A little tour of assembly methods

Camille Marchet & Antoine Limasset

Université de Lille, CNRS, France

EVOMICS Workshop on Genomics 2020 - Český Krumlov

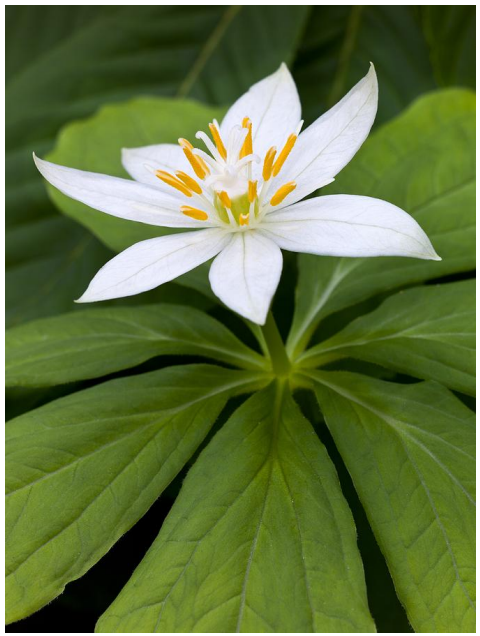
Content of this course

- How to reconstruct a genome with sequencing data?
- What are the main challenges?
- Which solutions have been proposed?

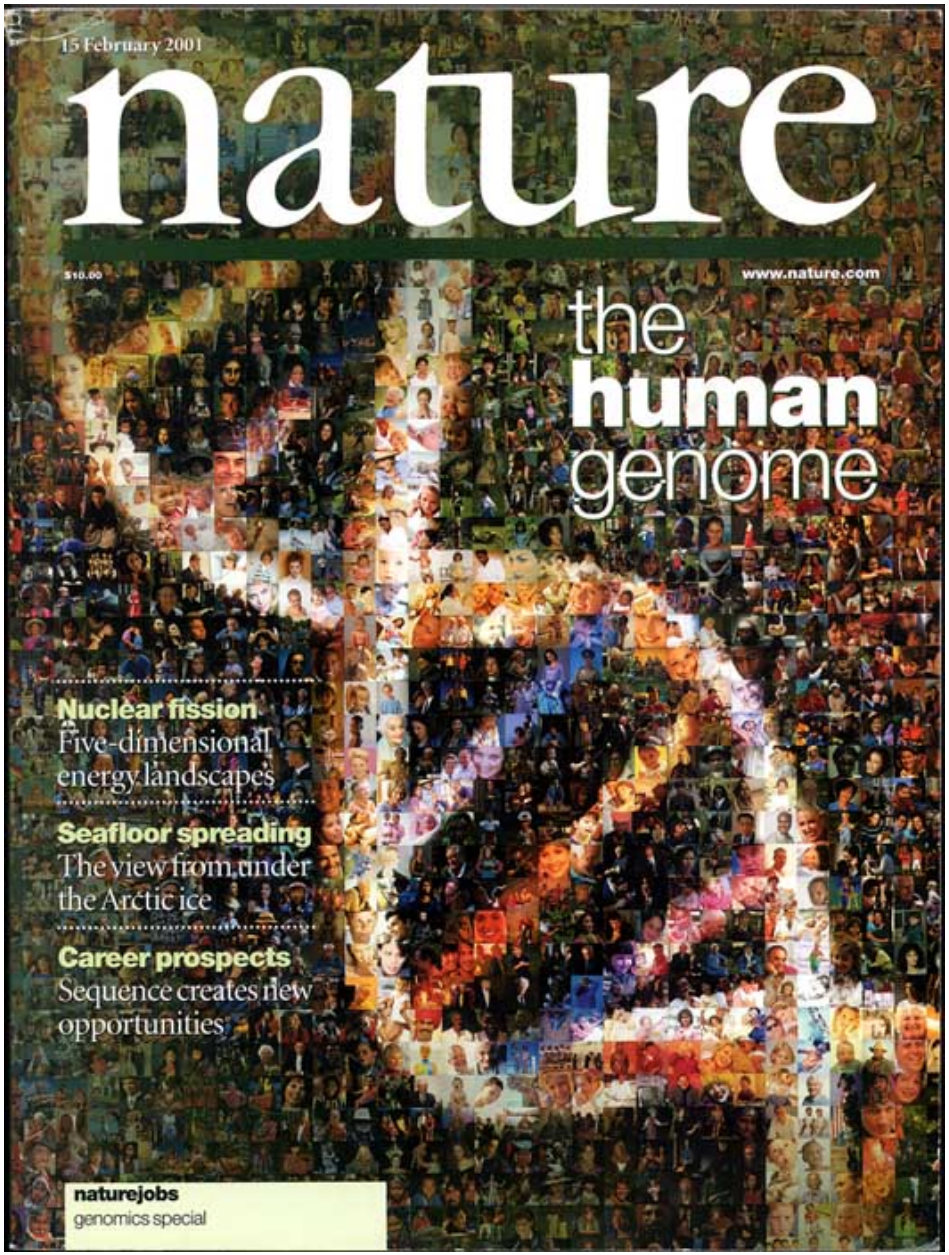
genome size: $\sim$ 32 gigabases

Content of this course

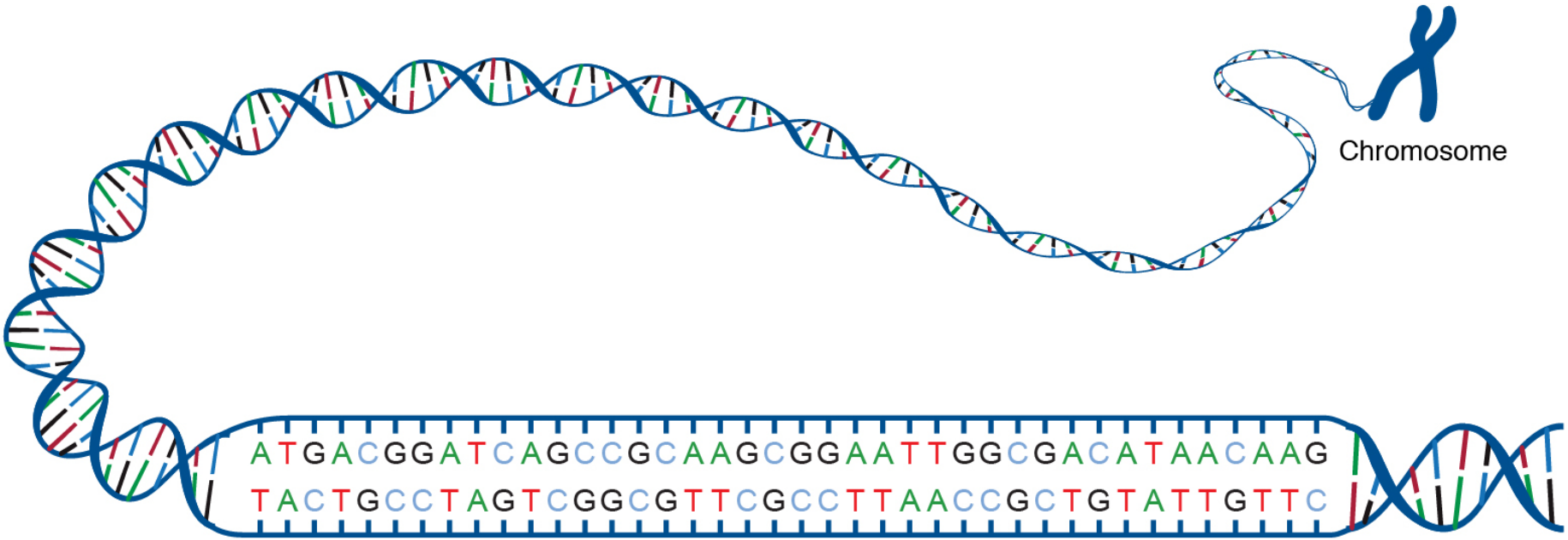
- What is assembly?
- What are the main problematics to bear in mind when conducting an assembly?
- What are the current limits of assembly?



Assembly

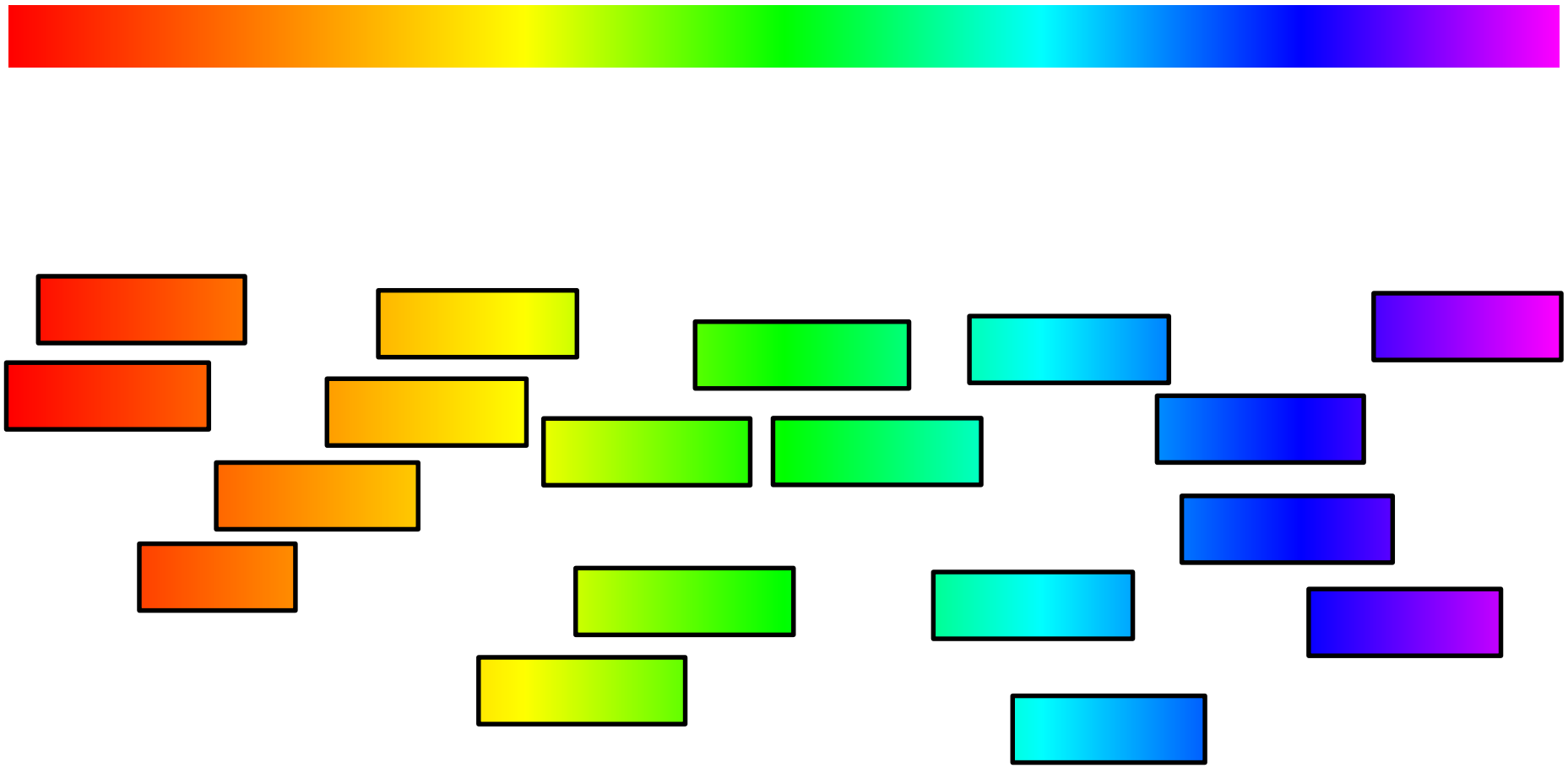

15 February 2001
nature
$10.00
www.nature.com
the human genome
Nuclear fission
Five-dimensional energy landscapes
Seafloor spreading
The view from under the Arctic ice
Career prospects
Sequence creates new opportunities
naturejobs
genomics special

Accessing a genome

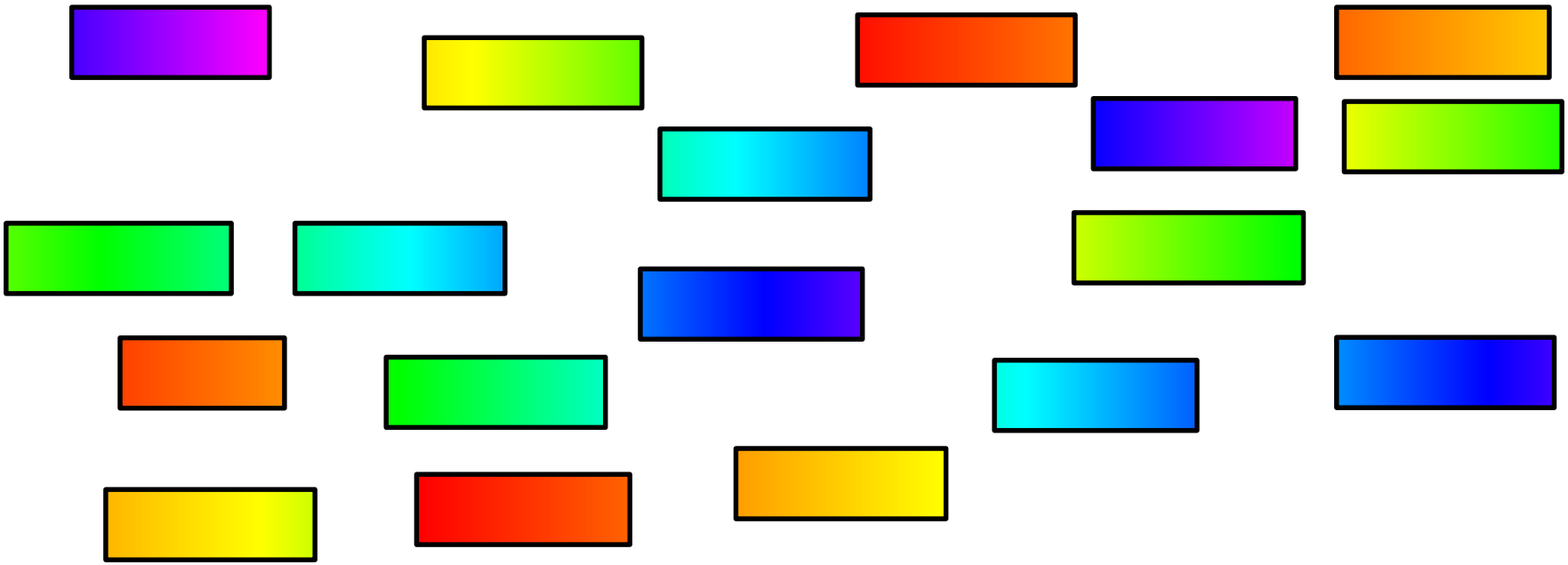


From `www.genome.gov/genetics-glossary/acgt`

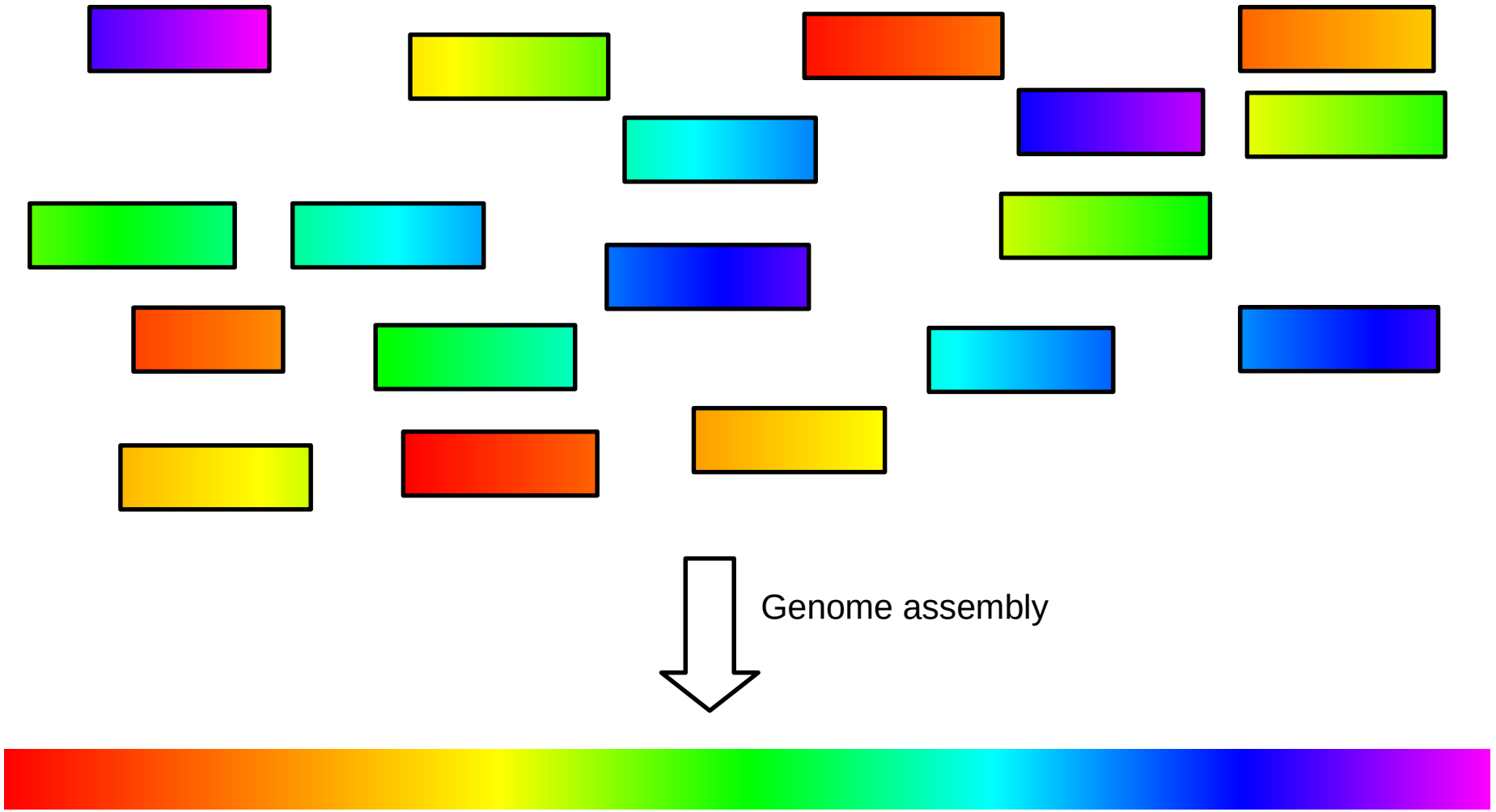
Reads are words from the genome



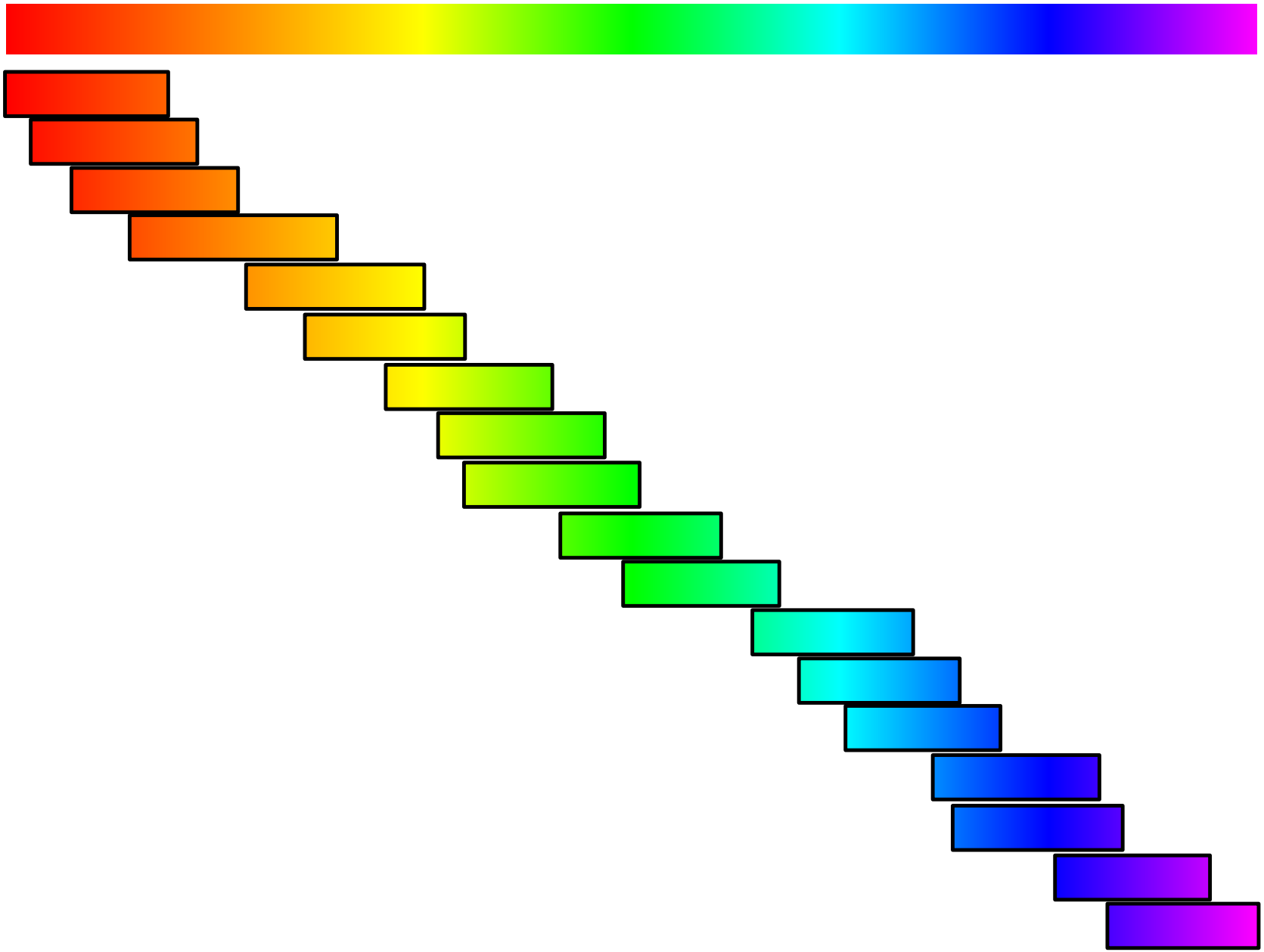
Reads are **shuffled** words from the genome



