A little tour of assembly methods

Antoine Limasset & Camille Marchet
CRIStAL, Université de Lille, CNRS, France
Evomics '23 – Český Krumlov

@Camillemrcht camille.marchet@univ-lille.fr
@BQPMalfoy antoine.limasset@univ-lille.fr

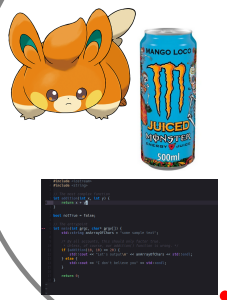
Antoine Limasset

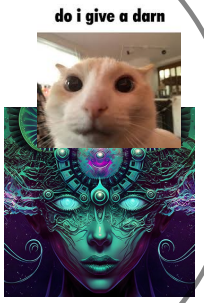
Antoine Limasset

Evomics

2020, 2022, 2023

Things he likes



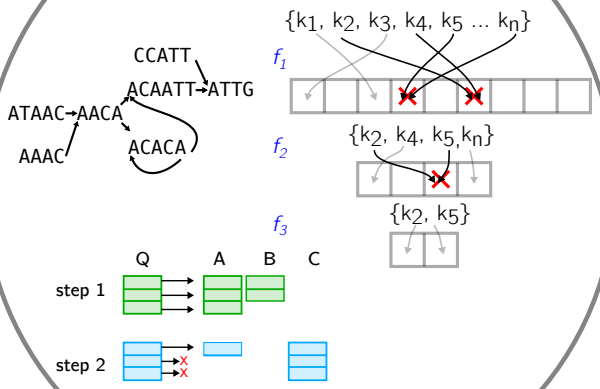


*

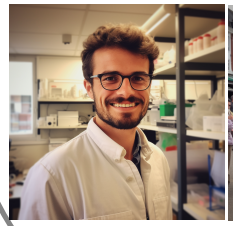
Random facts

- used to look like Drago Malfoy
- Amazon integrated one of his methods to their Alexa
- has done his PhD defense slides deck on an emergency lane
- has directed a movie about assembly
- will solve advanced data structure problems in his sleep

Research



According to MidJourney



Places

PennState

ULB UNIVERSITÉ LIBRE DE BRUXELLES

Université de Lille

UNIVERSITÉ DE RENNES 1

ENS CACHAN

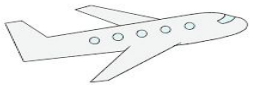
current

* Treasure hunt: find Antoine's fav animal in one of the slides

Camille Marchet

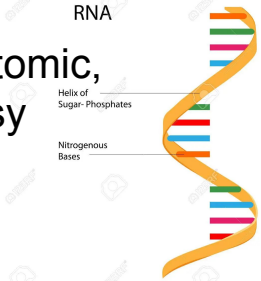
Can name every bird

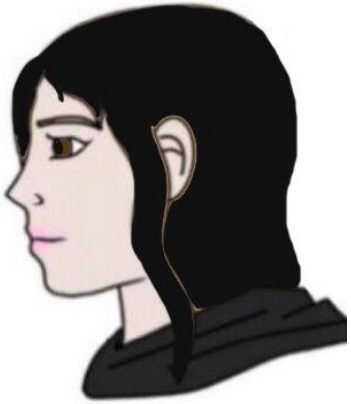
Learn novel language during a plane trip



Tries new recipes every day
100% success rate

Focus on transcriptomic,
Genomic is too easy



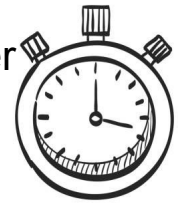


Manuscript rejected?
Let submit to a better journal

Can know if a restaurant is good
by the colors of the chairs

Parental Leave?
Goes to London and Krumlov
the same week with two child

Deadline is 4 day away?
Let's submit another paper



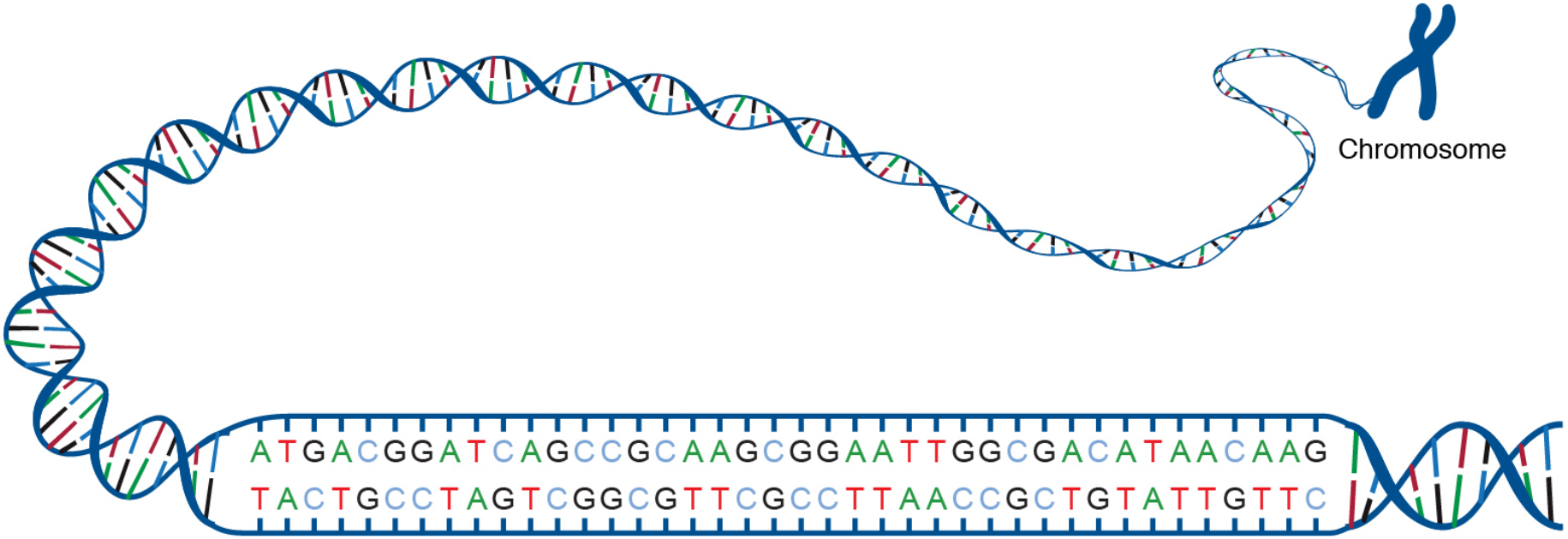
Read half of a CS book in her life
Develop state of the art data structures

Content of this course

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

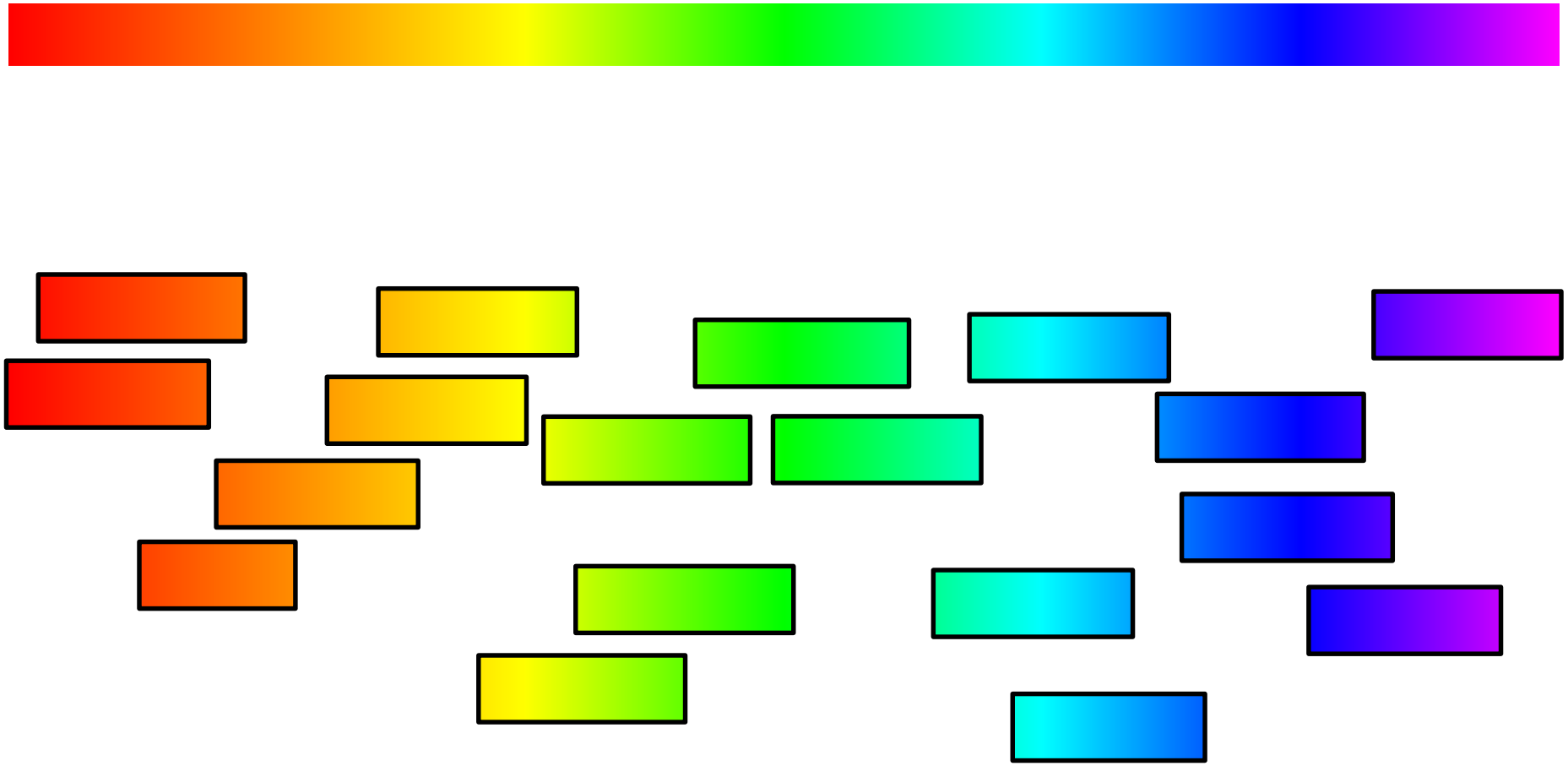
genome size: $\sim$ 32 gigabases

- Accessing a genome

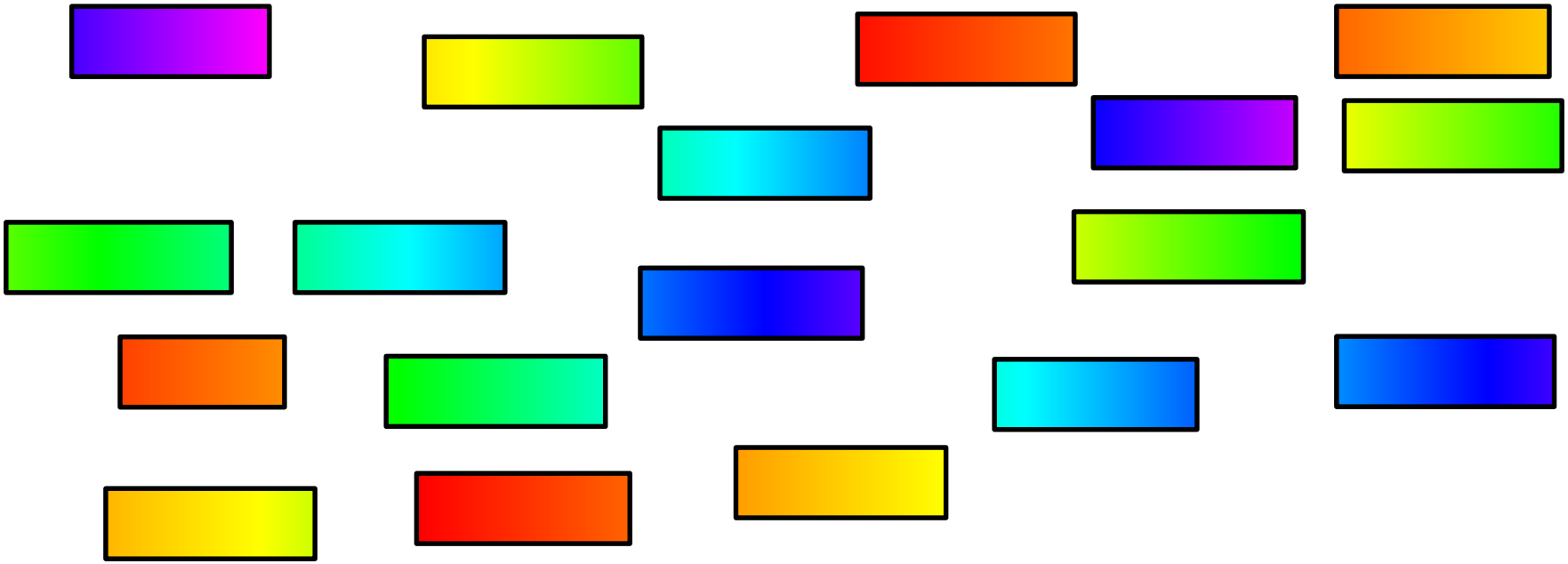


From `www.genome.gov/genetics-glossary/acgt`

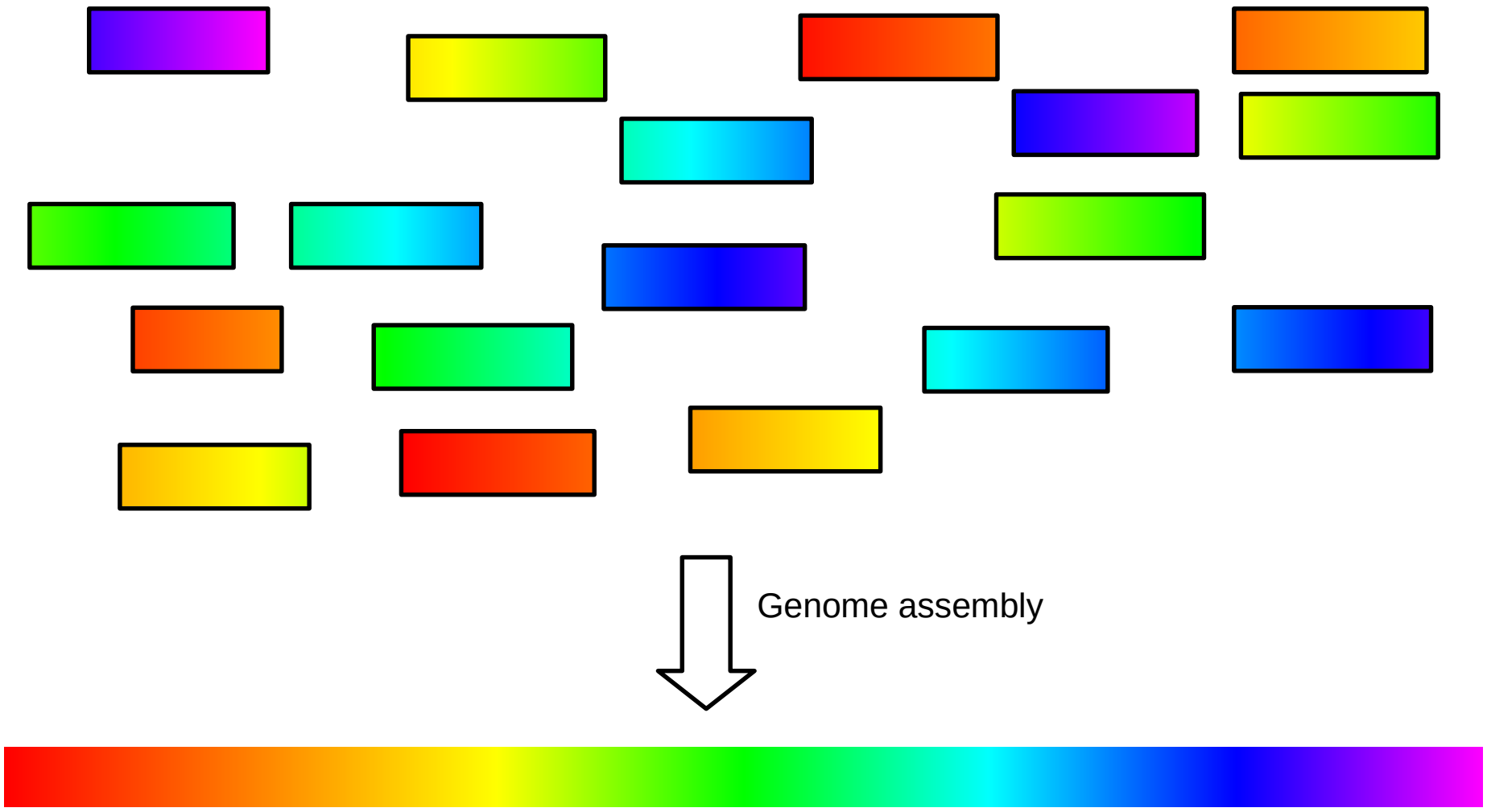
- Reads are subsequences from the genome



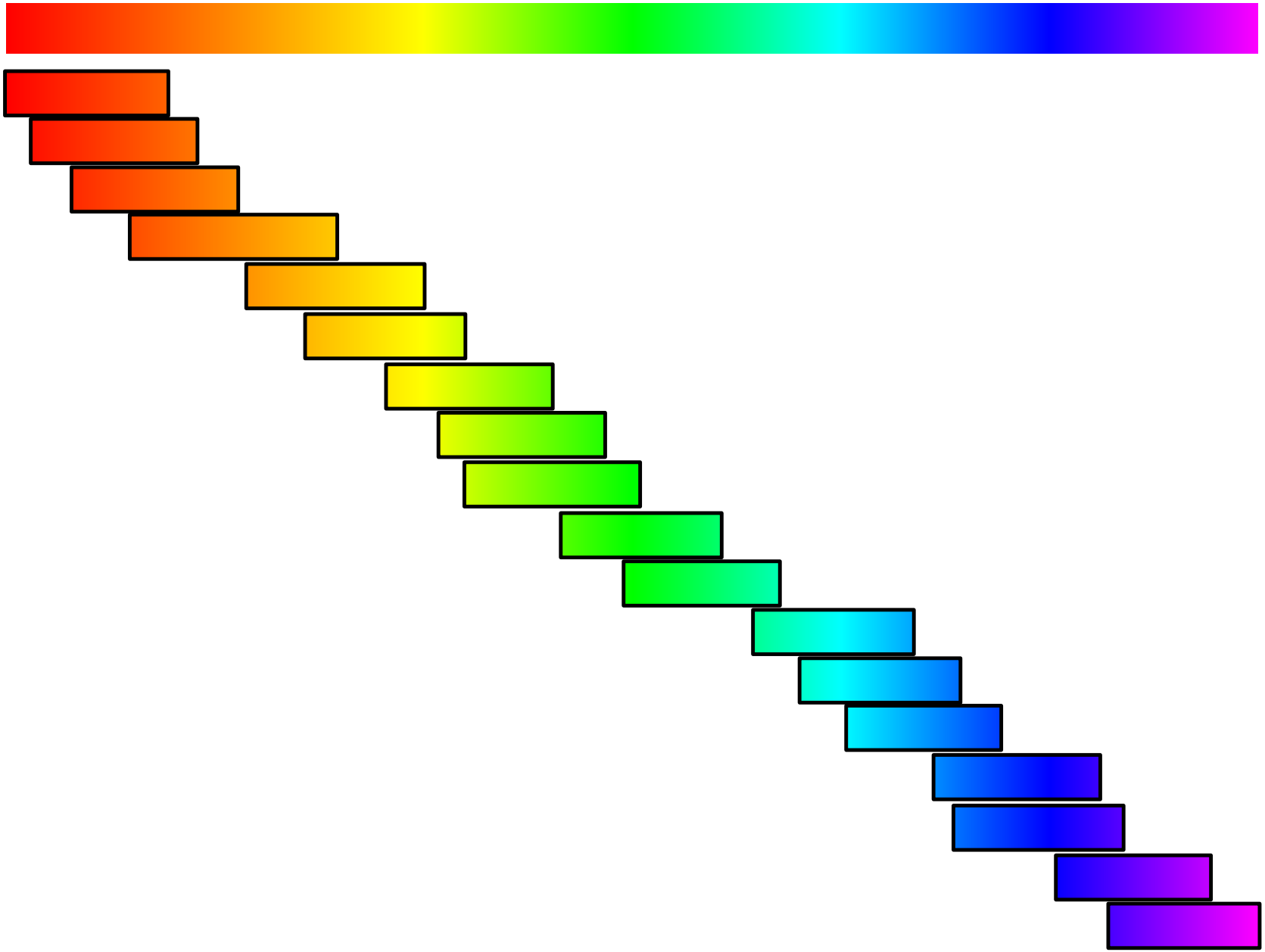
- Reads are **shuffled** subsequences from the genome



