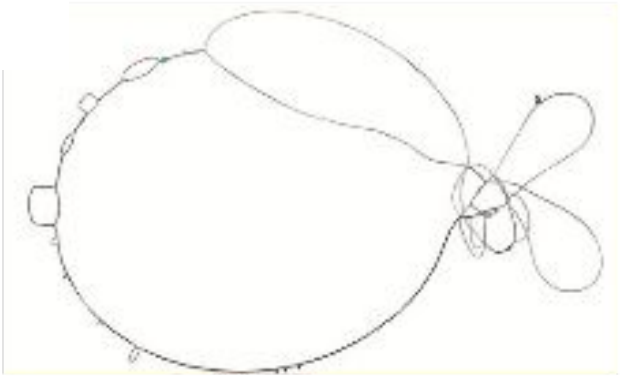
Improving *de novo* assemblies

- Paired-end & Mate-pair for long range continuity
- Hybrid approaches (combine Illumina with PacBio/Oxford Nanopore)

- Synthetic long reads: Illumina Synthetic Reads (Moleculo) or 10X Genomics
- Hi-C contact maps

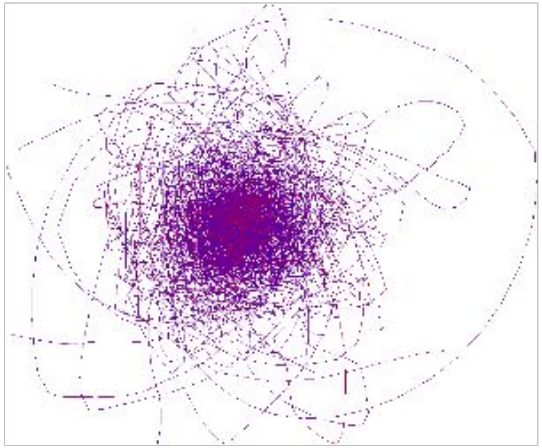
Two bacterial genomes *de Bruijn* graphs

Few repeats



Flicek & Birney, Nat.Methods 2009

"more" repeats

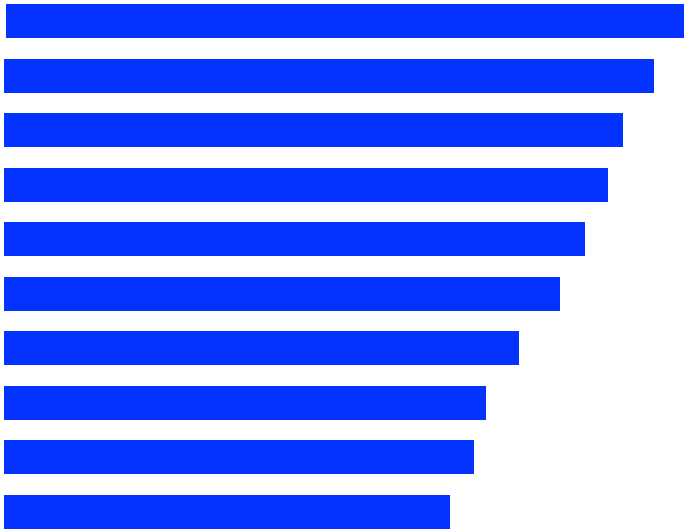


Zerbino, 2009

N50: Assembly quality

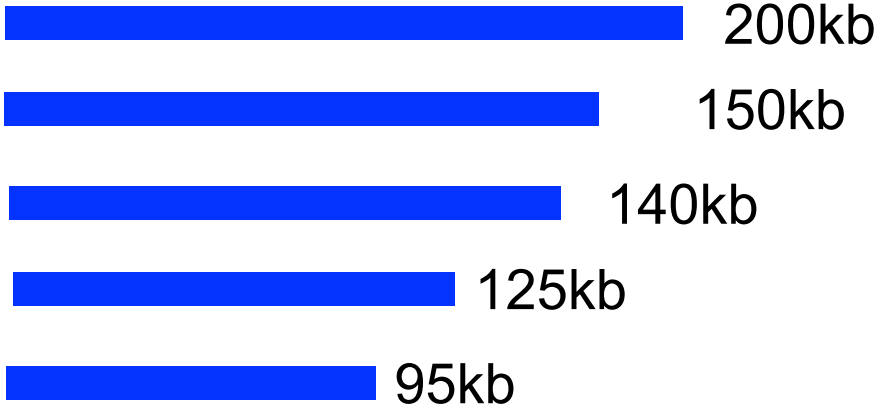
N50: What is the smallest piece in the largest half of the assembly?

- Calculate sum of assembly
- Order contigs by size
- Sum contigs starting by largest
- When half the sum is reached, N50 is the length of the contig



N50 example

5 scaffolds, calculate N50:



Sum: 200+150+140+125+95=710kb

Half: 710 / 2 = 355kb

200kb + 150kb = 350kb

350kb + 140kb = 490kb

490kb > 355kb => **N50: 140kb**

Some assemblers

- OLC: Canu, Newbler
- de Bruijn: Allpaths-LG, SPAdes, Velvet(best), SOAPdenovo, Megahit (very lean), …
- other: MIRA, SGA, Flye (very good for 3g NGS)

Used in exercises today

Exercise time!