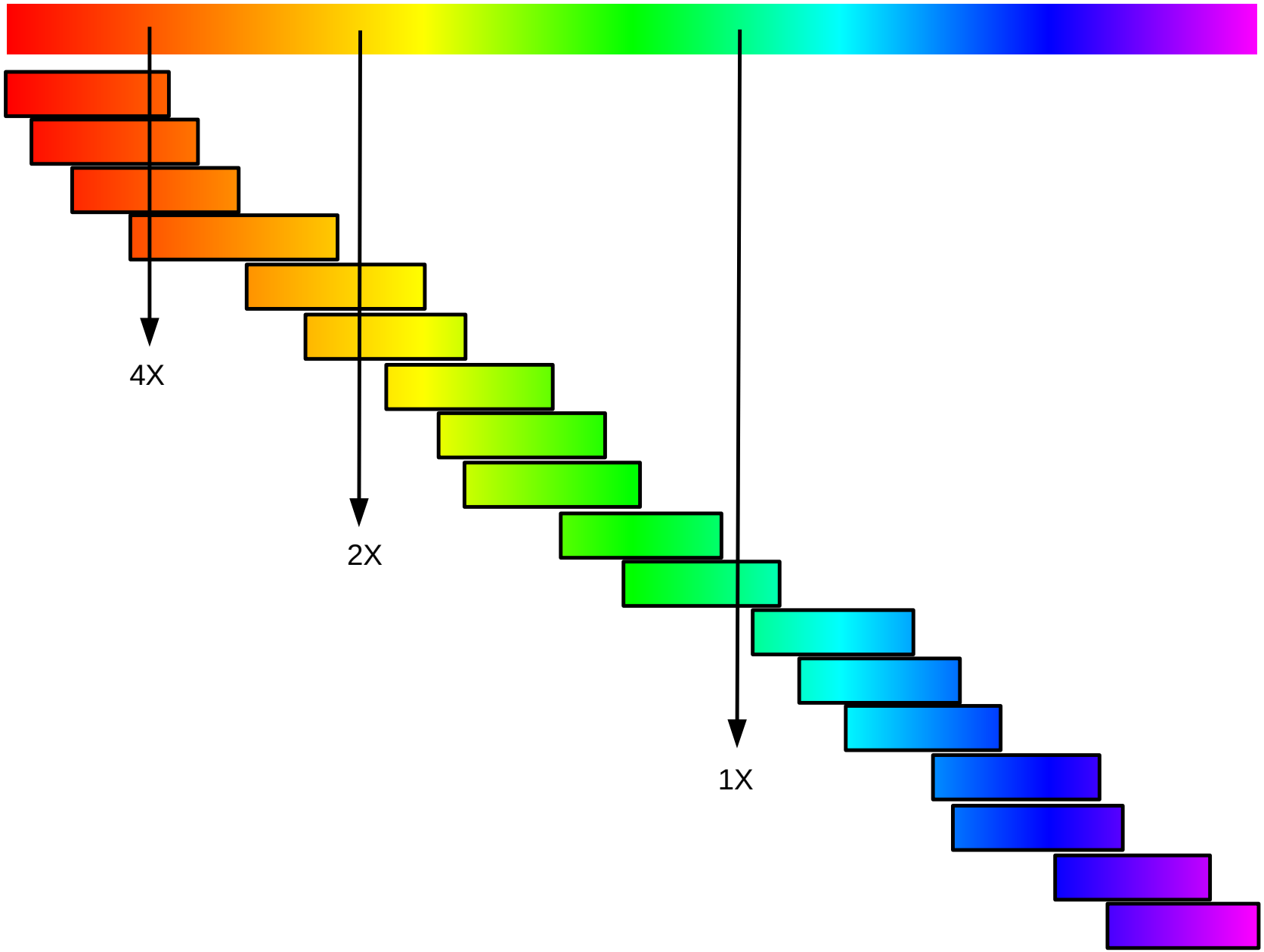
Genome assembly task



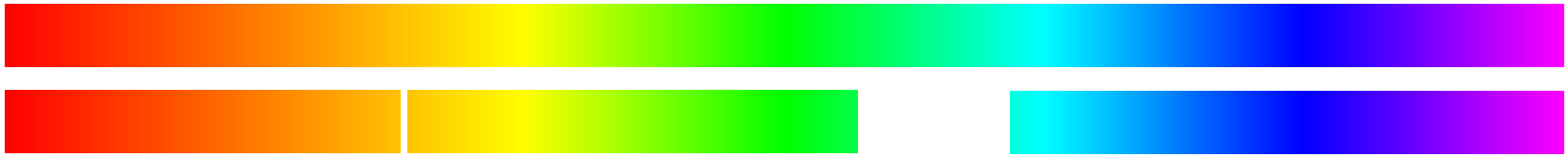
Using read overlaps



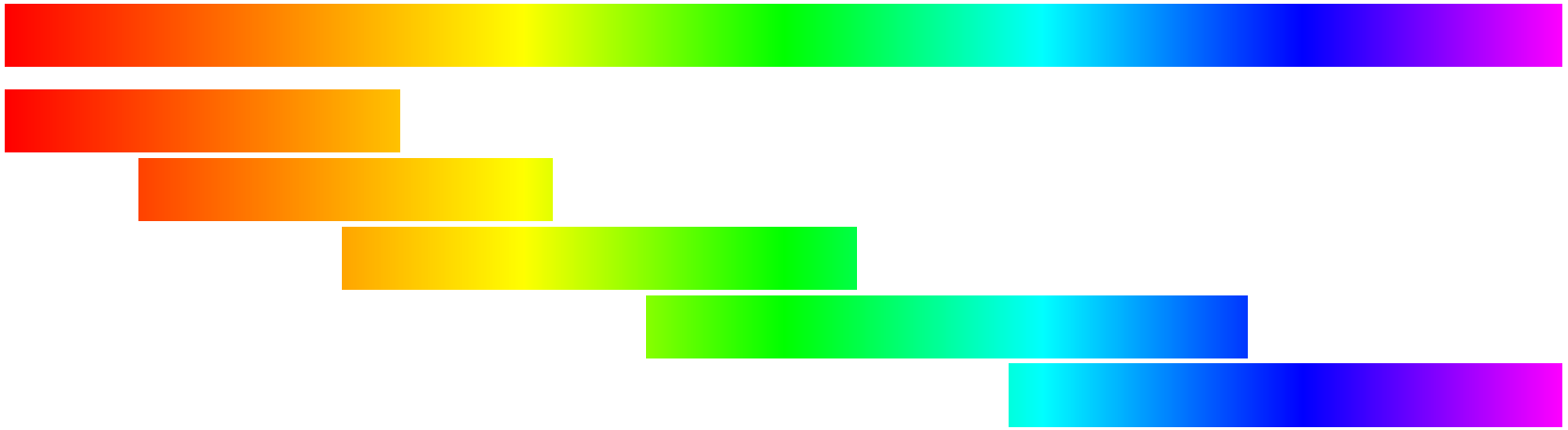
Genome sequencing: coverage



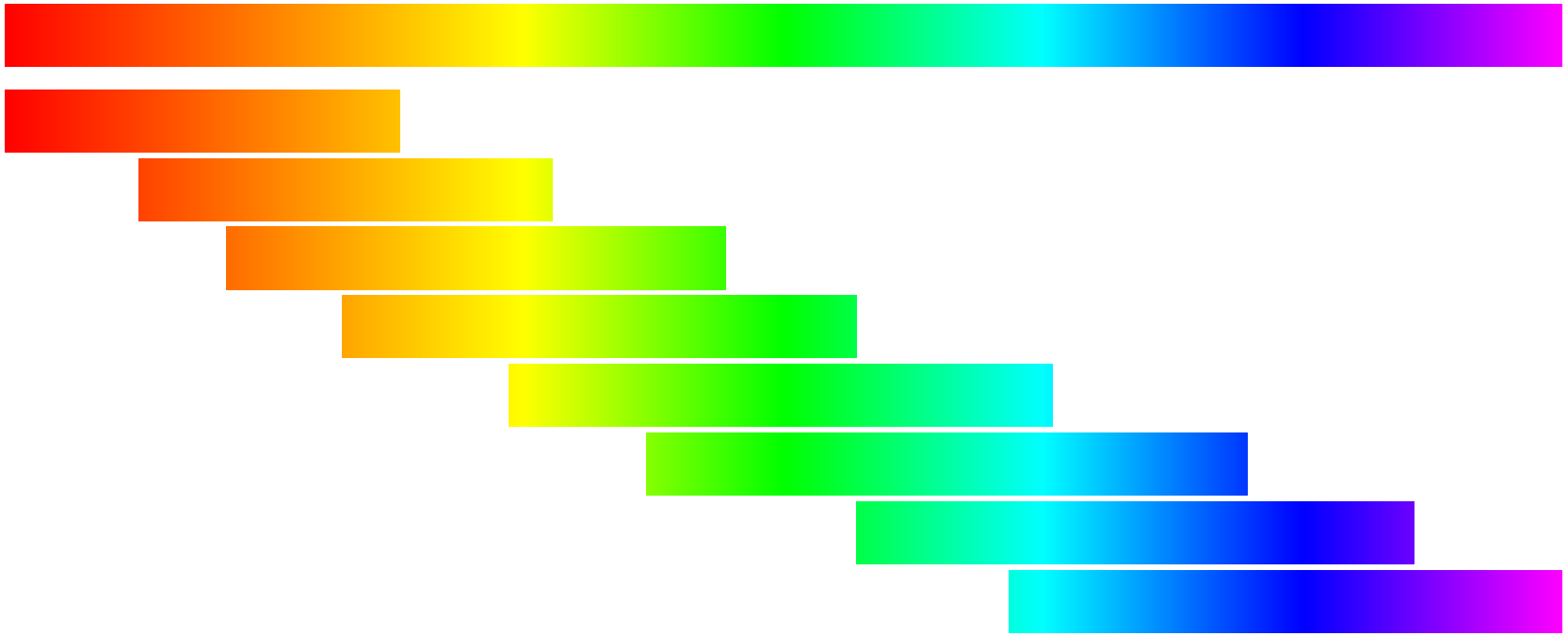
Genome sequencing: coverage



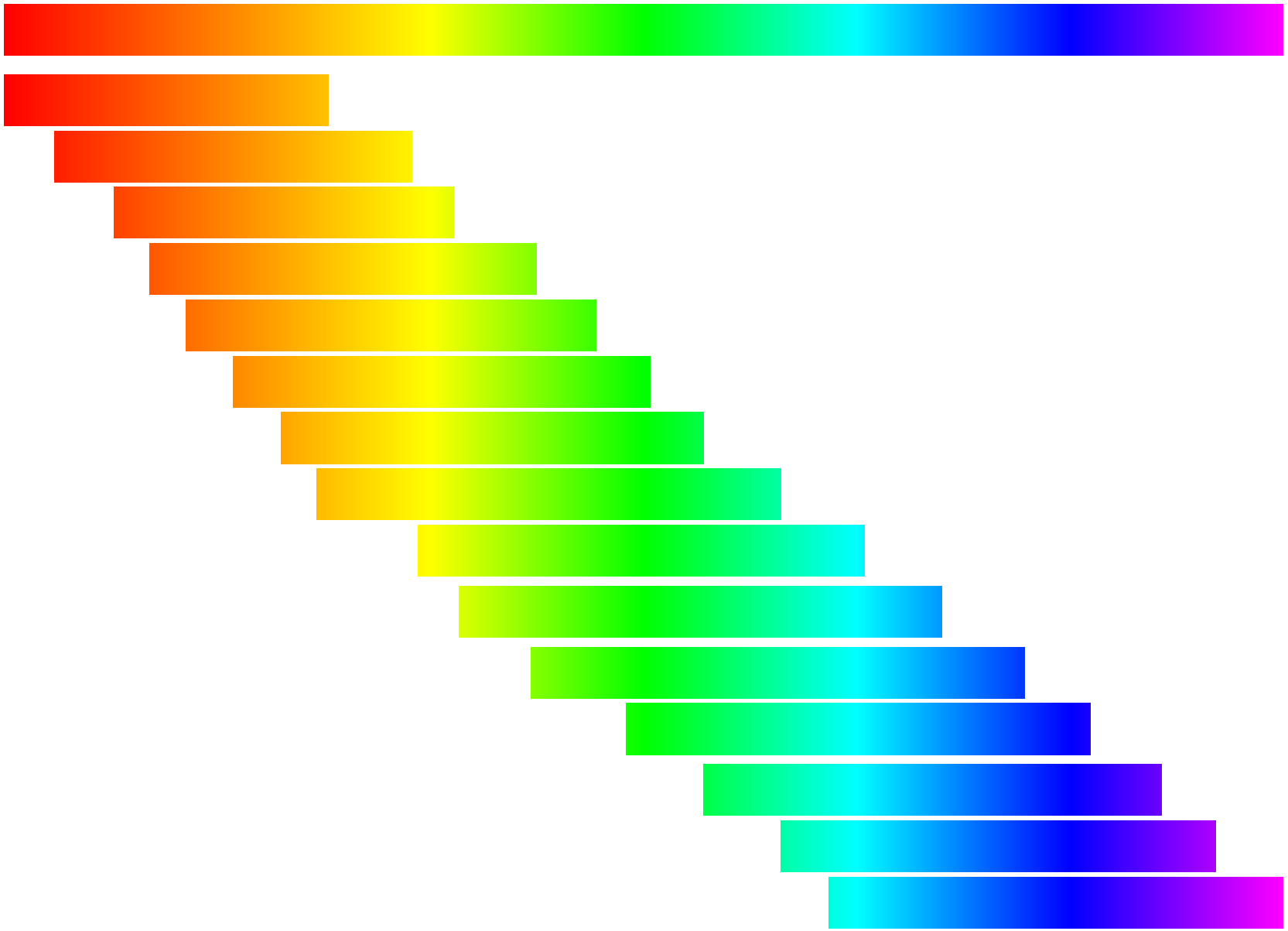
Genome sequencing: coverage



Genome sequencing: coverage



Genome sequencing: coverage



Assembly **idea number 1: Select the longest overlaps**

```
Reads:
1: ATCGGTATCG
2: GGTATCGTTA
3: ATCGTTACGG
4: GTTACGGTAT
5: ACGGTATACC
```

```
1: ATCGGTATCG                    Overlap length:7
2:    GGTATCGTTA

1: ATCGGTATCG                    Overlap length:4
3:       ATCGTTACGG

1: ATCGGTATCG                    Overlap length:1
4:           GTTACGGTAT

1: ATCGGTATCG                    No Overlap
5:            ACGGTATACC
```

Assembly idea number 1: assemble the longest overlaps

```
Reads:
1: ATCGGTATCG
2: GGTATCGTTA
3: ATCGTTACGG
4: GTTACGGTAT
5: ACGGTATACC

Best overlaps:
1: ATCGGTATCG
2:   GGTATCGTTA
3:      ATCGTTACGG
4:         GTTACGGTAT
5:            ACGGTATACC

Output "genome":
ATCGGTATCG + TTA + CGG + TAT + ACC
ATCGGTATCGTTACGGTATACC
```

Your time to shine!

Let assemble this genome!

Your read set:

```
1: ATTTACGGGT
2: TTACGGGTGG
3: ACGGGTCCTT
4: GTCCTTTCTT
5: TTTCTTACGG
```

For each read:
    Find the best overlap (length>5)
    Merge the two reads

The Greedy solution

```
The best overlaps:
ATTTACGGGT
  TTACGGGTGG

ACGGGTCCTT
    GTCCTTTCTT
        TTTCTTACGG
```

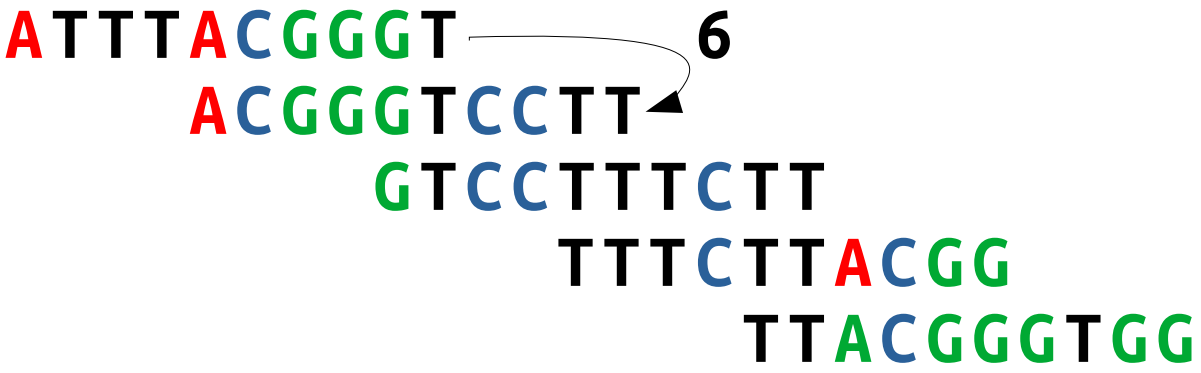
```
Output “genome”
ATTTACGGGTGG
ACGGGTCCTTTCTTACGG
```

The actual solution

The actual genome:

ATTTACGGGTCCTTTCTTACGGGTGG

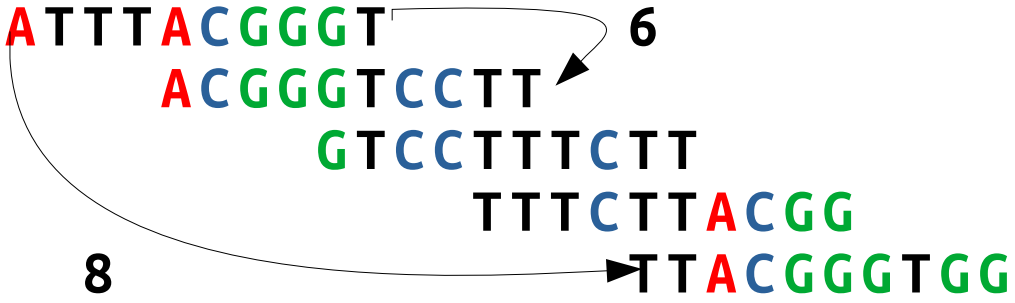
How the reads should be ordered:



What happened?

The actual genome:
ATTTACGGGTCCTTTCTTACGGGTGG

How the reads should be ordered:



ATTTACGGGT
TTACGGGTGG

ATTTACGGGTGG
Not in the genome

ACGGGTCCTT
GTCCTTTCTT
TTTCTTACGG

ACGGGTCCTTTCTTACGG
Not in the genome

Do we expect many repeats?

Probability to have NO repeated word of size 31 in a 5 megabases genome

Input interpretation:

$$\left(\frac{4^{31}-1}{4^{31}}\right)^{1/2\left(5\times10^6\left(5\times10^6-1\right)\right)}$$

Decimal approximation:

0.99999728949878430238317205542136383671202317193893202410 6...

From en.wikipedia.org/wiki/Birthday_problem

The burden of assembly: genomic repeats

Amount of repeats larger than a given size in *E. coli* genome

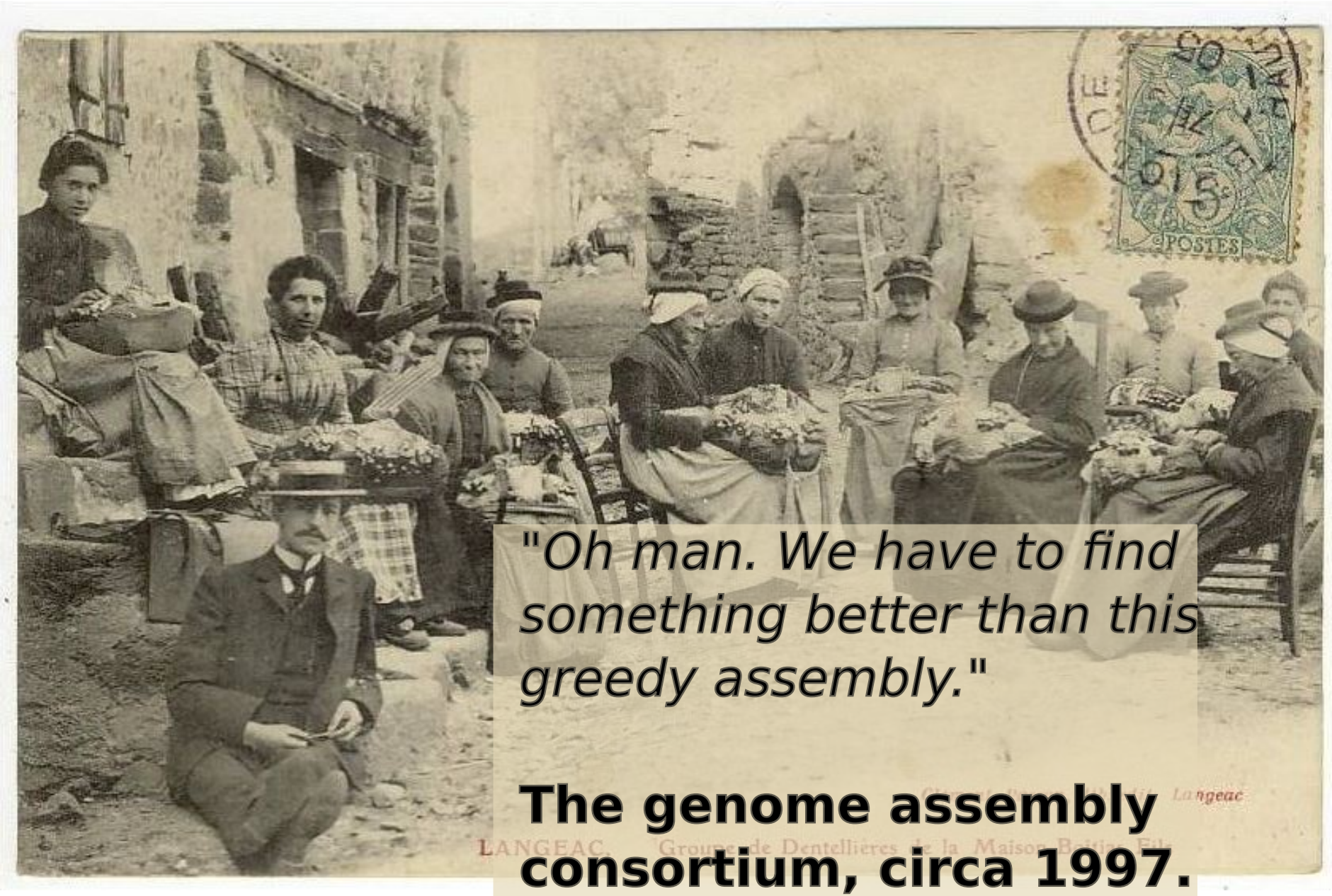
- 15: 44,994
- 21: 1,169
- 31: 559
- 41: 323
- 51: 225
- 61: 192

Genomic repeats are NOT random events

Greedy assemblers

- Simple and efficient scheme
- Rely on **local** best choice (greedy)
- May create errors because of local choices when there are repeats

History from the last century



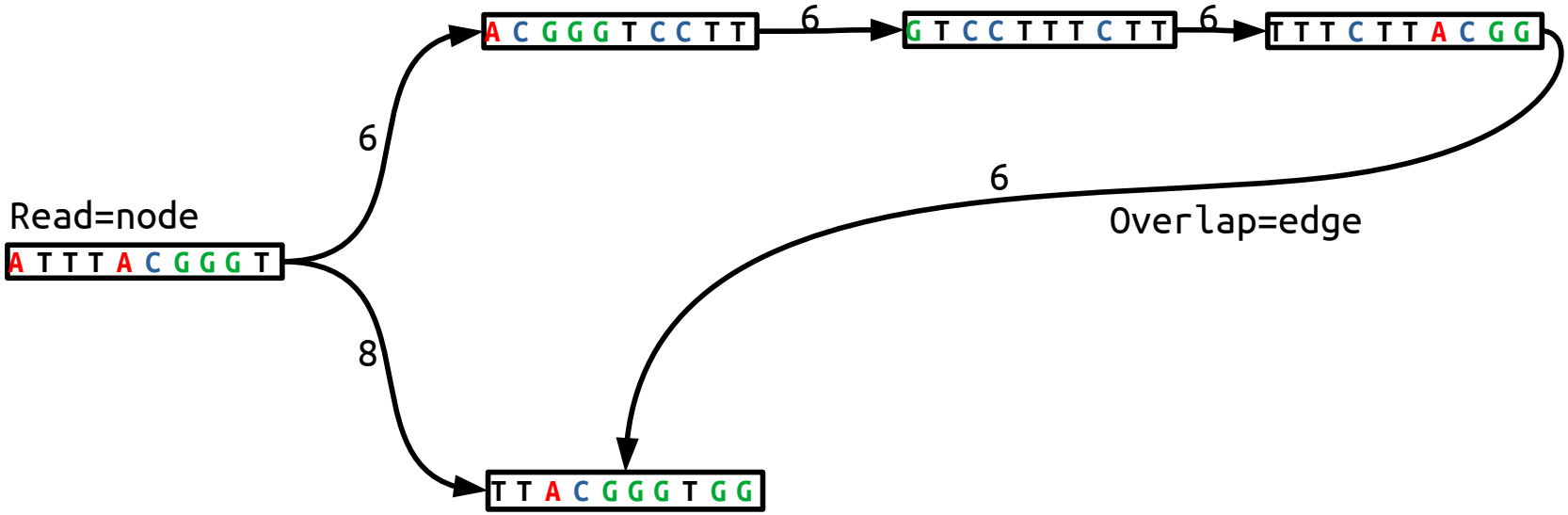
"Oh man. We have to find something better than this greedy assembly."

The genome assembly consortium, circa 1997.

Assembly **idea number 2: consider all overlaps**

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

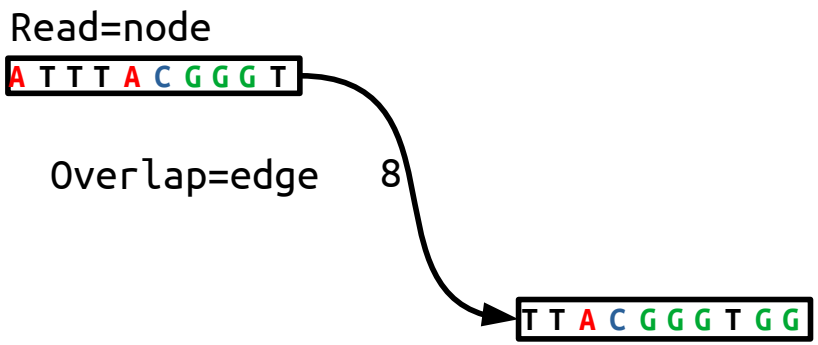
Overlap graph:



Greedy solution

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

Overlap graph:

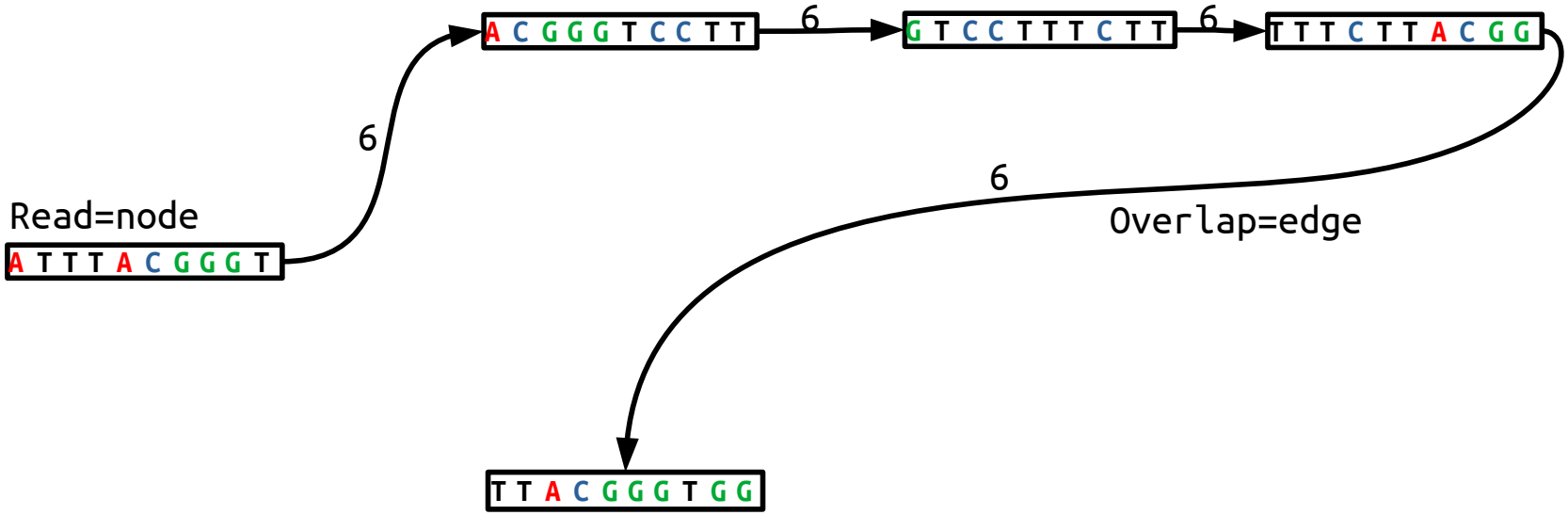


Greedy assembly output:
ATTTACGGGTGG
ACGGGTCCTTTCTTACGG

One piece solution

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

Overlap graph:

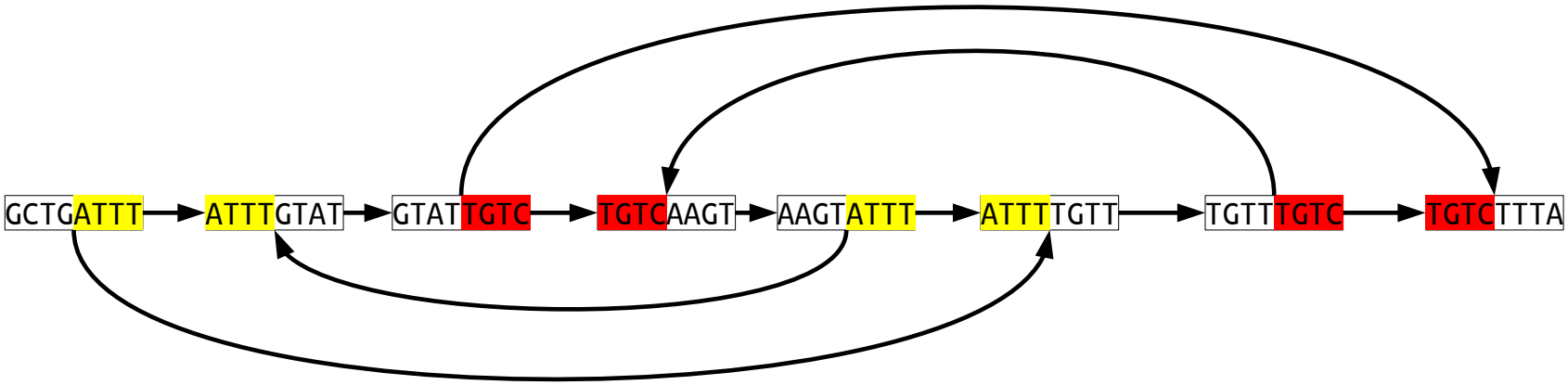


Overlap graph output:
ATTTACGGGTCCTTTCTTACGGGTGG

Multiple repeats

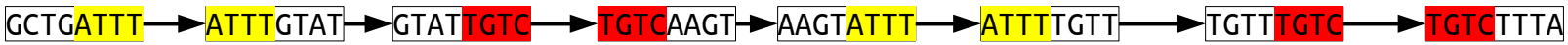
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:



First solution

```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA Overlap graph:
```

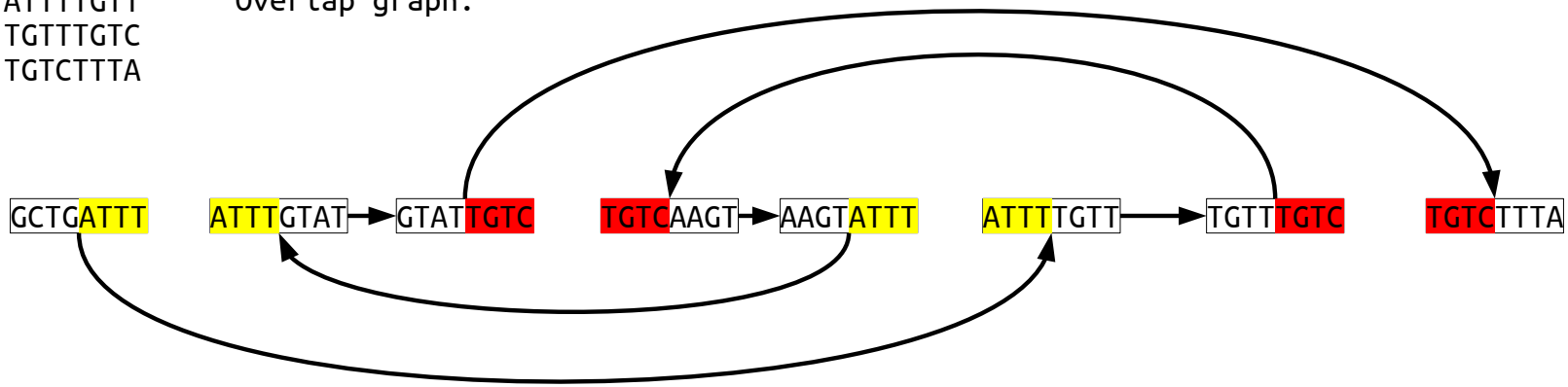



```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
```

Second solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:

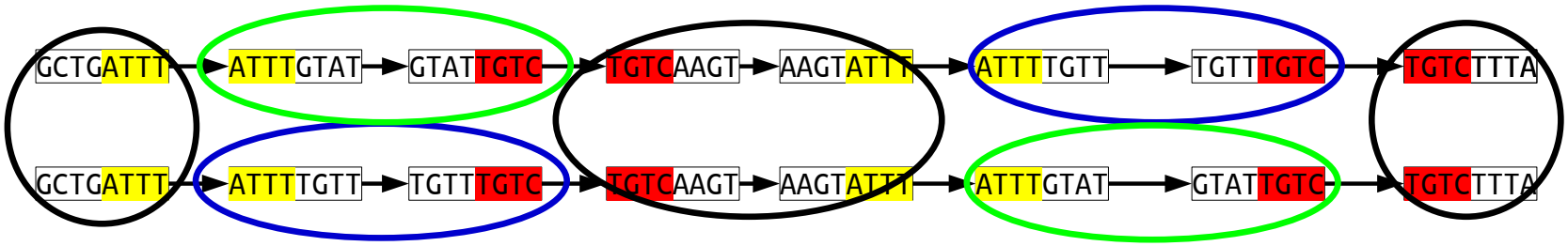


Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
GCTGATTTTGTTTGTCAAGTATTTGTATTGTCTTTA

Those two solutions are indistinguishable

Parsimonious solution: do not assemble

Possible assemblies:

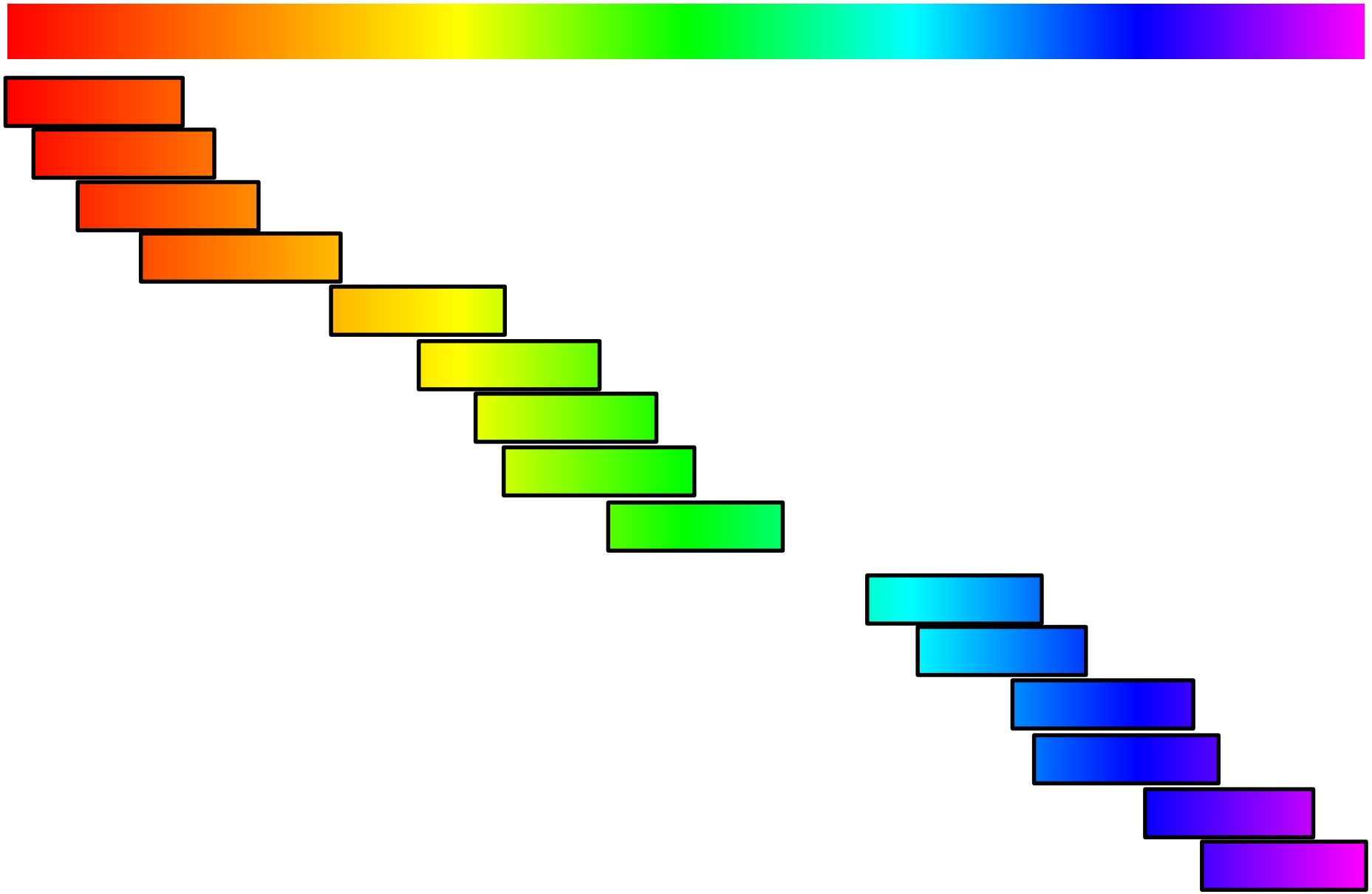


Genome pieces:

GCTGATTT ATTTGTATTGTC TGTCAAGTATTT ATTTTGTTTGTC

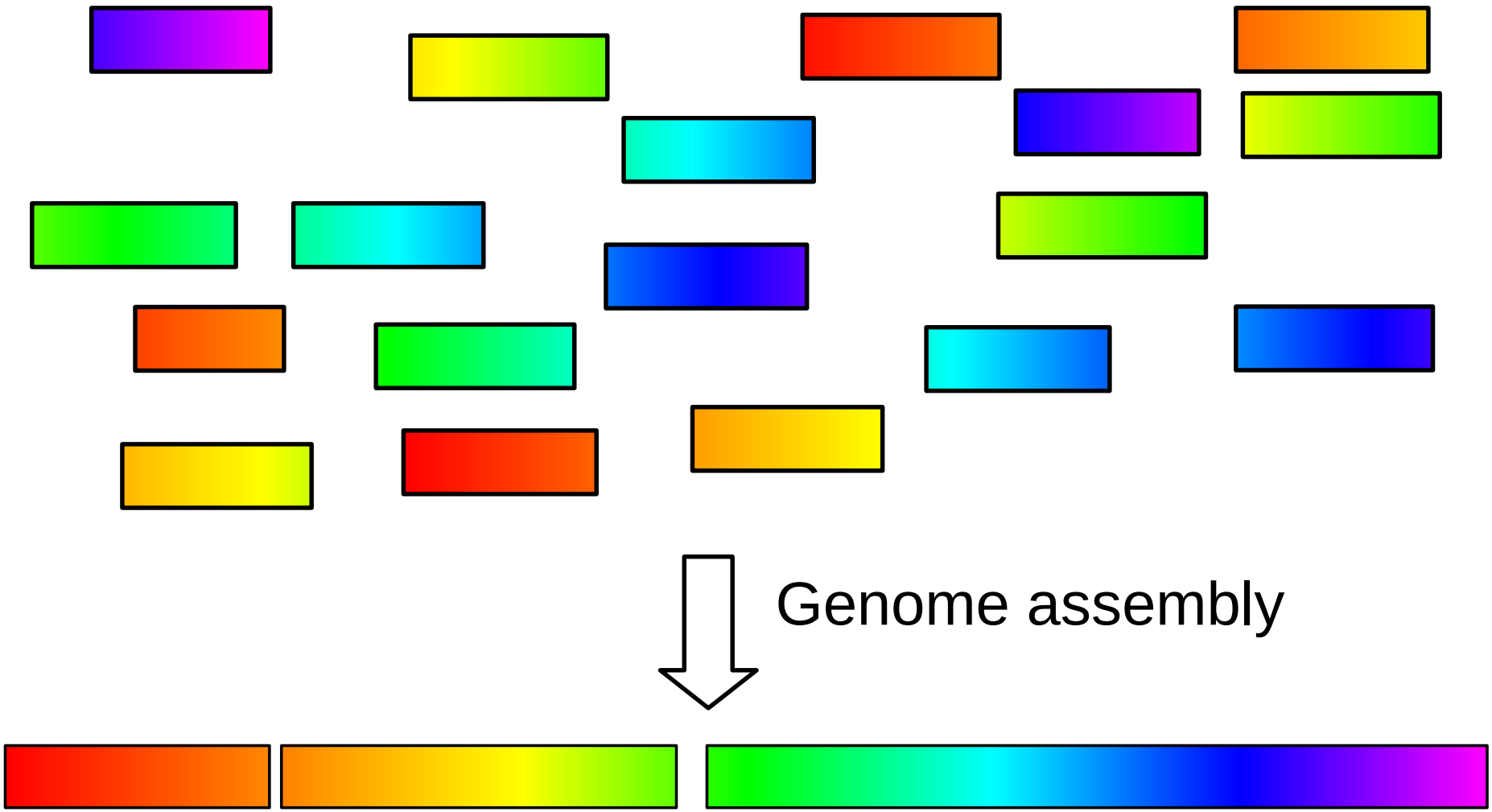
Repeats lead to the fragmentation of the assembly

Missing information also fragments the assembly

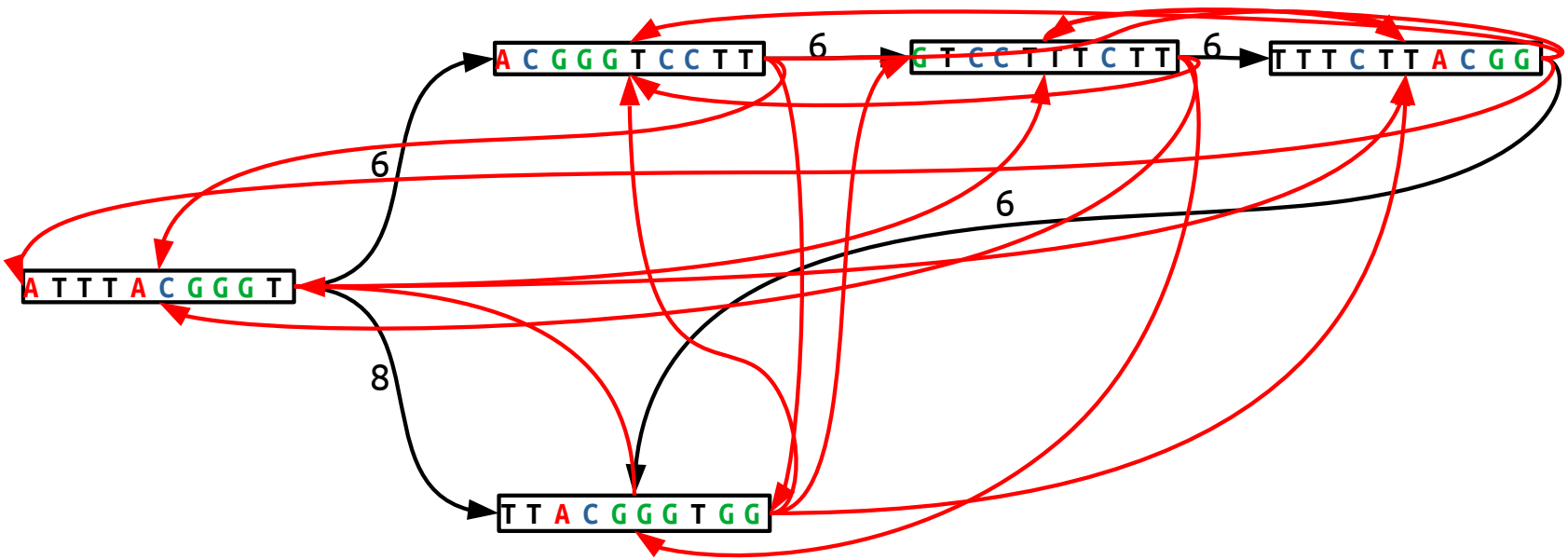


Assembly **concession number 1: output fragments**

In the real world, assemblers often provide pieces of genomes rather than complete ones

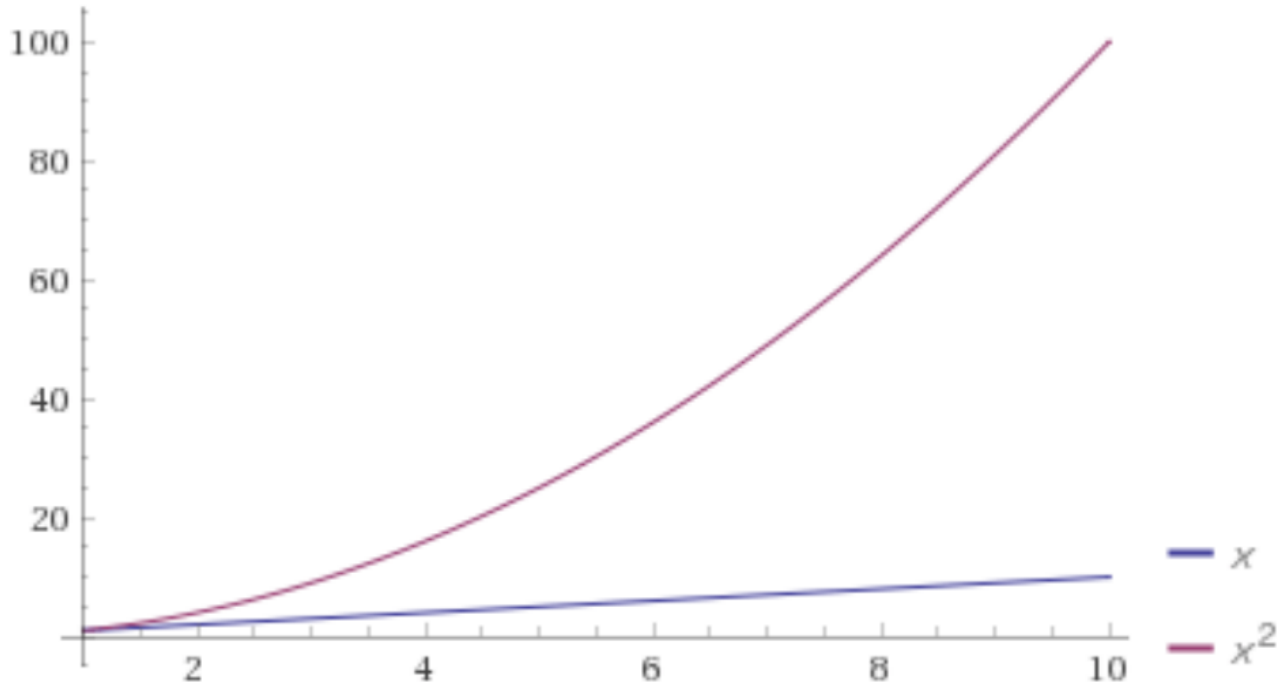


Overlap graph prerequisite : all overlaps



Overlap graph burden: number of reads

$n(n-1)/2 = \mathcal{O}(n^2)$ possible overlaps for n reads



Linear: 2X data 2X time
Quadratic: 2X data 4X time

Overlap graph burden: number of reads

$n(n-1)/2 = \mathcal{O}(n^2)$ possible overlaps for n reads

# Reads	# Overlaps
1000	499,500
10,000	50 million
100,000	5 billion
1 million	500 billion
10 million	50 trillion...

Most overlaps are too small to be considered...
The overlap computation is not linear

Overlap graphs in a nutshell

- Graphs of overlaps between the reads
- Can provide a global solution for assembly
- Can be difficult in real cases because it requires a lot of computation (overlaps)

S. cerevisiae, *D. melanogaster*, human could be assembled using overlap graphs approaches (Celera (Myers et al. 2000), SGA (Simpson & Durbin 2011), ...)

Fast forward

Assembly **idea number 3: Focus on genome words**

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC
GCCATGACC read

a read is a word from the genome

words of size 7 from the genome:

AGATACA
GATACAG
ATACAGC
TACAGCC
...

k-mer definition

A k-mer is a word of size k

Exercise

List all **distinct** 4-mers and 5-mers of this set of reads:
CATAATGCA
TGCACATAA
CATAATGCA

Solution

CATAATGCA, TGCA**CATAA**, **CATAATGCA**

5-mers:

CATAA, ATAAT, TAATC, AATGC, ATGCA, TGCAC, GCACA, CACAT, ACATA

CATAATGCA, **TGCA**CATAA, **CATAATGCA**

4-mers:

CATA, ATAA, TAAT, AATG, ATGC, TGCA, GCAC, CACA, ACAT

Genome words / read words

Let's select a word from the genome:

AGATACAGCCATGACCG**TAGCATG**CTAACTGTGACGGCATTAC

in the genome, after **TAGCAT**

only **AGCATG** appears

Genome words / read words

In real cases we don't have the genome

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

in the genome, after **TAGCAT**

only **AGCATG** appears

but we have the reads

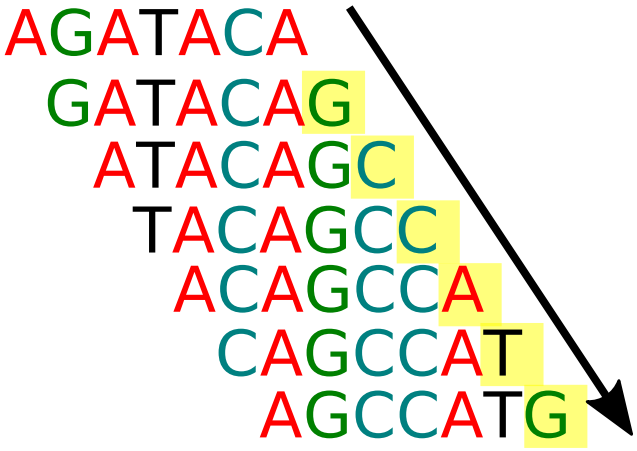
ATGACCGTAGCATGCT
ATGACCGTAGCATGCT
GACCGTAGCATGCTAA

in the reads, after **TAGCAT**

only **AGCATG** appears

Reconstitute larger genomic words

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC



AGATACA + G + C + C + A + T + G

AGATACAGCCATG a sequence from the genome

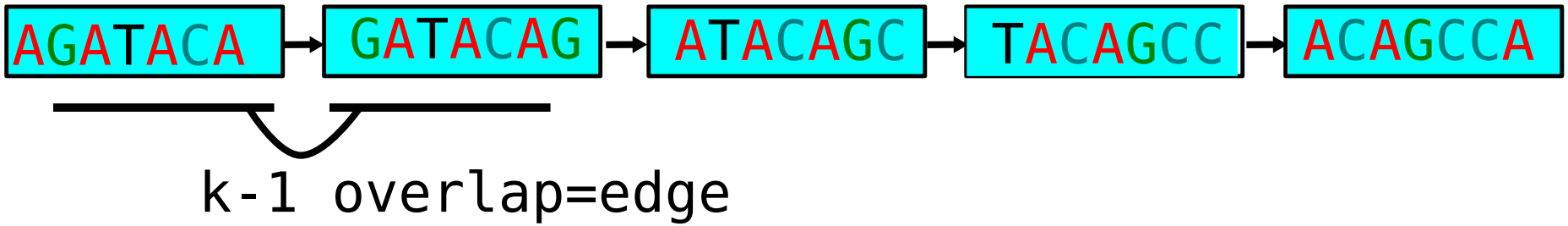
The De Bruijn graph

Read

AGATACAGCCA

De Bruijn graph

Kmer=node



AGATACA + G + C + C + A

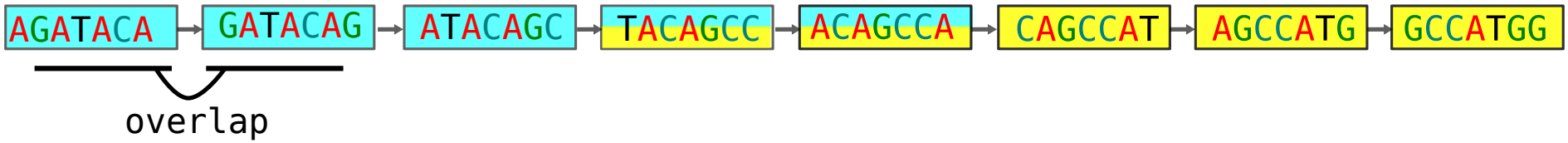
=AGATACAGCCA

De Bruijn graph assembly

Overlapping reads

AGATACAGCCA
TACAGCCATGG

De Bruijn graph



Resulting sequence

AGATACAGCCATGG

De Bruijn graph time!

Reads

GCCATGGGTTT
TACAGCCATGG
AGCCATGGGTT
GCCATGGGTTT
AGATACAGCCA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
CAGCCATGGGT

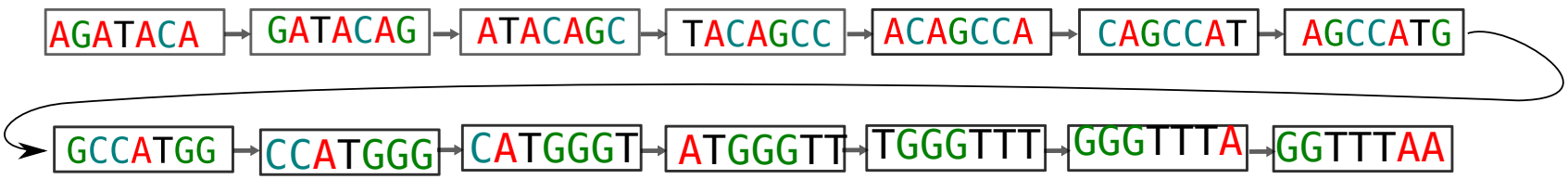
Hint: Use 7-mers

Solution

Overlapping reads

```
AGATACAGCCA
 GATACAGCCATG
 GATACAGCCATG
   TACAGCCATGG
    ACAGCCATGGG
    ACAGCCATGGG
     CAGCCATGGGT
      AGCCATGGGTT
       GCCATGGGTTT
       GCCATGGGTTT
         CATGGGTTTAA
         CATGGGTTTAA
```

De Bruijn graph



Resulting sequence

```
AGATACAGCCATGGGTTTAA
```

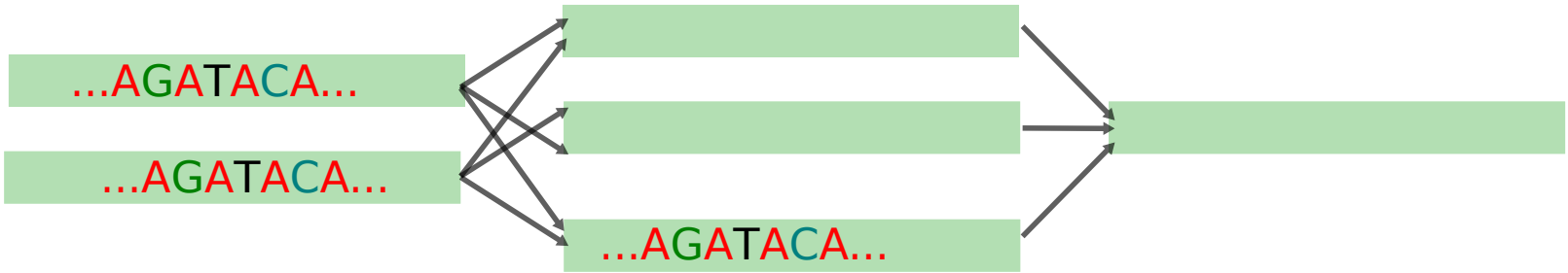
De Bruijn graph versus overlap graph

AGATACA
GATACAG TACAGCC AGCCATG
ATACAGC ACAGCCA CAGCCAT
...

words from the reads

word graph (De Bruijn graph)

Overlap graph from the reads



De Bruijn graphs abstract redundancy

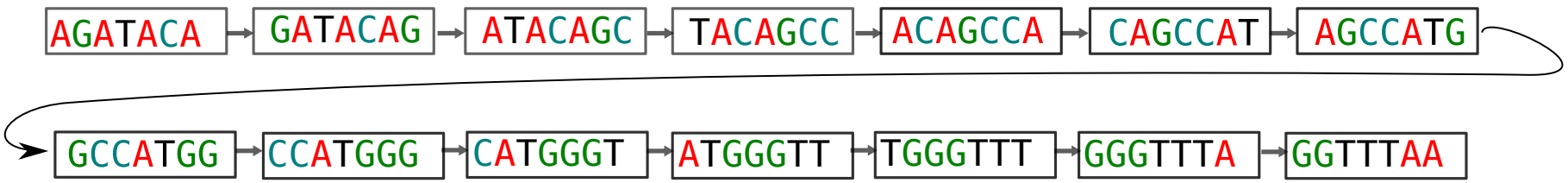
Overlapping reads

```
AGATACAGCCA
 GATACAGCCATG
 GATACAGCCATG
   TACAGCCATGG
    ACAGCCATGGG
    ACAGCCATGGG
     CAGCCATGGGT
      AGCCATGGGTT
       GCCATGGGTTT
       GCCATGGGTTT
         CATGGGTTTAA
         CATGGGTTTAA
```

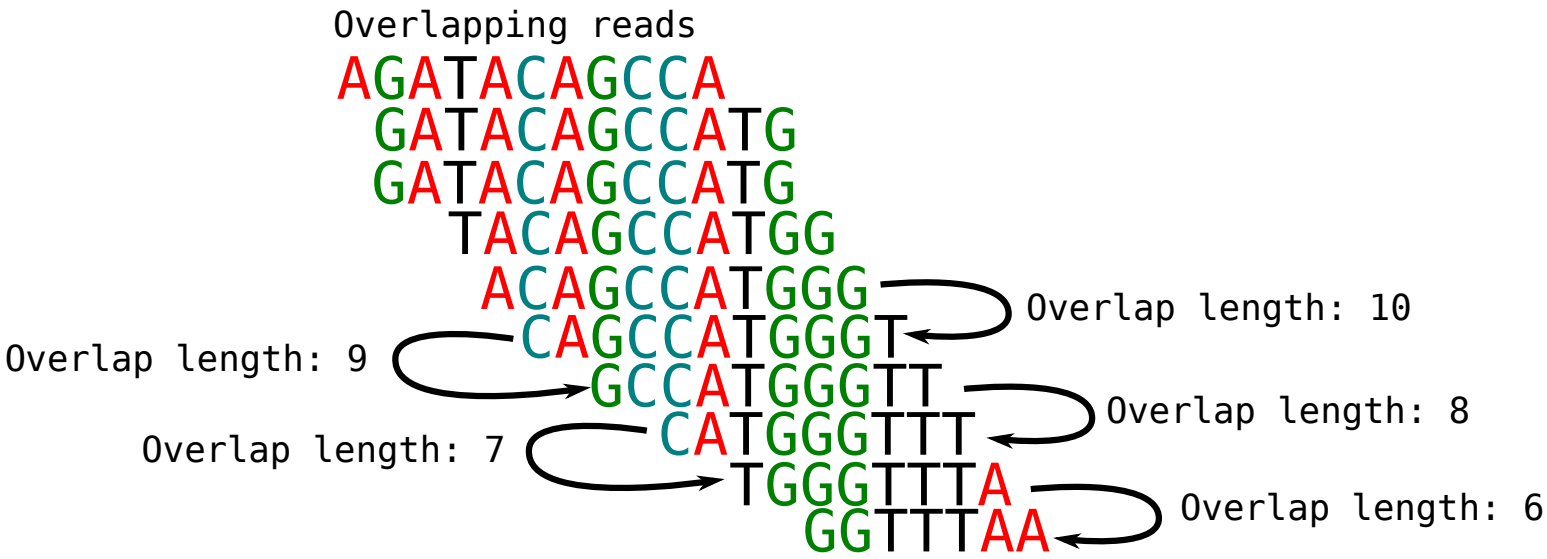
62 **(non distinct)** 7-mers in the reads

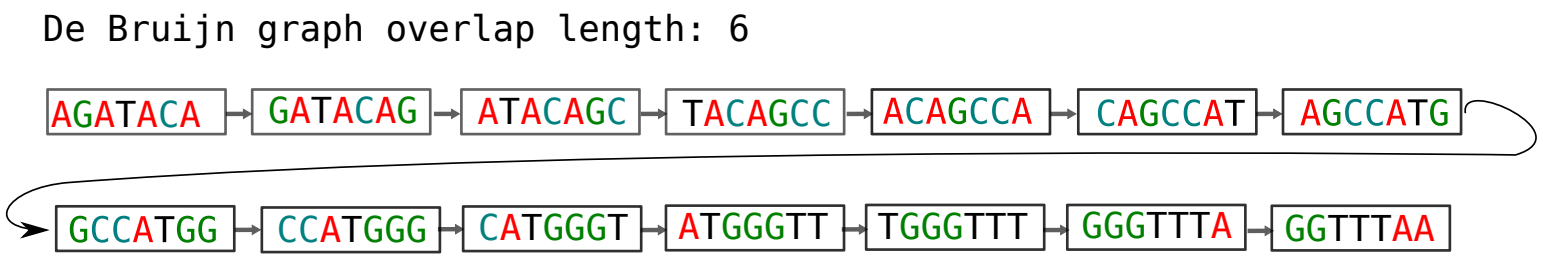
De Bruijn graph

14 **distinct** 7-mers in the De Bruijn graph



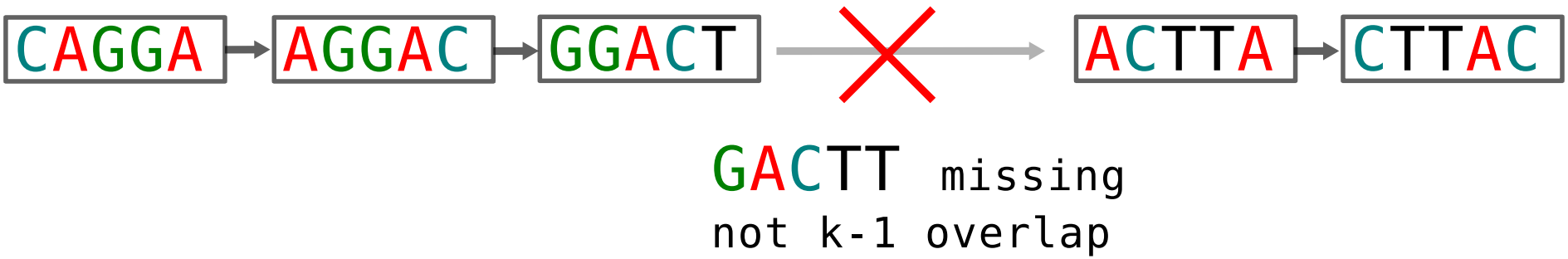
De Bruijn graphs only rely on $k-1$ overlaps





De Bruijn graphs limitation

Fixed overlaps

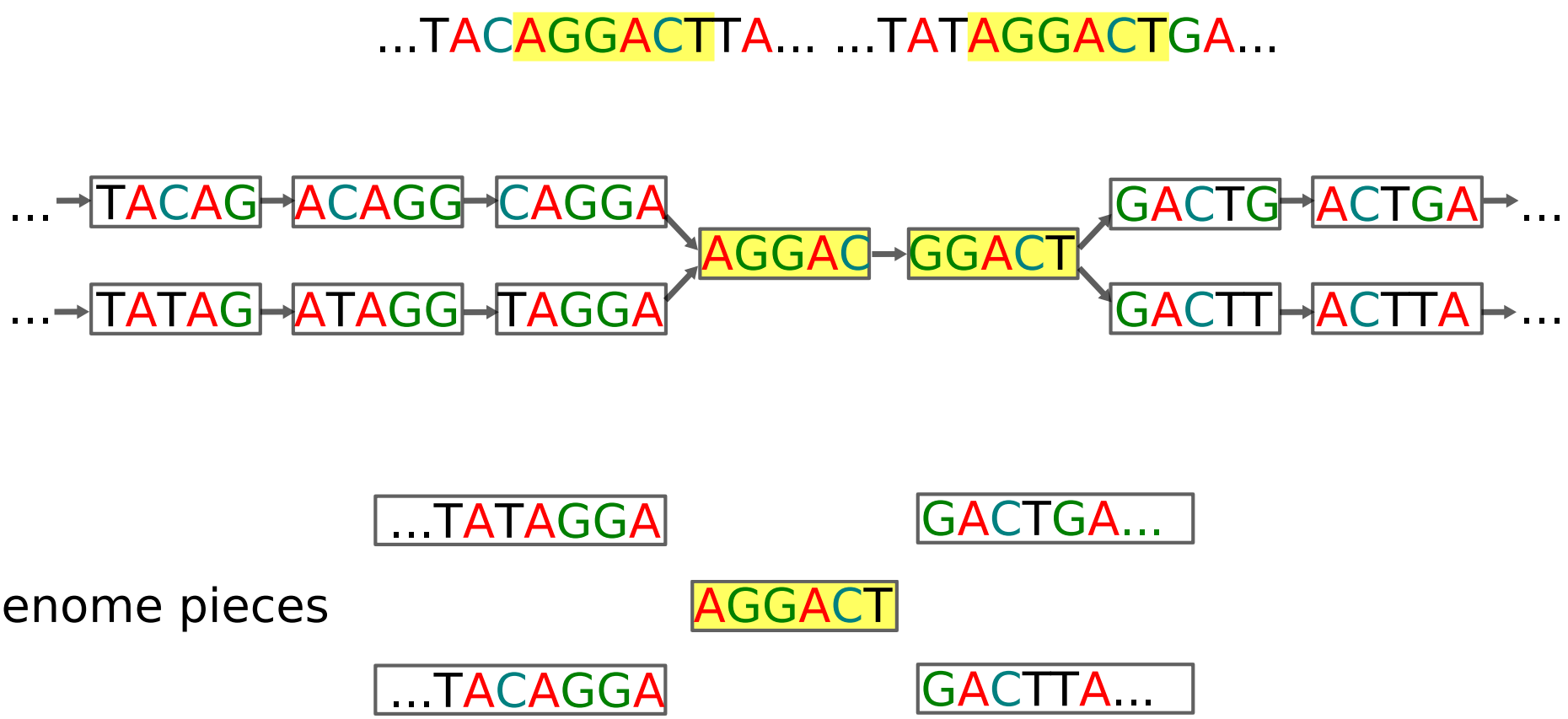


De Bruijn graphs limitation

Repeats...