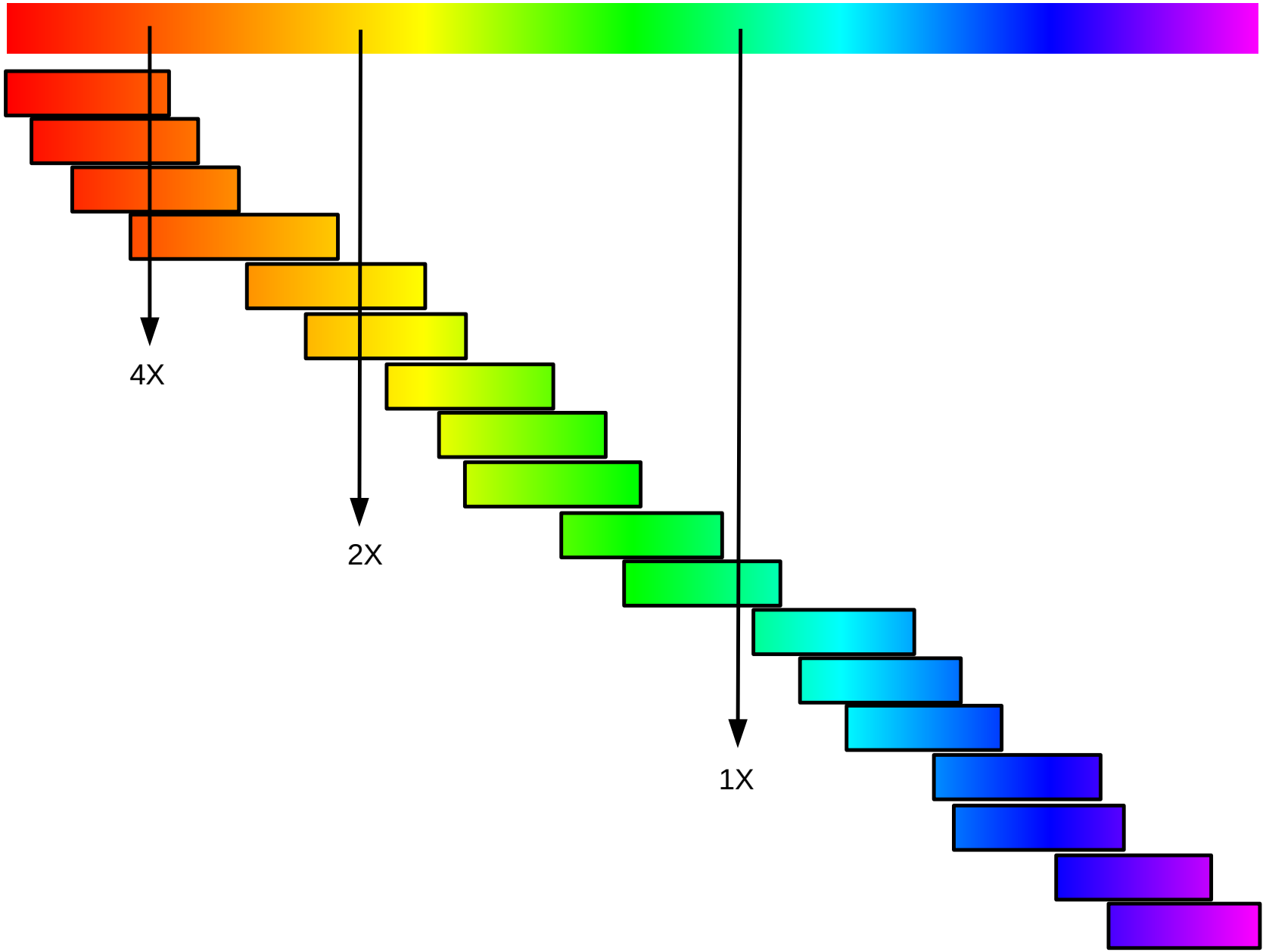
- Genome assembly task



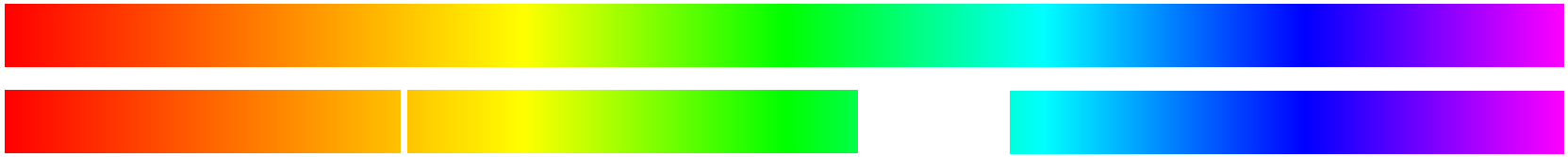
- Using read overlaps



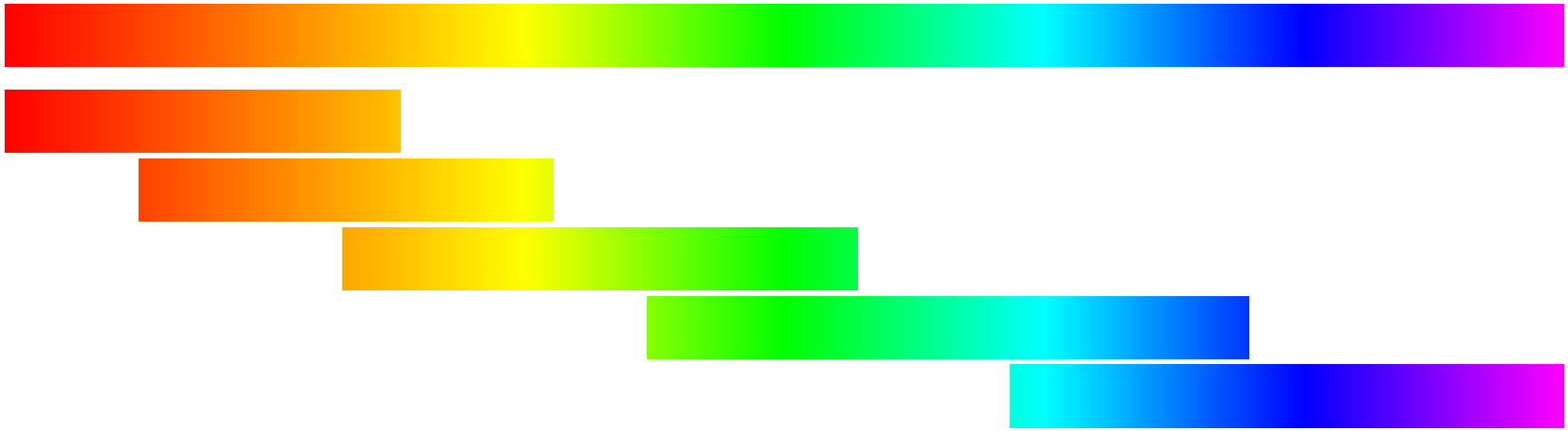
Genome sequencing: coverage



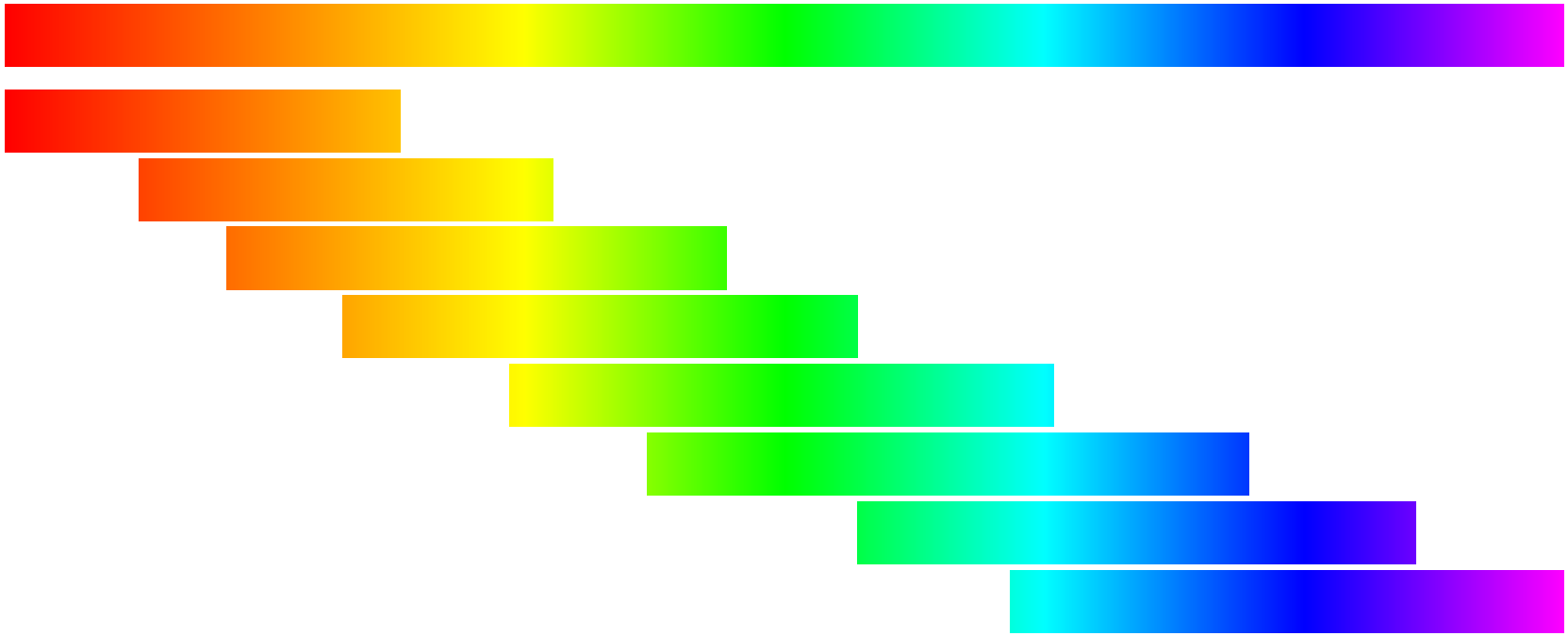
- Genome sequencing: coverage



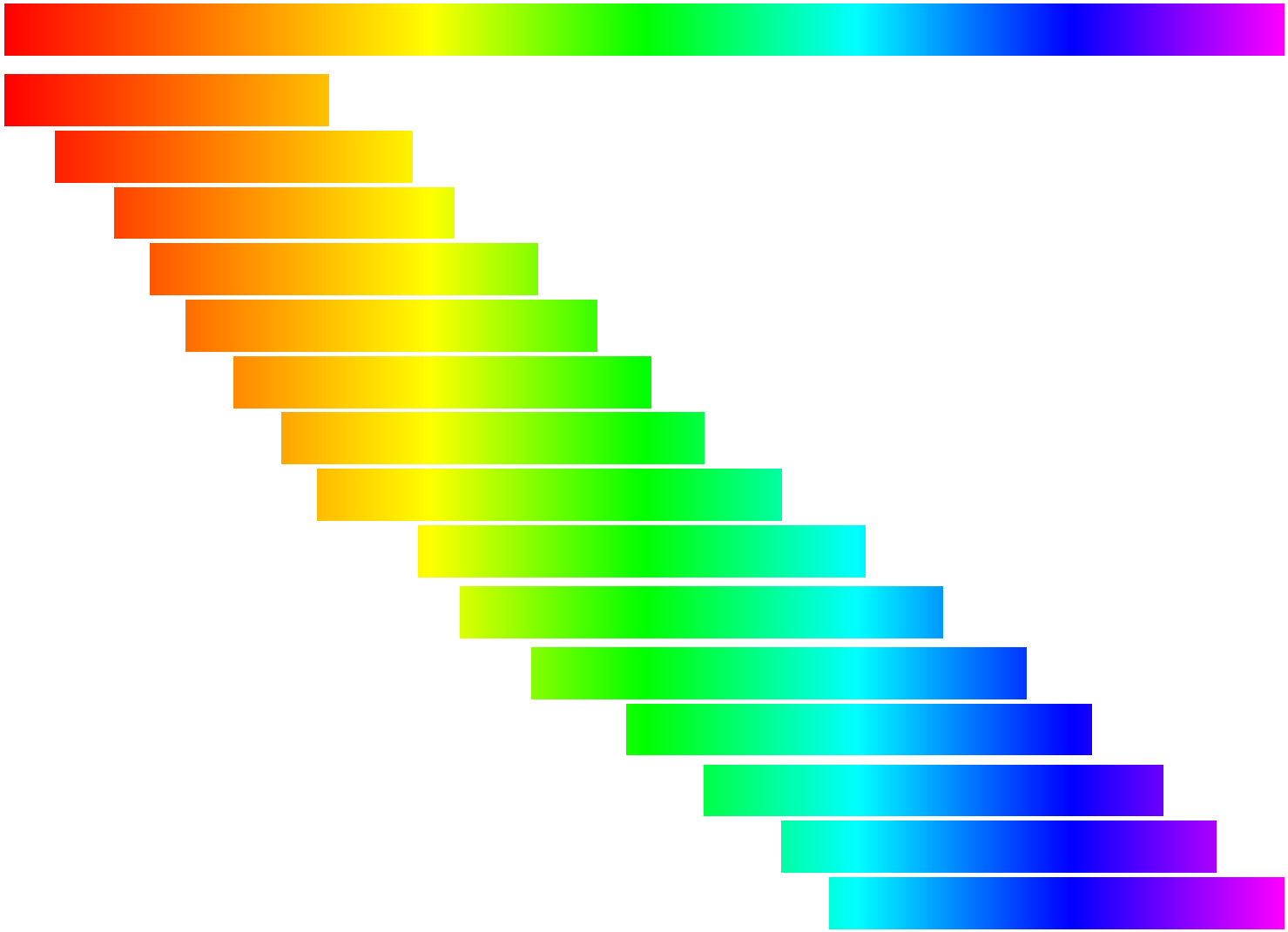
- Genome sequencing: coverage



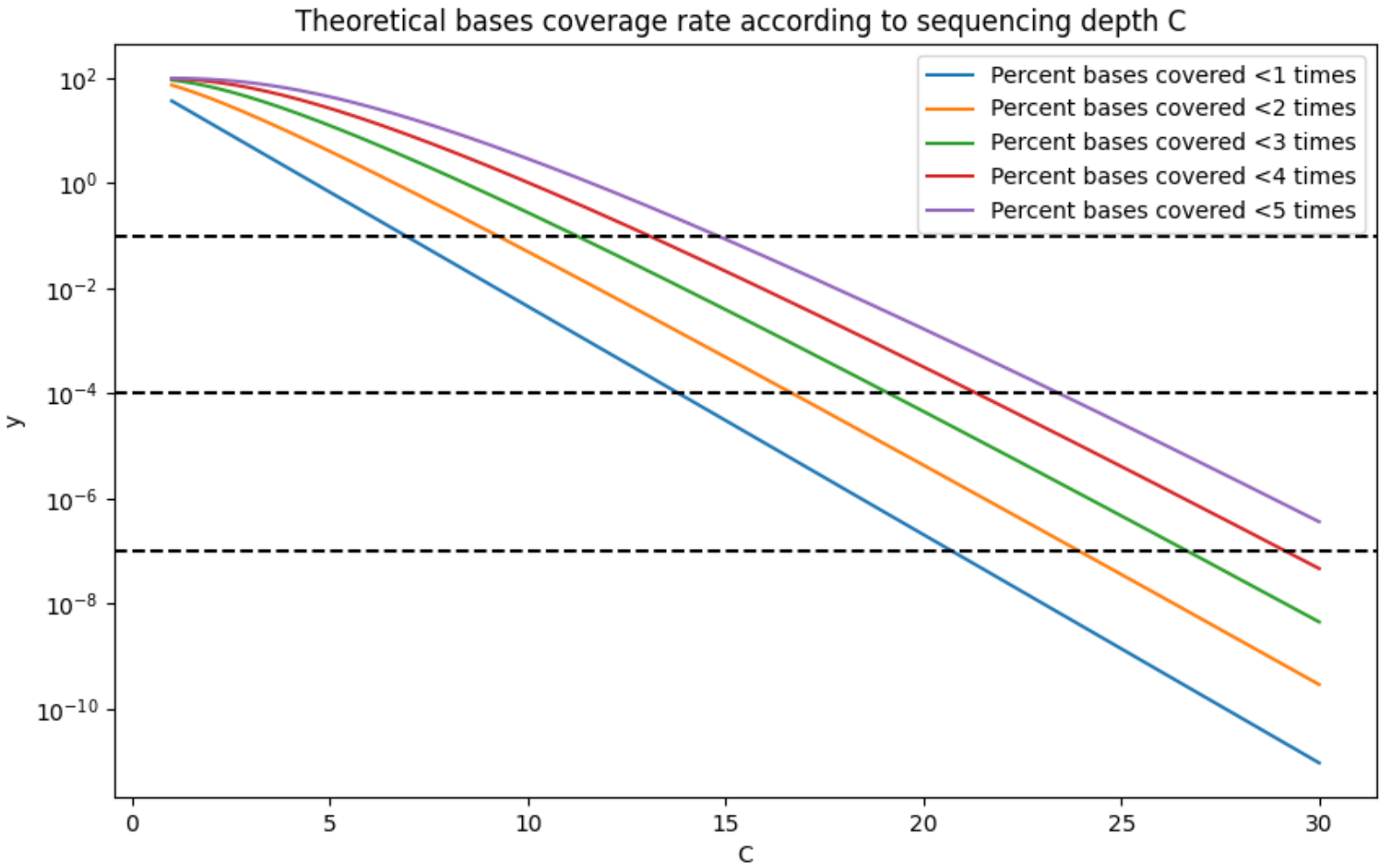
- Genome sequencing: coverage



Genome sequencing: coverage



Genome sequencing: coverage



30-100X are often required for assembly projects

- Which overlap?

```
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
```

supplementary information brought by each read

Best overlaps:

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

- 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:

```
ATTTACGGGT
    ACGGGTCCTT
    6   GTCCTTTCTT
            TTTCTTACGG
                TTACGGGTGG
```

~~longest overlap we found~~

```
ATTTACGGGT
    TTACGGGTGG
    8
```

- What happened?

The actual genome:

ATTTACGGGTCCTTTCTTACGGGTGG

How the reads should be ordered:

```
ATTTACGGGT
    ACGGGTCCTT
    6   GTCCTTTCTT
            TTTCTTACGG
                TTACGGGTGG
```

longest overlap we found

```
ATTTACGGGT
    TTACGGGTGG
        8
```

ATTTACGGGTGG
not in the genome

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:

$$\left(\frac{4^{31}-1}{4^{31}}\right)^{1/2\left(5\times 10^6\left(5\times 10^6-1\right)\right)}$$

Decimal approximation:

0.999997289498784302383172055421363836712023171938932024106...

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

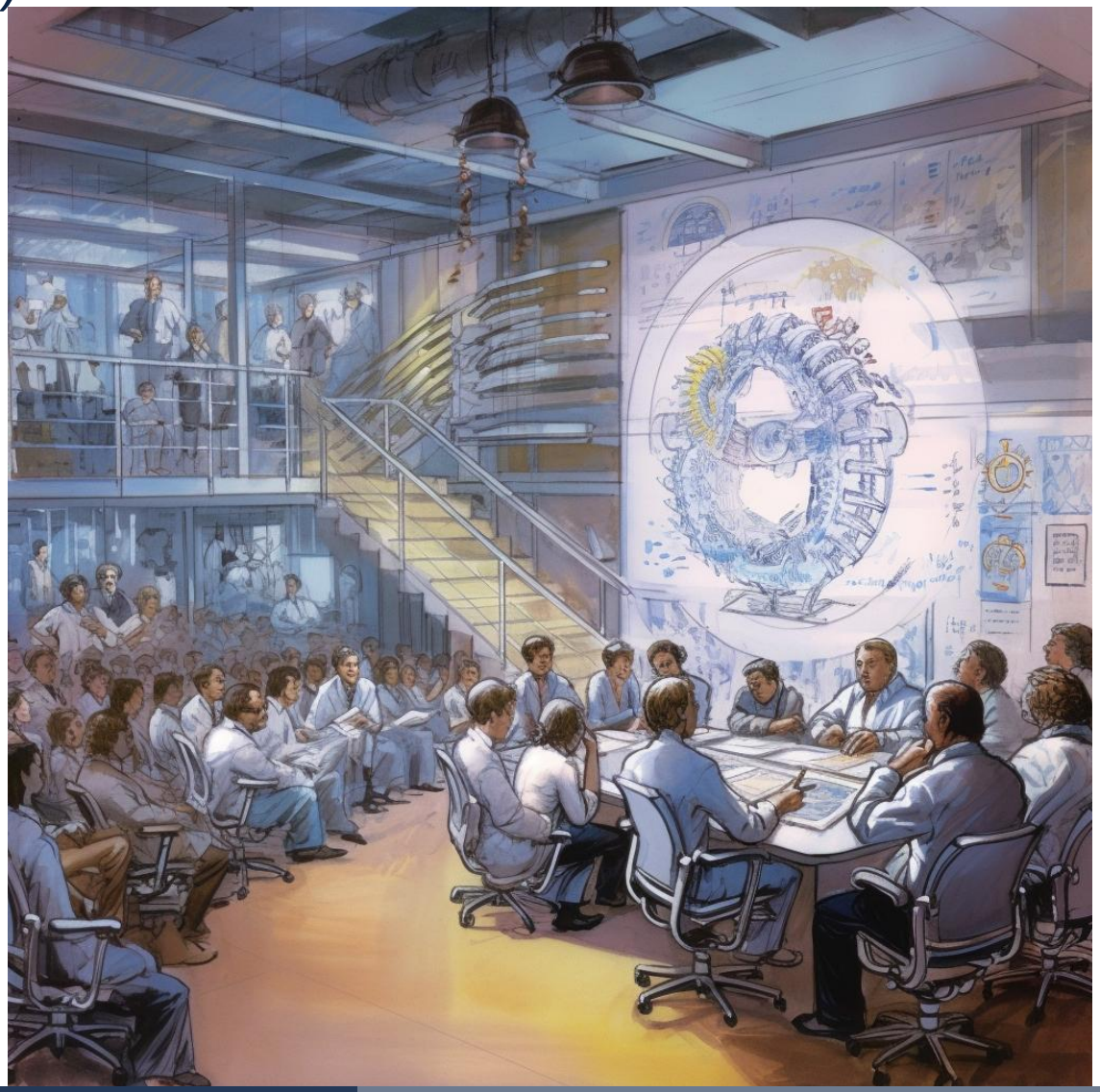
- 15: 44,994
- 21: 1,169
- 31: 559
- 41: 323
- 51: 225
- 61: 192

Genomic repeats are NOT random events

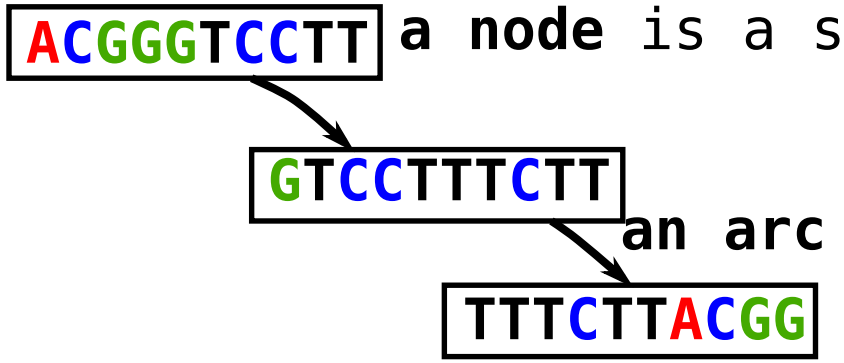
Greedy assemblers

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

- History: the human genome project while finding a successor to the greedy approach (according to MidJourney)



Graph representation



a node is a sequence

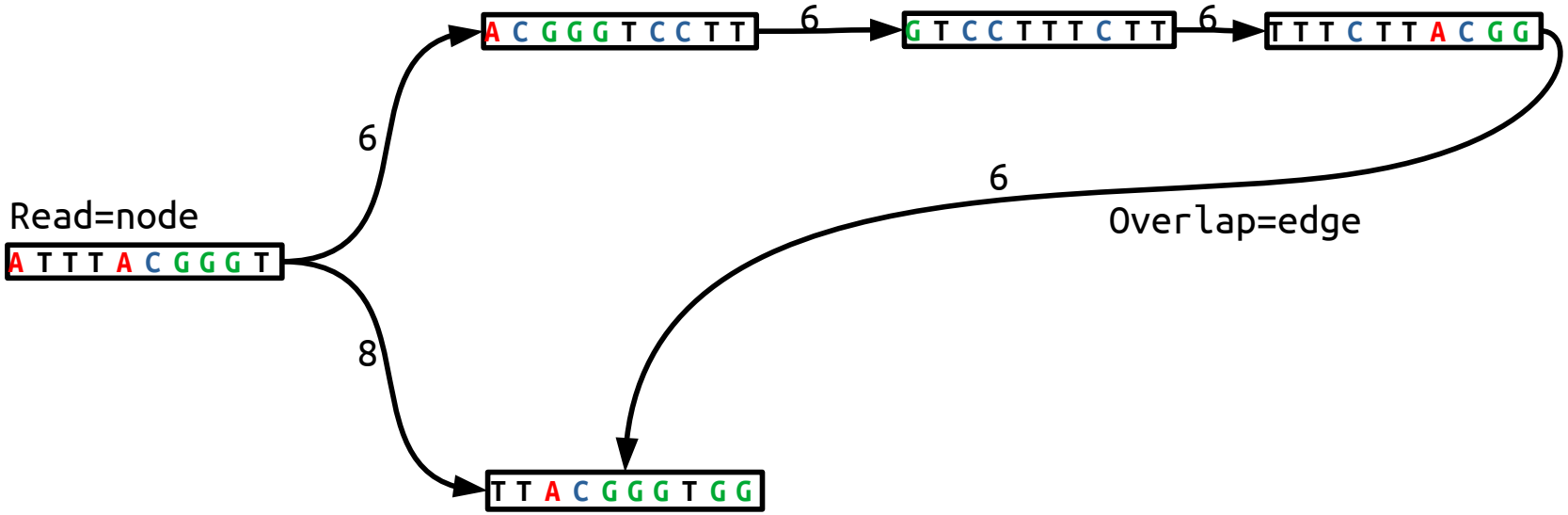
an arc oriented between
a source node and
a sink node

an arc means there is an overlap between the end of the source node and the beginning of the sink node

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

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

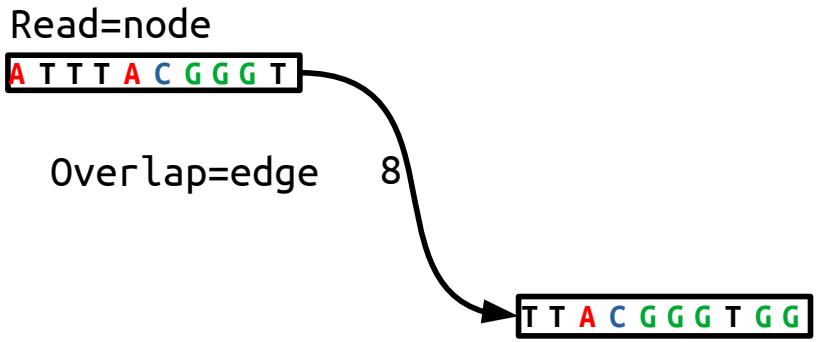
Overlap graph:



Greedy solution

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

Overlap graph:

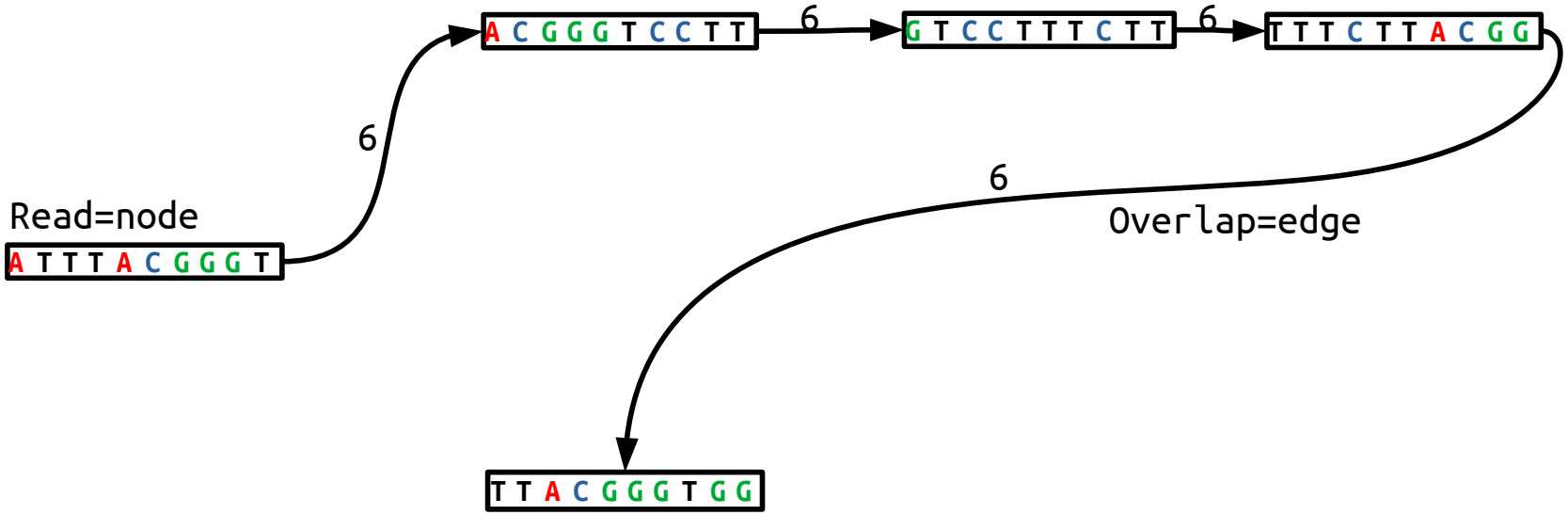


Greedy assembly output:
ATTTACGGGTGG
ACGGGTCCTTTCTTACGG

One piece solution

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

Overlap graph:

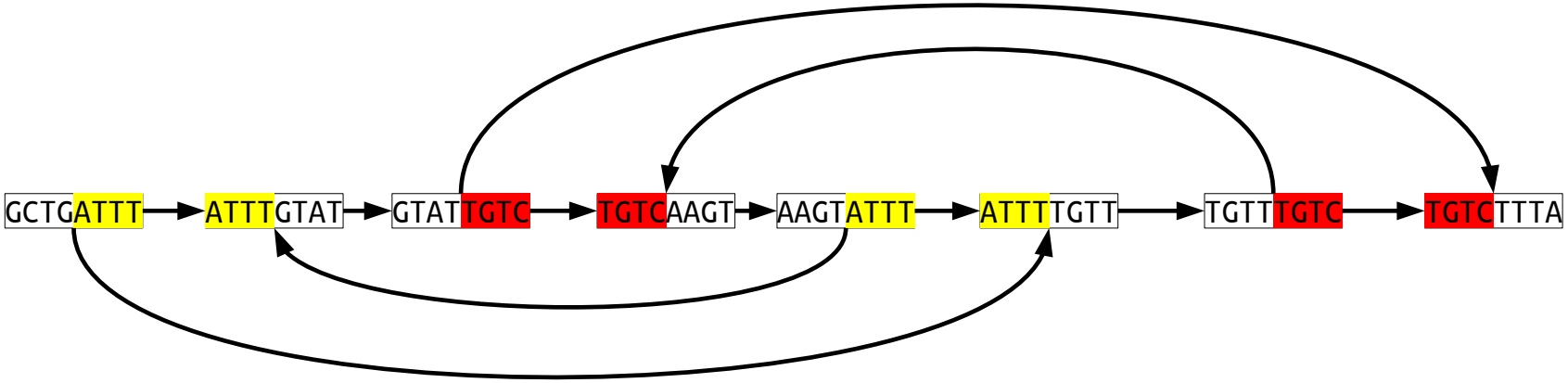


Overlap graph output:
ATTTACGGGTCCTTTCTTACGGGTGG

Multiple repeats

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

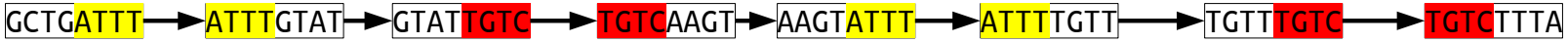
Overlap graph:



First solution

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

Overlap graph:

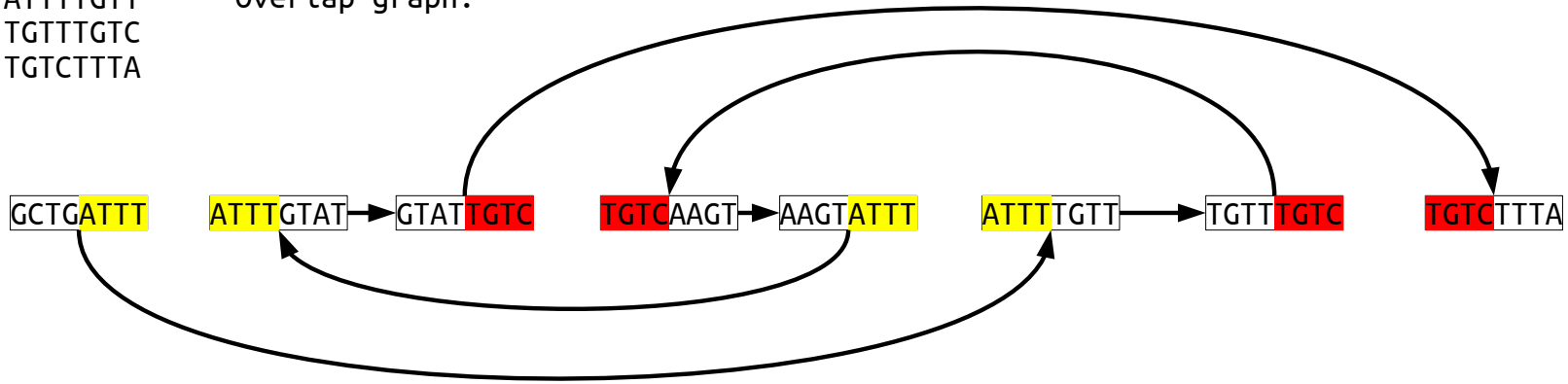



```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
```

Second solution

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

Overlap graph:

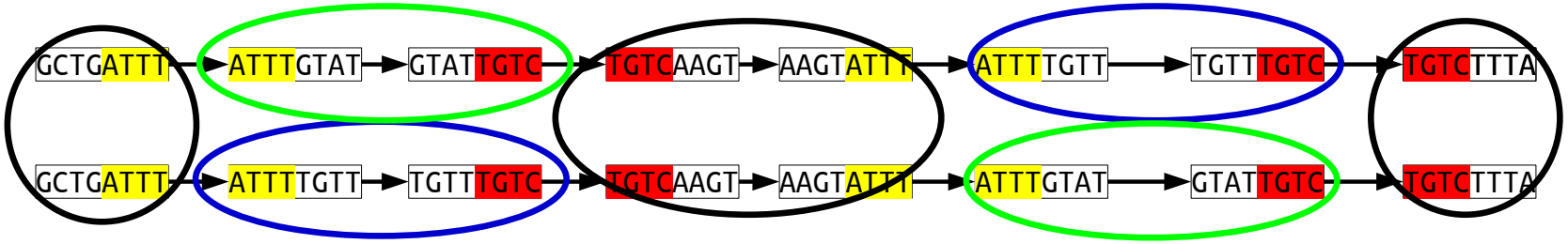



```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
GCTGATTTTGTTTGTCAAGTATTTGTATTGTCTTTA
```

Those two solutions are indistinguishable

Parsimonious solution: do not assemble

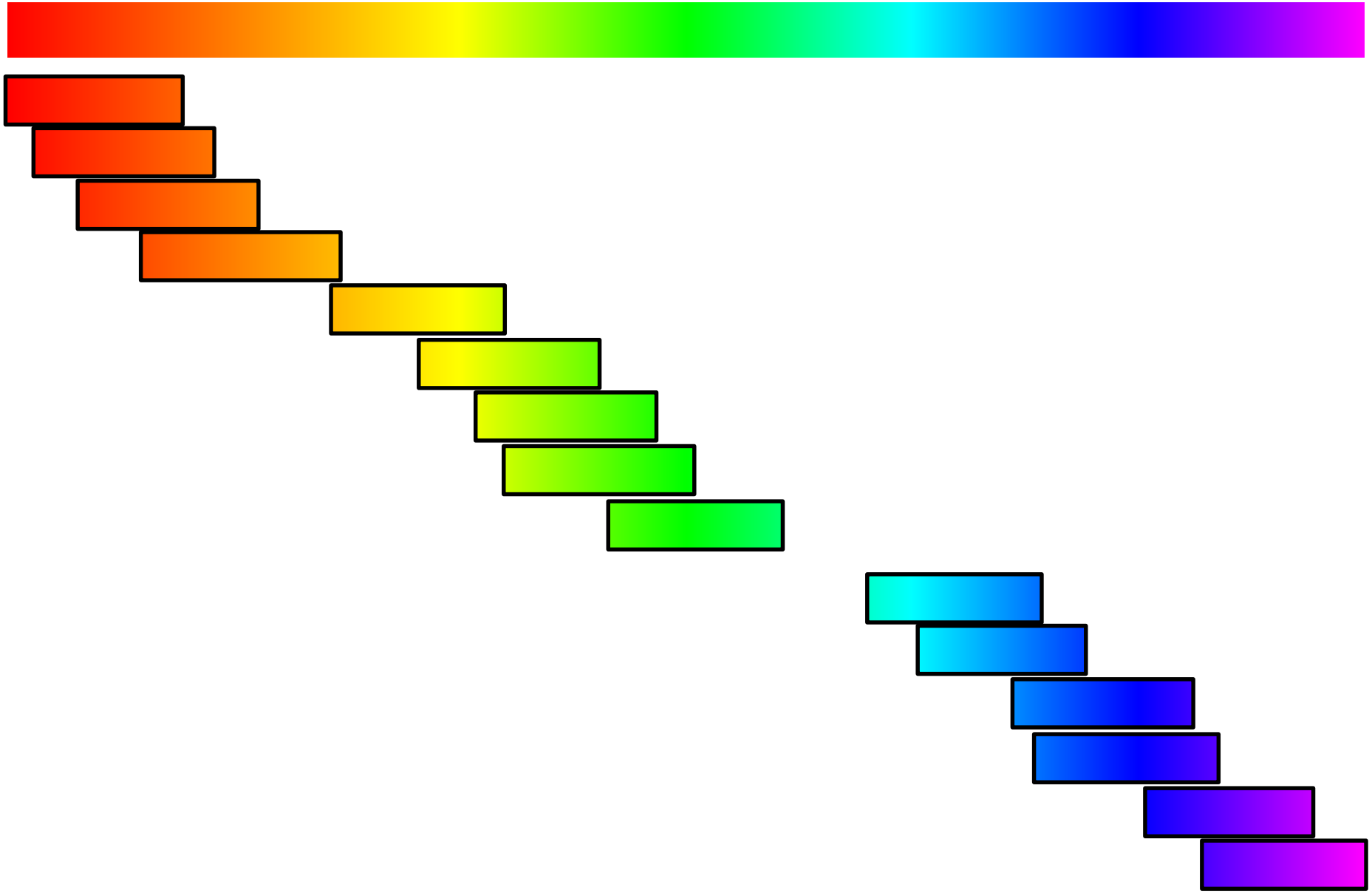
Possible assemblies:



Genome pieces:

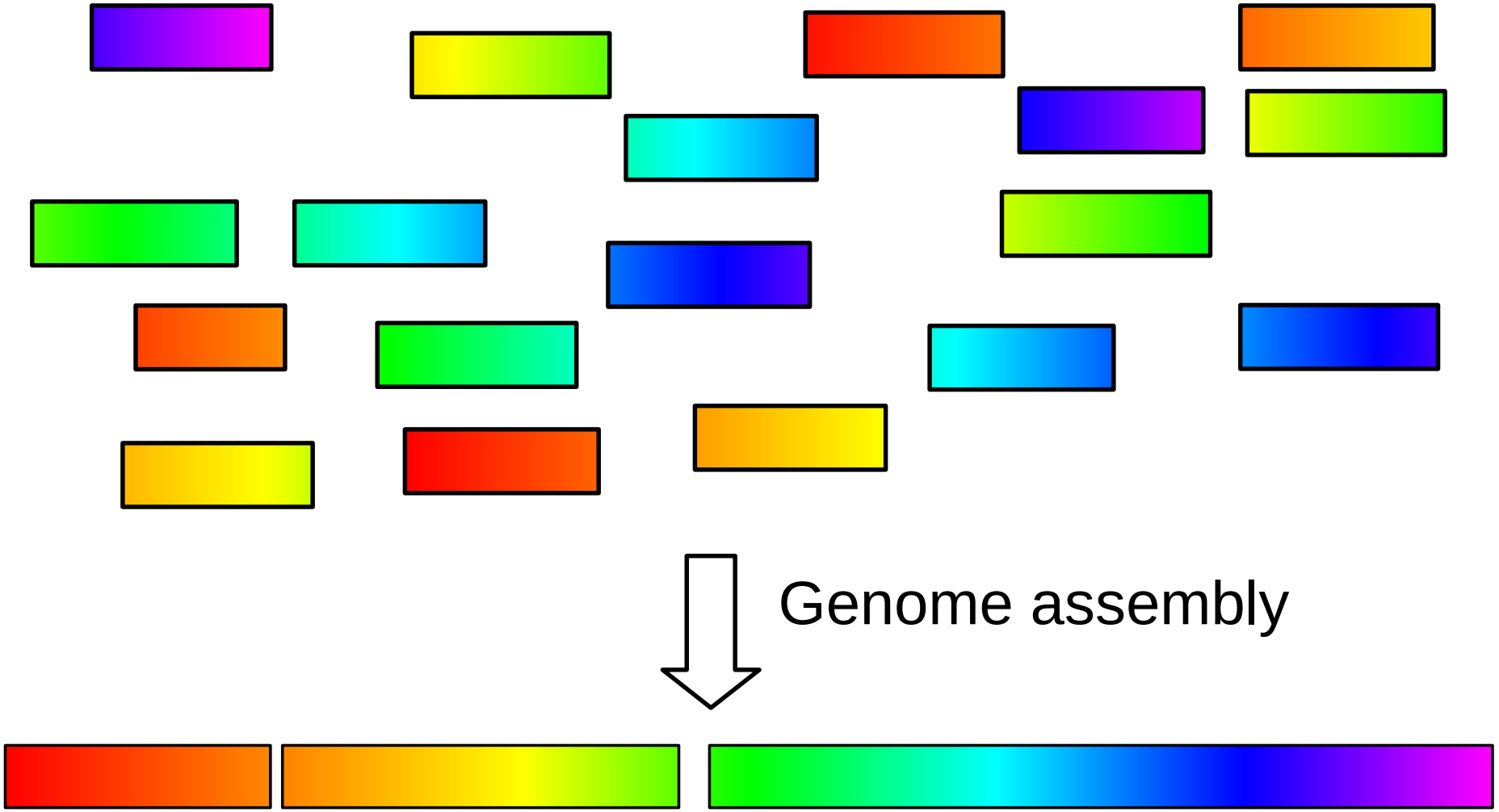
Repeats lead to the fragmentation of the assembly

- Missing information also fragments the assembly

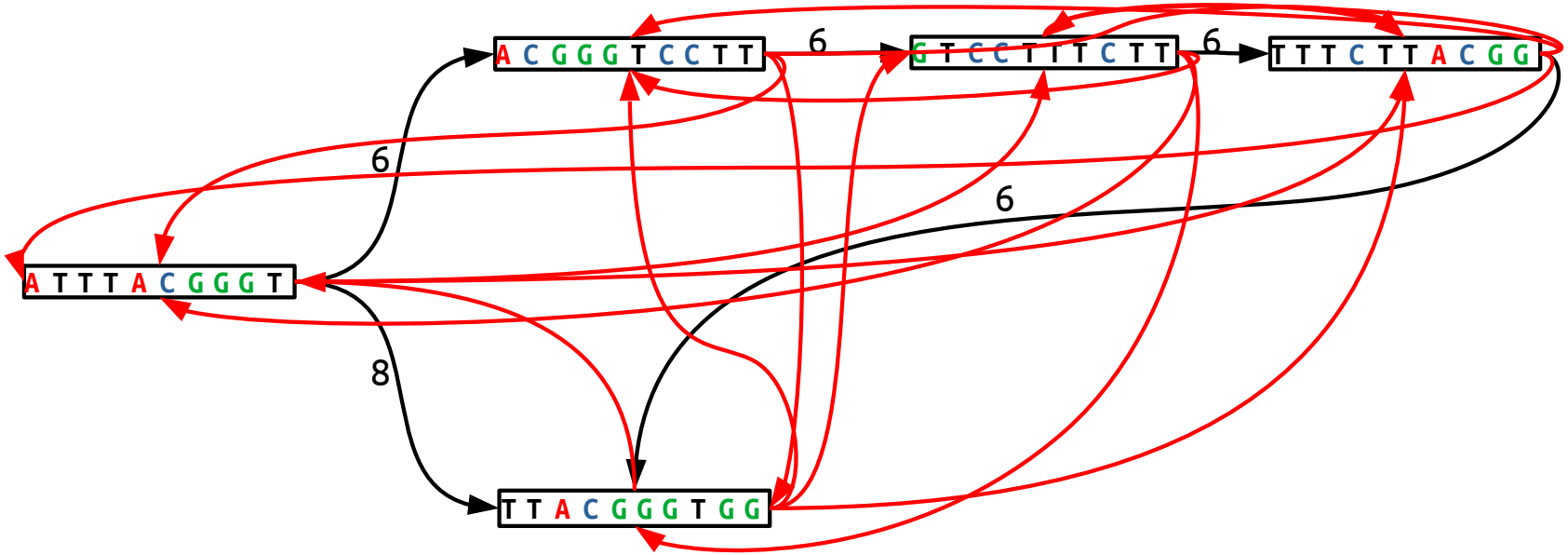


• Assembly **concession number 1: output fragments**

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

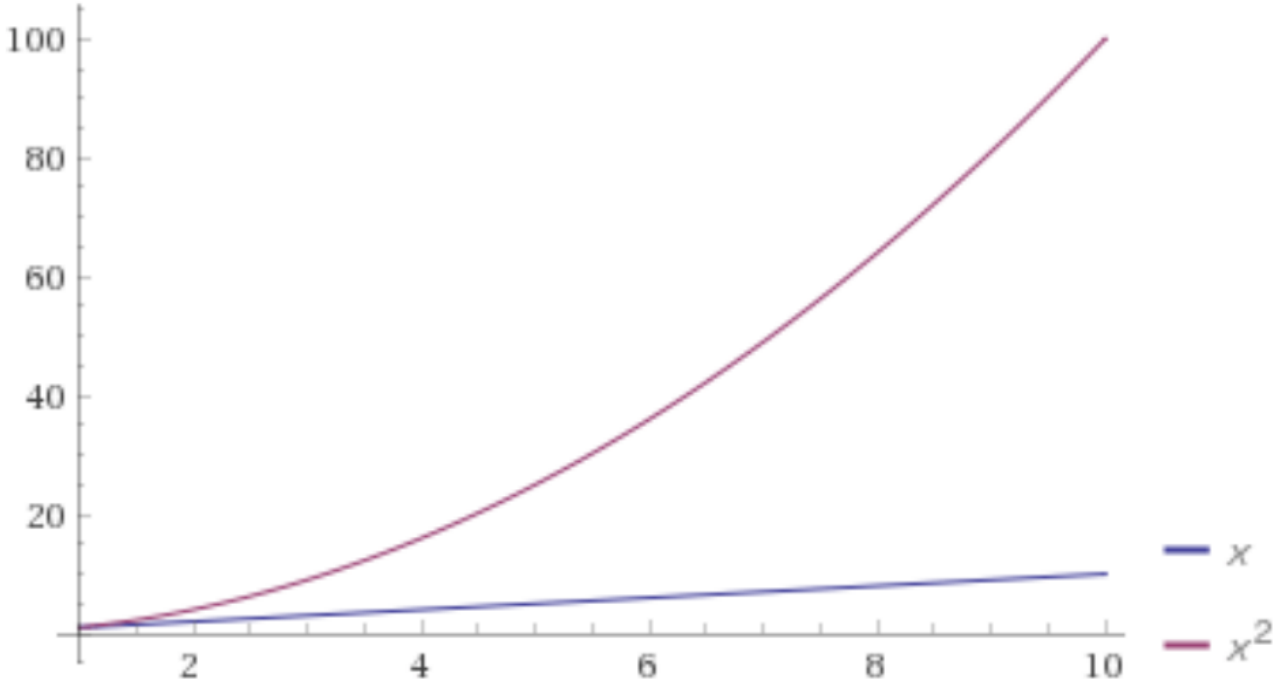


- Overlap graph prerequisite : all overlaps



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

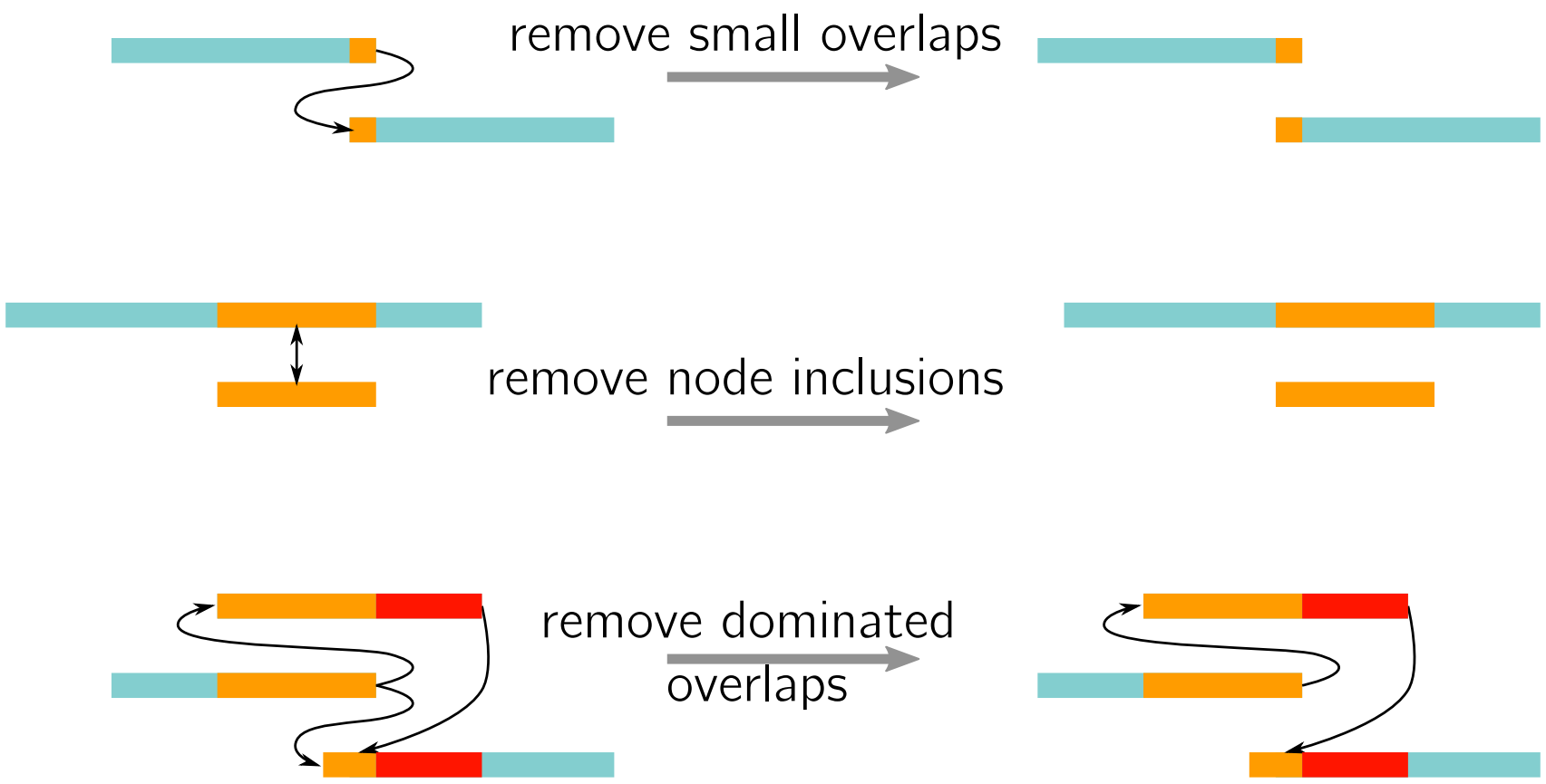
Overlap graph burden: number of reads

$n(n-1)/2 = \mathcal{O}(n^2)$ 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...

The overlap computation is not linear
Talking about CPU years on large genomes...
We have to be efficient here and focus on "relevant" overlaps

Overlap graph simplifications



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), ...)

- History: the introduction of another graph structure for assembly, according to MidJourney

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

a genome: a succession of words

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

GCATGC

CATGCT

- Find consecutive genome words in read words

genome

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

read TGACCGTAGCATGCT

TGACCG
GACCGT
...
GCATGC
CATGCT
?

k-mers
(k=6)

next nucleotide

- How to connect read words?

genome

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

reads TGACCGTAGCATGCT 1
GACCGTAGCATGCTA 2

k-mers (k=6)

GCATGC 1 2
CATGCT 1 2
ATGCTA 2

next nucleotide

- Reconstitute larger genomic words

genome

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

reads

TGACCGTAGCATGCT (1)
GACCGTAGCATGCTA (2)

TGACCG (1)
GACCGT (1) (2)
...
TAGCAT (1) (2)
AGCATG (1) (2)
GCATGC (1) (2)
CATGCT (1) (2)
ATGCTA (2)

extract the read's *k*-mers (*k*=6)

TGACCGTAGCATGCTA
a sequence from the genome

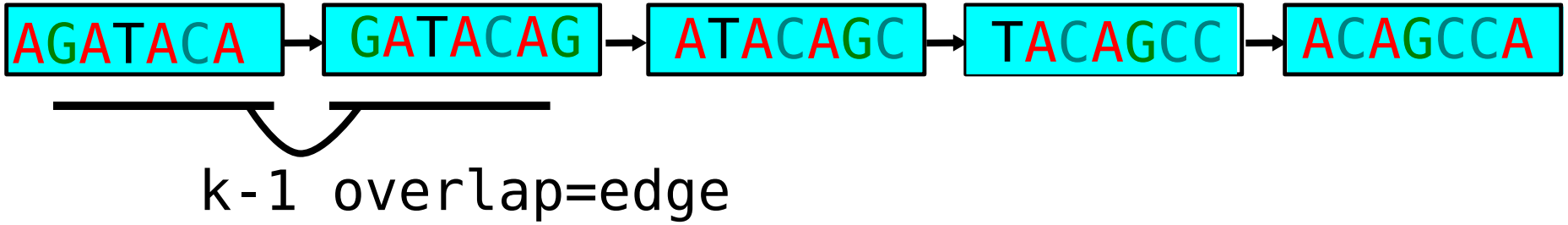
The de Bruijn graph

Read

AGATACAGCCA

De Bruijn graph

Kmer=node



AGATACA + G + C + C + A

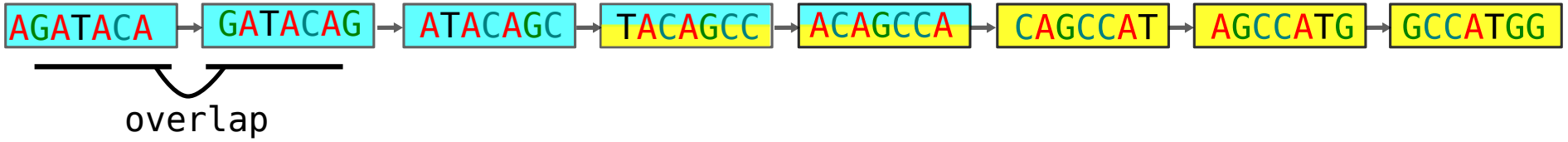
=AGATACAGCCA

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

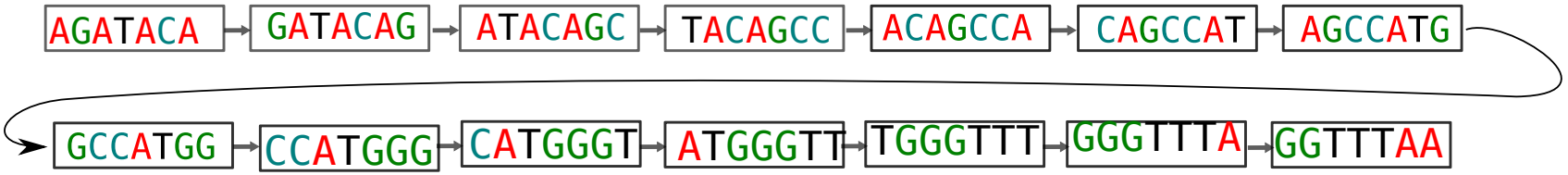
Hint: Use 7-mers

Solution

Overlapping reads

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

De Bruijn graph



Resulting sequence

AGATACAGCCATGGGTTTAA

de Bruijn graphs abstract redundancy

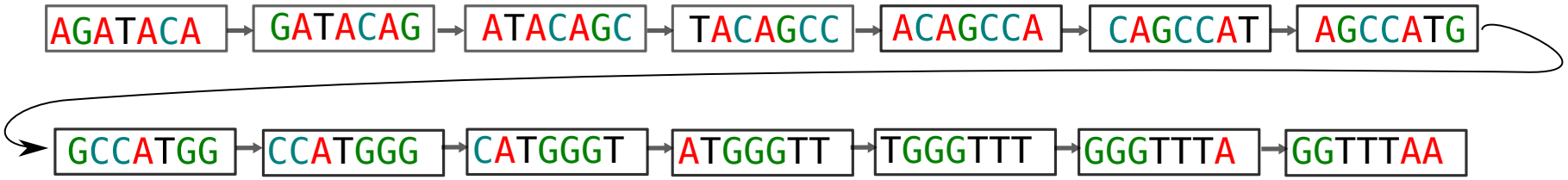
Overlapping reads

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

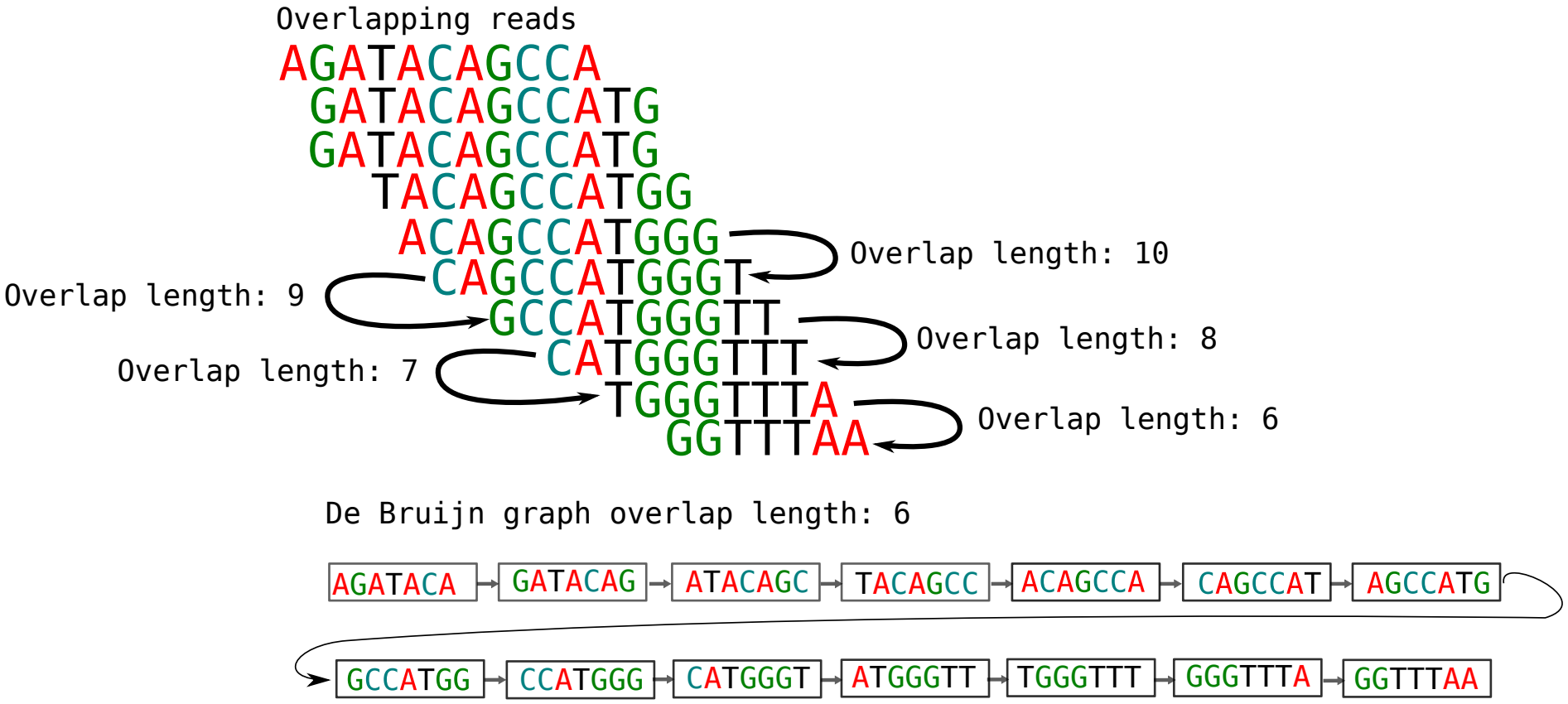
62 **(non distinct)** 7-mers in the reads