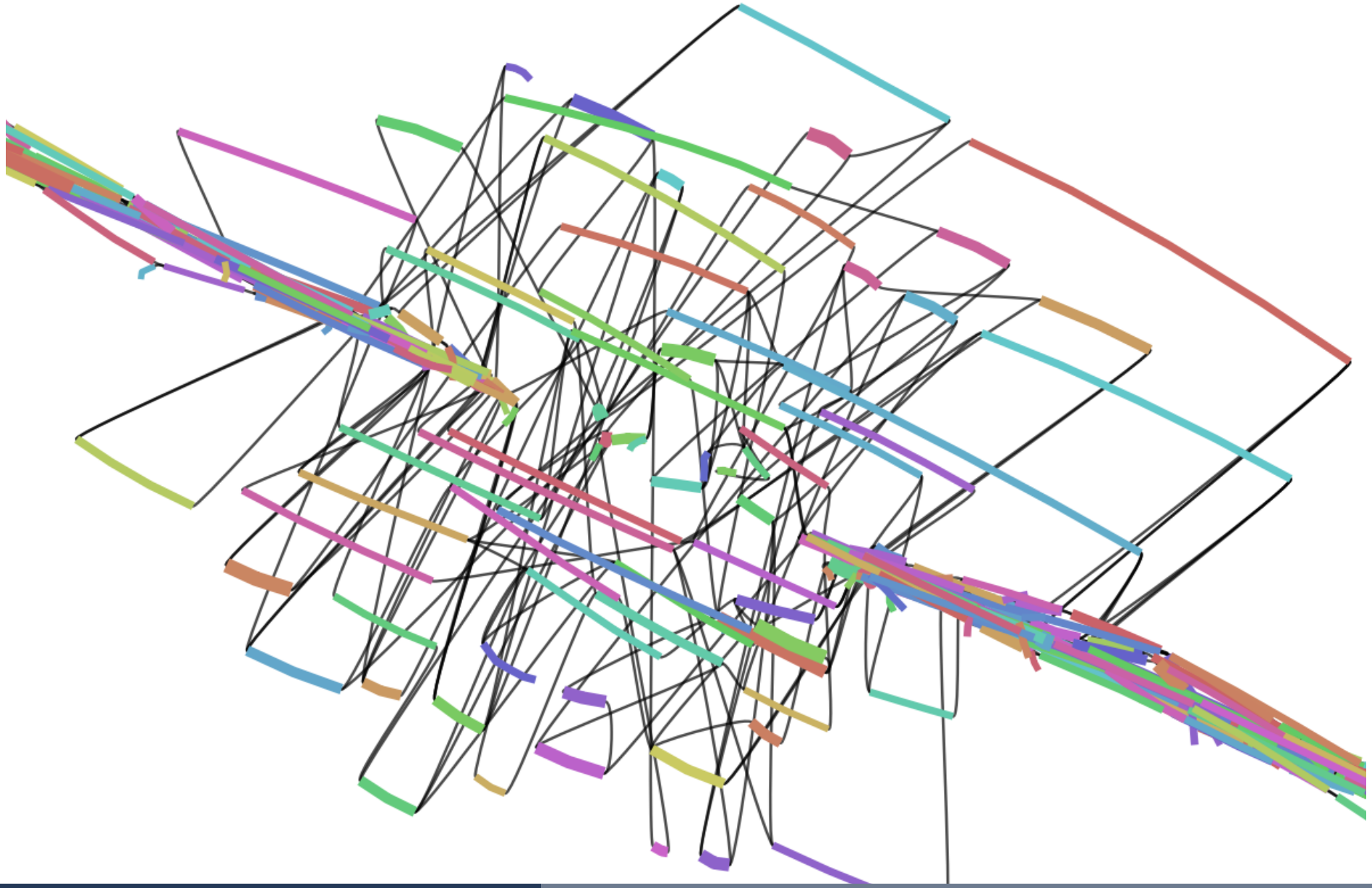
...TACAGGACTTA... ...TATAGGACTGA...

De Bruijn graph limitation



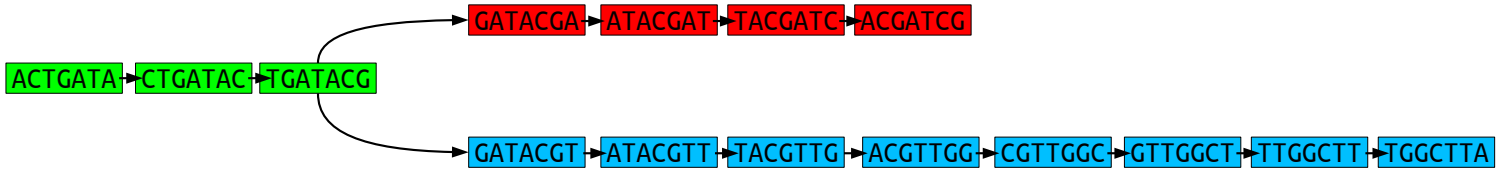
De Bruijn graph on a real dataset

De Bruijn graph on real dataset ZOOMED IN

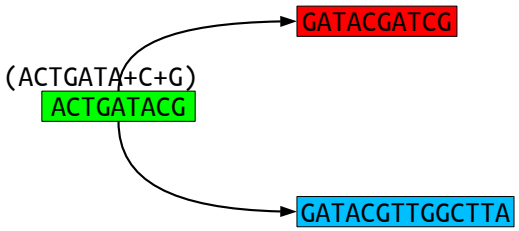


On the representation of De Bruijn graphs

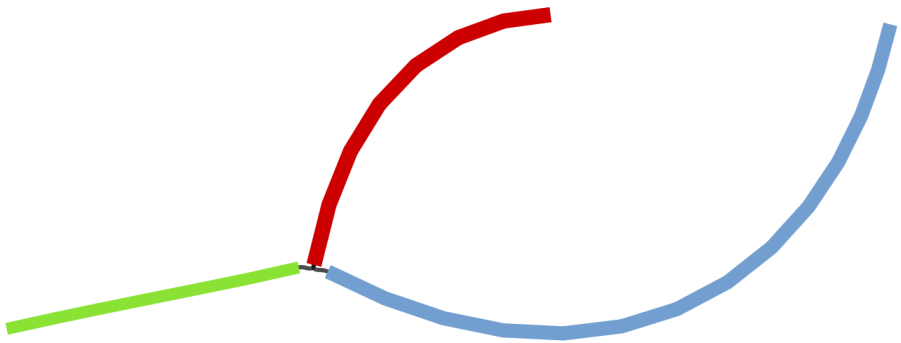
De Bruijn graph:



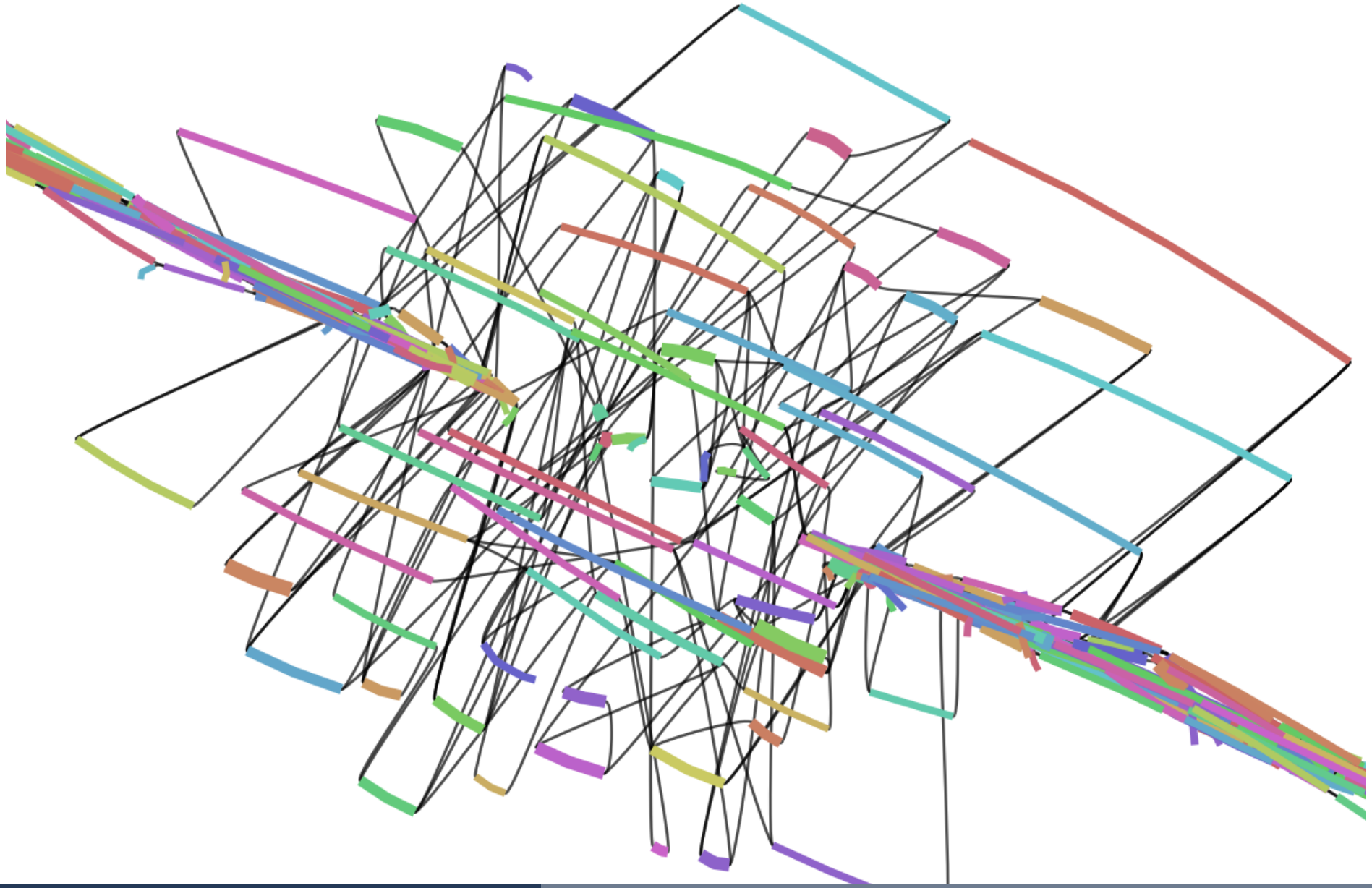
Compacted De Bruijn graph:



Graphical representation
(.gfa plot using Bandage):



De Bruijn graph on a real dataset ZOOMED IN



Dealing with sequencing errors

```
Genome:
 ATCGGTATCGTTACGGTATACC

Reads:
 ATCGCTATCG
    GGTTTCGTTA
       ATCGATACGG

TCGCTA
GGTTTC
 ATCGAT
...
Are not genomic kmers...
```

Erroneous k-mers vs genomic k-mers

Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACA

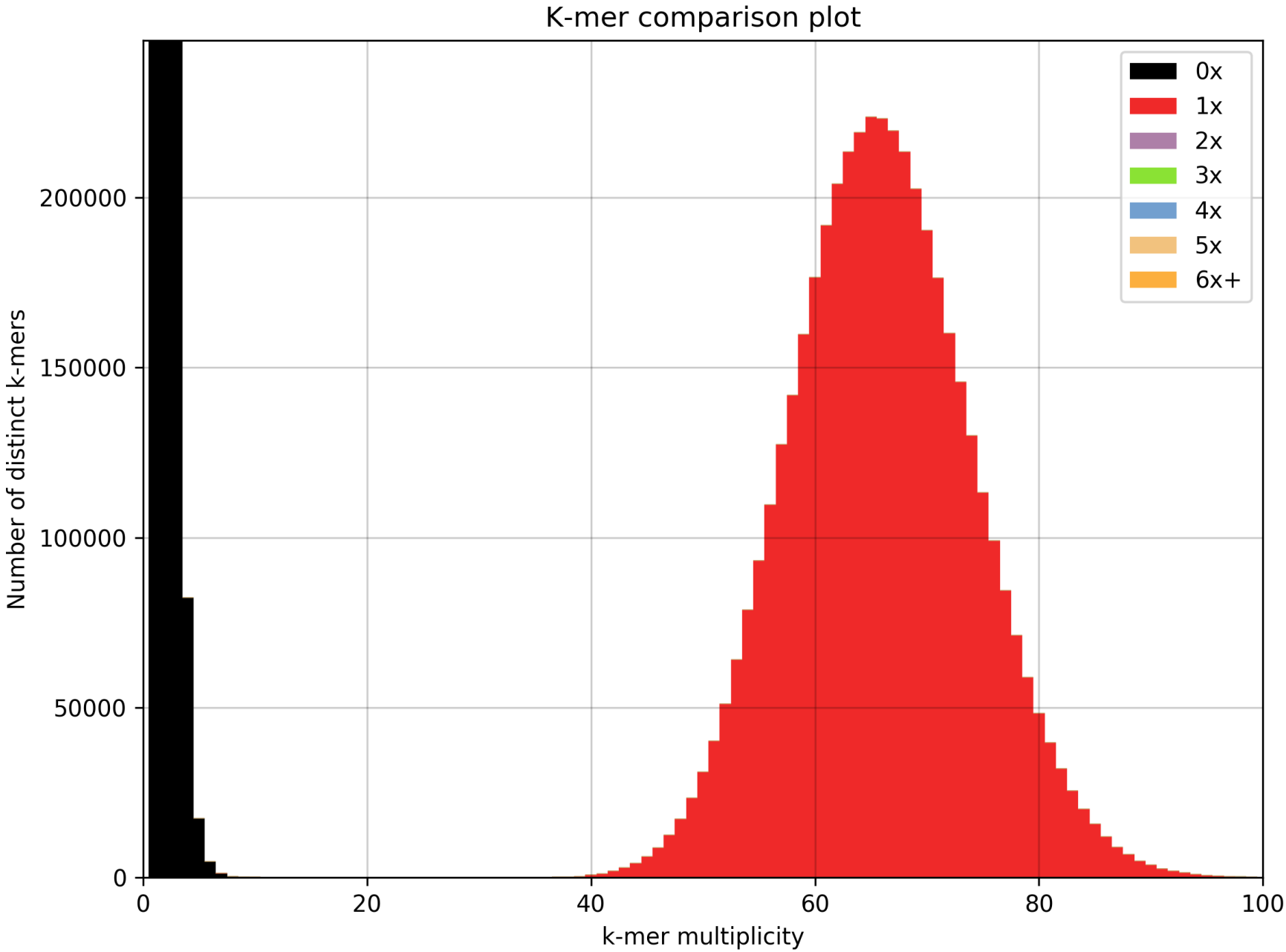
Reads:

9 times TCTGAAT
1 time TCTAAAT

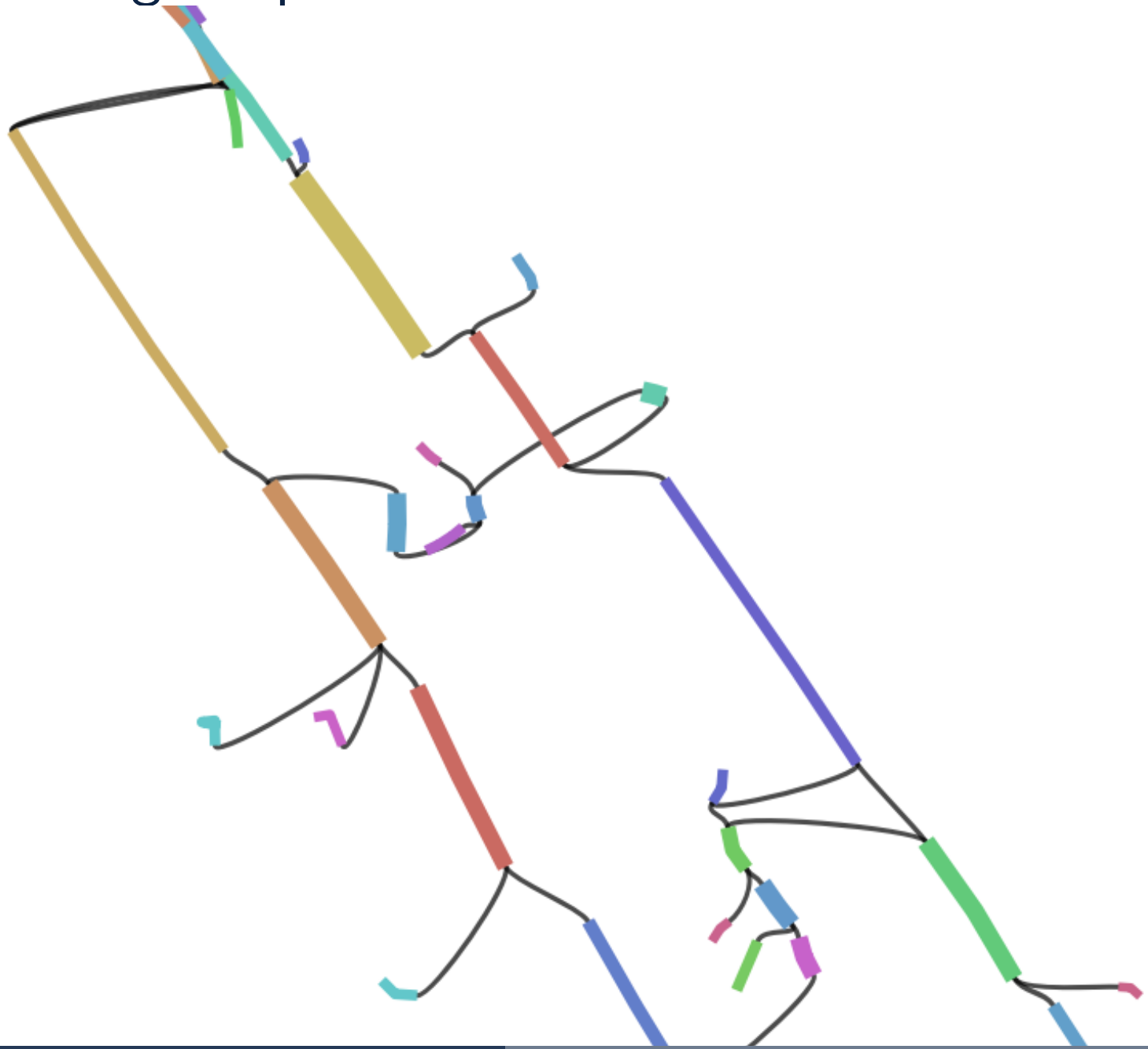
6 times CAACGGA
1 time CAACGGT

Erroneous k-mers are seen less than genomic ones

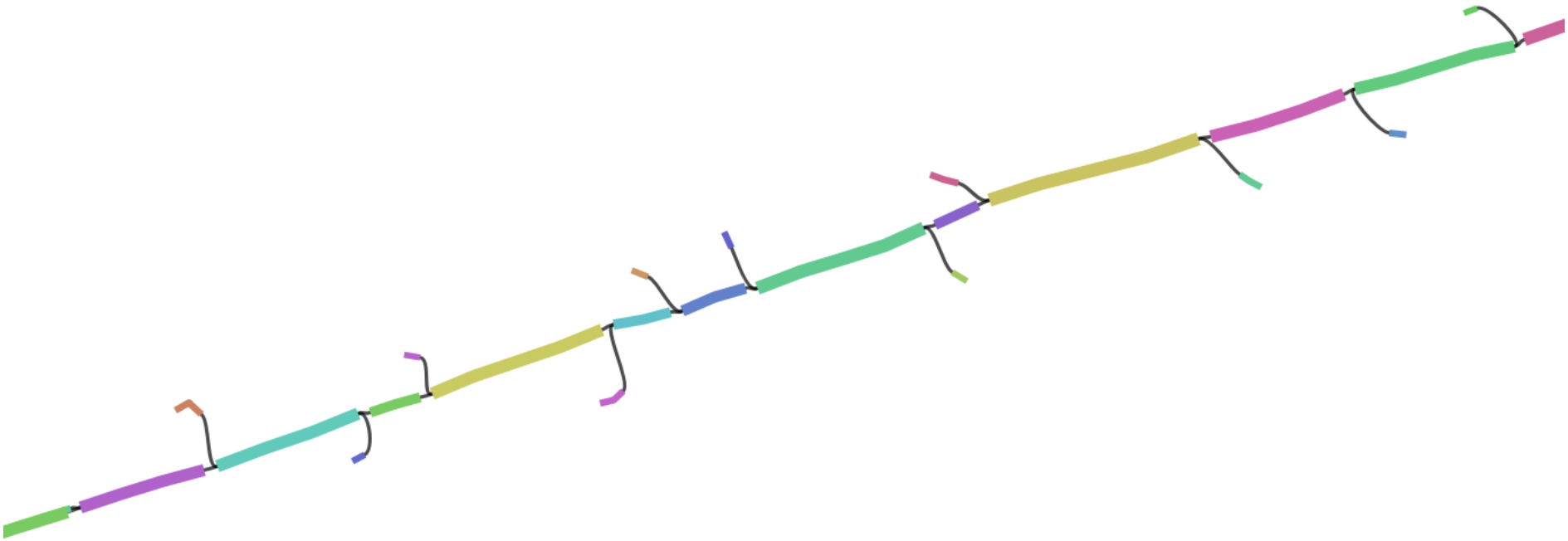
K-mer histogram



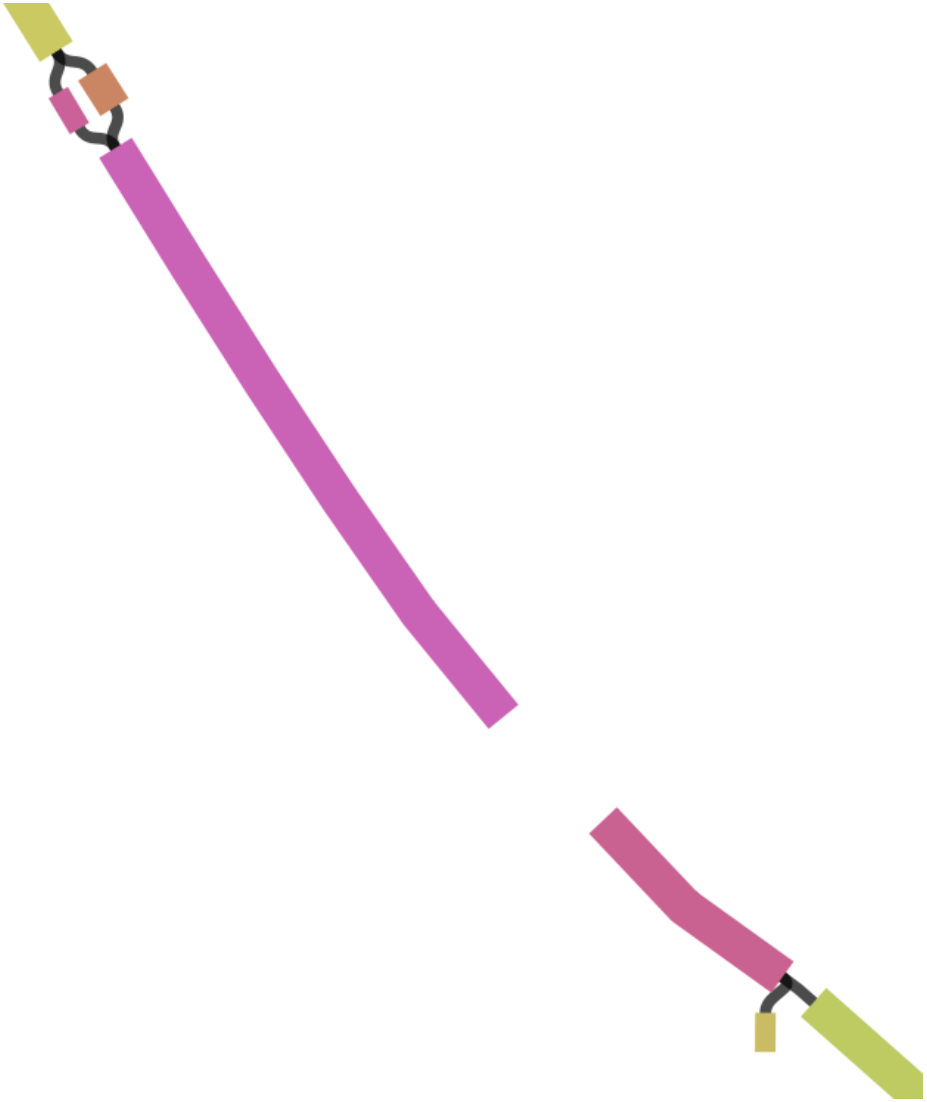
Removing unique k-mers



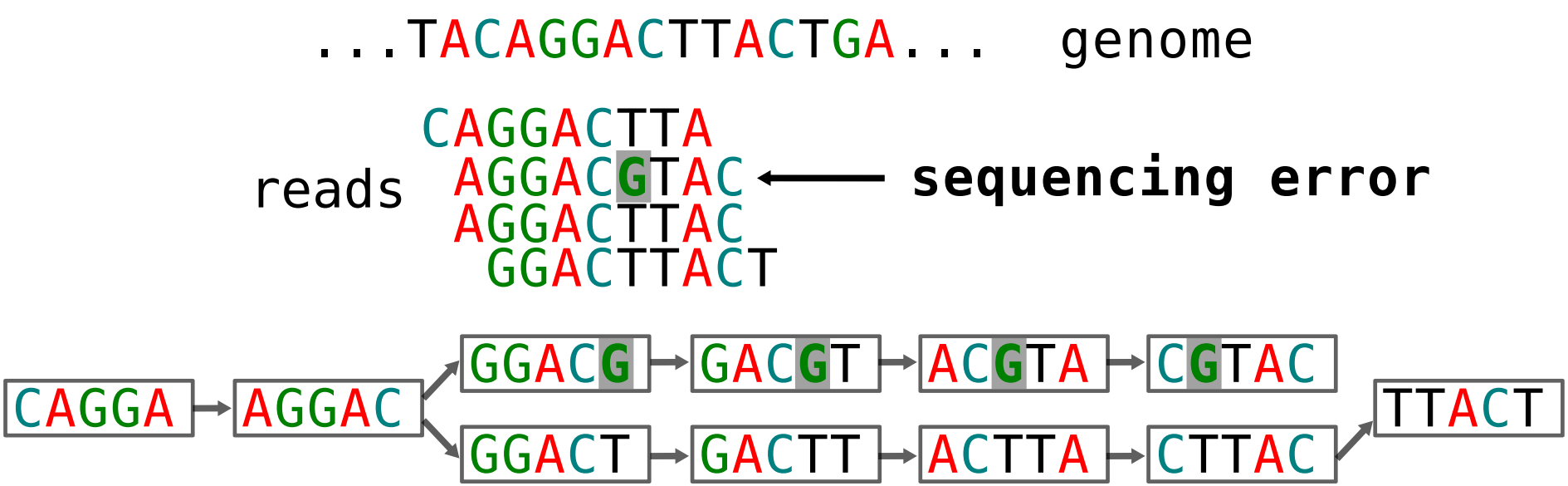
Removing k-mers seen less than 3 times



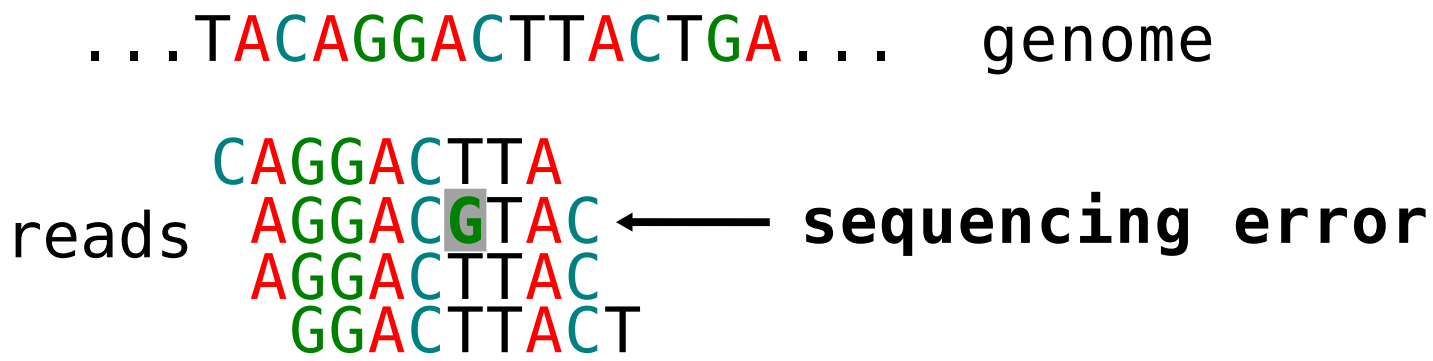
Removing k-mers seen less than 4 times

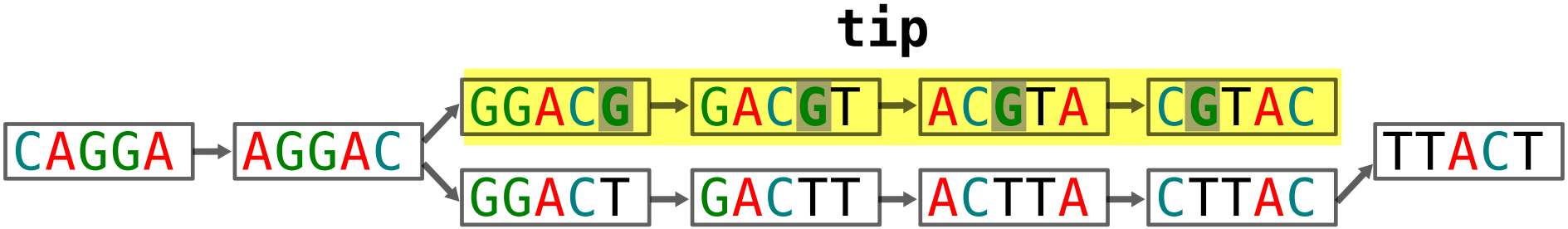


Errors in De Bruijn graphs

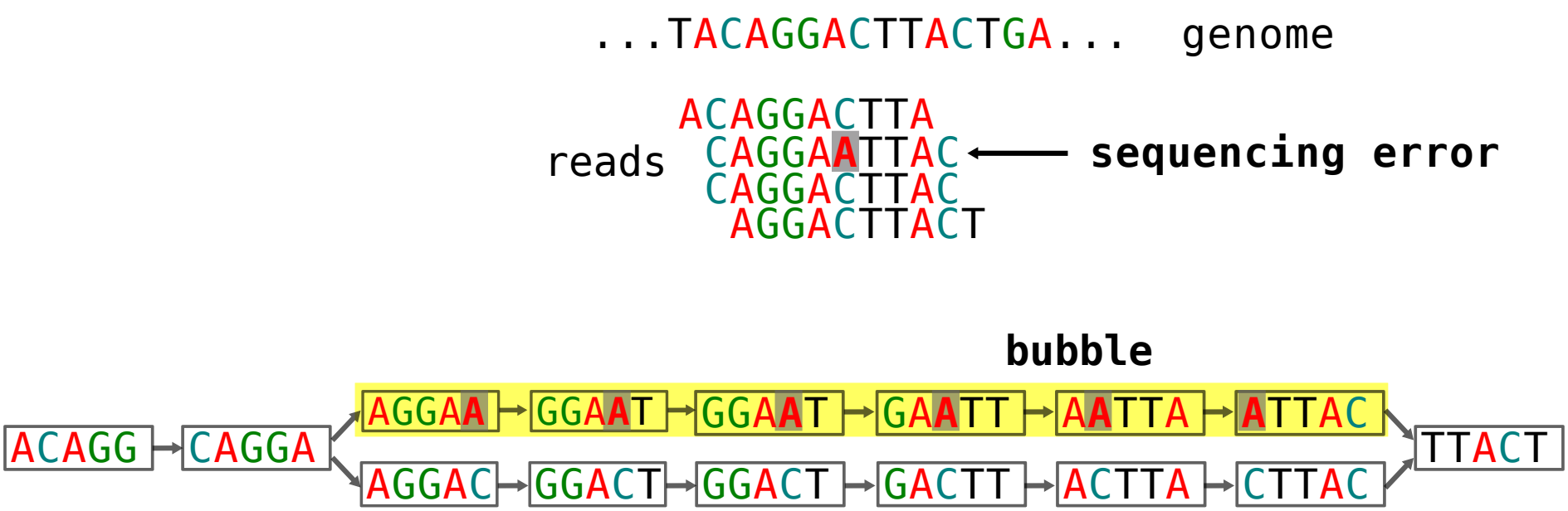


Errors in De Bruijn graphs

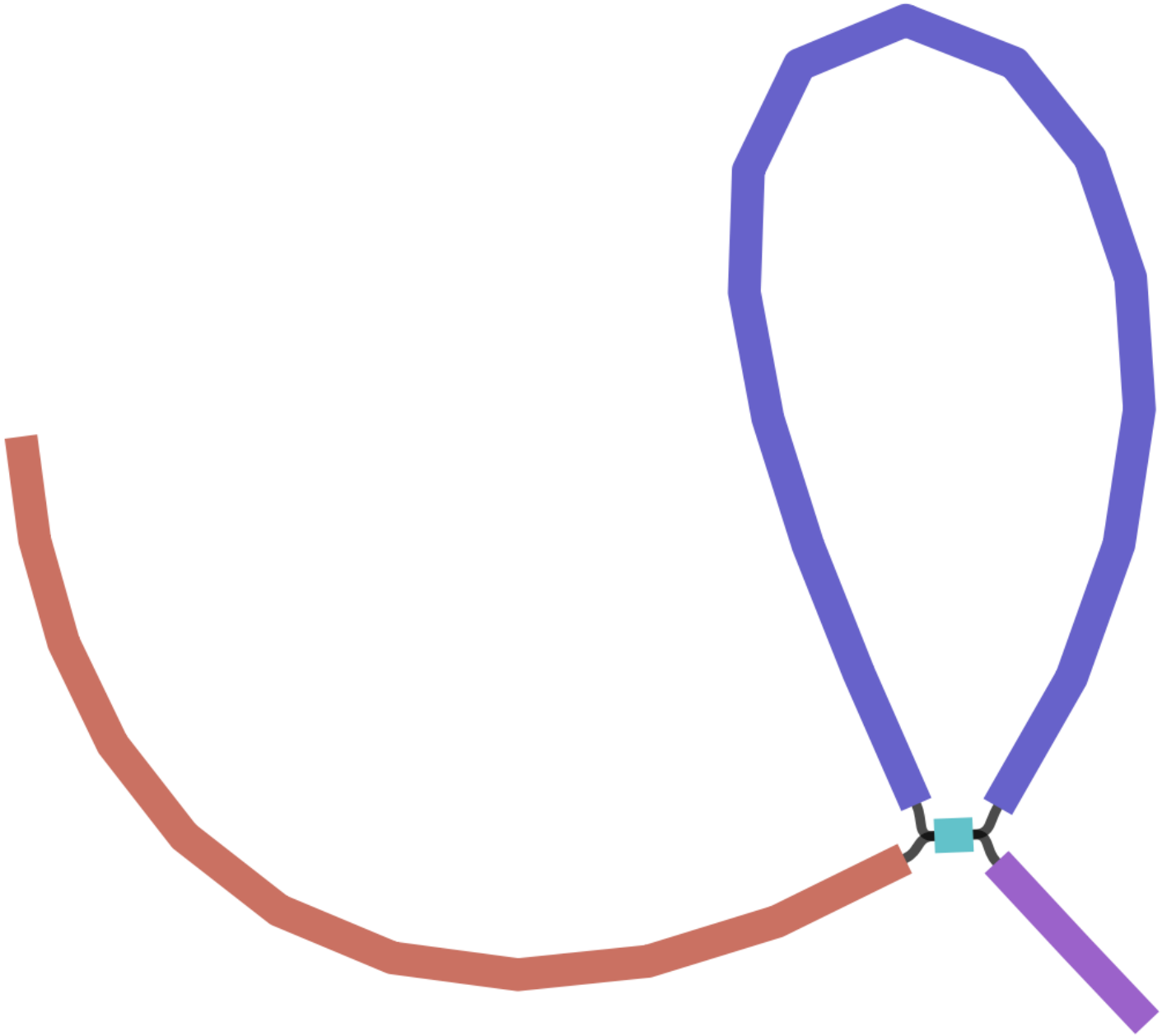




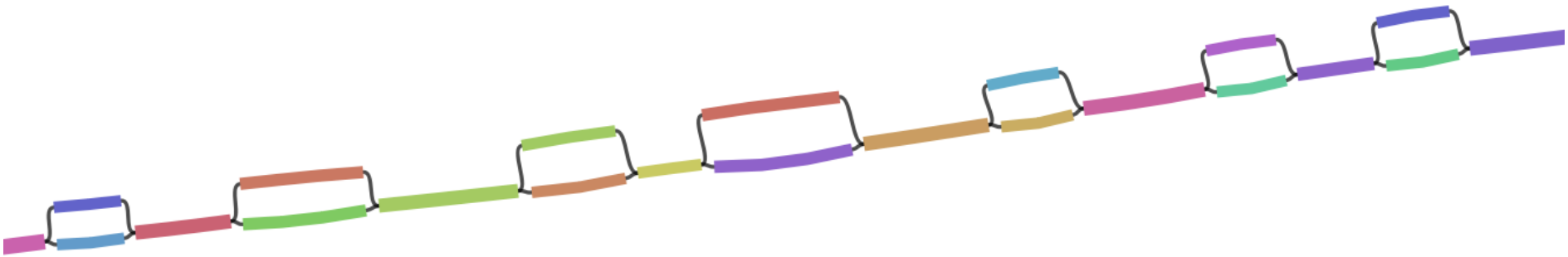
Errors in De Bruijn graphs



(Almost assembled phage !)



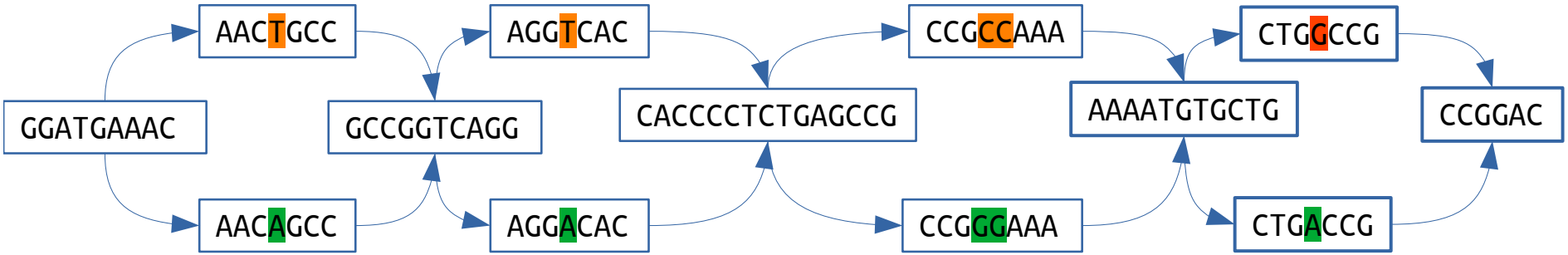
De Bruijn graph on an eukaryota



Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

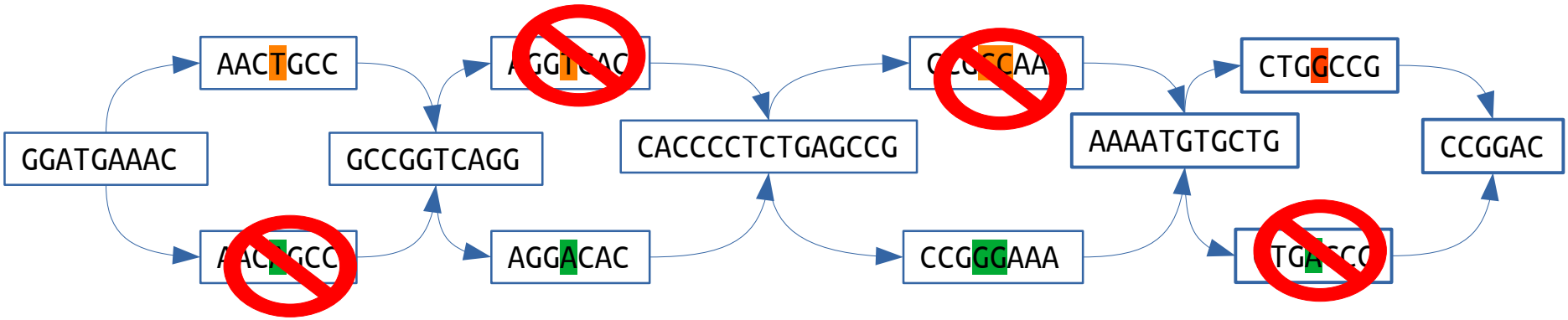
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

Assembly **concession number 2**: collapse variability

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

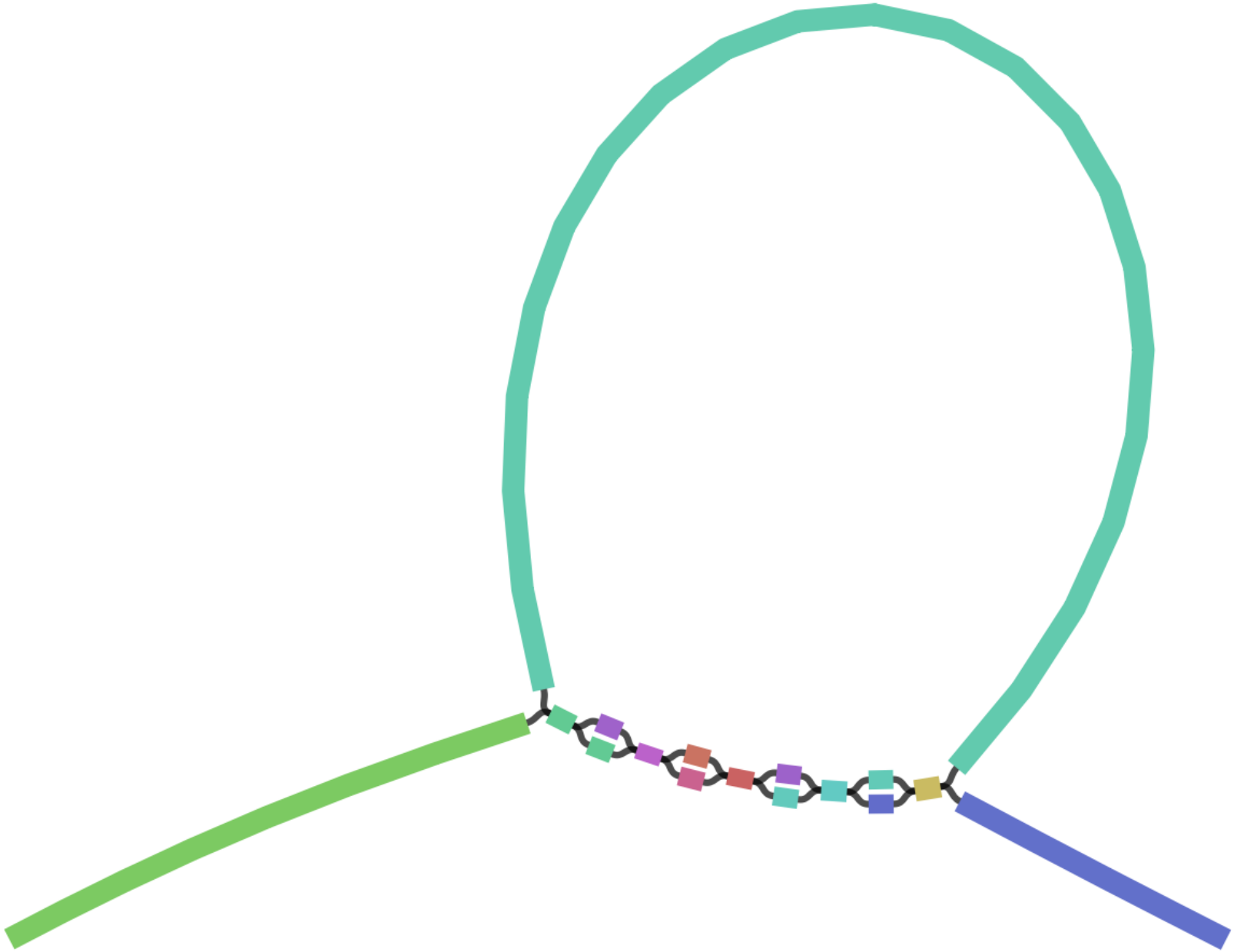
Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

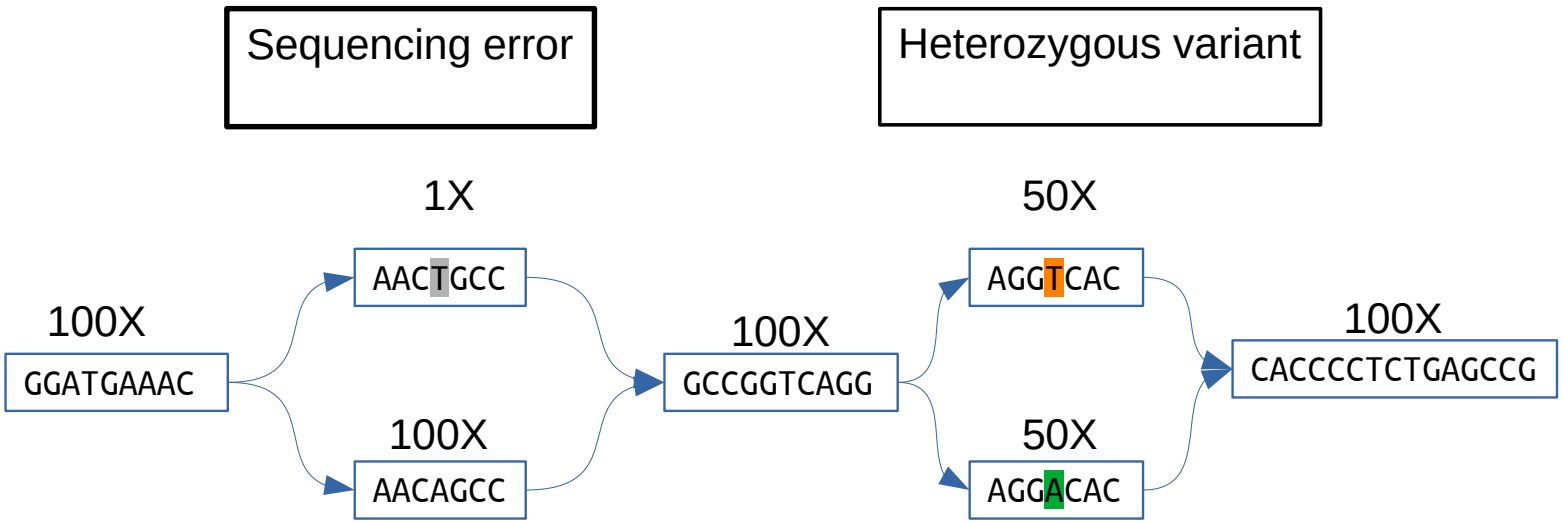
Reads:

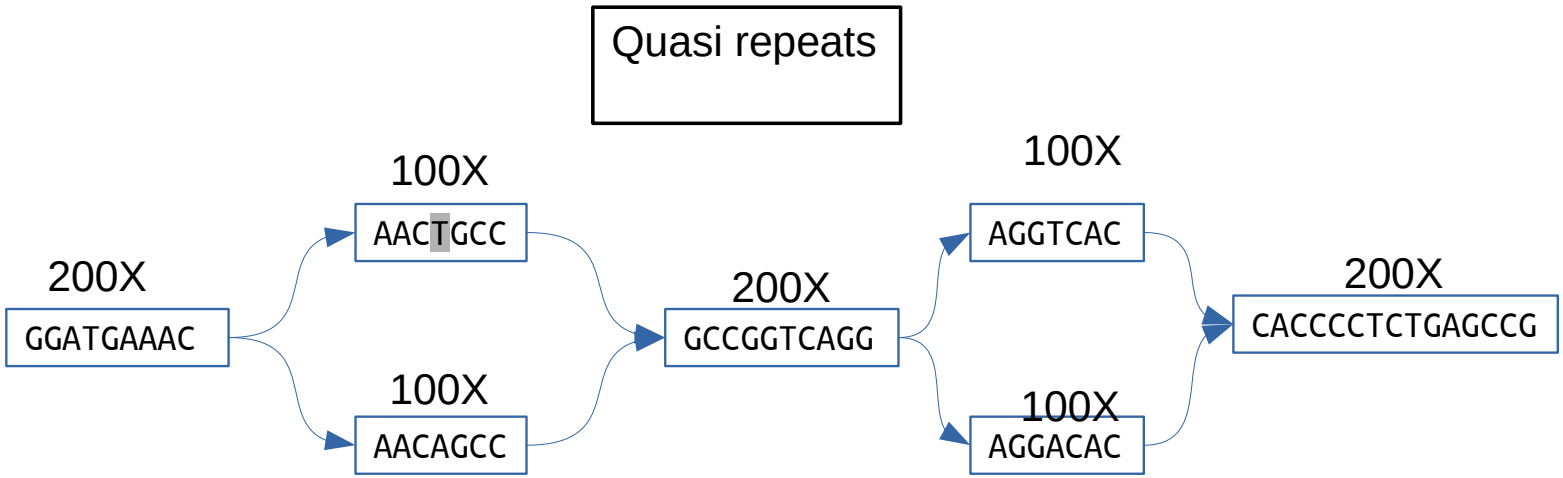
```
GATGAAACTG
 ATGAAACAGC
  TGAAACAGCCG
   GAAACTGCCGG
    AAACTGCCGGT
     AACAGCCGGTC
      ACAGCCGGTCA
       CTGCCGGTCAG
```

Paralog genes/repeats



Paralog genes/repeats in graph





An assembler is a set of heuristics

Graph cleaning heuristics

- Nodes coverage
- Graph local/global topology
- Reads that can be mapped on nodes
- Estimated coverage/genome size
- ...

An assembly is a model

Different tools can produce very similar assemblies

A single tool can produce very different assemblies with small changes of parameters(!)

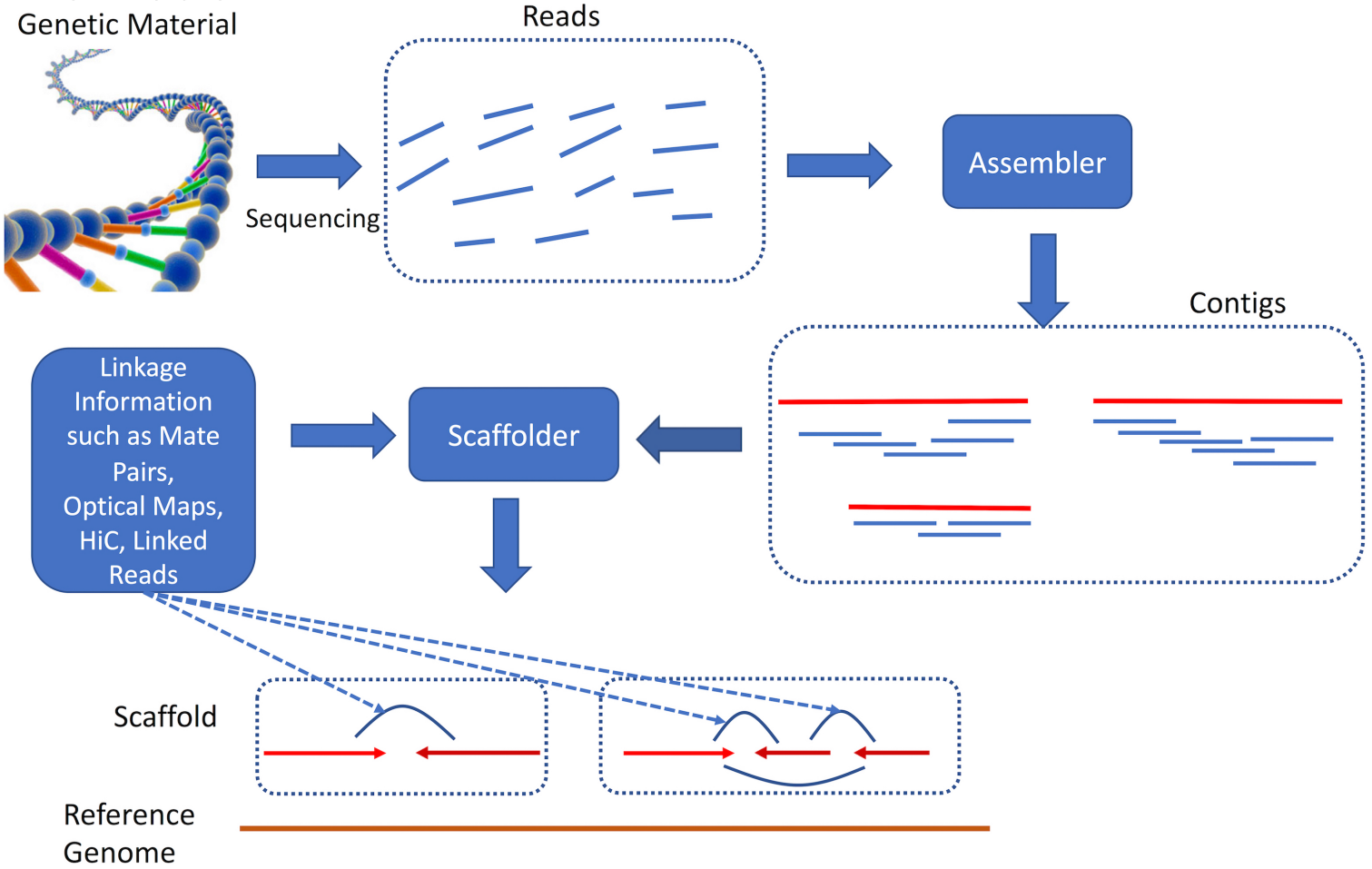
De Bruijn graphs in a nutshell

- Graph of words of size k, k-1 overlaps
- Collapses identical k-mers
- Very successful, have replaced the overlap graphs with high throughput sequencing data
- Still outputs fragments of the genome

white spruce, 20 gigabases

Scaffolding

Softwares can improve the assembly continuity by using other kinds of information



From "Modern technologies and algorithms for scaffolding assembled genomes" Plos Computational Biology

Second generation sequencing

NextSeq Series

HiSeq 4000 System

HiSeq X Series‡

NovaSeq 6000 System

- Short reads $\approx 150bp$
- Low error rate $\approx 1\%$
- High throughput (up to billion of reads per run)
- GC bias

Mainly assembled using De Bruijn graphs

State-of-the-art

Your toolkit for the practical session

- SPAdes
- Megahit
- Minia

Other notable assemblers

- SGA
- Discovar denovo
- Abyss

Third generation sequencing

- Long reads $\approx 10 - 100kbp$
- High error rate $\approx 10 - 12\%$
- High throughput (up to millions of reads per run)

Nanopore VS Pacbio

Nanopore

- Portable
- Ultra long reads (100kbases, some reads reach the megabase level)
- Mostly deletions

Pacbio

- More mature
- HiFi reads (99% identity)
- Mostly insertions

Long reads killed the assembly star

Laura Landweber @LandweberLab · Jan 2

Our newest version of Oxytricha's somatic genome is out (rdcu.be/bZNfC) and has 18,617 distinct chromosomes. That's 2000 more than we previously published in doi.org/10.1371/journa.... PacBio captured most chromosomes in single reads: Genome sequence, No assembly required

Repeats spanning

Genome:

GGTAATGGTTTTTTGGTGCTAATGCGTTTTTTCATGGATGTCGTAATTTTTTATCTG

Reads:

```
GGTAATG  TTTTTT  GTGCTAAT  GTTTTTT  ATGGATG      TTTTTTA
    ATGGTTT          AATGCGTT          ATGTCGT      TTTATCTG
           TTTGGTG                TTTTCATG      CGTAATTT
```

Contexts of the repeat:

```
...ATGG                                  ATCT...

                  ???TTTTTT???           GGTG...
...TGCG

                                         CATG...
...GTAA
```

Repeats spanning

Genome:

GGTAATGGTTTTTTGGTGCTAATGCGTTTTTTCATGGATGTCTGAATTTTTTATCTG

Reads:

GGTAATG TTTTTT GTGCTAAT GTTTTTT ATGGATG TTTTTTA
ATGGTTT AATGCGTT ATGTCTG TTTATCTG
TTTGGTG TTTTCATG CTGAATTT

Long reads:

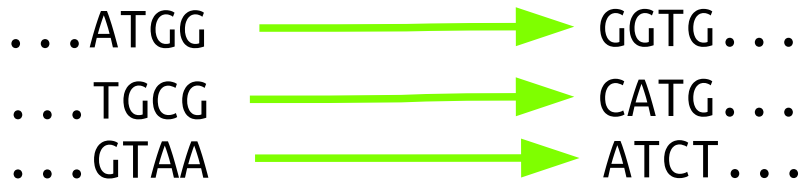
TGGTTTTTTGGT TGCCTTTTTTCAT TGAATTTTTTATCT

Contexts of the repeats:

...ATGG → GGTG... ...TGCG → CATG... ...GTAA → ATCT...

Repeats spanning

Contexts of the repeats:



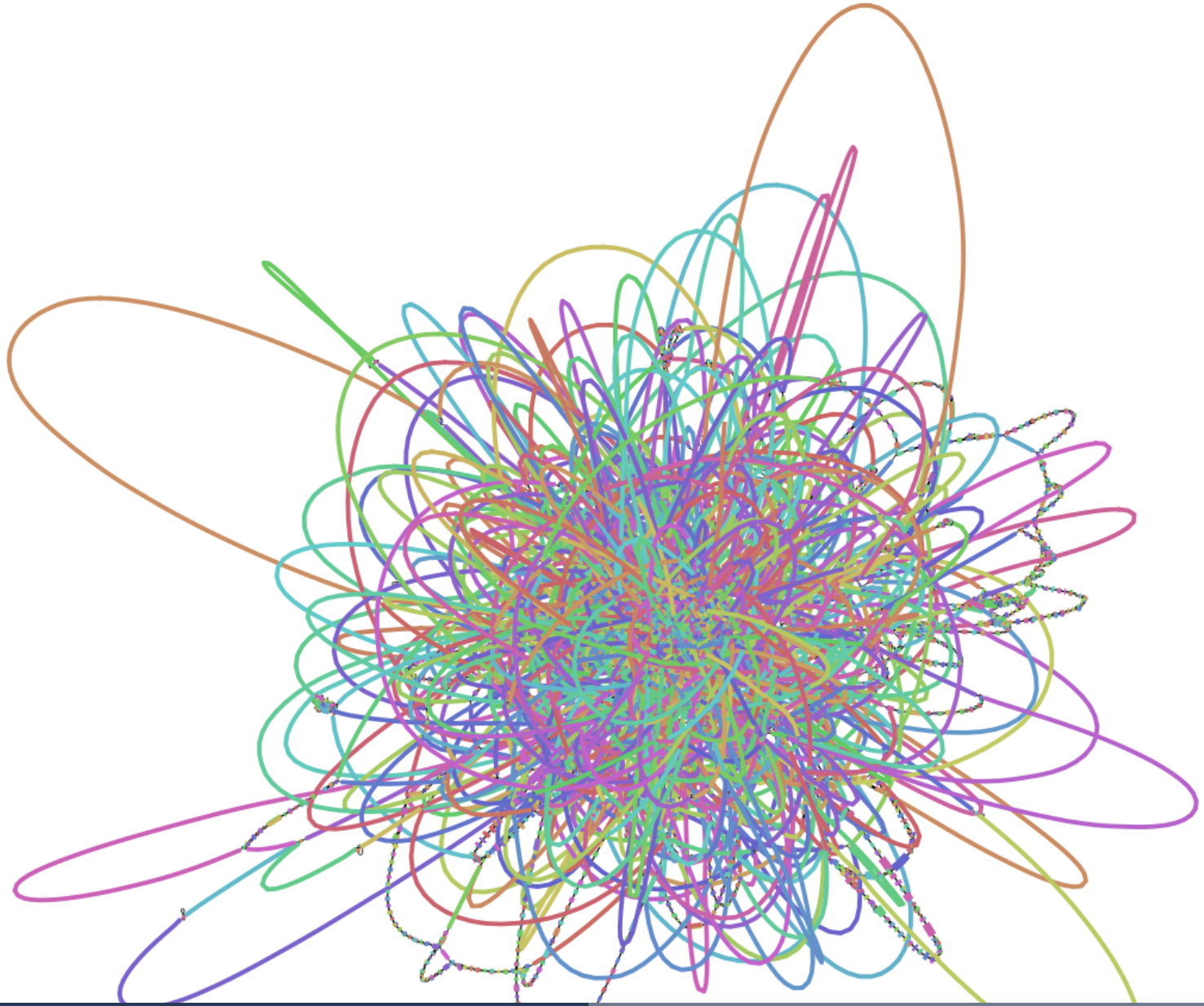
Overlapping Reads:

GGTAATG GTGCTAAT
ATGGTTT AATGCGTT ATGTCTG
TGGTTTTTTGGT TGCCTTTTTTCAT CTGAATTT
TTTGGTG TTTTCATG TGAATTTTTTATCT
ATGGATG TTTATCTG

Output assembly:
GGTAATGGTTTTTTGGTGCTAATGCGTTTTTTCATGGATGTCTGAATTTTTTATCTG

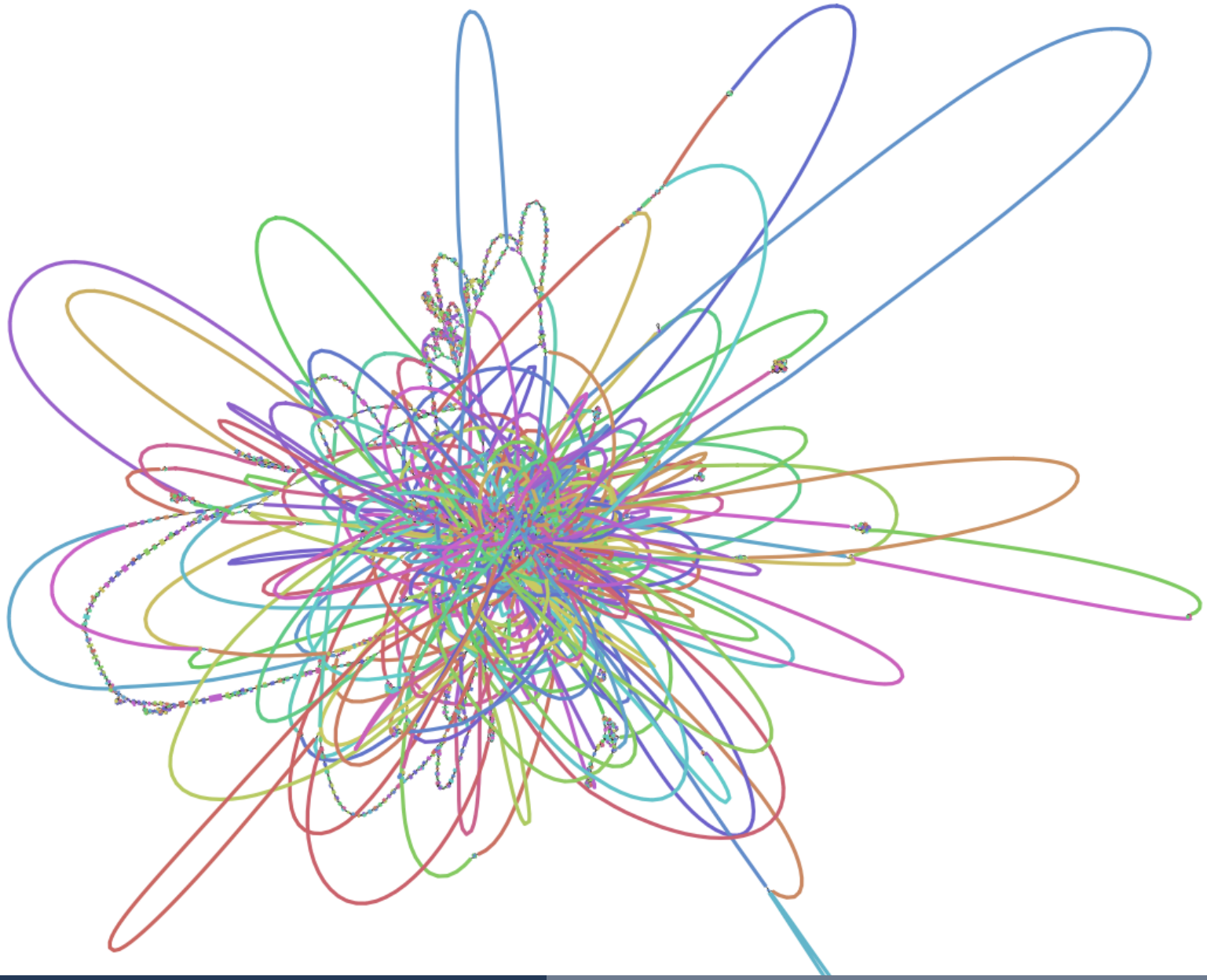
Read length matter

Read size=21



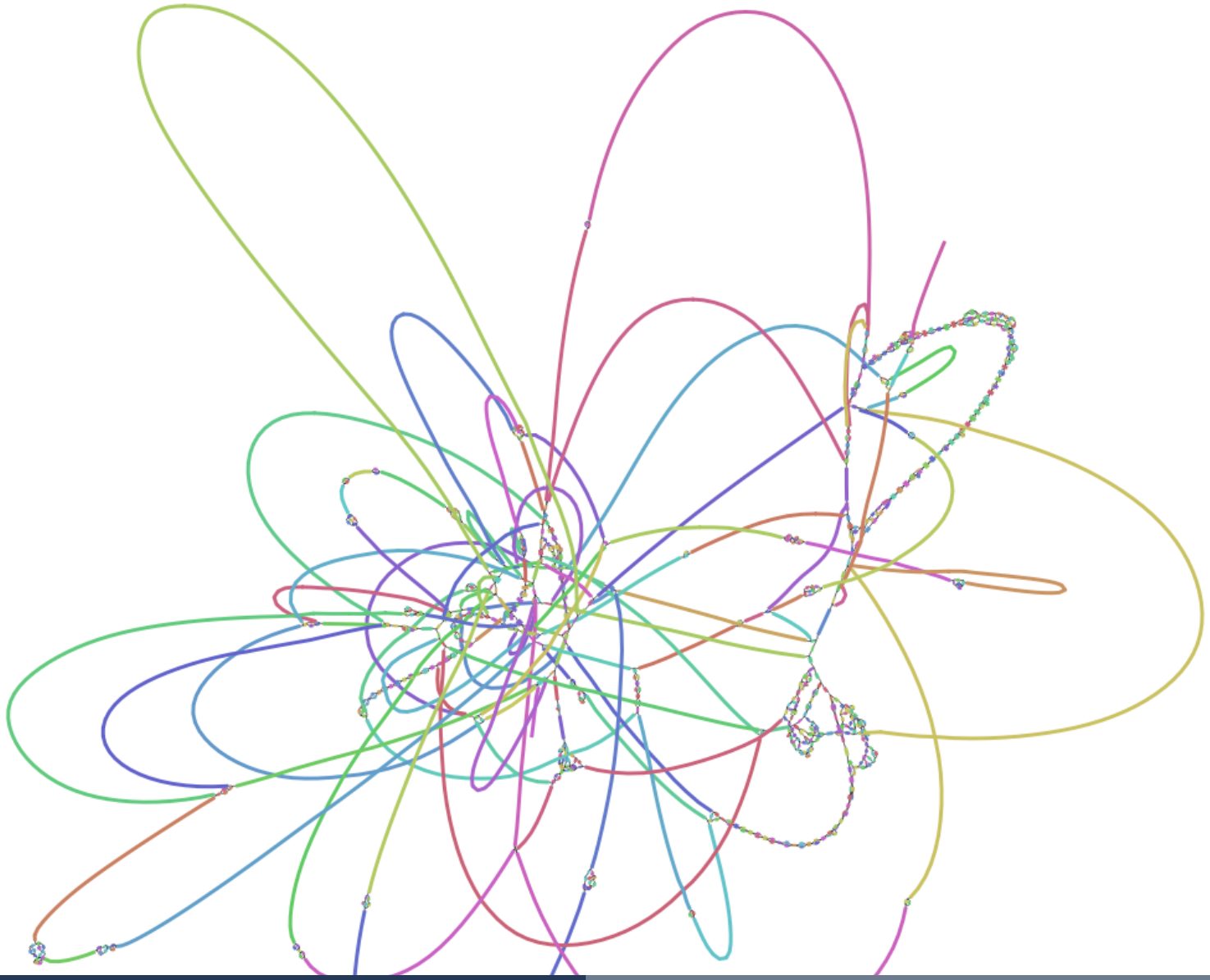
Read length matter

Read size=31



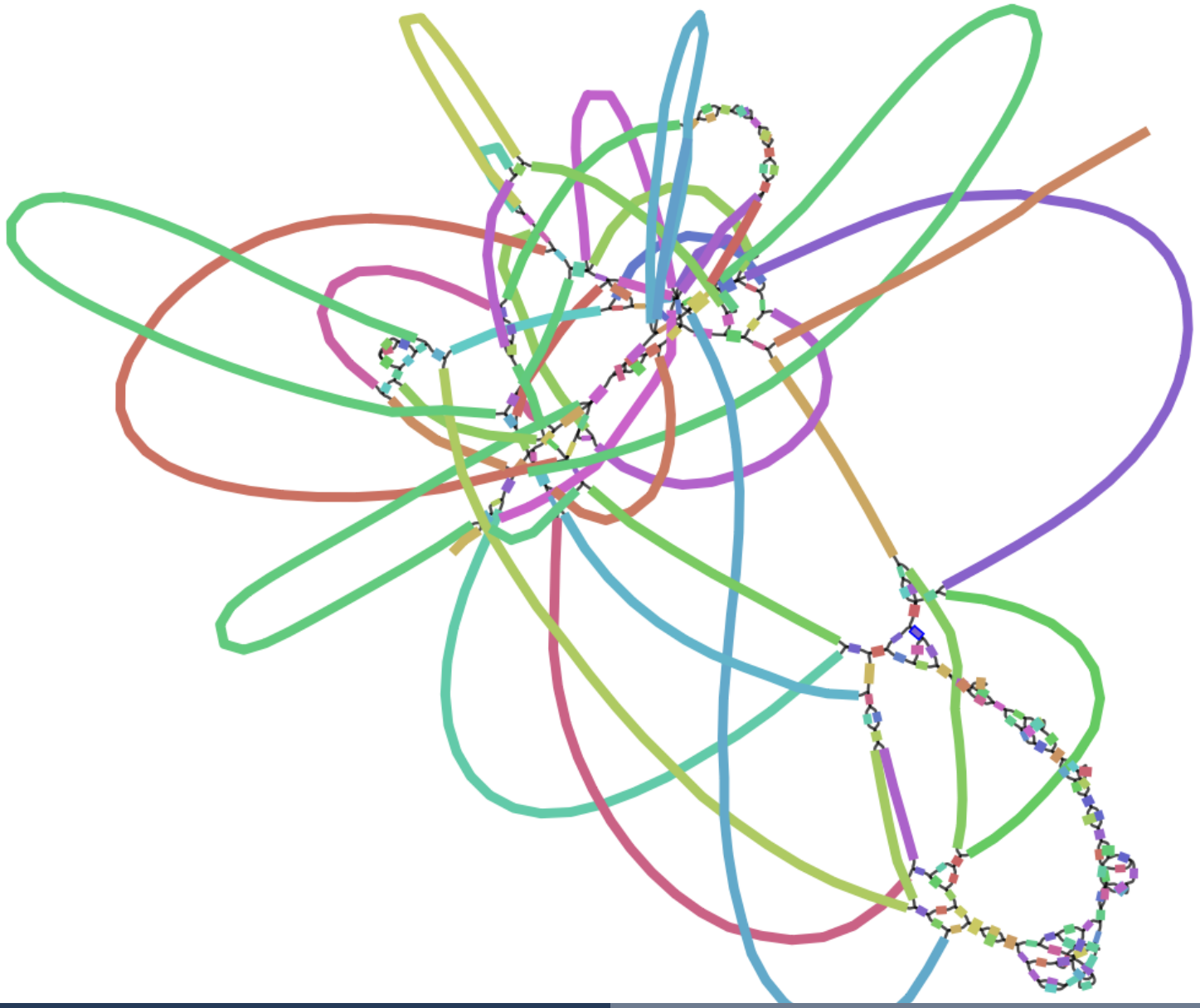
Read length matter

Read size=63



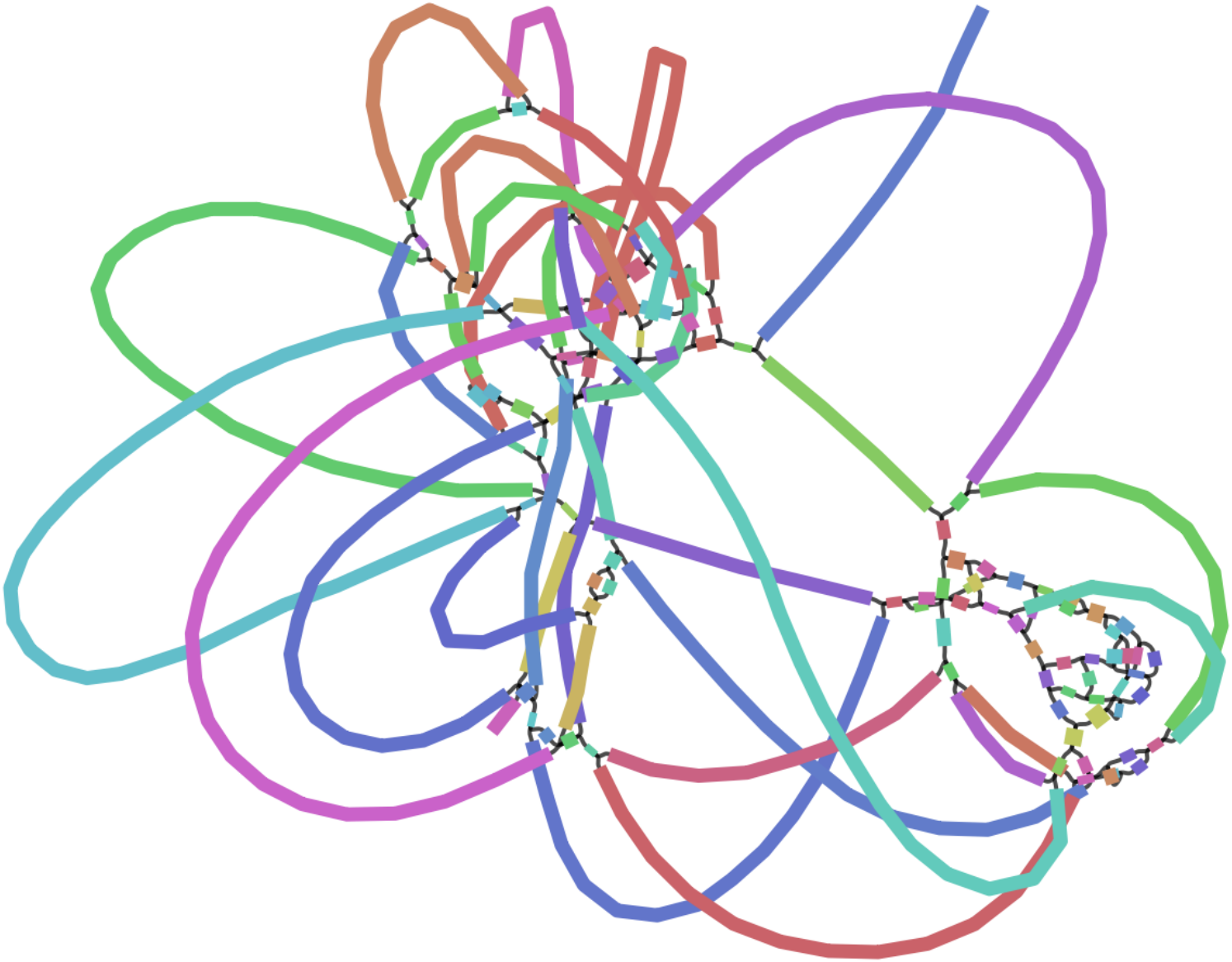
Read length matter

Read size=255



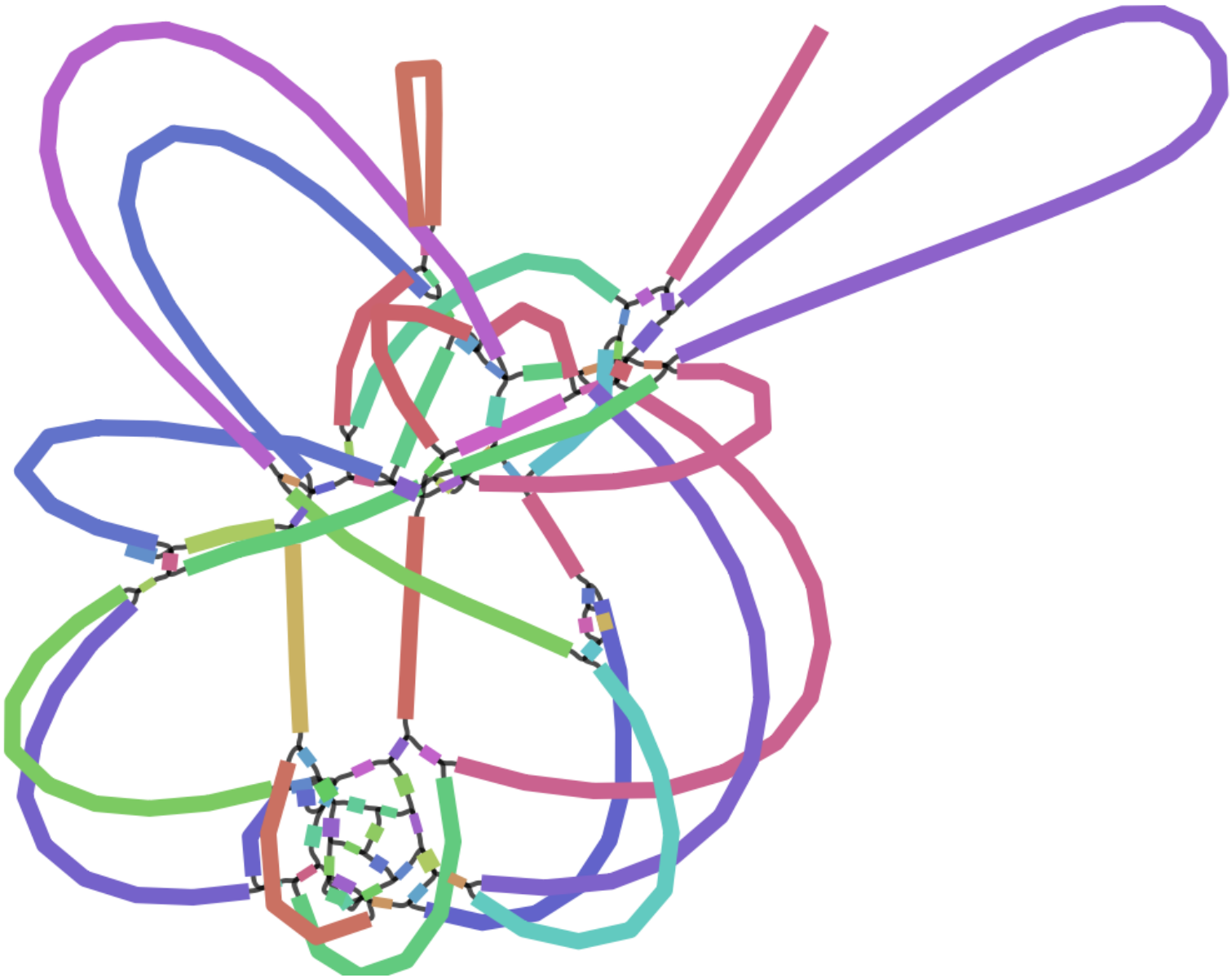
Read length matter

Read size=500



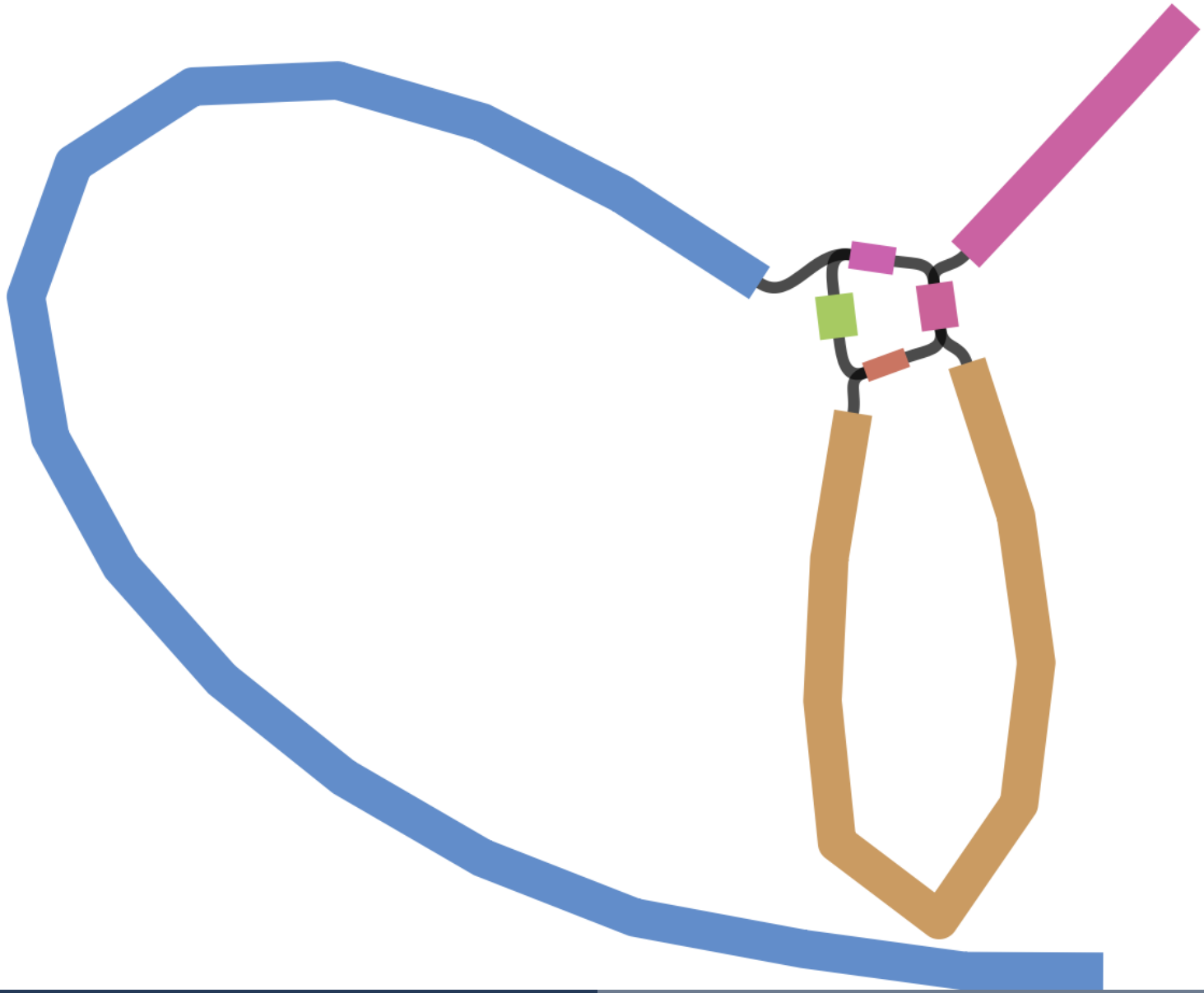
Read length matter

Read size=1000

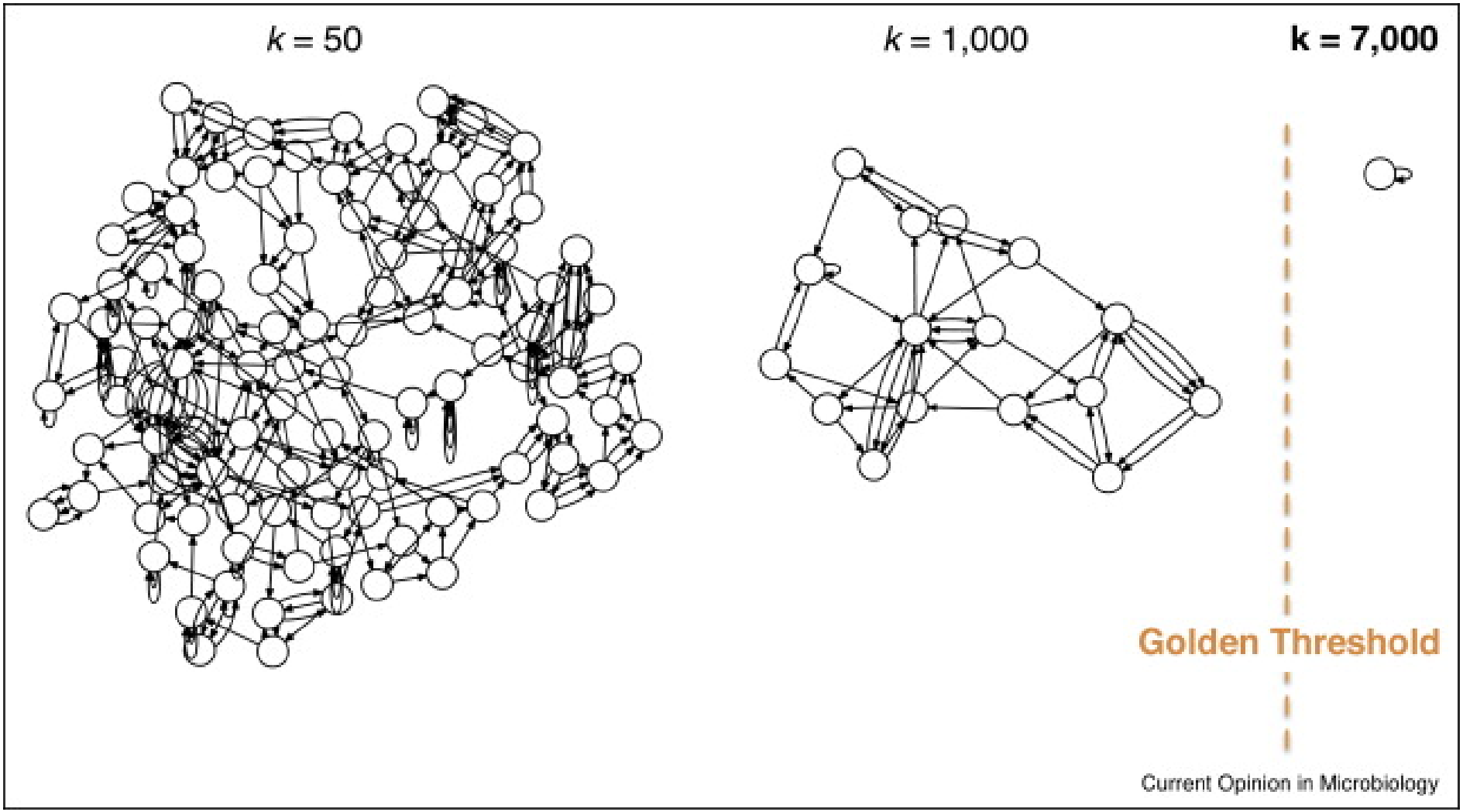


Read length matter

Read size=2000

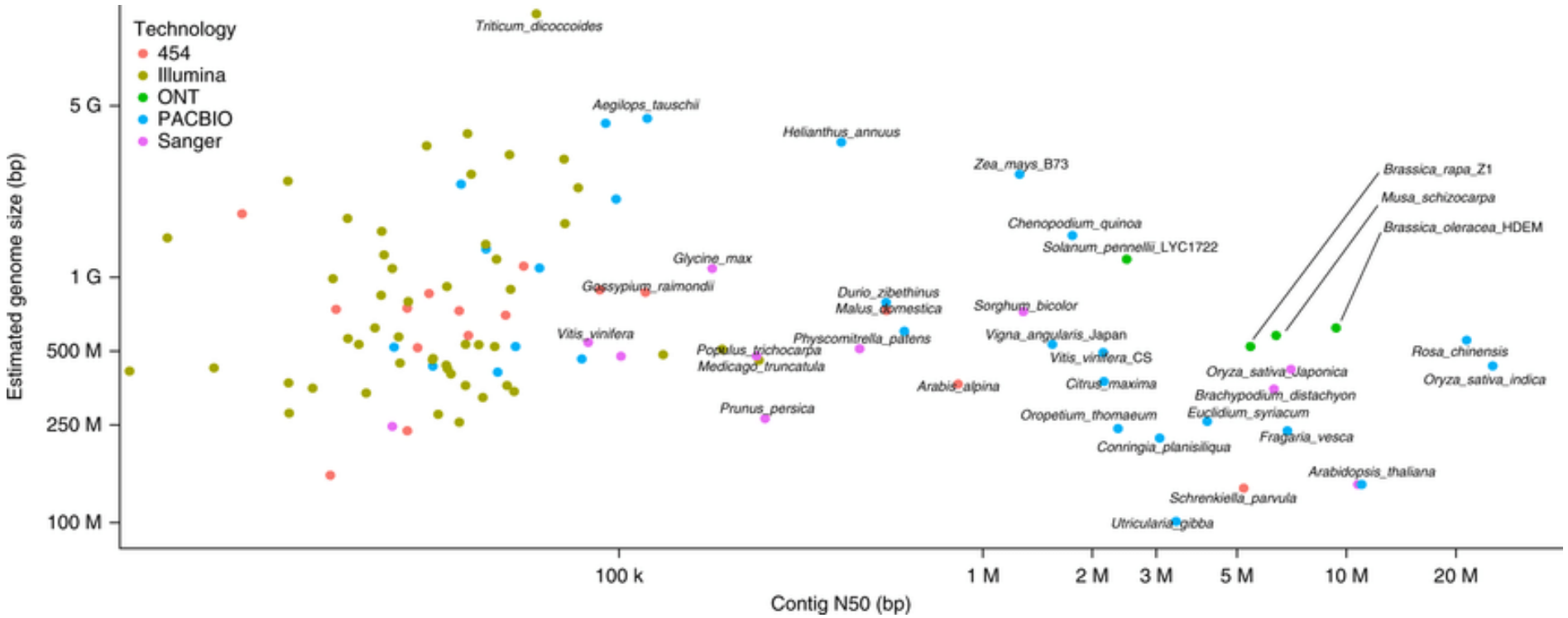


Great hope for assembly



From "One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly" Current Opinion in Microbiology 2015

Great hope for assembly



From "Chromosome-scale assemblies of plant genomes using nanopore long reads and optical maps" Nature Plants 2018

Which assembly strategy is best suited?