De Bruijn graph

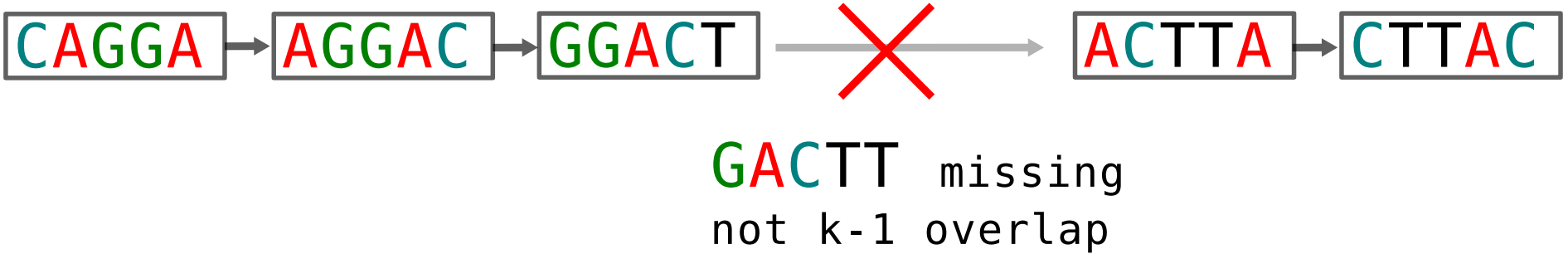
14 **distinct** 7-mers in the De Bruijn graph



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



- de Bruijn graphs limitation 1: Fixed overlaps



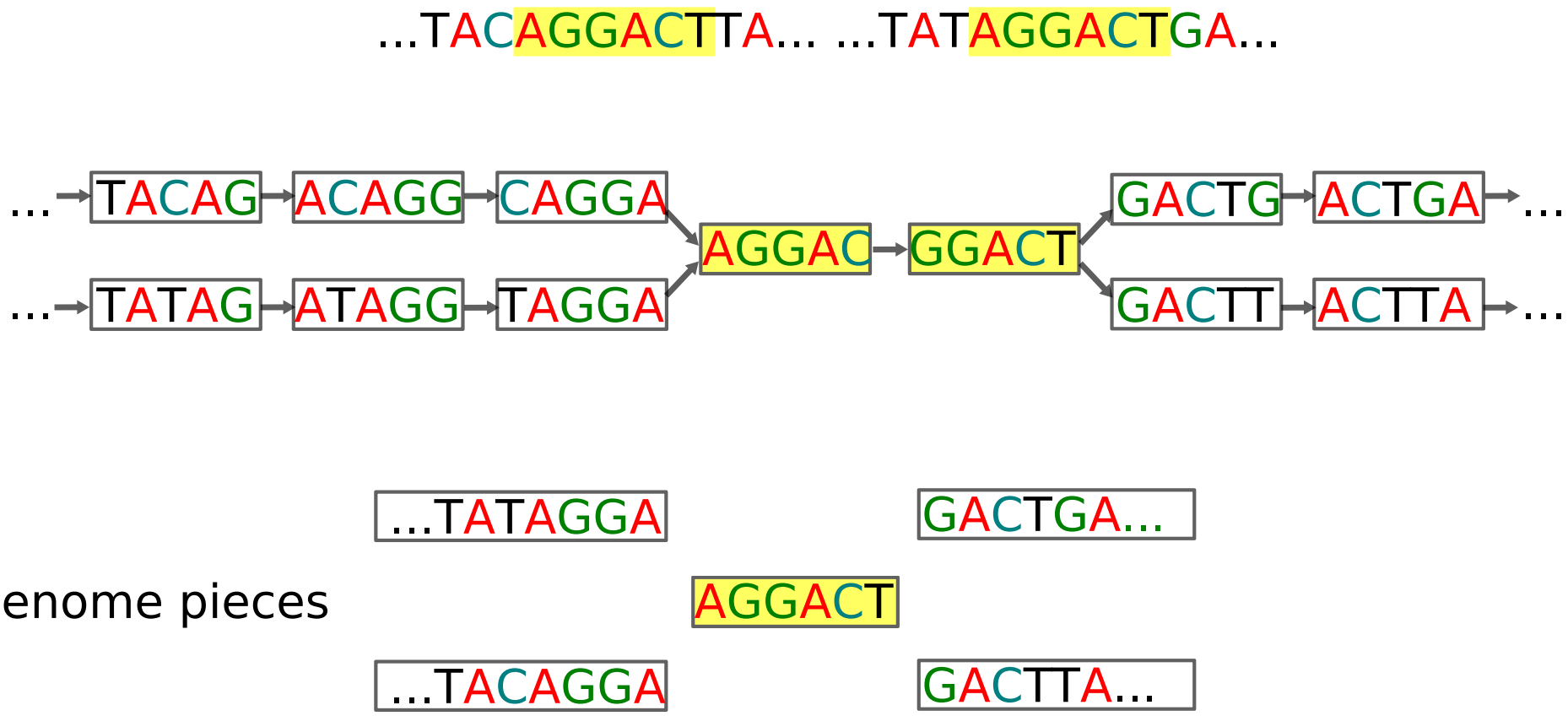
GGACT and ACTTA overlap is only of size 3 !

de Bruijn graphs limitation 2: Repeats

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

each k-mer appears only once in a de Bruijn graph

de Bruijn graph limitation



de Bruijn graph versus overlap graph

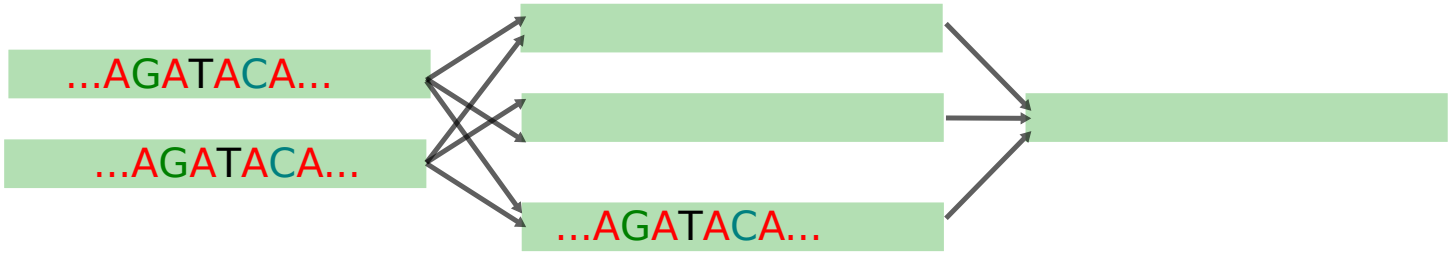
reads

words from the reads

AGATACA
GATACAG TACAGCC AGCCATG
ATACAGC ACAGCCA CAGCCAT
...

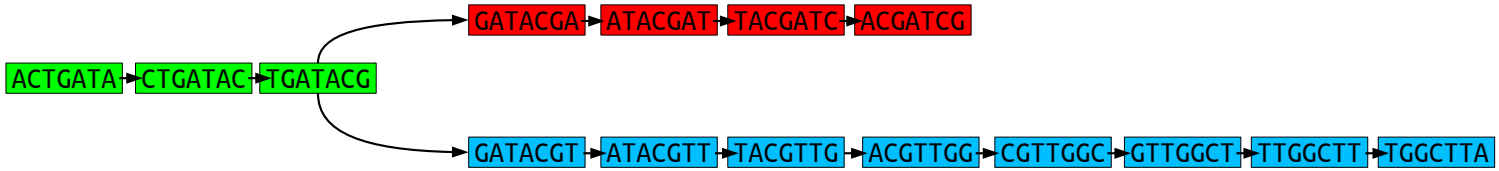
word graph (de Bruijn graph)

Overlap graph from the reads

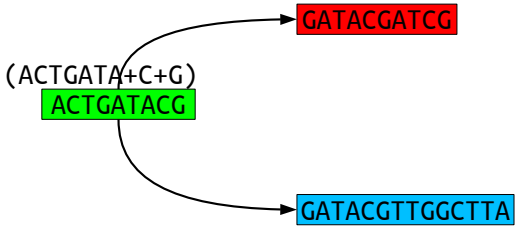


On the representation of de Bruijn graphs

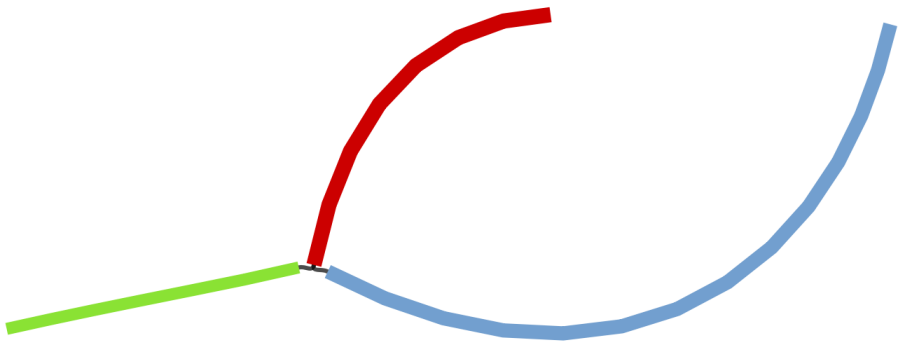
De Bruijn graph:



Compacted De Bruijn graph:

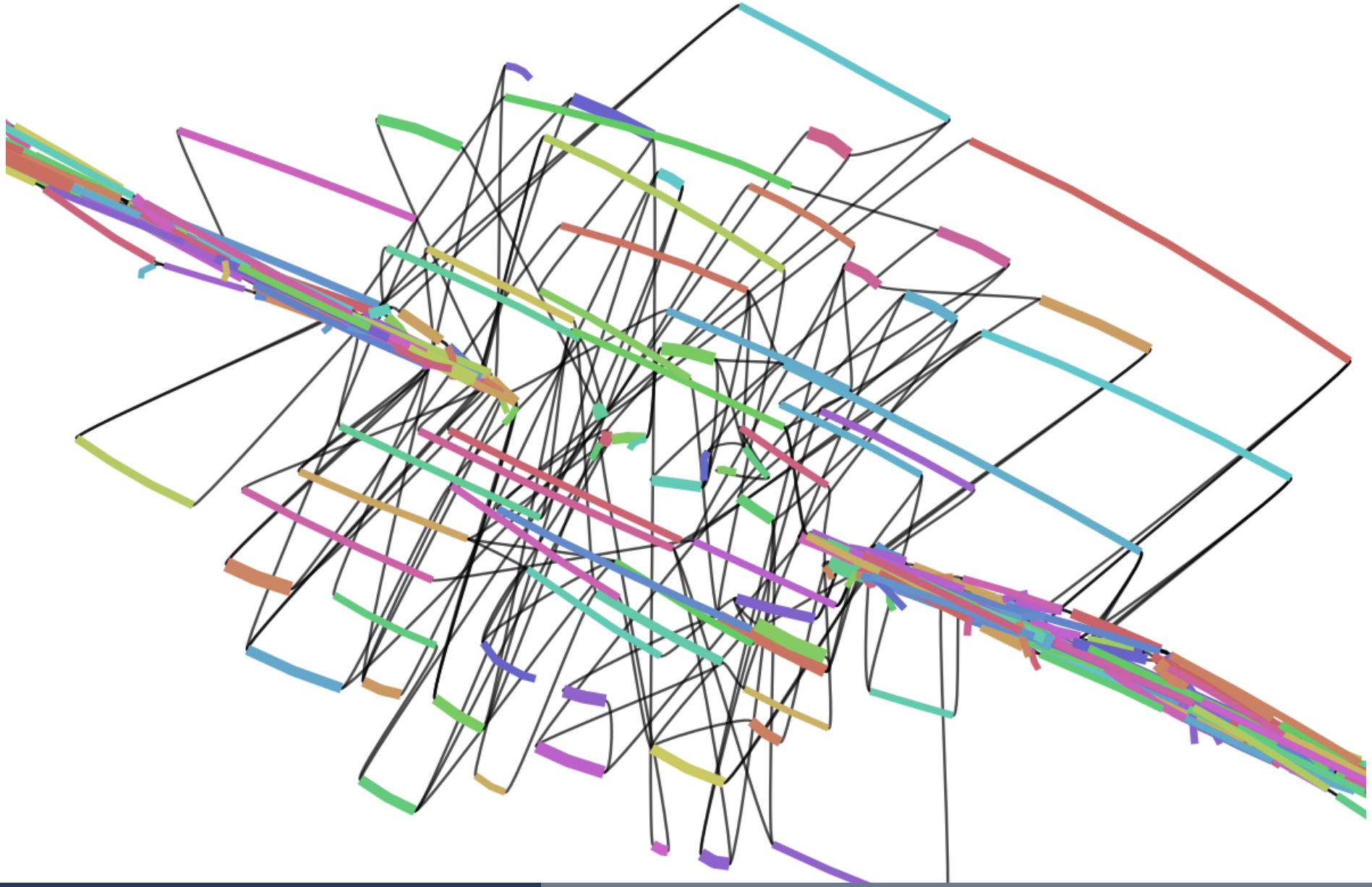


Graphical representation
(.gfa plot using Bandage):



- de Bruijn graph on a real dataset

- de Bruijn graph on a real dataset ZOOMED IN



- Sequencing errors

```
Genome:
 ATCGGTATCGTTACGGTATACC

Reads:
 ATCGCTATCG
    GGTTTCGTTA
        ATCGATACGG

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

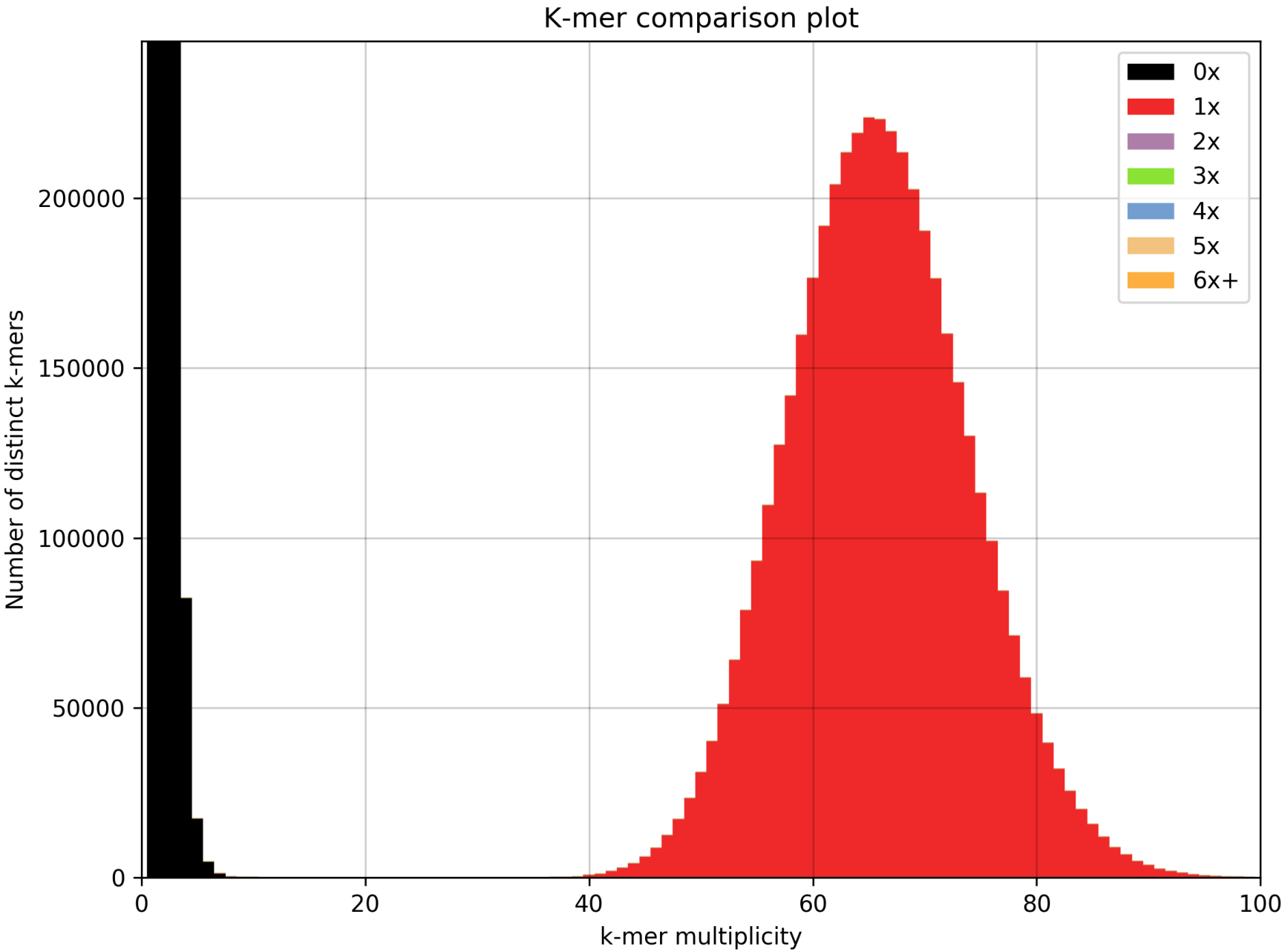
Erroneous k-mers vs genomic k-mers

```
Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACA

Reads:
TAAGAAAGCTCTGAATCA
 AAGAAAGCTCTAAATCAAC
  AGAAAGCTCTGAATCAACG
   GAAAGCTCTGAATCAACGGA
    AAAGCTCTGAATCAACGGAC
     AAGCTCTGAATCAACGGACT
      AGCTCTGAATCAACGGACTG
       GCTCTGAATCAACGGTCTGC
        CTCTGAATCAACGGACTGCG
         TCTGAATCAACGGACTGCGA
```

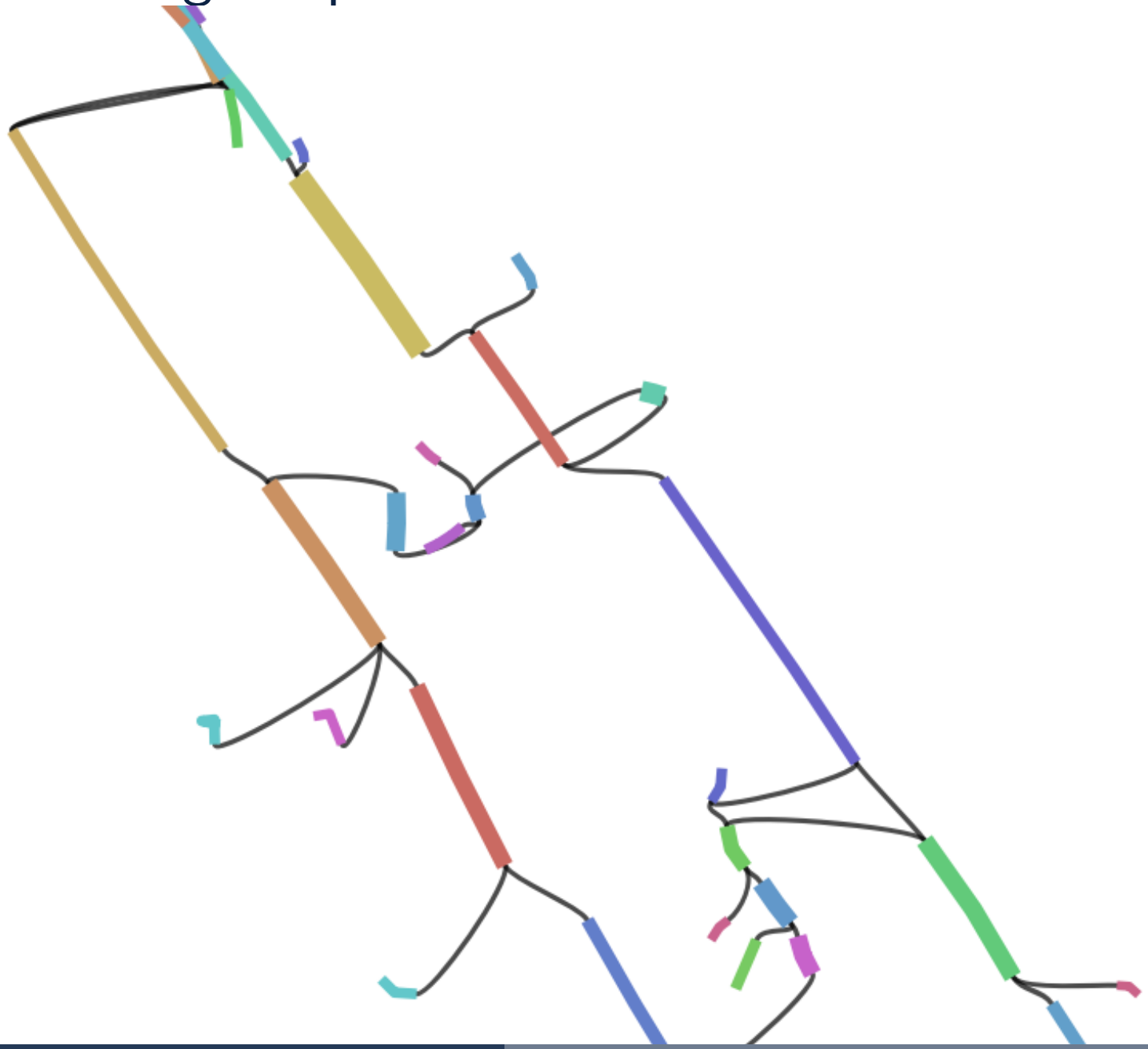
```
9 times   TCTGAAT
1 time    TCTAAAT

6 times           CAACGGA
1 time            CAACGGT
```

Erroneous k-mers are seen less than genomic ones

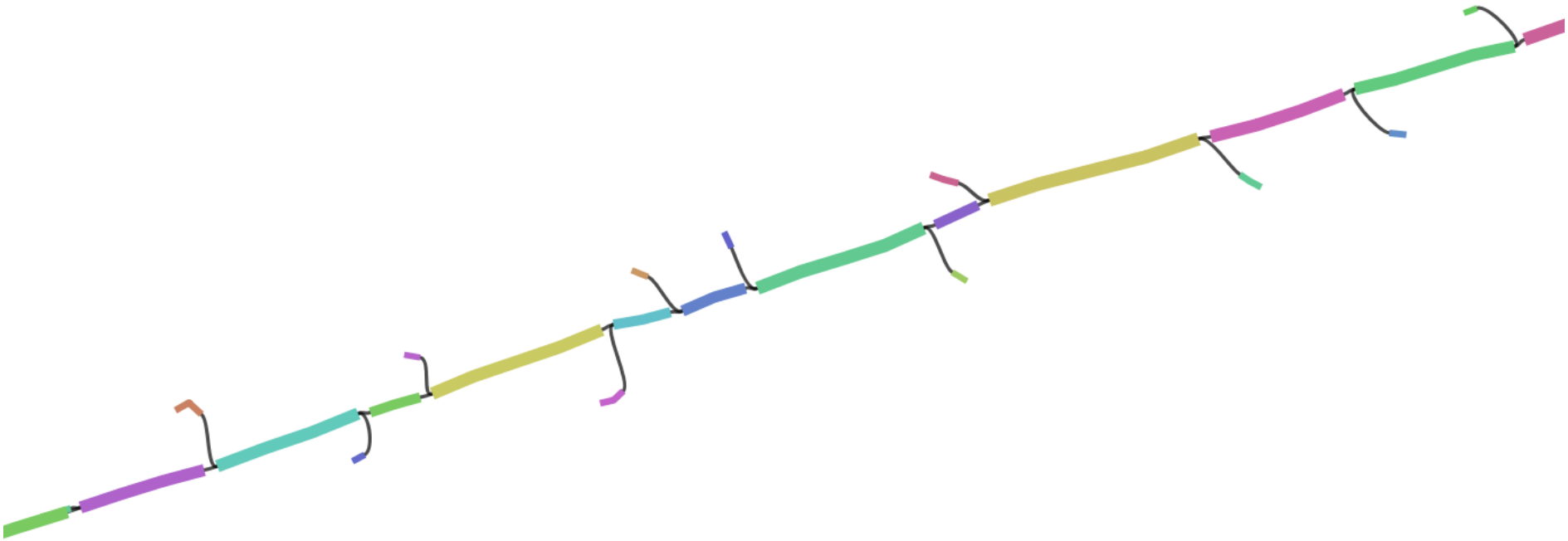
K-mer histogram



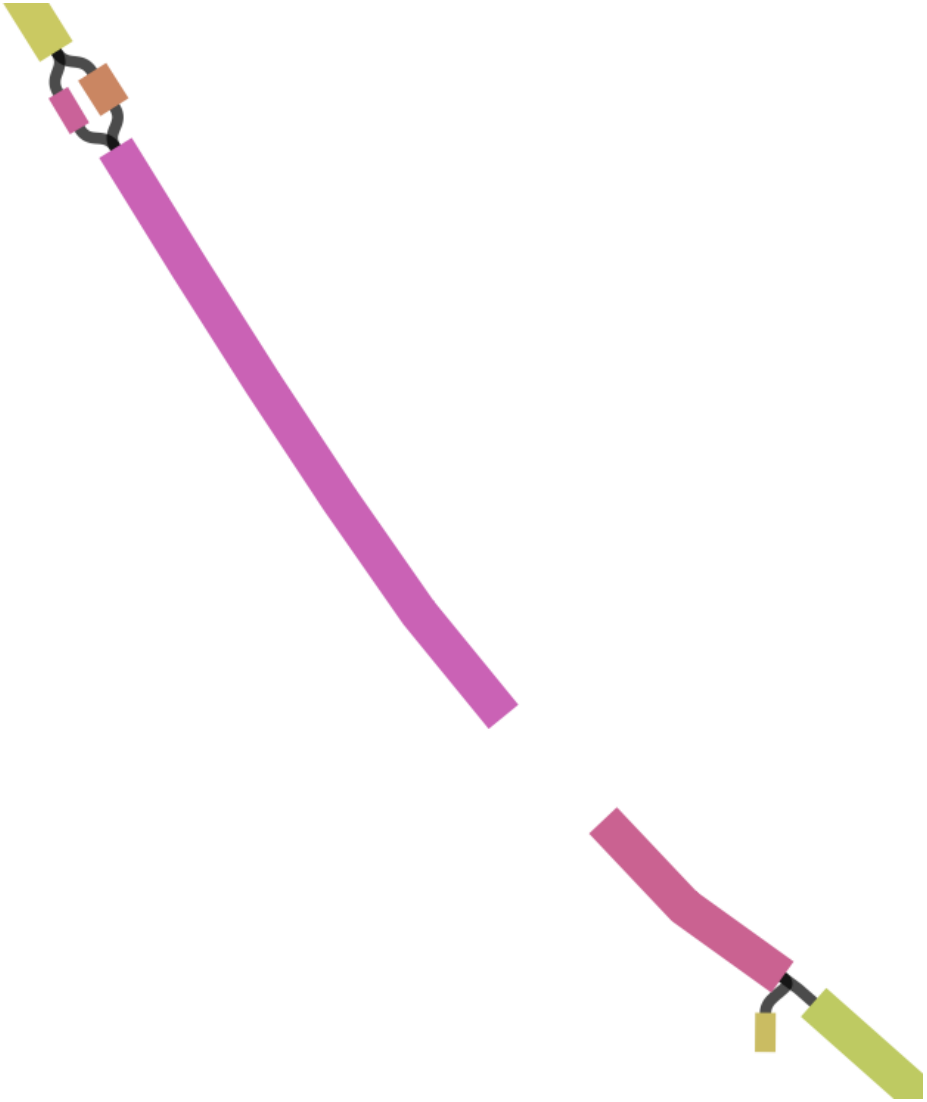
- Removing unique k-mers



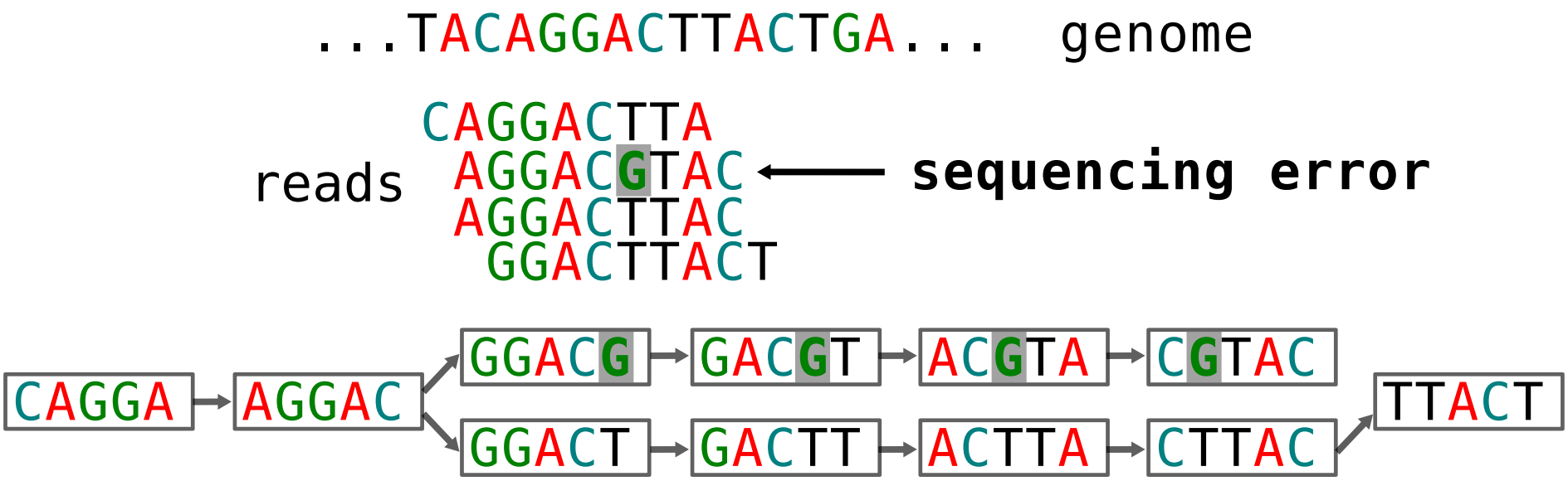
- Removing k-mers seen less than 3 times



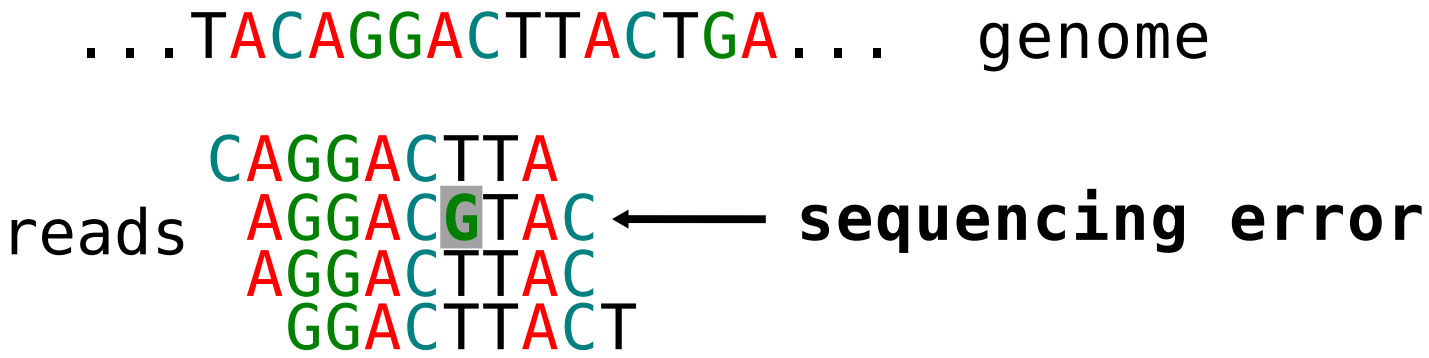
- Removing k-mers seen less than 4 times

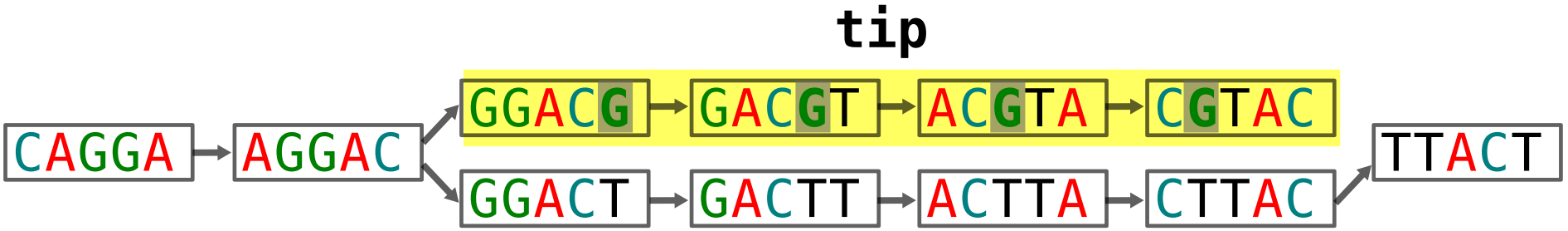


Errors in de Bruijn graphs

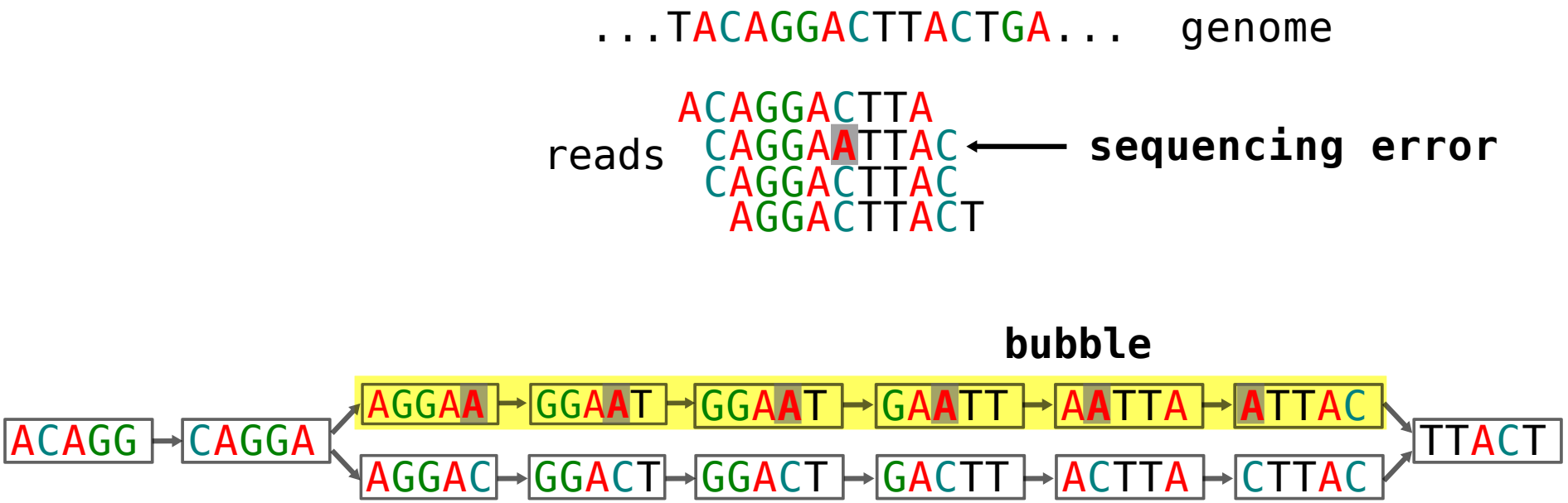


Errors in de Bruijn graphs

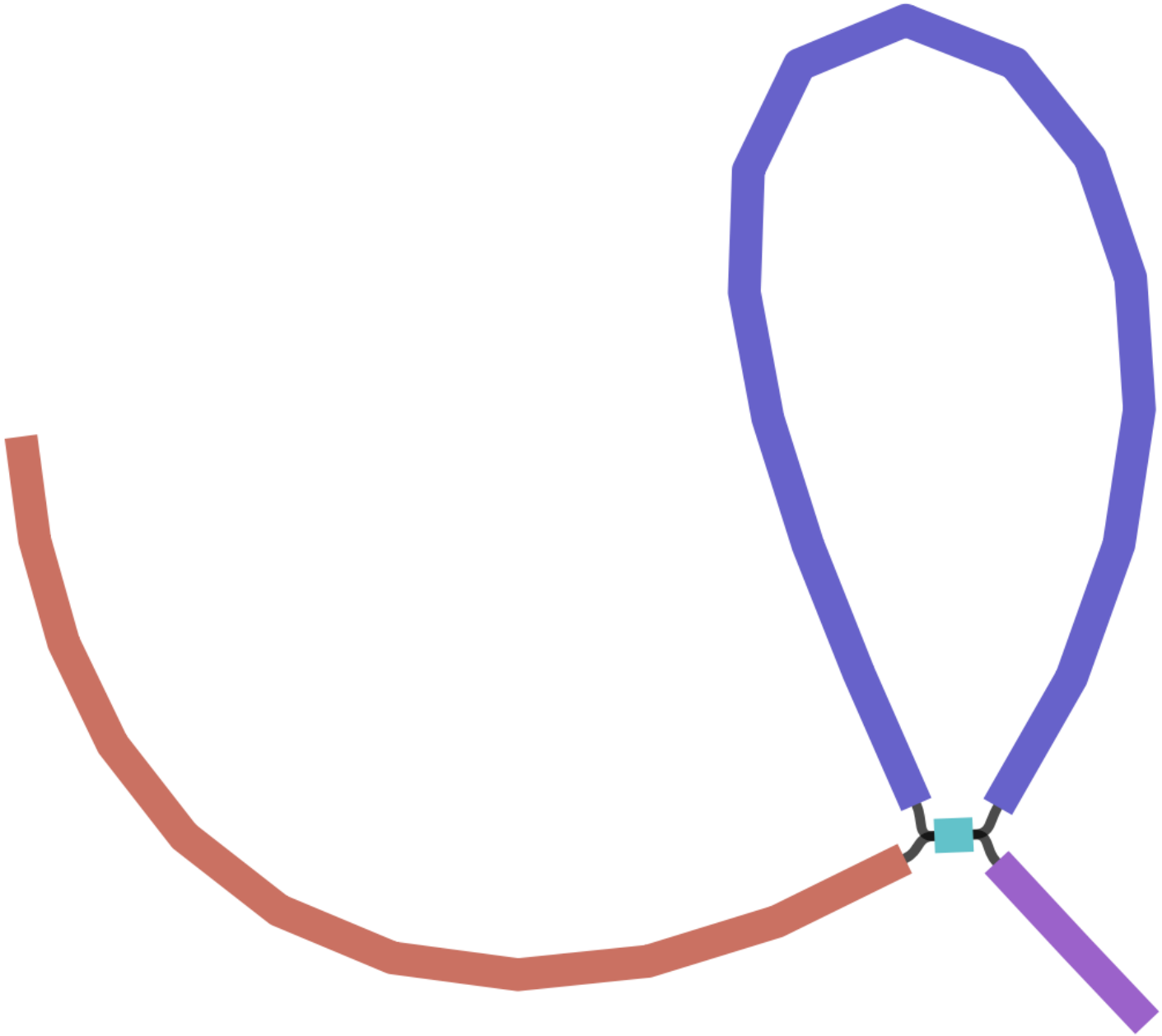




Errors in de Bruijn graphs



- (Almost assembled phage !)



de Bruijn 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

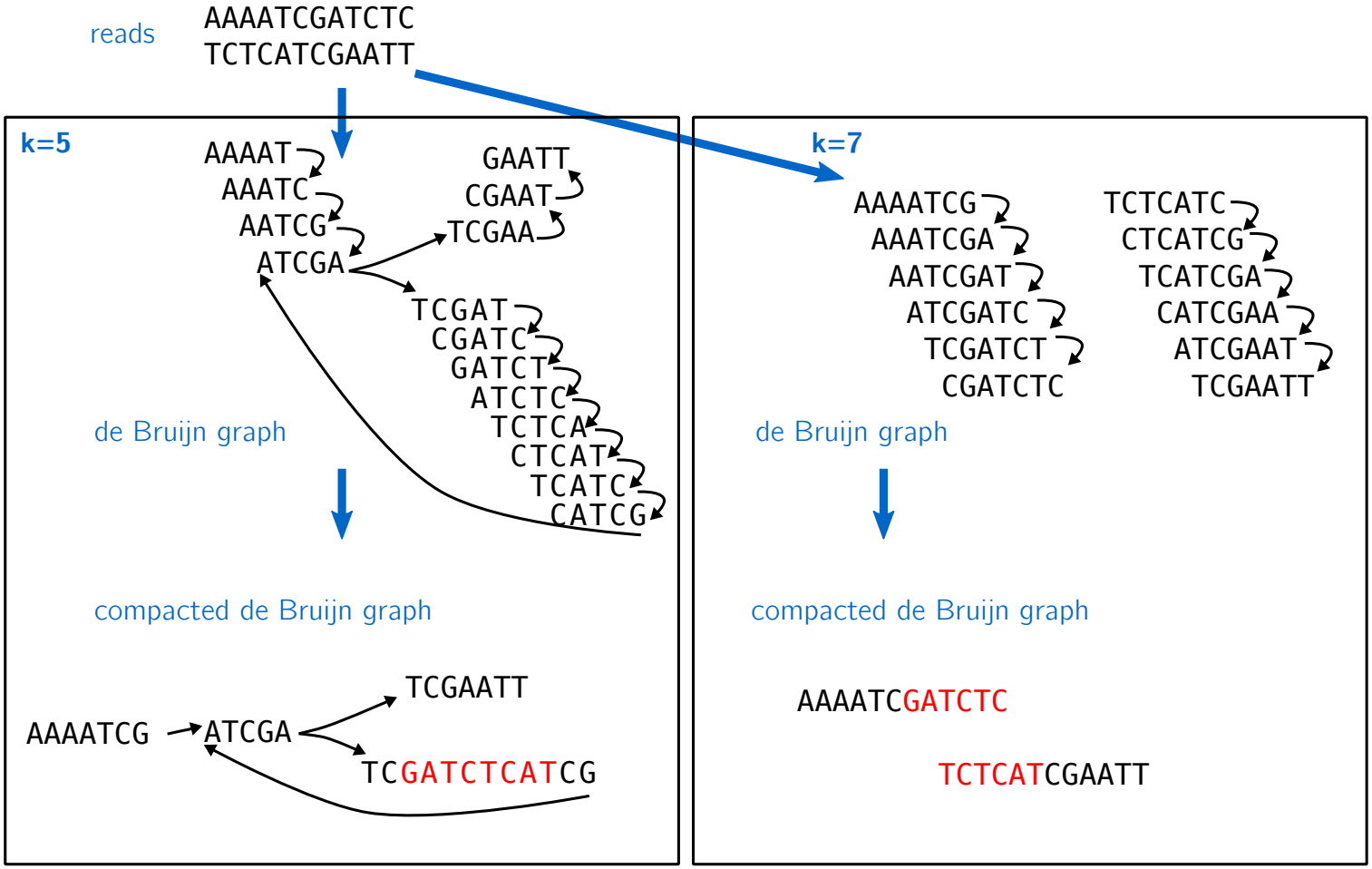
Multiple k assembly

Most de Bruijn graph assemblers can now perform several assemblies with different k-mer sizes to produce an improved "super" assembly

Exercice

Build DBG with k=5 and k=7 from those reads
AAAATCGATCTC
TCTCATCGAATT

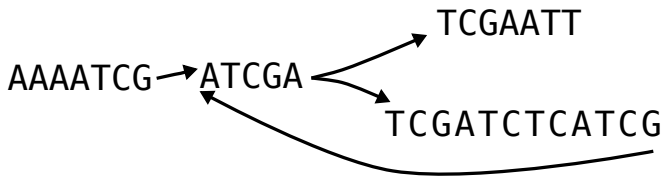
• Multiple k assembly



We are missing GATCTCA and ATCTCAT in the second graph.
But they are present in the first graph!

Multiple k assembly

compacted de Bruijn graph
k=5



k-mers k=7
from cdBG k=5