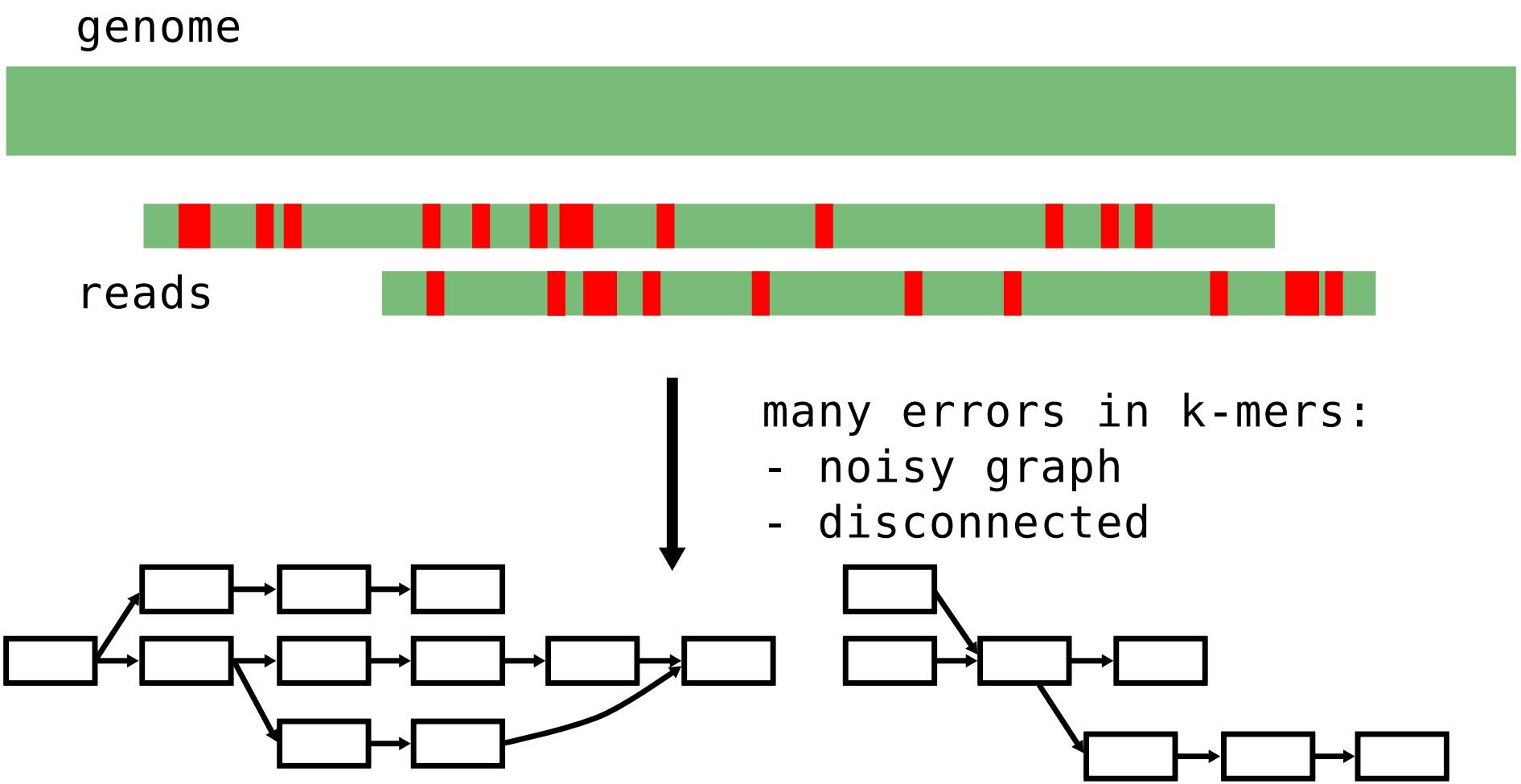
- Long reads $\approx 10kbp$
- High error rate $\approx 12\%$
- High throughput (up to millions of reads per run)

Based on long reads properties, which assembly solution would you choose and why?

Vote!

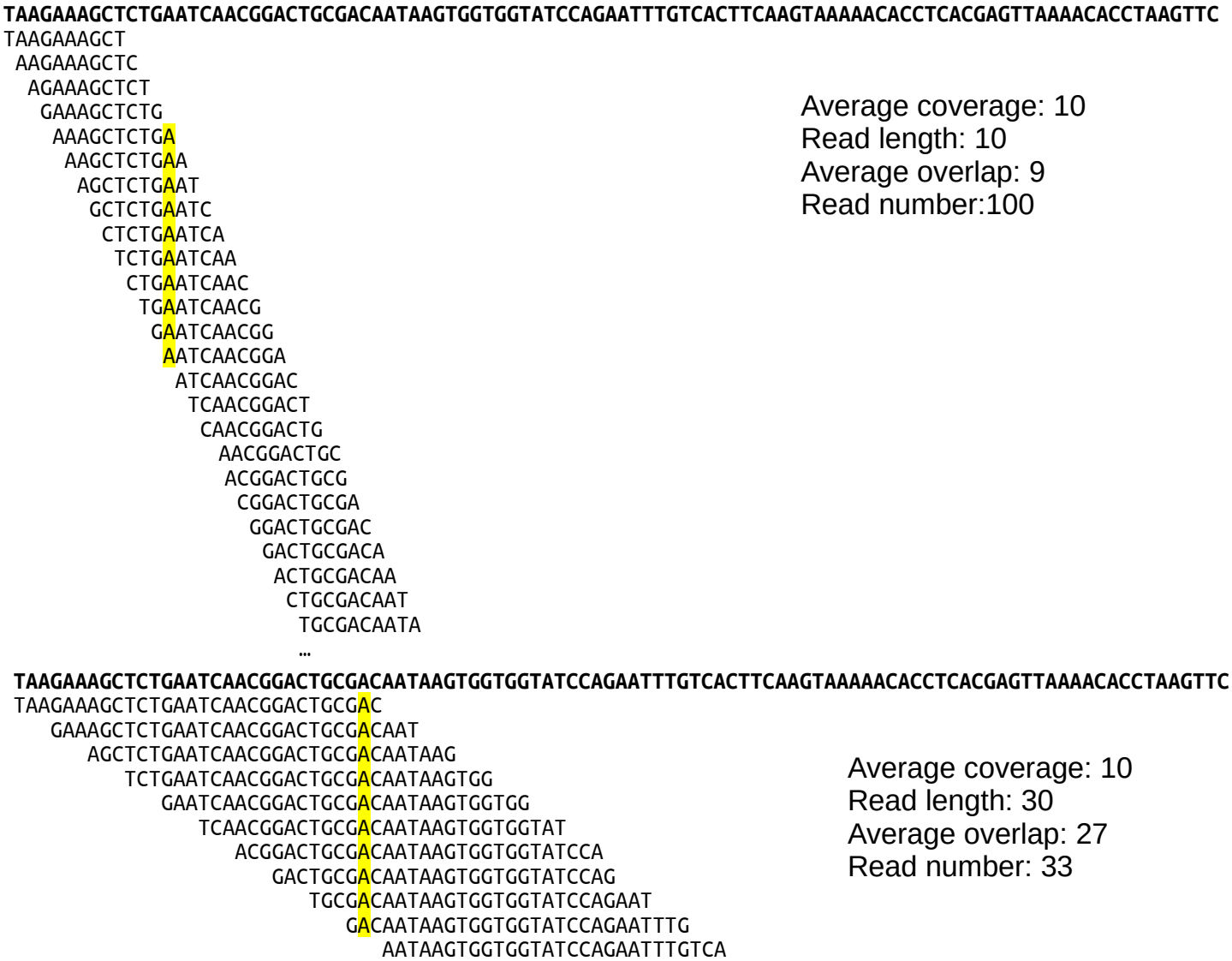
- ~~Fish~~
- Greedy
- Overlap graph
- ~~Birbs~~
- De Bruijn graph

Long reads for assembly: De Bruijn graph?



Long reads for assembly: overlap graph?

Supposed to be super expensive!



Average coverage: 10
Read length: 10
Average overlap: 9
Read number:100

Average coverage: 10
Read length: 30
Average overlap: 27
Read number: 33

Longer reads, better overlaps

- Less reads for the same coverage
- Larger overlaps

5Mb bacteria example with 100X coverage

Short reads

- 5 million 100bp reads
- 99 bp average overlap

Long reads

- 50,000 10kbp reads
- 9,900 bp average overlap

Very long reads

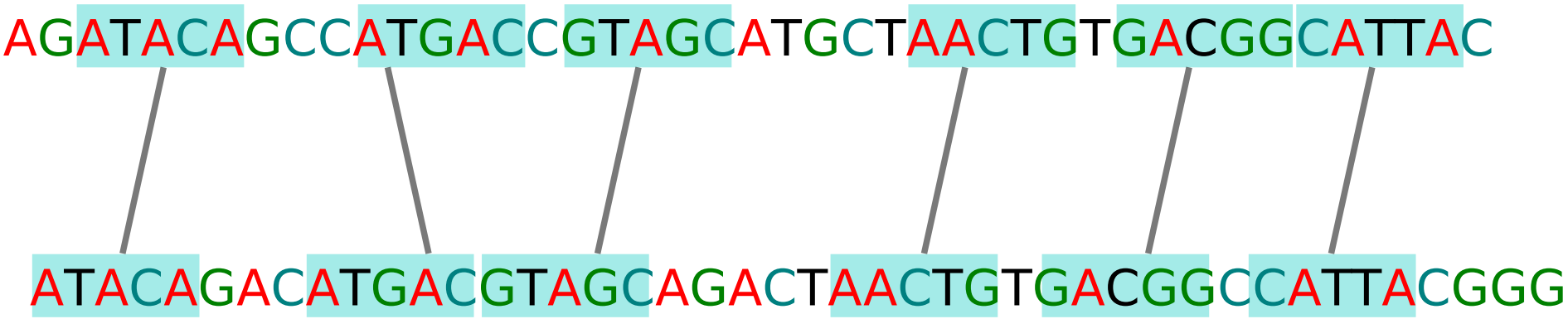
- 5,000 100kbp reads
- 99,000 bp average overlap

Anchors chaining in overlap graph

GCCATGACC

CATGACCGA

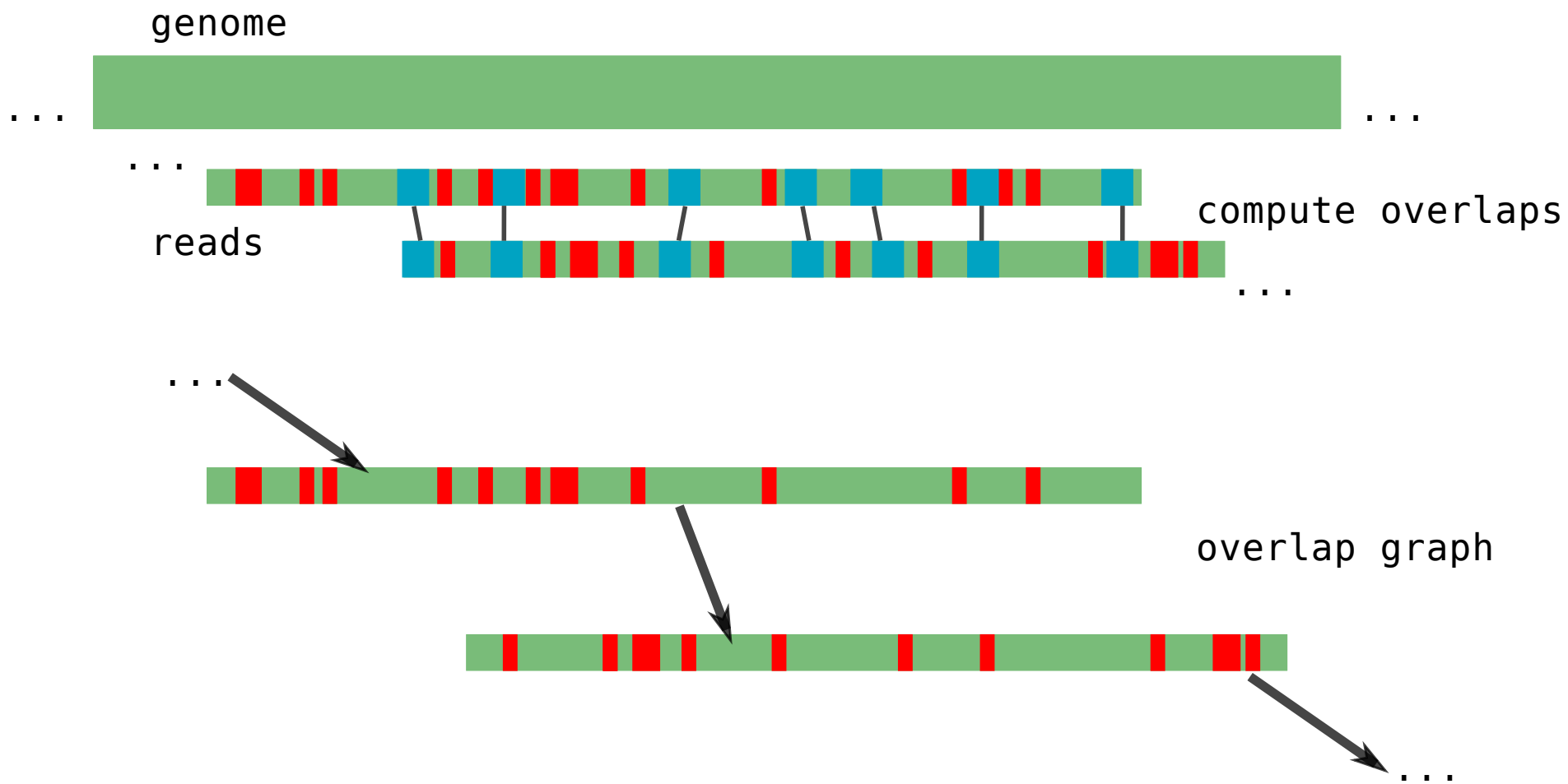
short reads



long reads

Minimap's job.

Long reads for assembly: overlap graphs



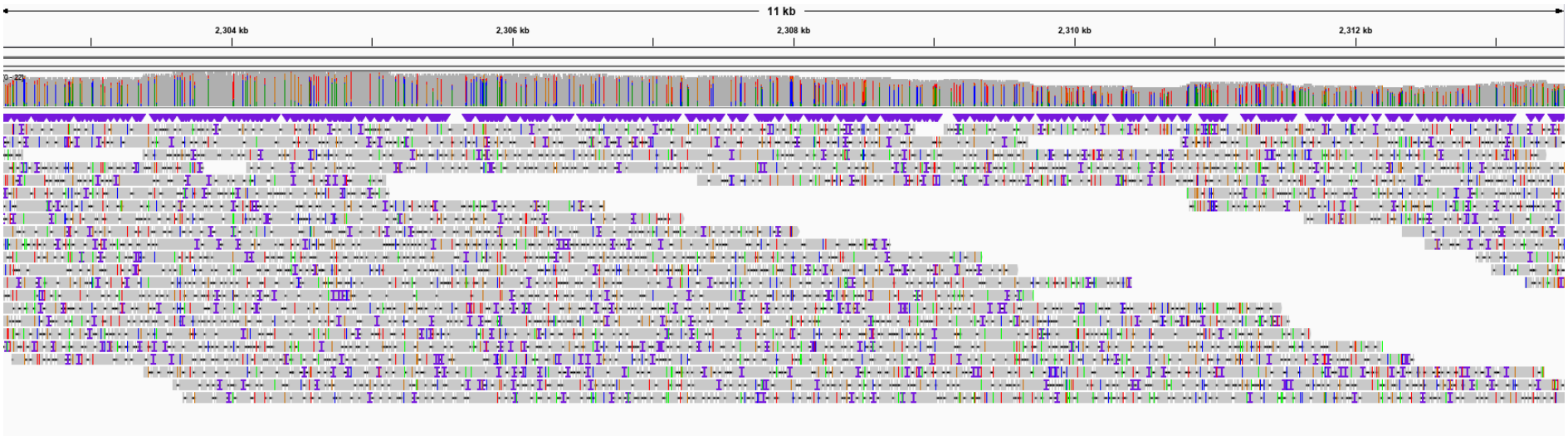
Sequencing errors

```
Genome:
 ATCGGTATCGTTACGGTATACC

Reads:
 ATCGCTATCG                (Substitution)
   GGTATCGTCTA             (Insertion)
      AT GTTACGG           (Deletion)
```

Insertion and deletion made calling almost impossible

Sequencing errors



Miniasm

Long reads can be assembled without taking care of the sequencing errors ("Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences", Bioinformatics '16)

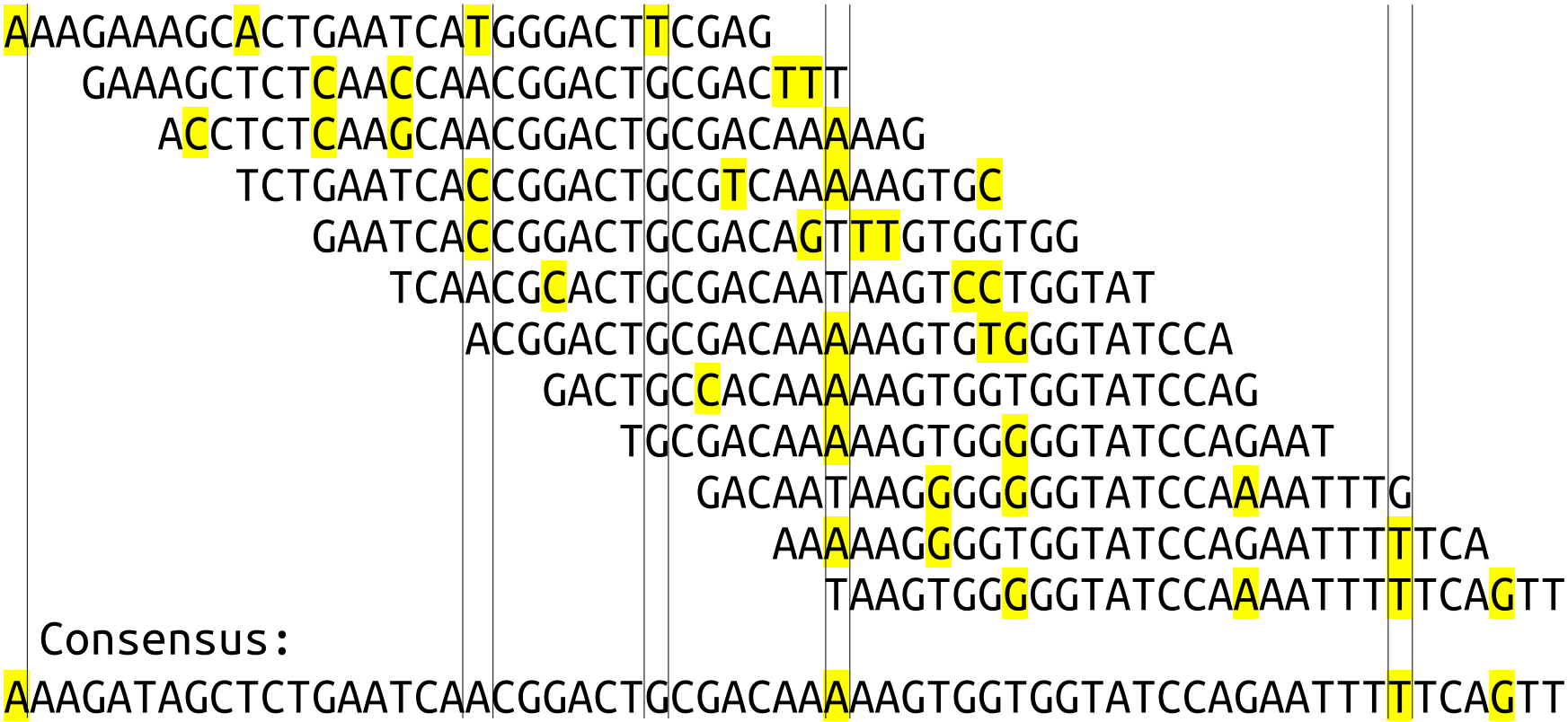
Genome characteristics and structure can be quickly estimated

Using coverage to remove noise: Consensus

Genome:

TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACTT

Reads:



Consensus:

AAAGATAGCTCTGAATCAACGGACTGCGACAAAAAGTGGTGGTATCCAGAATTTTTCAGTT

1/1 4/7 9/10 6/11 3/4

Exercise: Perform a consensus

Erroneous reads:
TAAGAAAGCTCTGAATCAAACGGTACTGCGA
GAAAGCTTGAATCAAACGGACTGCGACAA
AGCTCTGAATCAACGGACTGCGACAATAA

Contig to polish:
TAAGAAAGCTTGAATCAACGGAATGGCGACAATAA

Exercise: Perform a consensus - solution

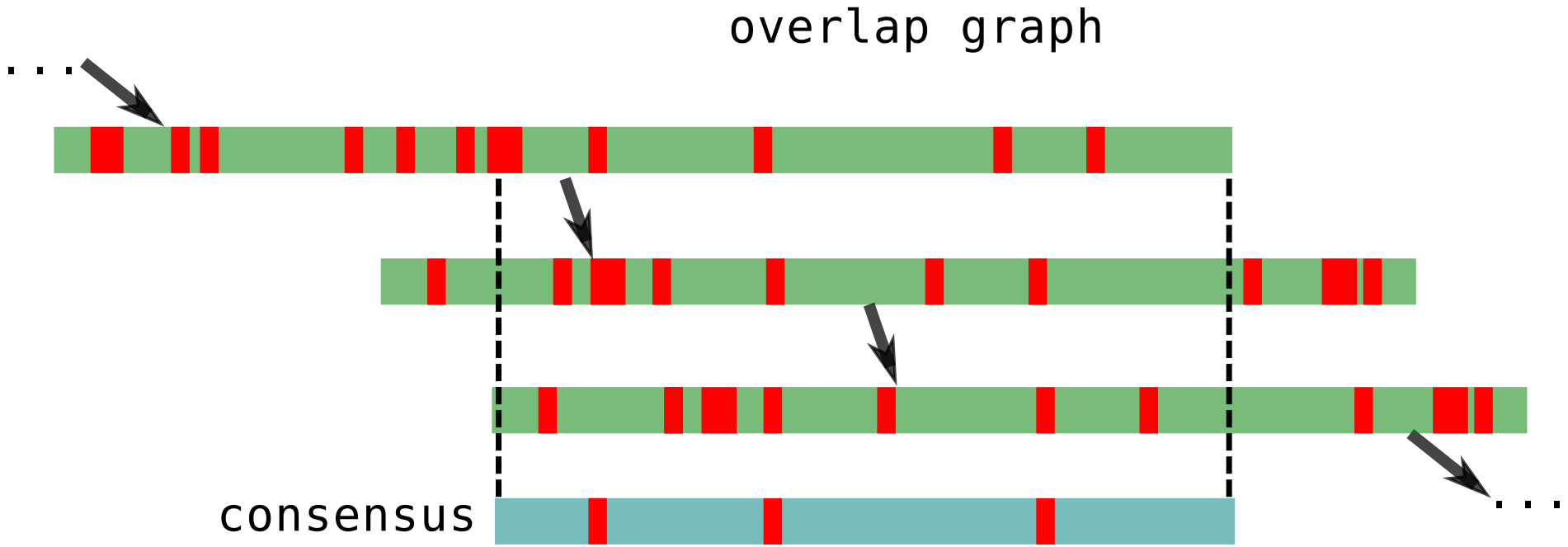
Correct contig:

TAAGAAAGCTCTGAATCAA-CGGACTG-CGACAATAA

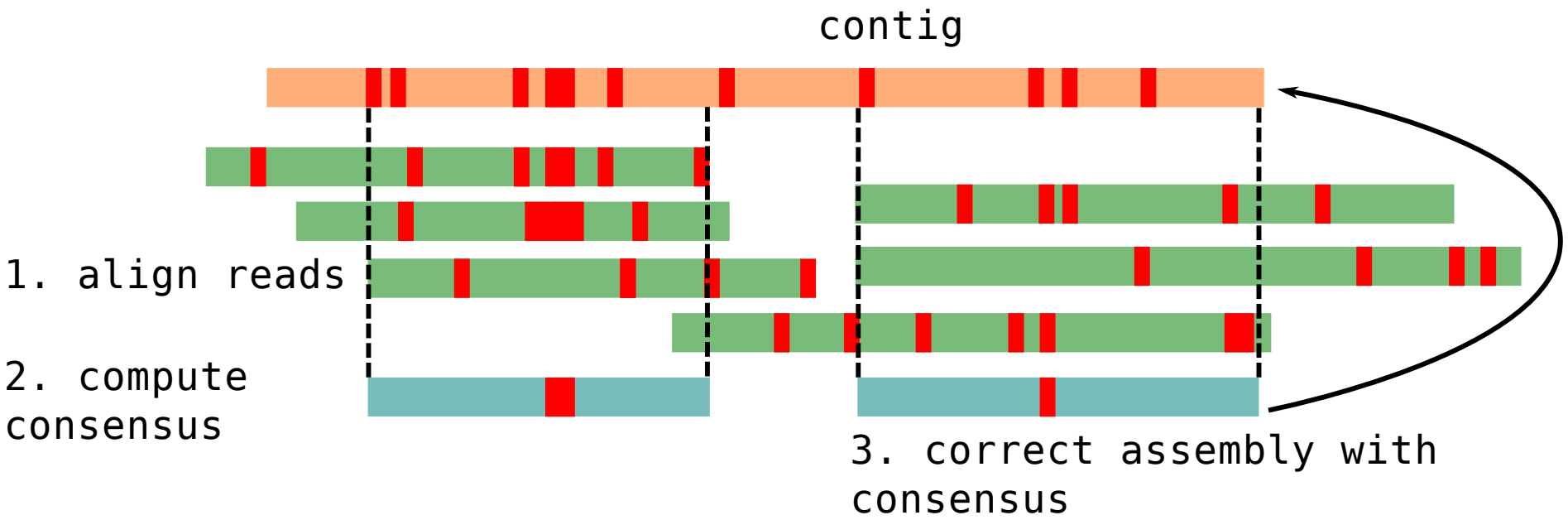
Aligned reads:

```
TAAGAAAGCTCTGAATCAAACGGTACTGCGA
   GAAAGCT-TGAATCAAACGGACTG-CGACAA
      AGCTCTGAATCAA-CGGACTG-CGACAATAA
```