```
AAAATCG
 AAATCGA
          TCGATCT
           CGATCTC
            GATCTCA
TCGAATT      ATCTCAT
              TCTCATC
               CTCATCG
```

+

k-mers k=7
from the reads

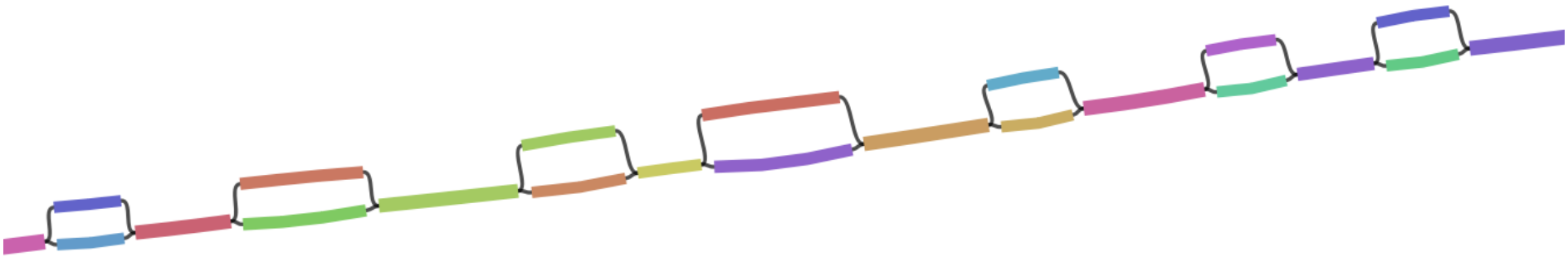
```
AAAATCG TCTCATC
 AAATCGA CTCATCG
  AATCGAT TCATCGA
   ATCGATC CATCGAA
    TCGATCT ATCGAAT
     CGATCTC TCGAATT
```

→

compacted de Bruijn graph
k=7

AAAATCGATCTCATCGAATT

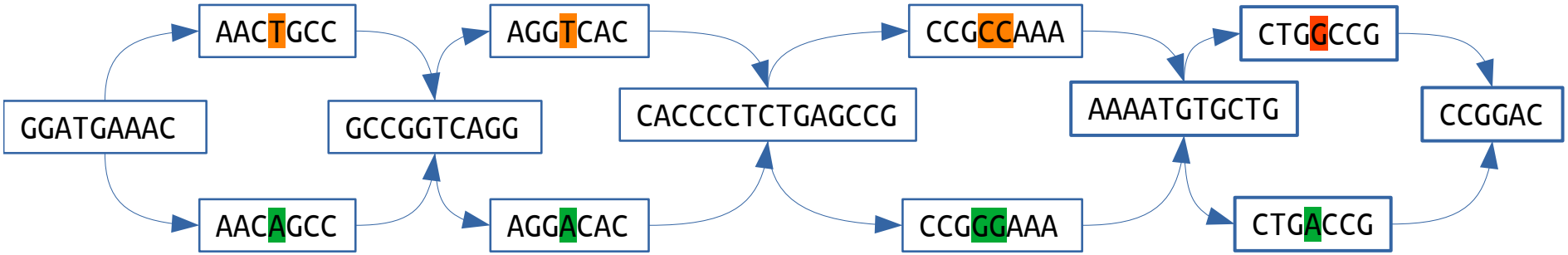
- de Bruijn graph on an eukaryota



- Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

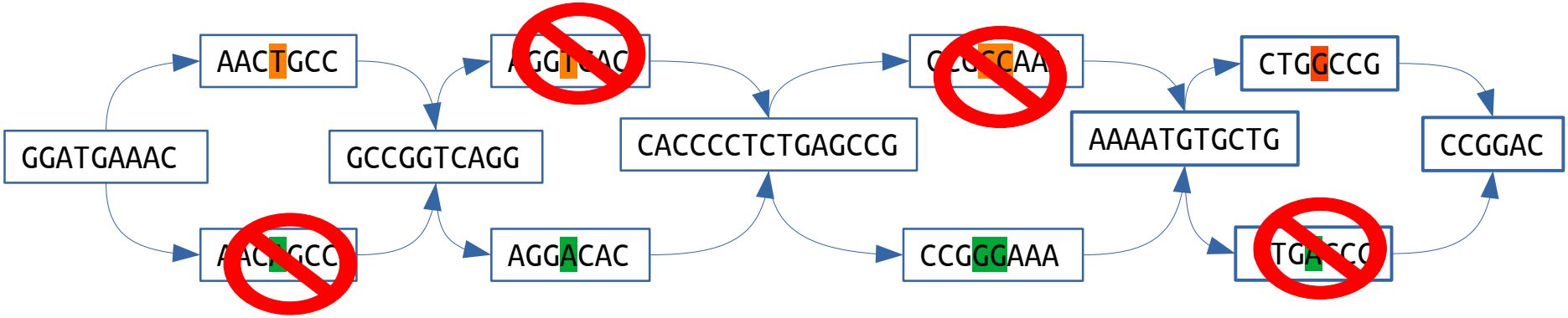
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

- Assembly **concession number 2**: collapse variability

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

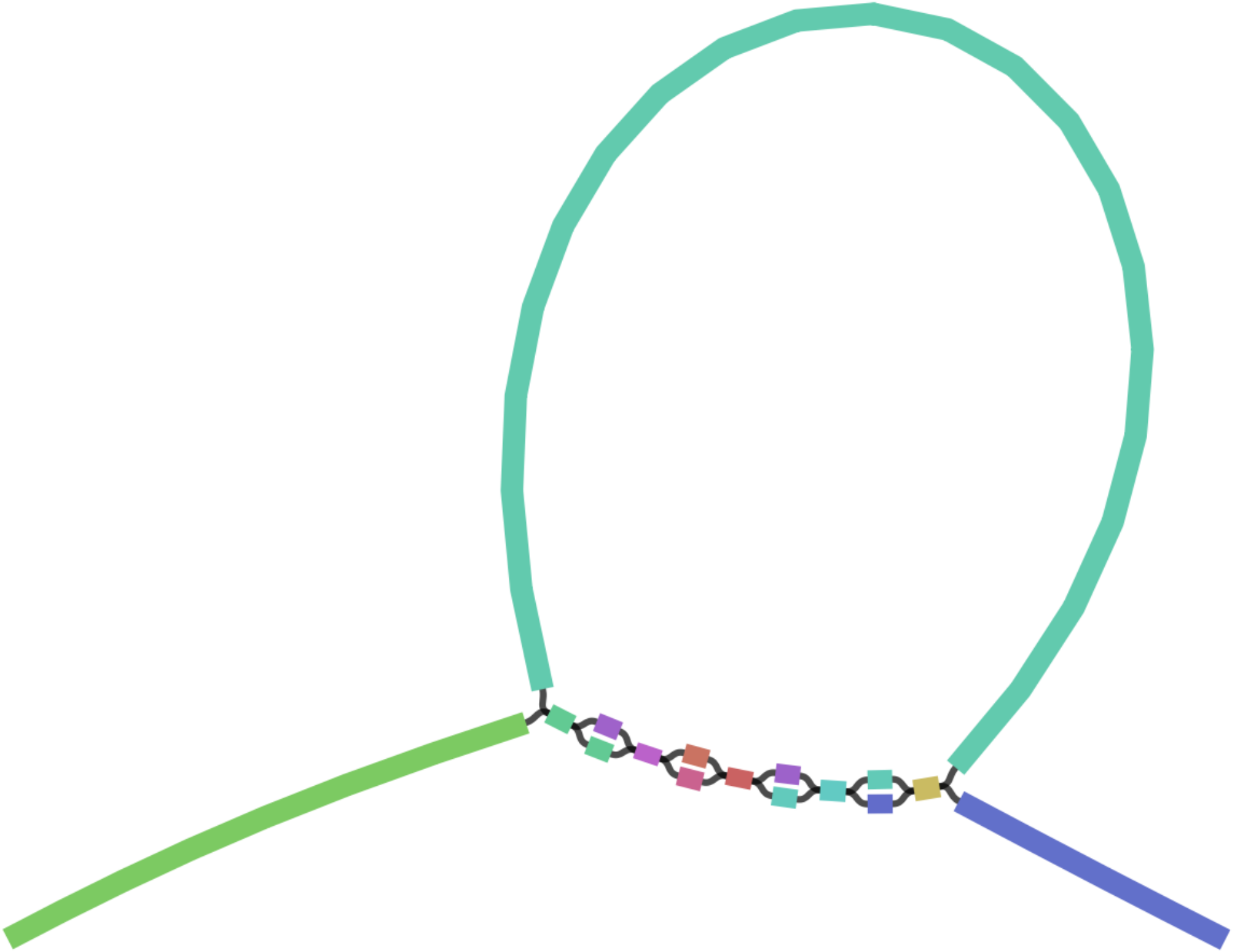
Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

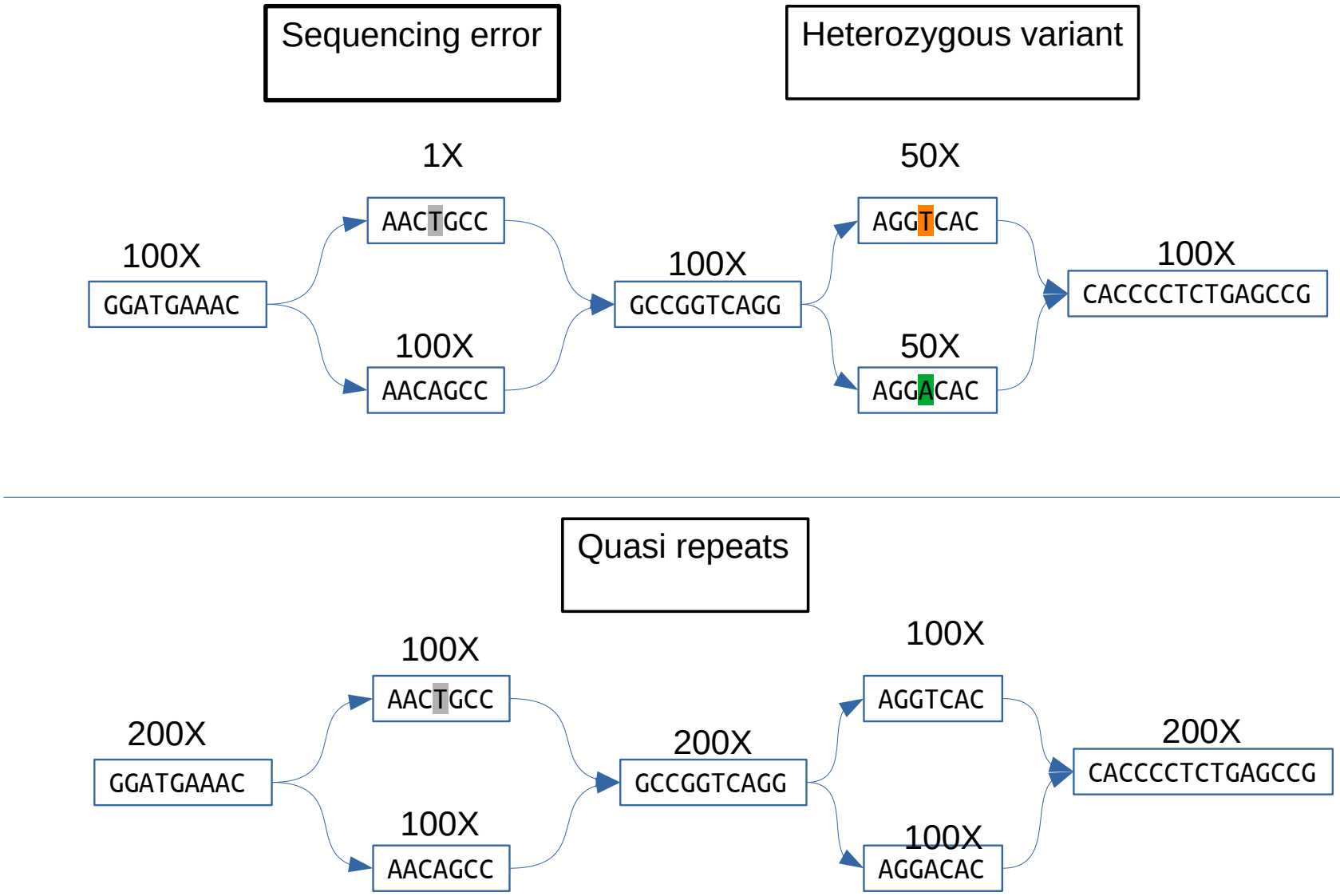
Reads:

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

- Paralog genes/repeats



Paralog genes/repeats in graph



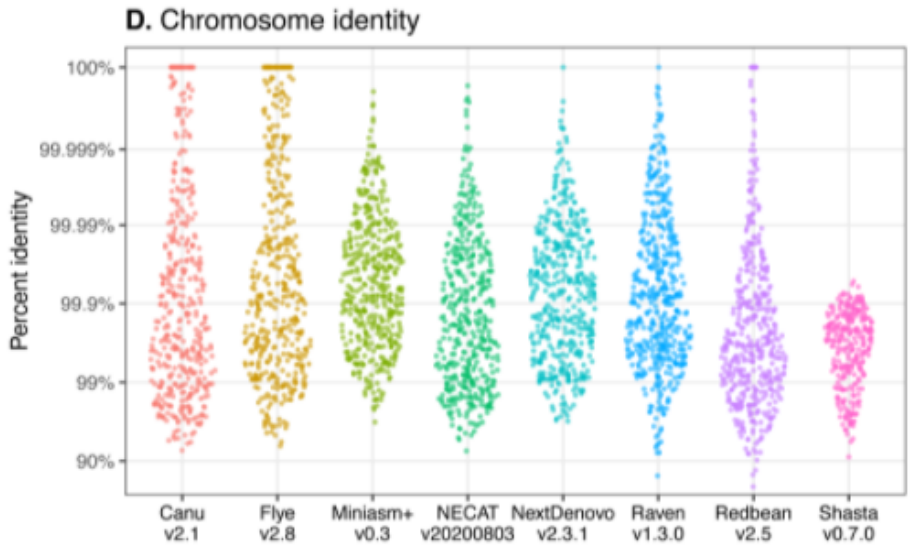
- An assembler is a set of heuristics

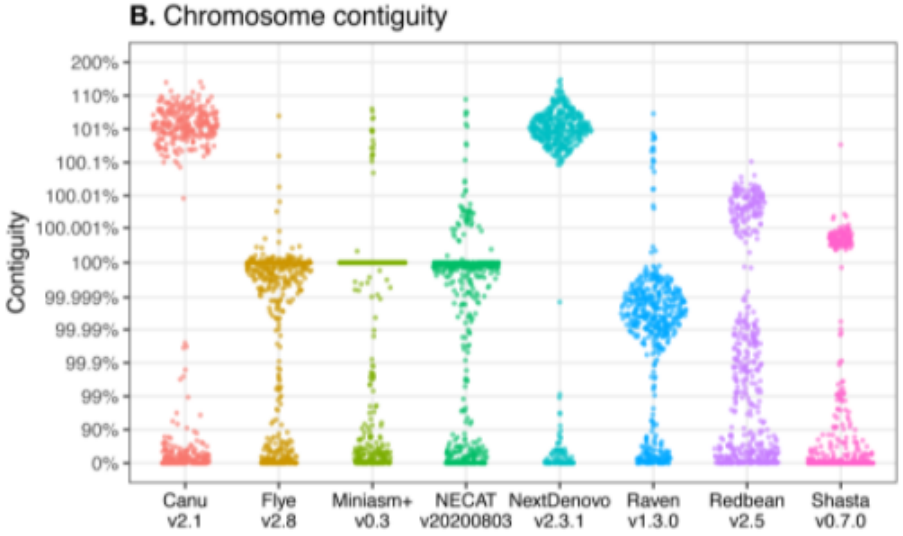
Graph cleaning heuristics

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

An assembly is a model

1. Assemblies contain errors
2. Different tools can produce very similar assemblies
3. A single tool can produce very different assemblies with small changes of parameters(!)





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

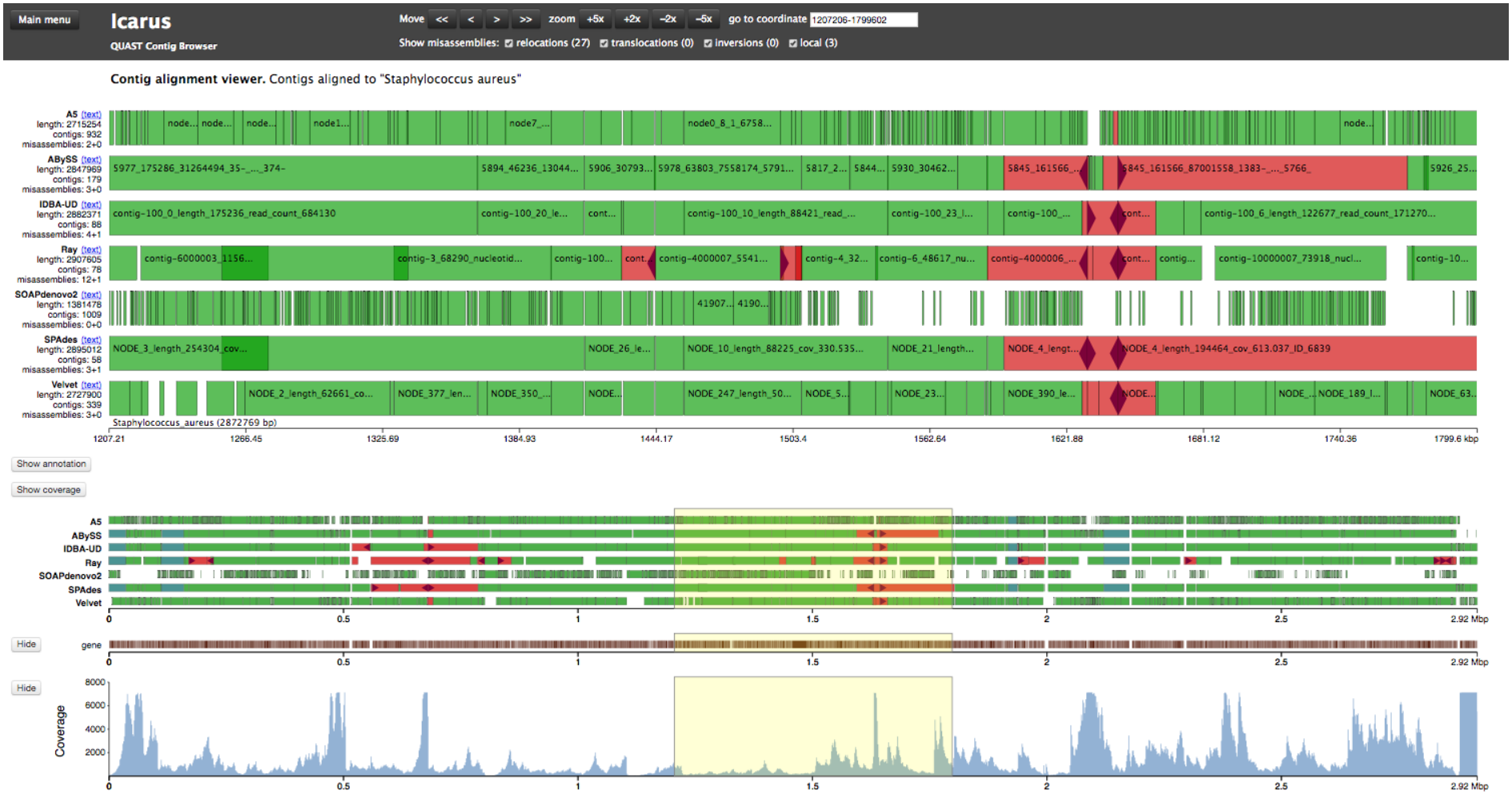
- What do we do post-assembly?

1. Assess its quality
2. Improve it
3. Use it!

Two ways to polish an assembly according to Mid-Journey

Evaluate assembly according to a reference

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



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

Assembly statistics

Alignment-based statistics	ABySS	MEGAHIT	SPAdes	Velvet
Genome fraction (%)	98.661	98.424	98.113	97.997
Duplication ratio	1.043	1	1	1
# genomic features	4525 + 75 part	4511 + 64 part	4489 + 50 part	4486 + 56 part
Largest alignment	248 481	235 933	285 096	264 944
Total aligned length	4 776 214	4 568 317	4 553 809	4 550 150
NGA50	69 801	122 647	133 309	112 446
LGA50	21	14	12	14
Misassemblies				
# misassemblies	4	0	0	4
Misassembled contigs length	231 767	0	0	435 515
Per base quality				
# mismatches per 100 kbp	2.09	2.69	1.03	3.19
# indels per 100 kbp	0.57	1.31	0.29	1.98
# N's per 100 kbp	24.59	0	17.55	94.19
Statistics without reference				
# contigs	176	95	92	90
Largest contig	248 481	235 933	285 196	264 944
Total length	4 777 853	4 571 292	4 557 363	4 552 266
Total length (>= 1000 bp)	4 757 929	4 562 458	4 548 710	4 544 453
Total length (>= 10000 bp)	4 562 801	4 478 614	4 466 223	4 475 223
Total length (>= 50000 bp)	3 248 113	3 833 793	3 812 315	3 817 904
BUSCO completeness				
Complete BUSCO (%)	98.65	98.65	98.65	98.65
Partial BUSCO (%)	0	0	0	0
Predicted genes				
# predicted genes (unique)	3717	3595	3587	3576

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.

- The catsembler

genome

ACGGATGATAGATTTGATACGA

reads

GATTTGATAC
ACGGATGATA
TTTGATACGA

concatenate the reads: super N50!

GATTTGATACACGGATGATATTTGATACGA

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

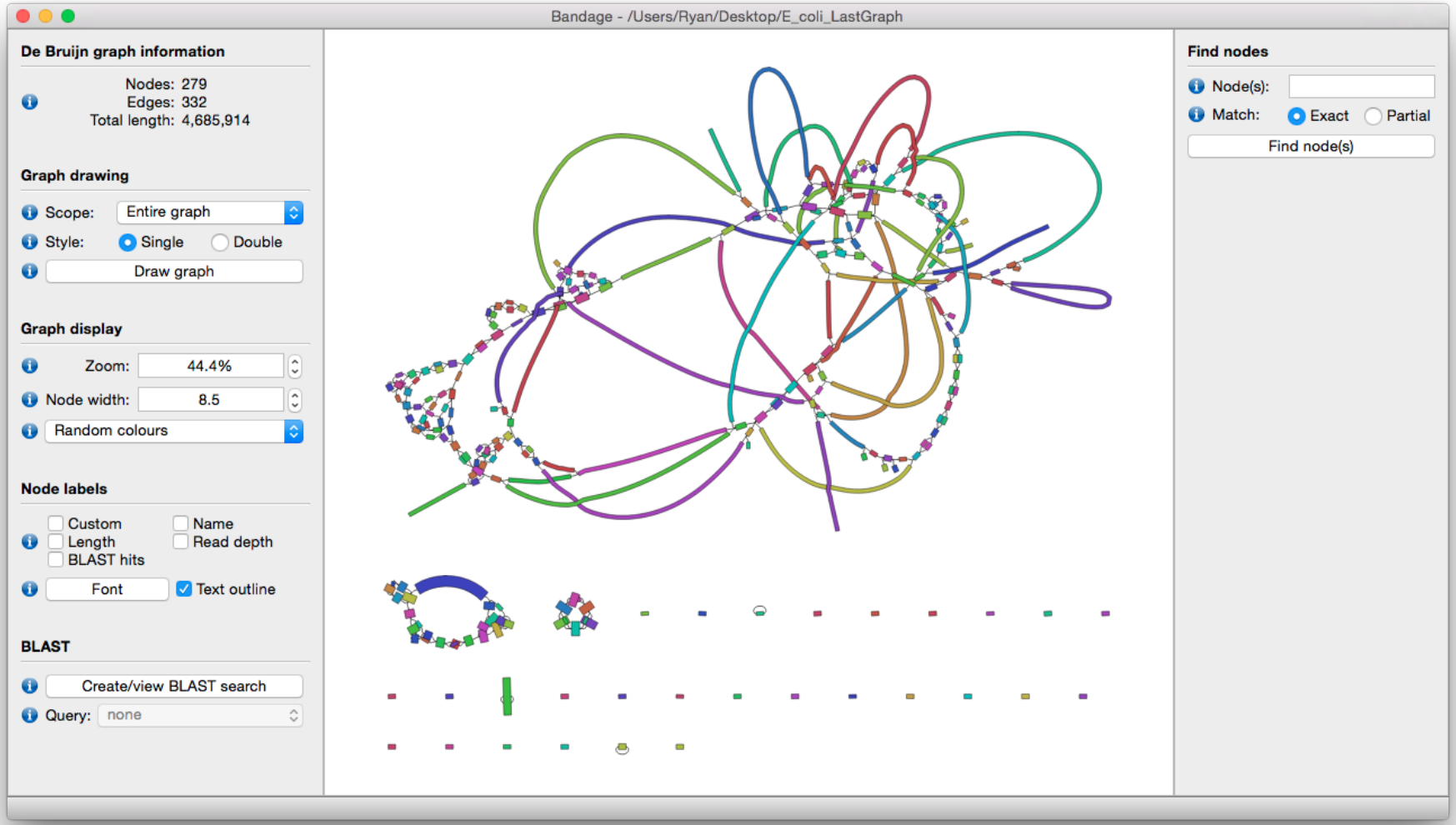
NGA50

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

Misassemblies

• Visualize assembly

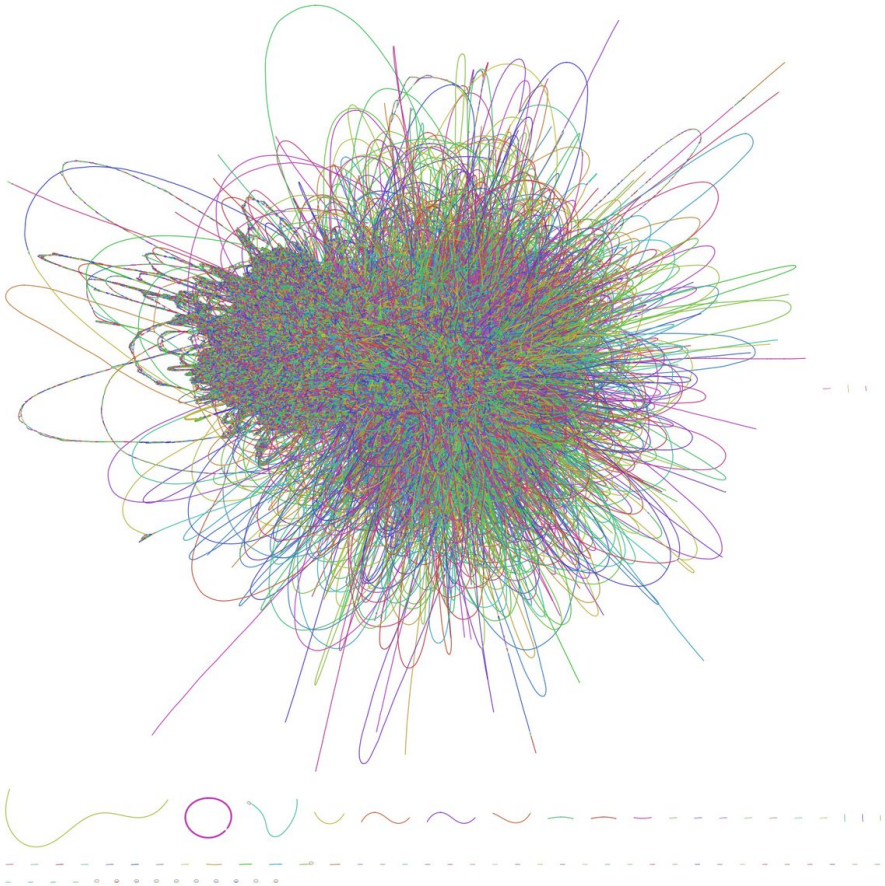
Bandage tool can visualize assembly graphs (GFA)



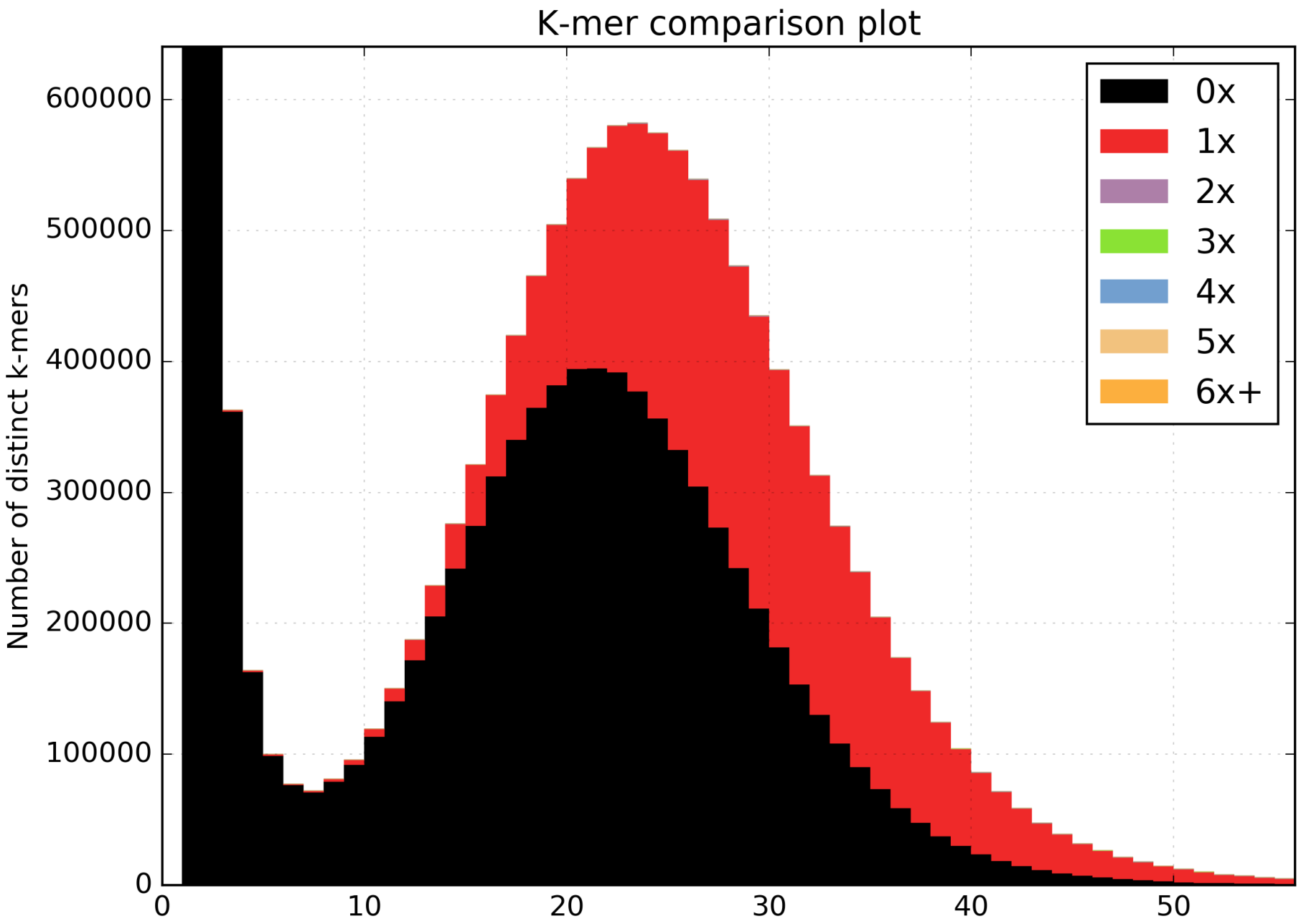
From `rwick.github.io/Bandage`

• Visualize assembly

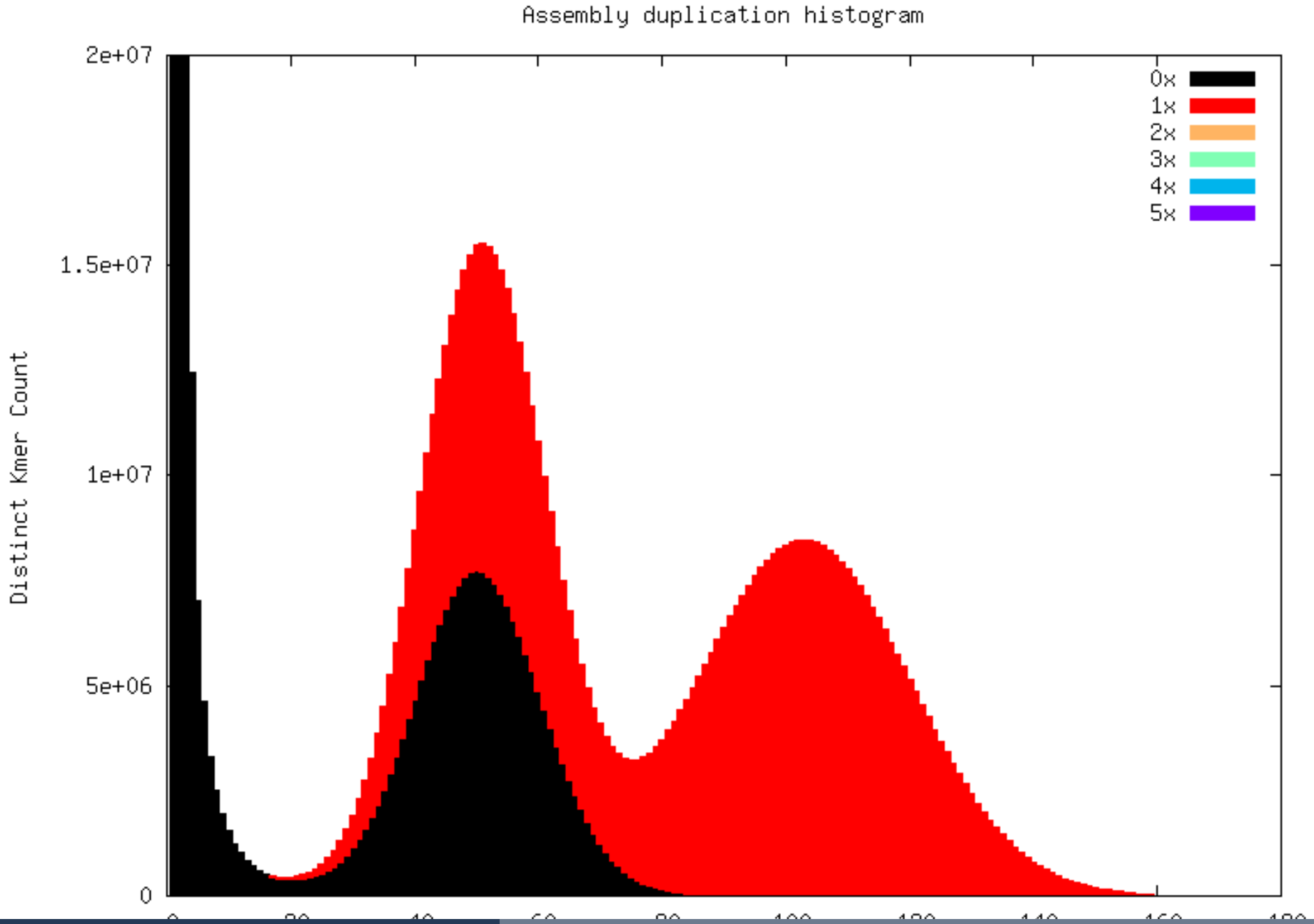
Bandage tool can visualize assembly graphs (GFA)



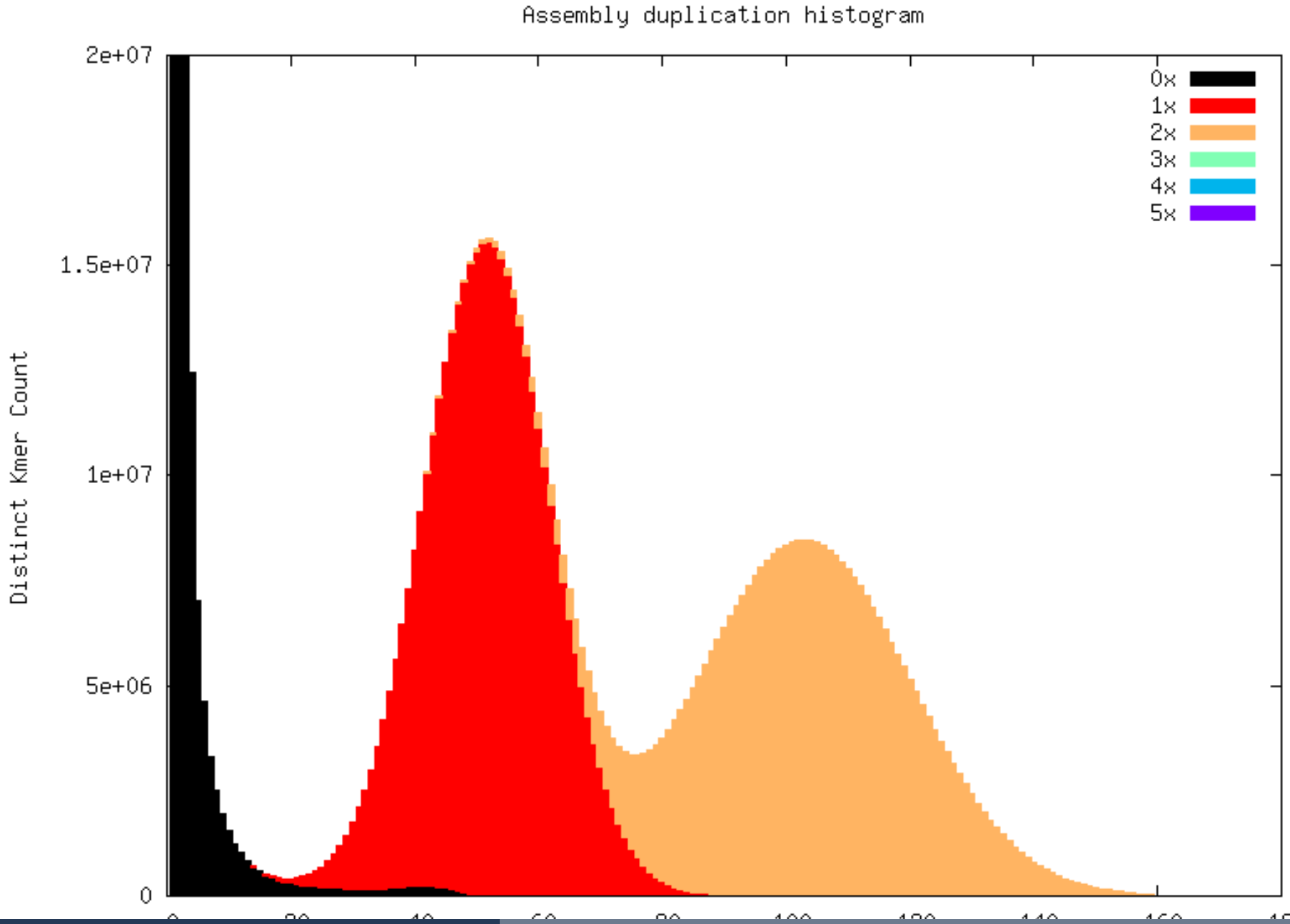
K-mer spectrum visualization with KAT



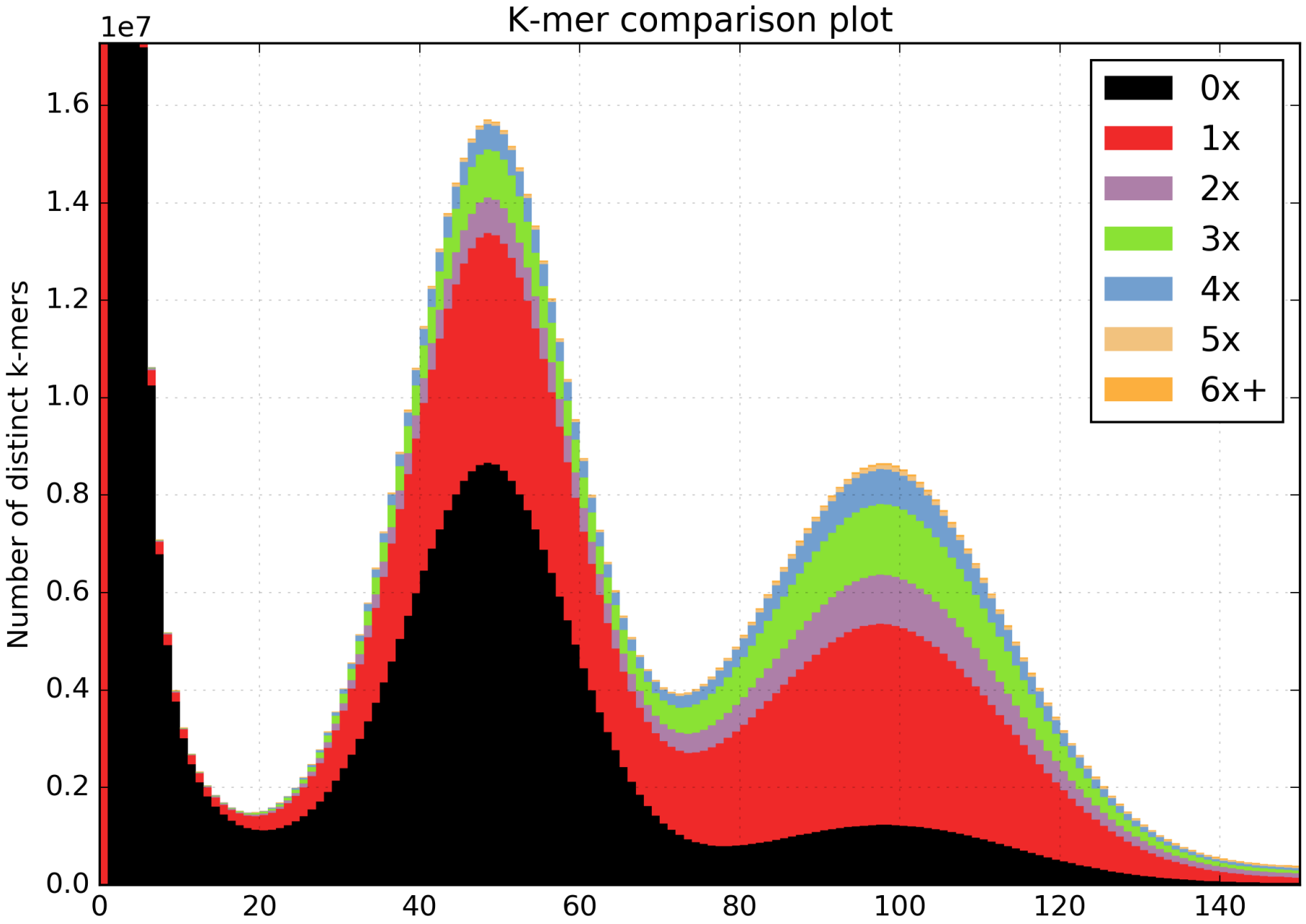
- *K*-mer spectrum visualization with KAT



- K-mer spectrum visualization with KAT

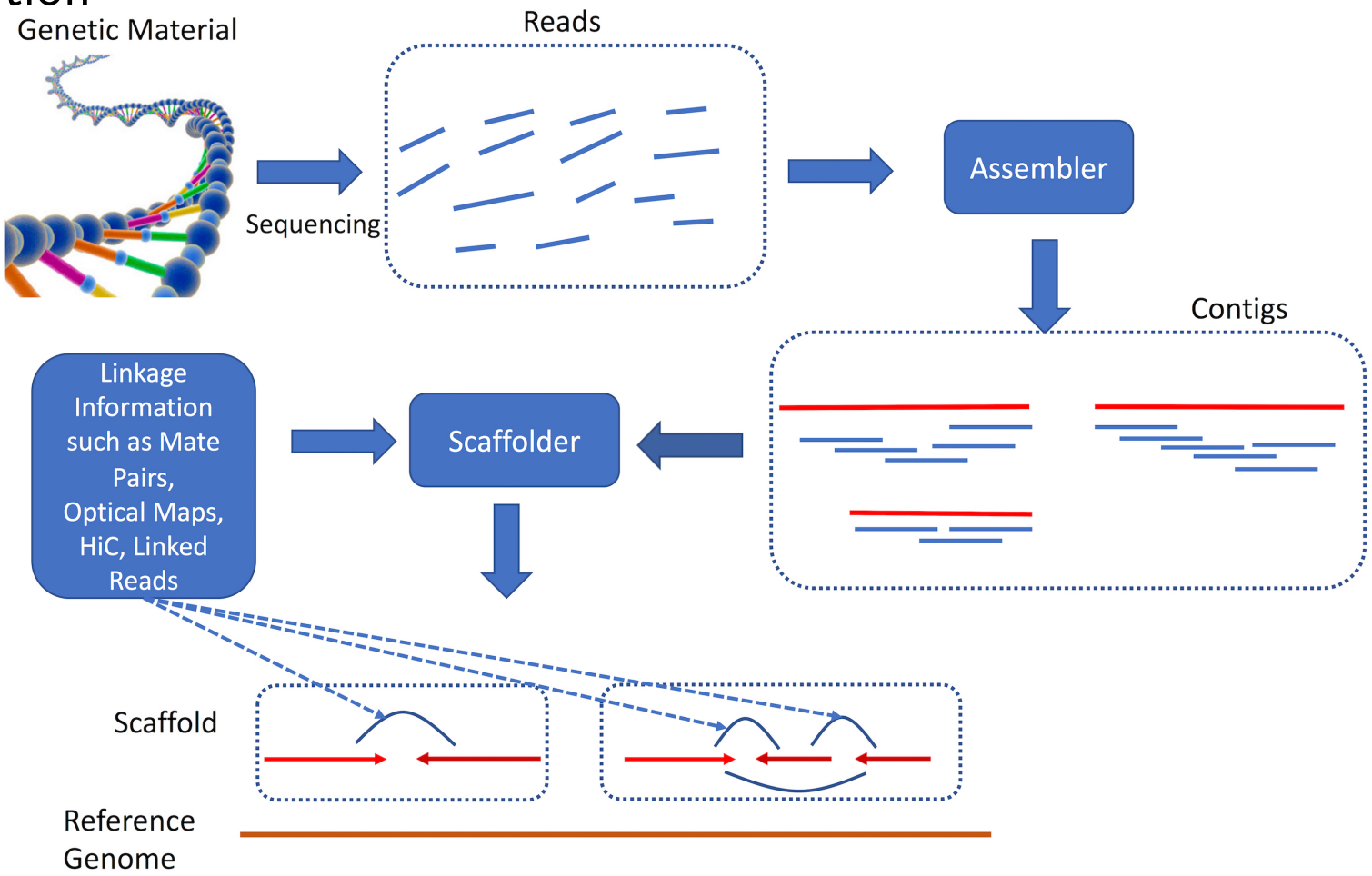


K-mer spectrum visualization with KAT



Scaffolding

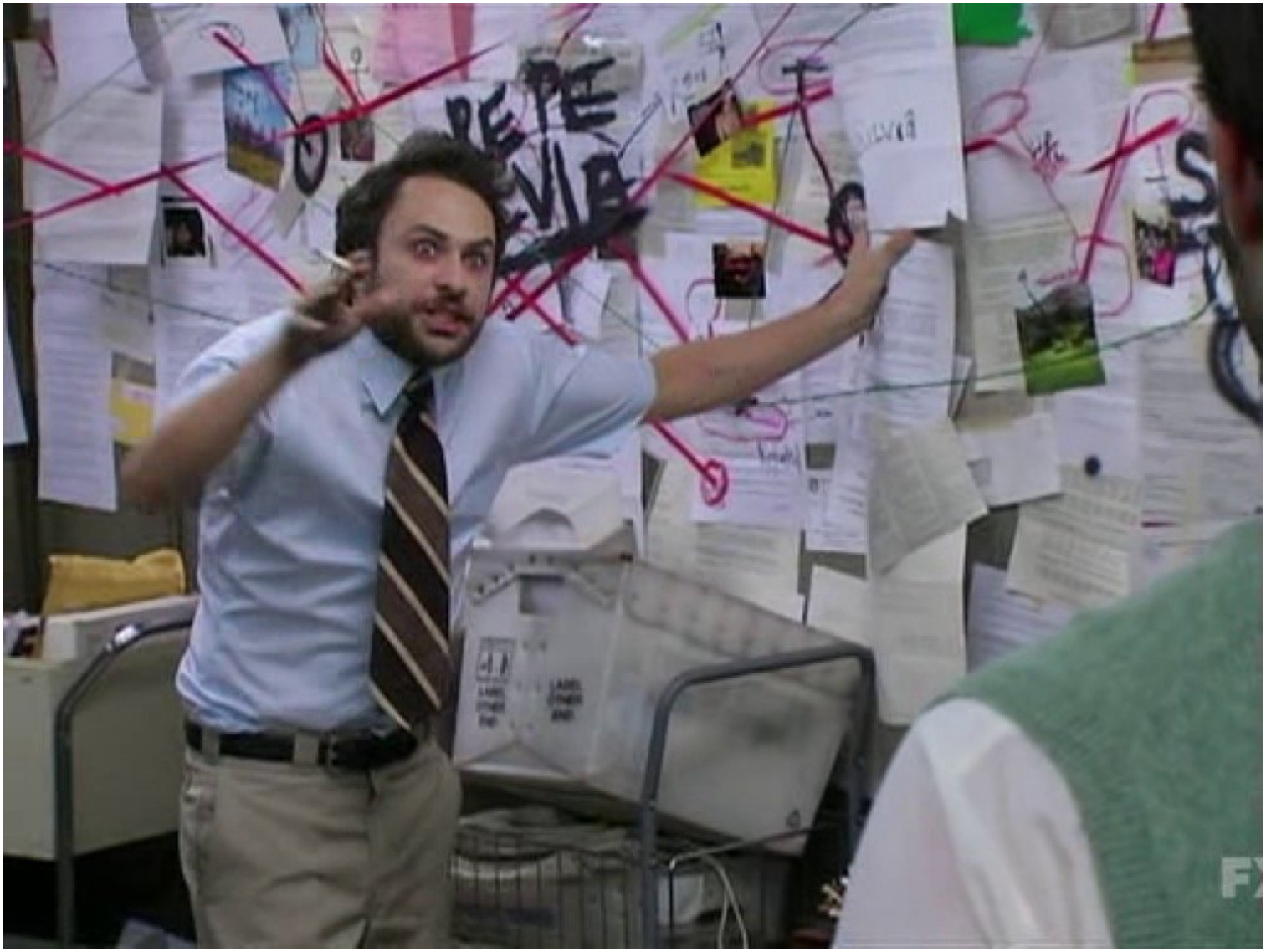
Softwares can improve the assembly continuity by using other kinds of information



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

The end

. . . of the theoretical part (or is it?)



Intermission

The workshop's team during the annual lungfish genome assembly session according to MidJourney

- Sanger

- Medium reads $\approx 1000bp$
- Very low error rate $\approx 0.01\%$
- Low throughput (up to billion of reads per run)
- Costly ($500/Mb)

No longer used for assembly

Second generation sequencing

NextSeq Series

HiSeq 4000 System

HiSeq X Series‡

NovaSeq 6000 System

- Short reads $\approx 150bp$
- Low error rate $< 1\%$
- High throughput (up to billion of reads per run)
- Cheap ($0.50/Mb)
- GC bias

Which assembly strategy is best suited?

- Short reads $\approx 100bp$
- Low error rate below 1%
- High throughput (up to billions of reads per run)

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

Vote!

- Greedy
- Overlap graph
- de Bruijn graph

Paradigm Breakdown

Greedy

Does not handle repeats $\times$
Scalable $\checkmark$

Overlap graph

Handles repeats smaller than reads size $\checkmark$
Extremely expensive to handle billion reads $\times$

De Bruijn graph

Handle repeat smaller than $k \approx$ reads size $\checkmark$
Scalable $\checkmark$

State-of-the-art

Well performing assemblers

- SPAdes [Bankevich 2012]
- Megahit [Li 2015]
- IDBA [Peng 2012]

Other notable assemblers

- SGA [Simpson 2012]
- Discovar denovo [Weisenfeld 2014]
- Abyss [Simpson 2009]

Third generation sequencing

- Long reads $\approx 10 - 100kbp$
- High error rate (up to 10%)
- 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 (99.9% identity)
- Mostly insertions

Repeats spanning

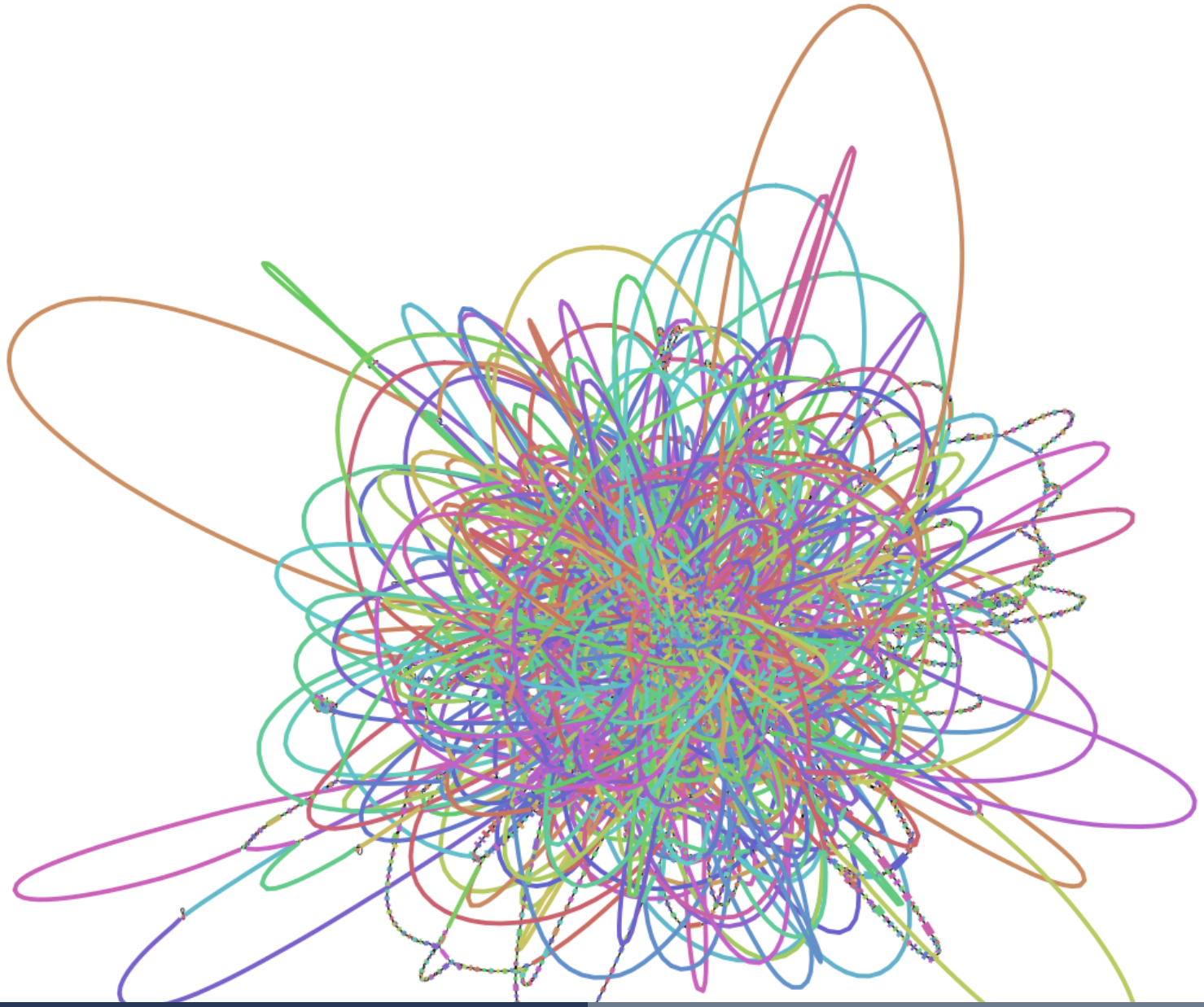
```
Genome:
GGTAATGGTTTTTTGGTGCTAATGCGTTTTTTCATGGATGTCGTAATTTTTTATCTG