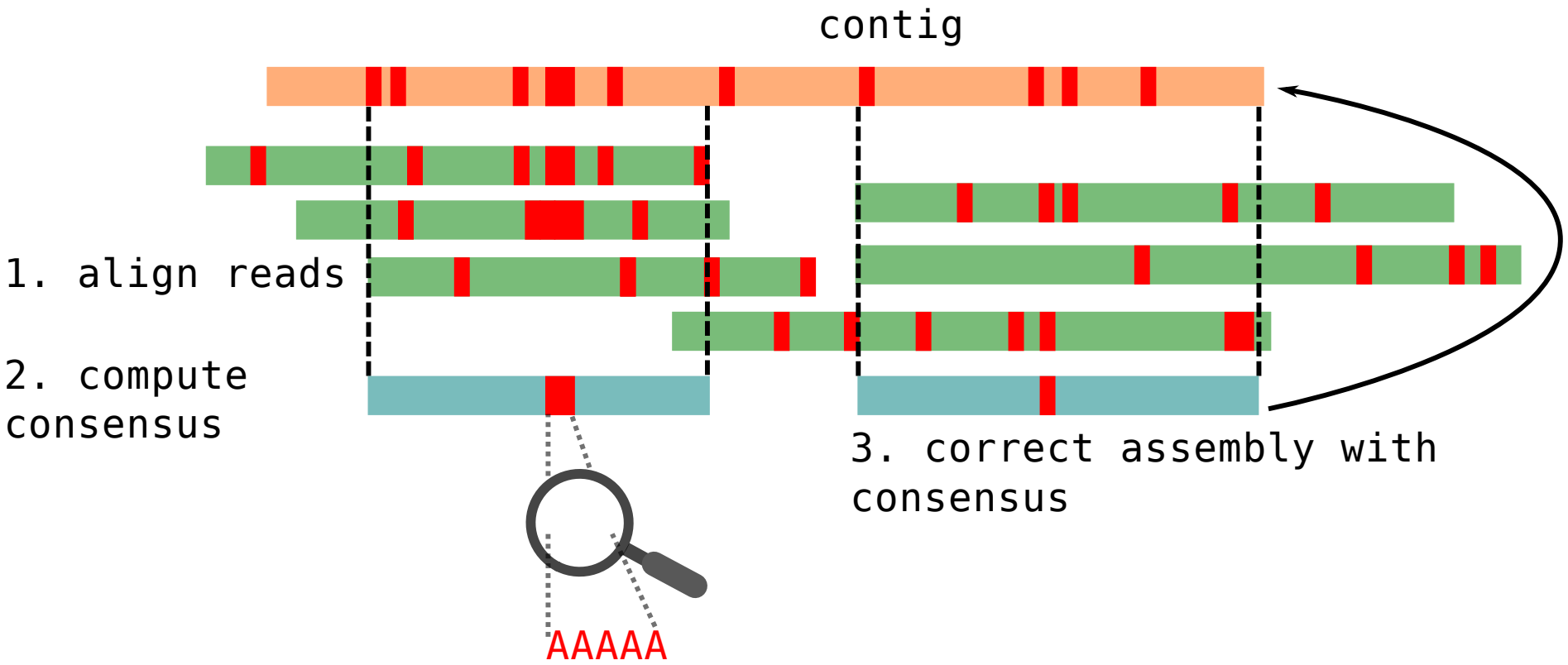
Consensus during assembly



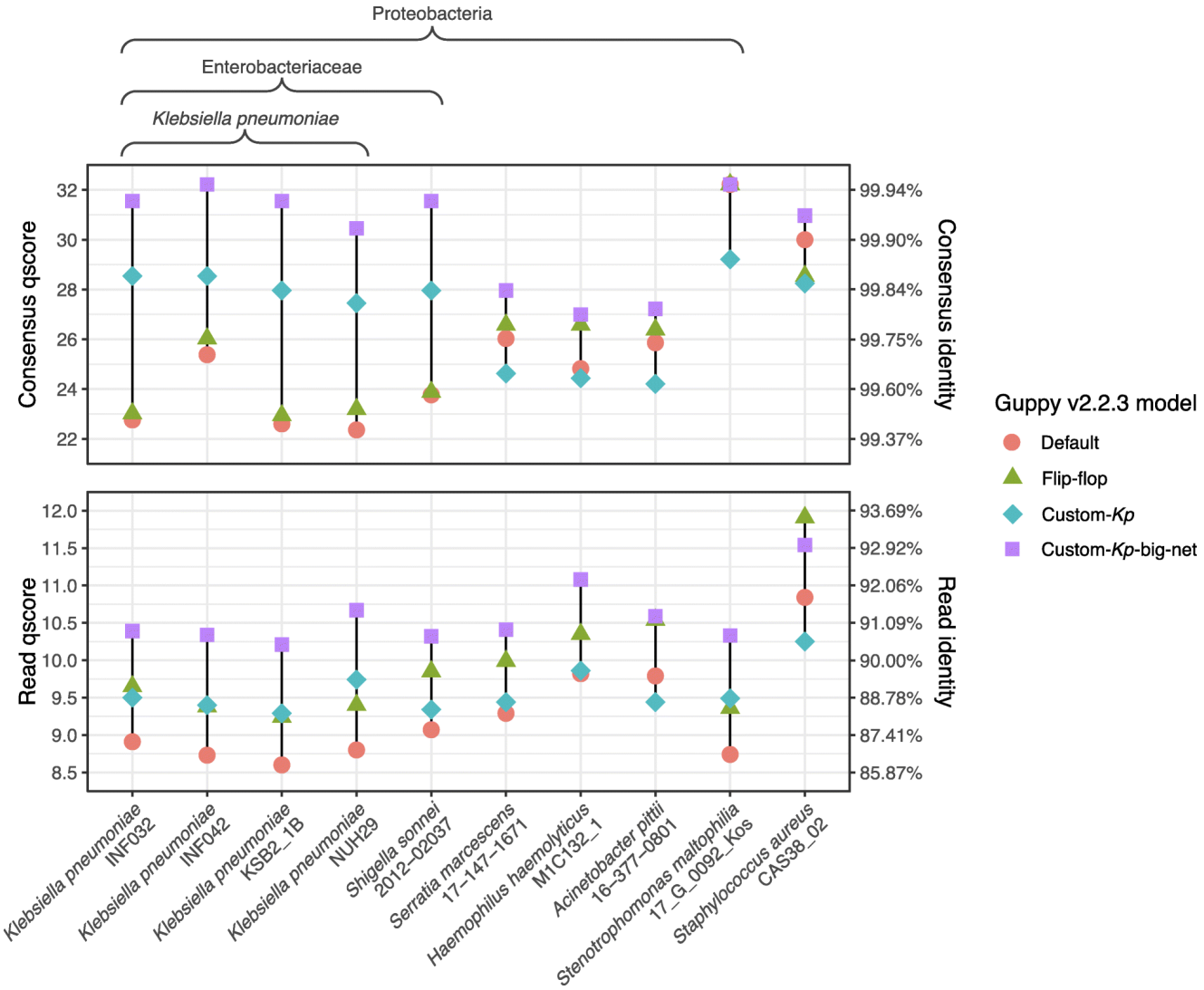
Consensus after assembly: polishing



Consensus after assembly: polishing



Systematic errors

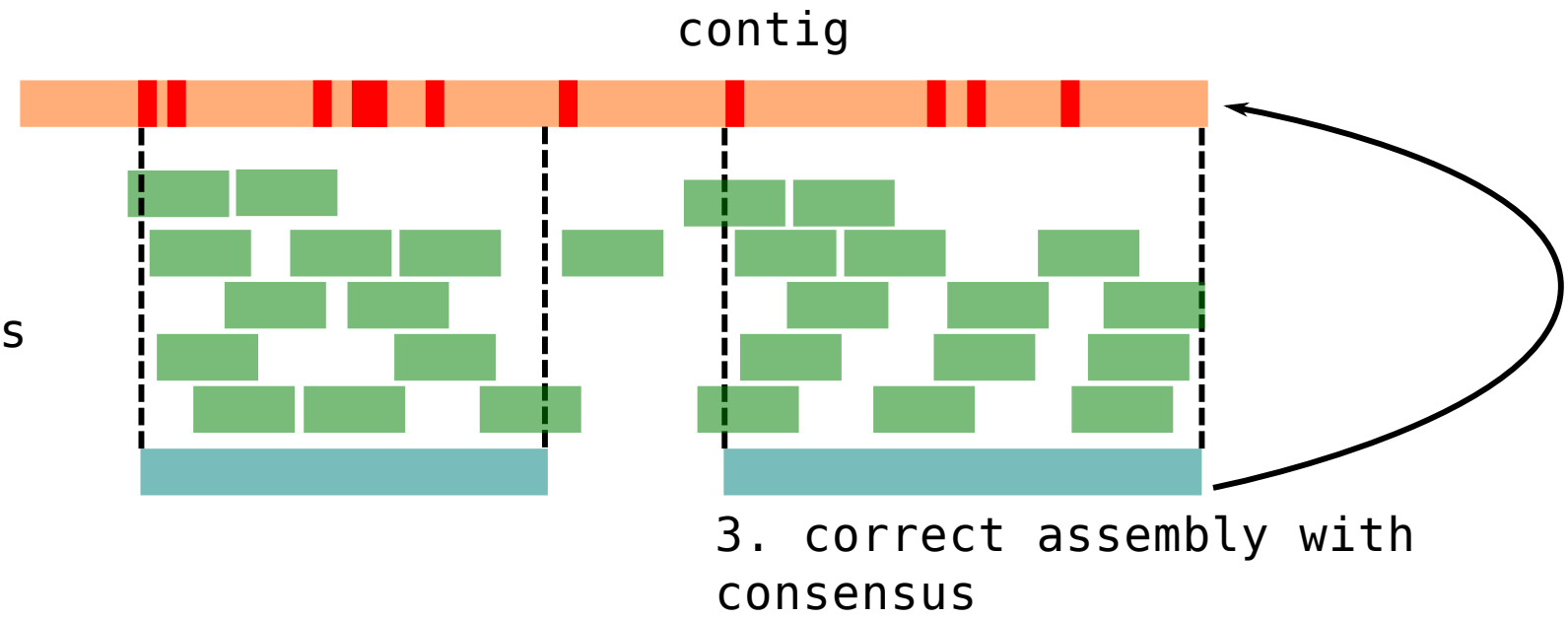


From "Performance of neural network basecalling tools for Oxford Nanopore sequencing" Genome biology 2019

Polishing using accurate reads

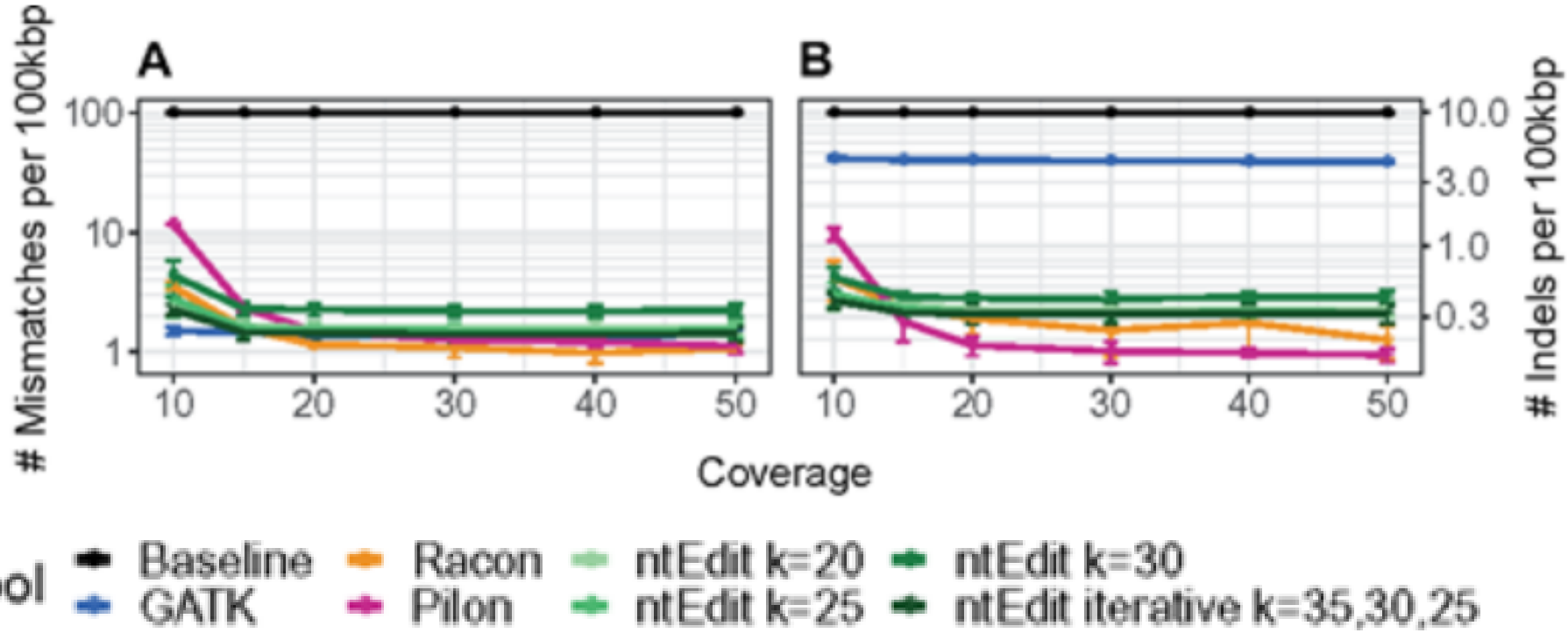
1. align **short** reads

2. compute consensus



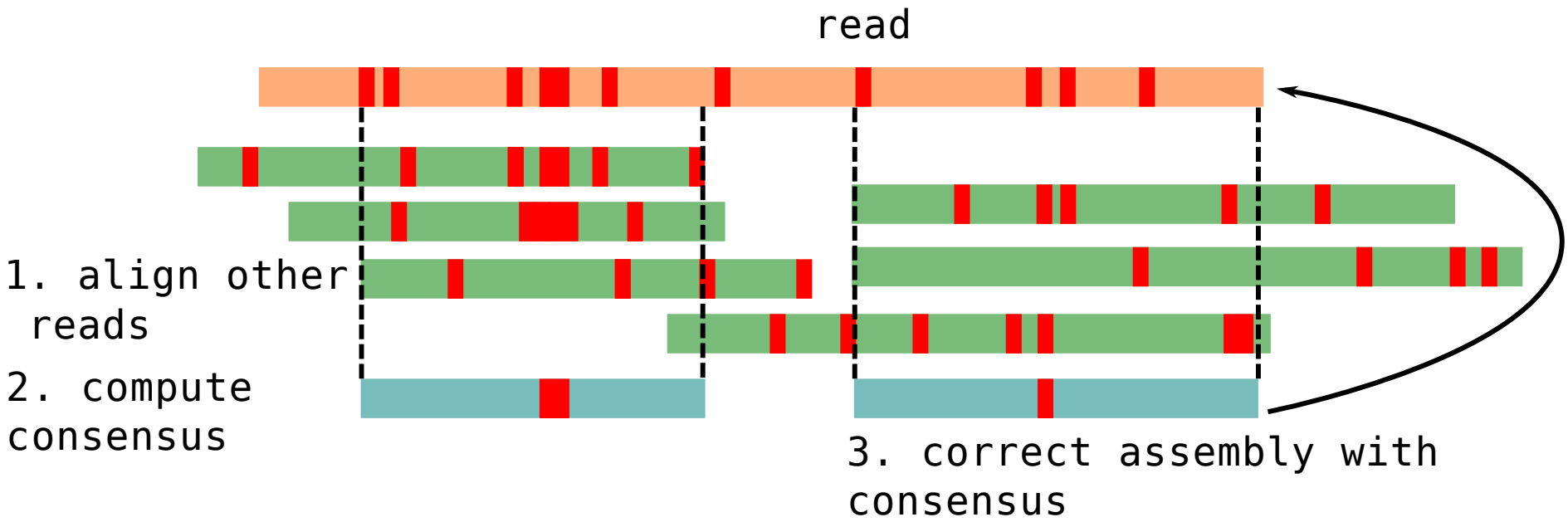
Systematics errors

Polishing with Illumina data can improve the final error rate



From ntEdit: scalable genome sequence polishing Bioinformatics 2019

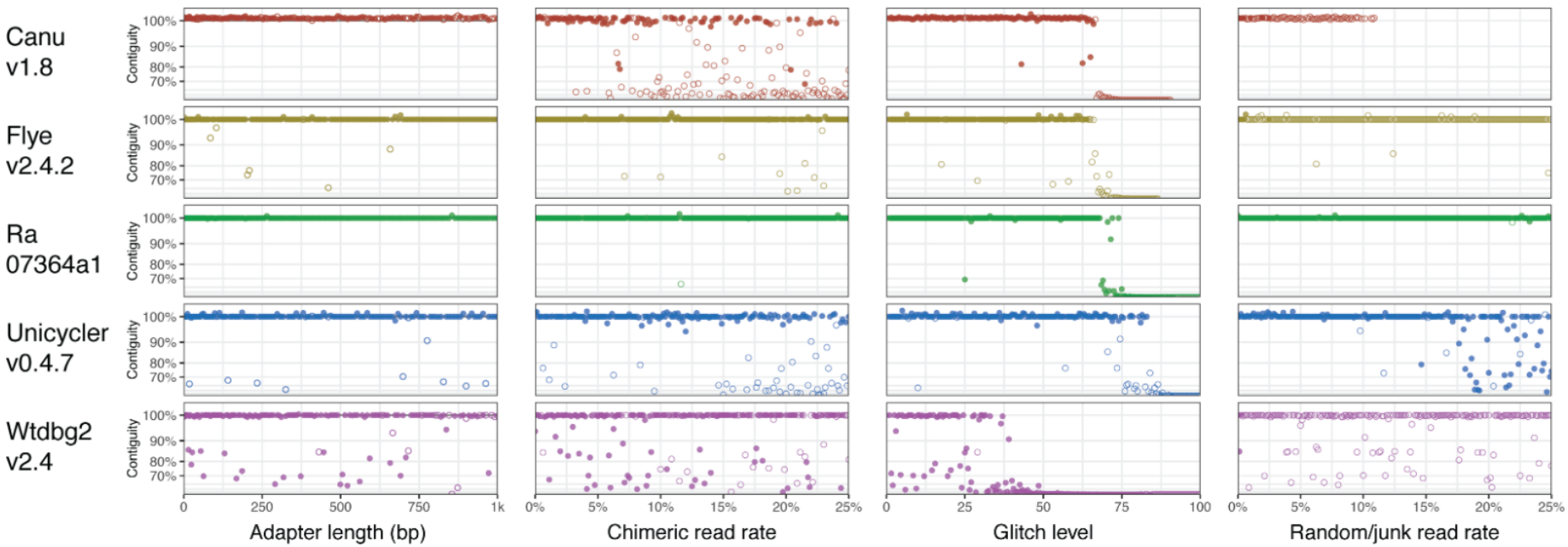
Correction before assembly



Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometime the software fail



From `github.com/rrwick/Long-read-assembler-comparison`

Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the data cannot solve the problem

- Very large repeated region
- Low local coverage
- Chimeric/noisy reads

Long reads assemblers

Your toolkit for the practical

- Flye
- Miniasm
- Raven

Other notable assemblers

- Canu
- Mecat
- Redbean
- ...

Long read assembly summary

- Not resolved: correction before or after assembly (polishing)
- Overlap graphs with quick overlap computation
- Long reads can span repeats and improve assemblies
- Methods to polish contigs

Conclusions

Take home messages

- Short reads: De Bruijn graphs / Long reads: Overlap graph
- Repeats are the core issue
- Output fragments of genomes (**contigs**)
- Several steps and heuristics in practice

Challenges

- Difficult to reconstruct haplotypes
- Scaling on large genomes
- Robustness to noisy data

Topics we did not review today

- Read correction
- Multi k assembly
- Use of paired-end, mate-pair, 10X, Hi-C . . .)

The end

PopGenGoogling
@popgengoogling

i trust you to figure out your own genome

Traduire le Tweet
3:01 AM · 14 déc. 2019 · Twitter for iPhone

37 Retweets **267** J'aime

Practical session

Datasets

Short Illumina reads 150bp
Long PacBio reads

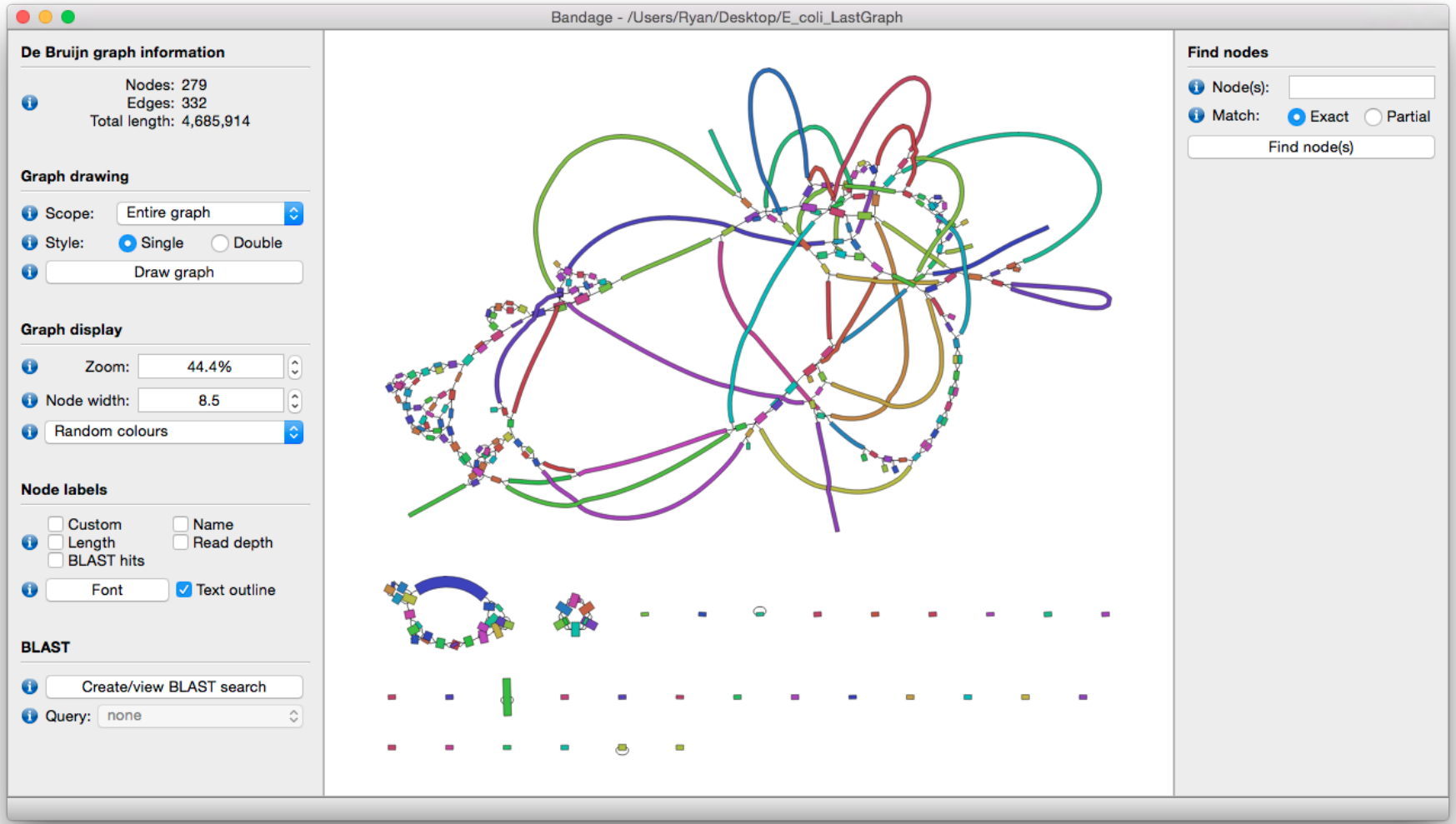
Steps

- Perform a draft assembly
- Evaluate it
- (Try to) Perform a better assembly

You will read the assembler website and papers to decide which one you want to use

Visualize assembly

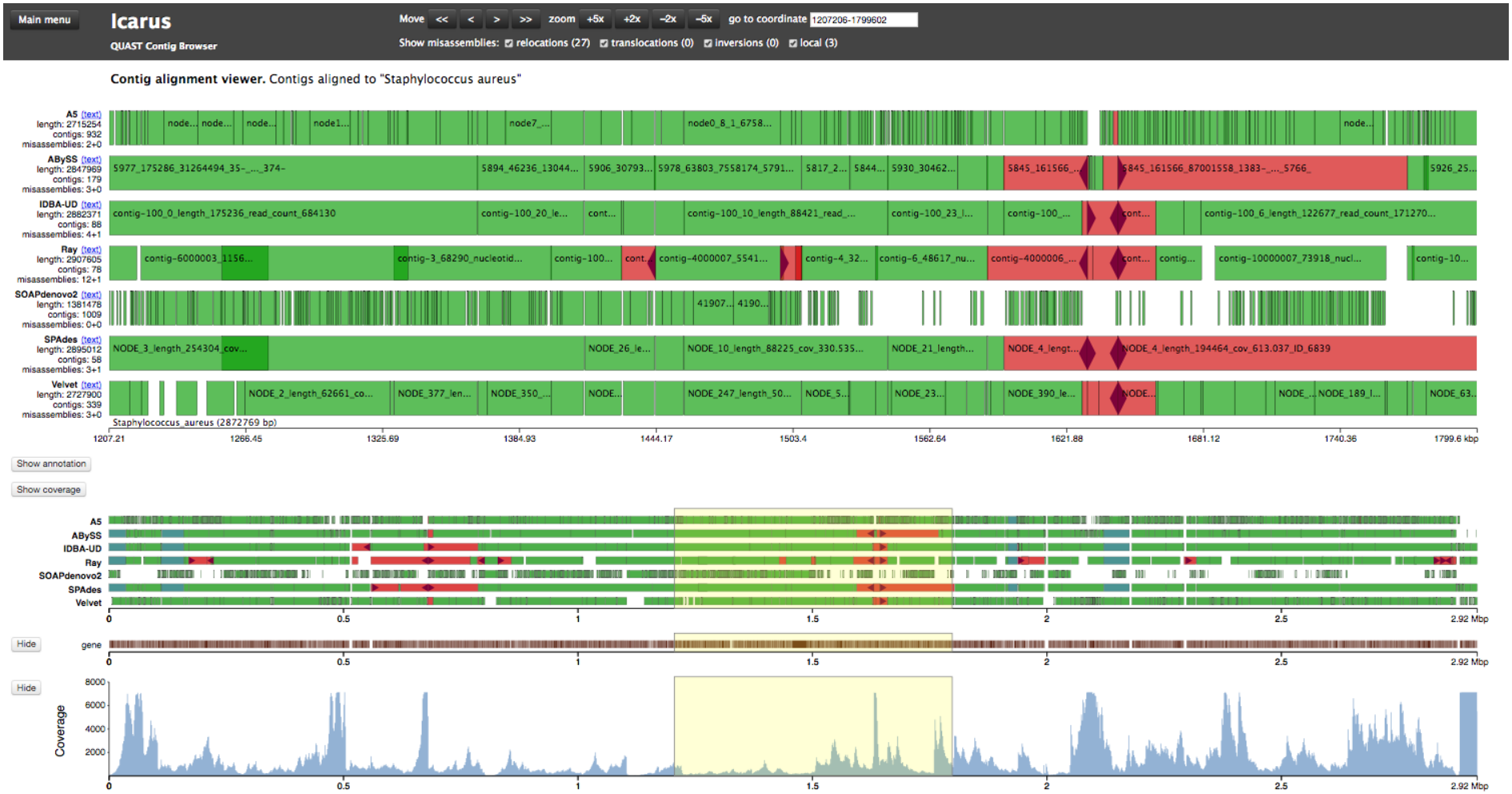
Bandage tool can visualize assembly graphs (GFA)



From `rwick.github.io/Bandage`

Evaluate assembly

Contigs can be mapped and compared to a reference/closely related genome



From `quast.bioinf.spbau.ru/manual.html`

Assembly continuity

N50

N50 can be described as a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value.

N75

N75 is the same statistic for 75% of the assembly

NGA50

Similar to the N50 but only takes into account contigs/scaffolds that can be aligned on the reference genome and consider 50% of the **genome size** instead of the assembly size

Misasemblies

Multiple k assembly

Most De Bruijn graph assemblers can now perform several assemblies with different k-mer sizes to produce an improved "super" assembly
(will be discussed in metagenomic session)

Advice

Using a single size of k-mer will allow the assembly to go way faster during the practical

Fig. 1. The workflow of MEGAHIT