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...
```

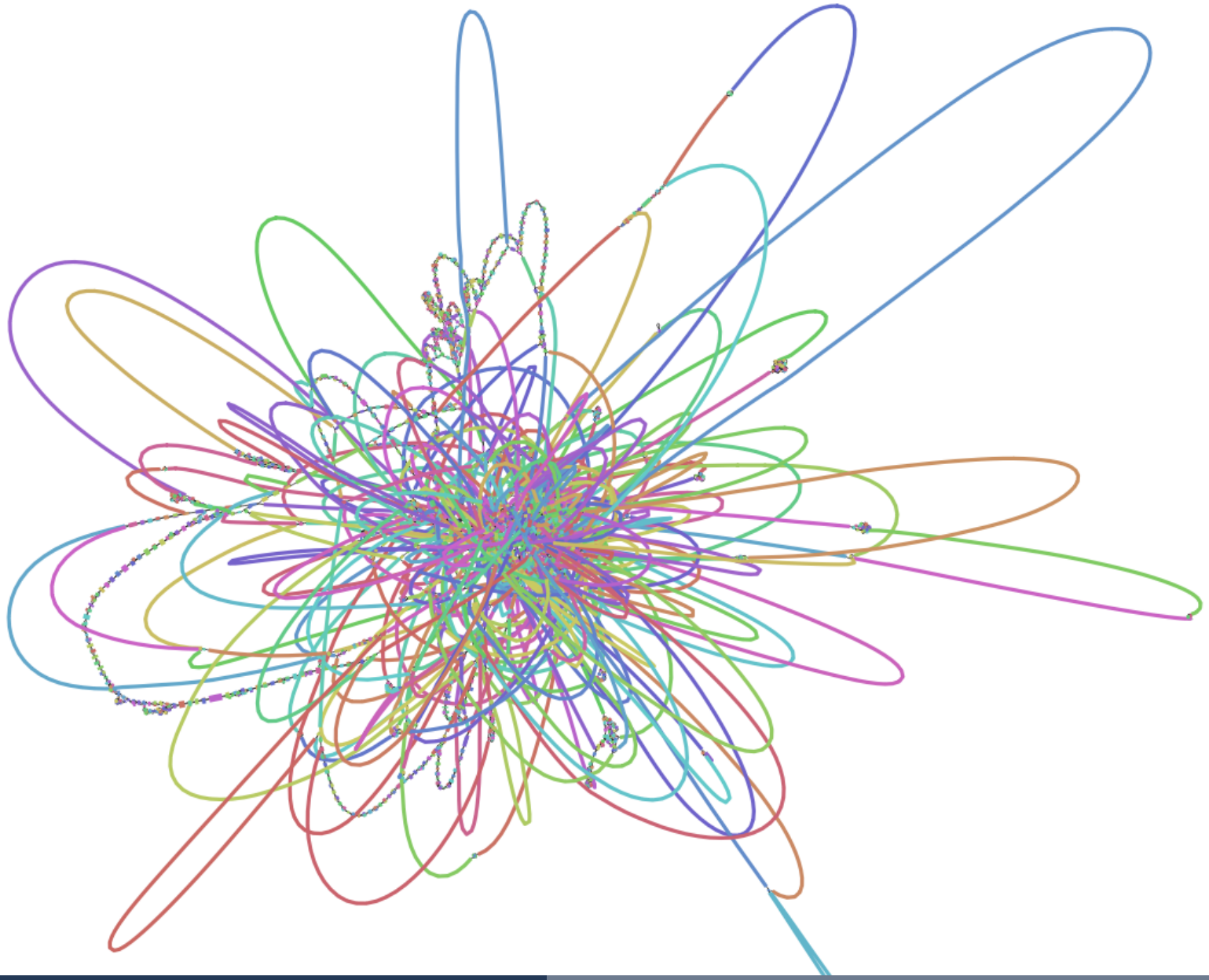
• Read length matters

Read size=21



• Read length matters

Read size=31

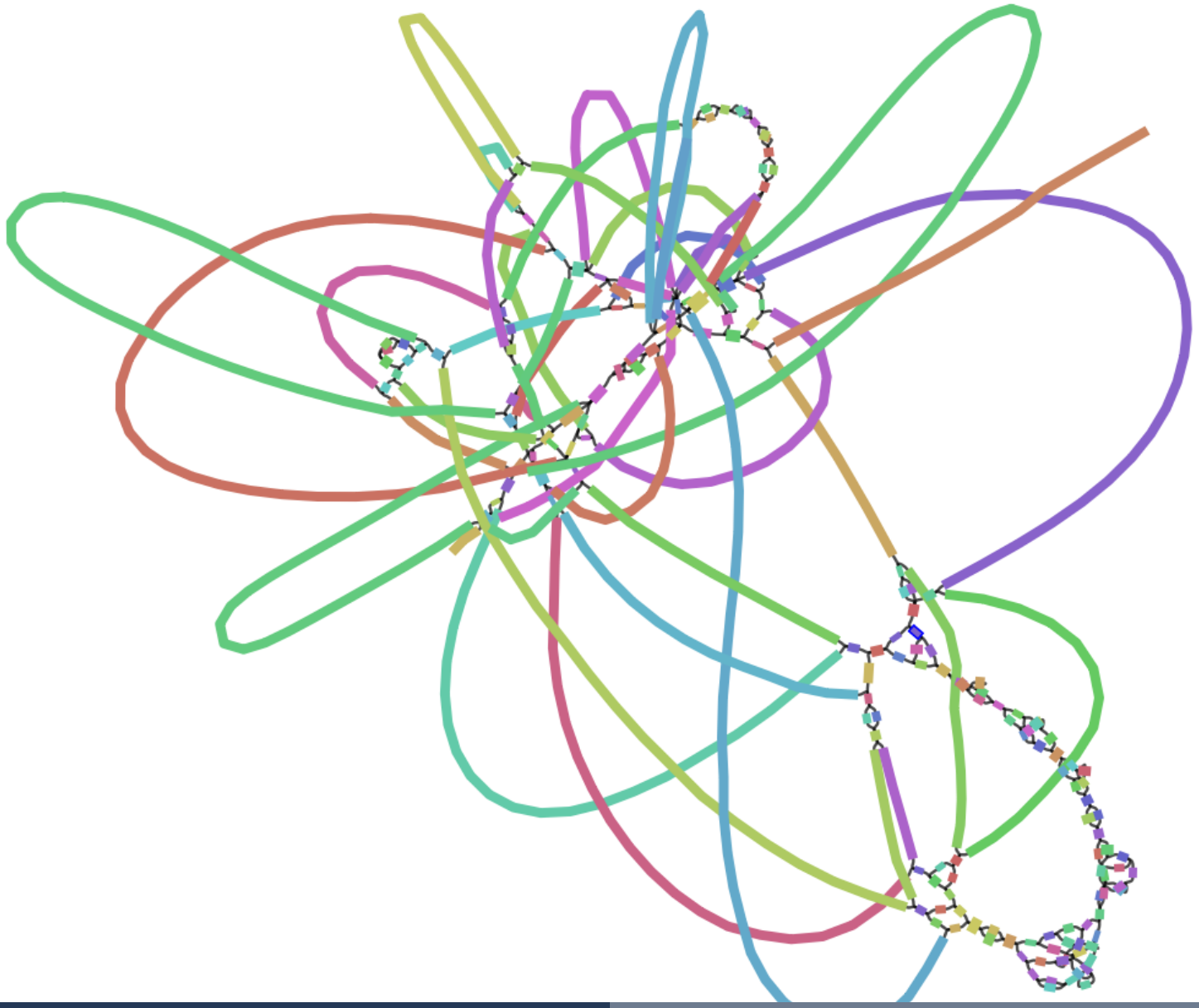


• Read length matters

Read size=63

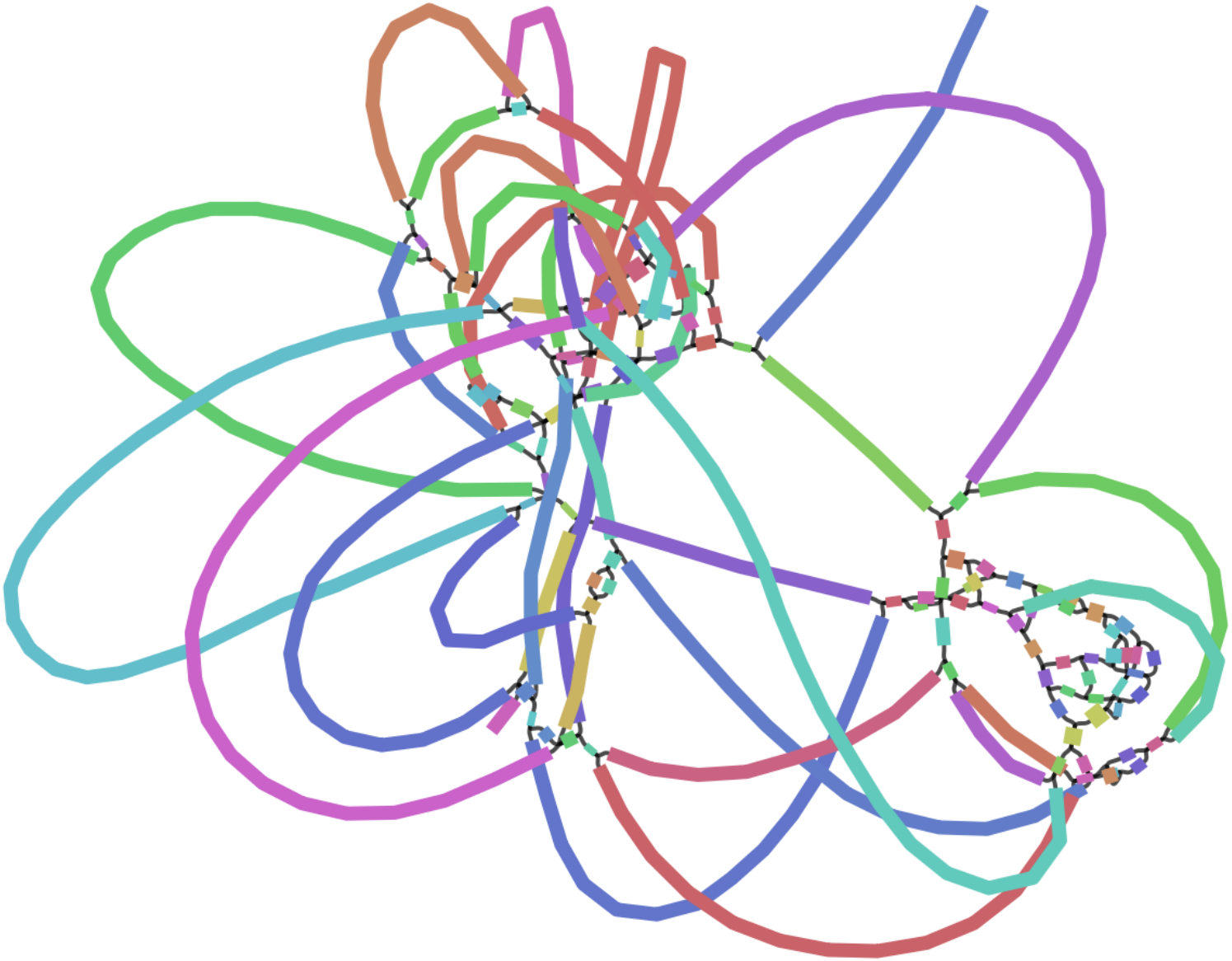
- Read length matters

Read size=255



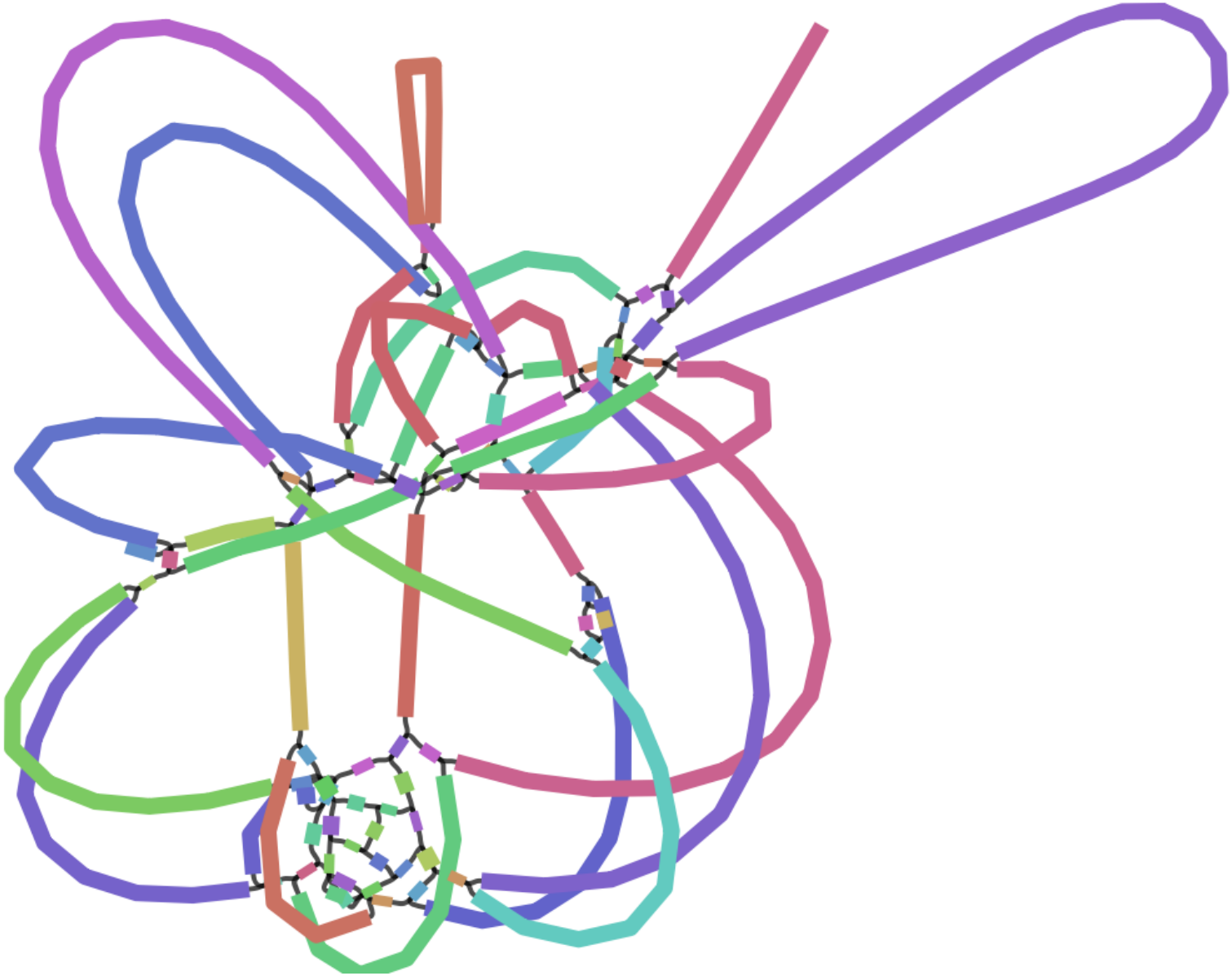
- Read length matters

Read size=500



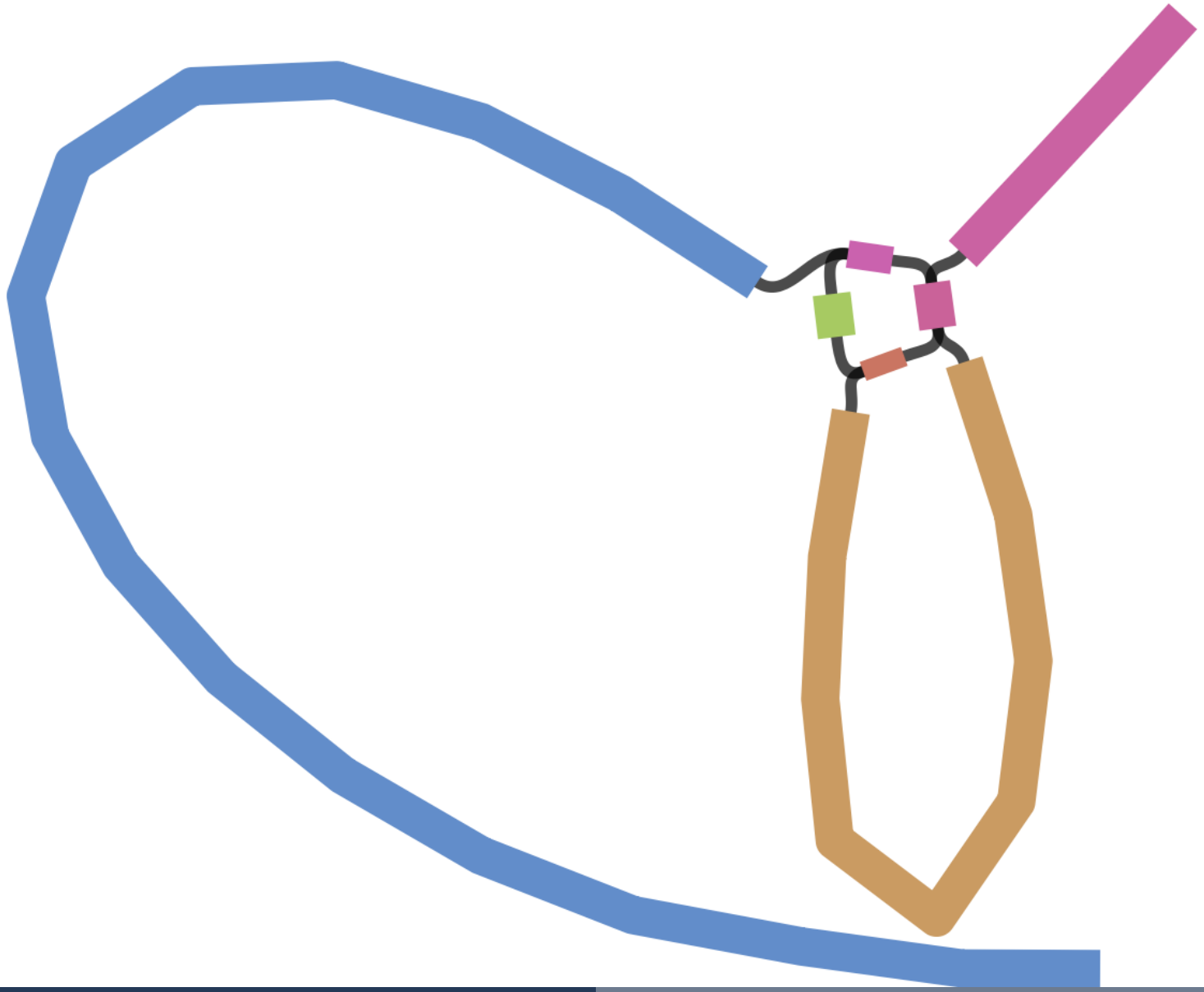
• Read length matters

Read size=1000

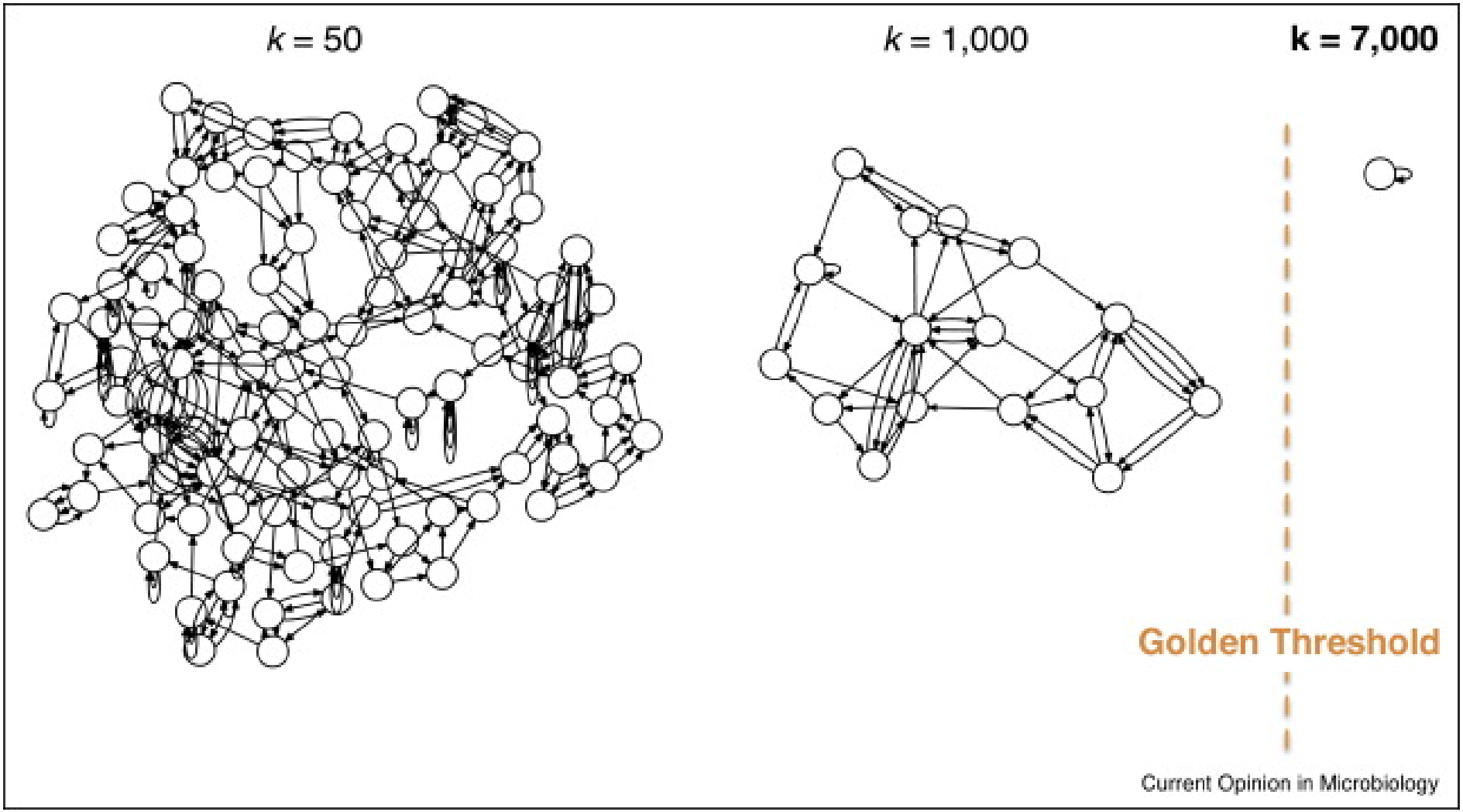


- Read length matters

Read size=2000

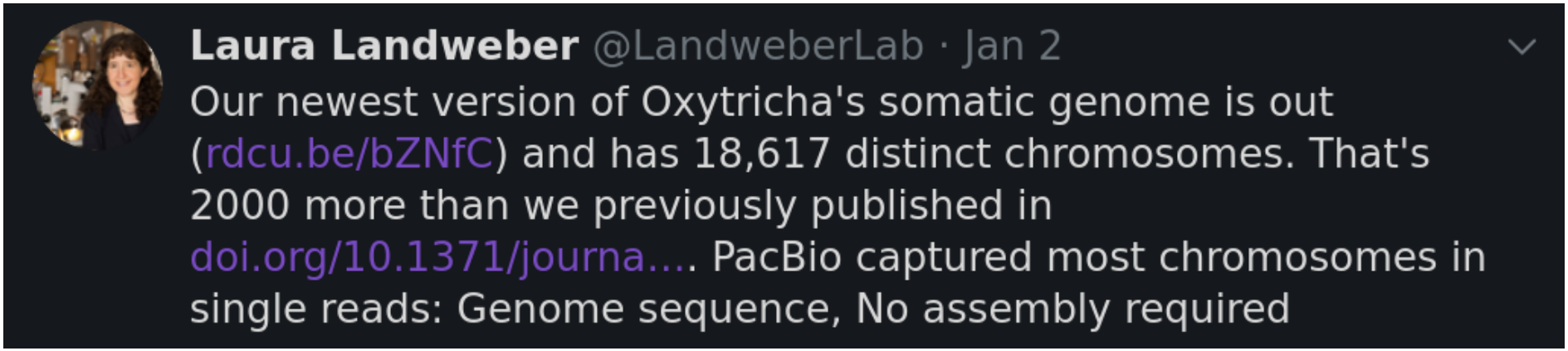


- Great hope for assembly

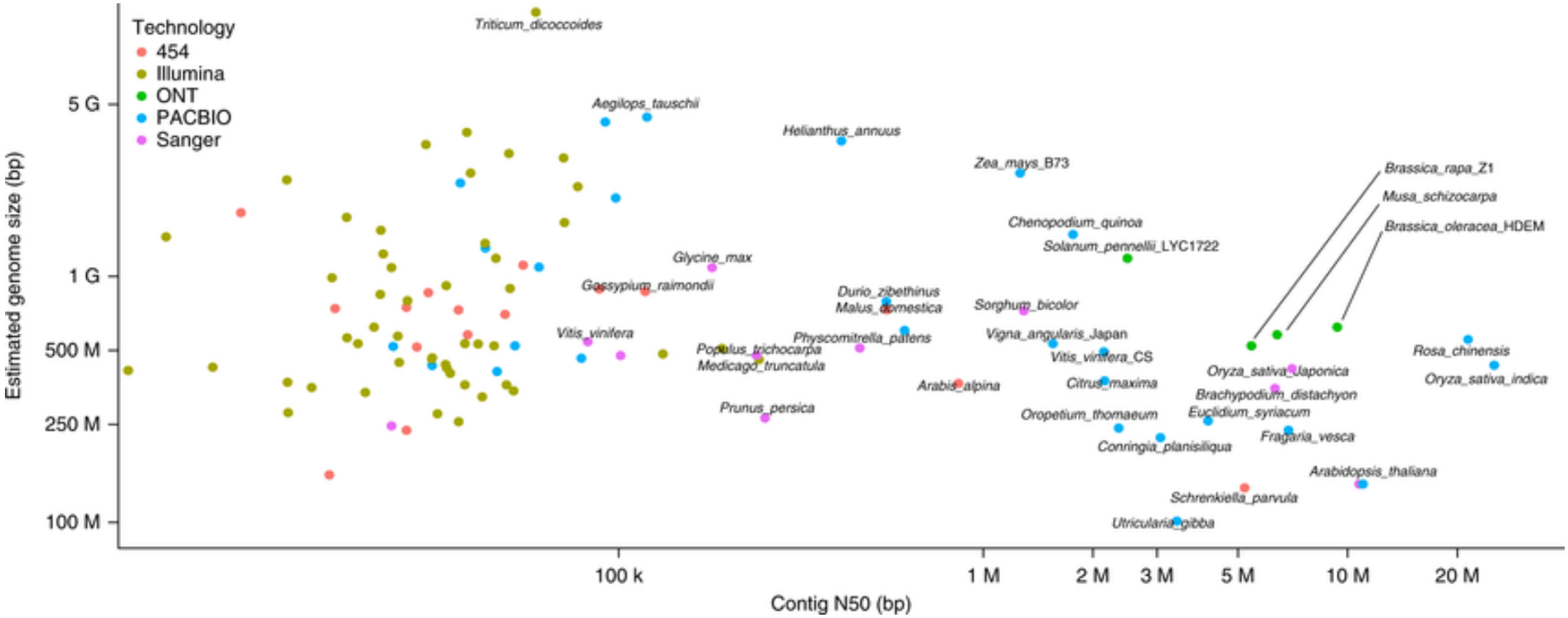


From "One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly" Current Opinion in Microbiology 2015

- Long reads killed the assembly star



- 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?

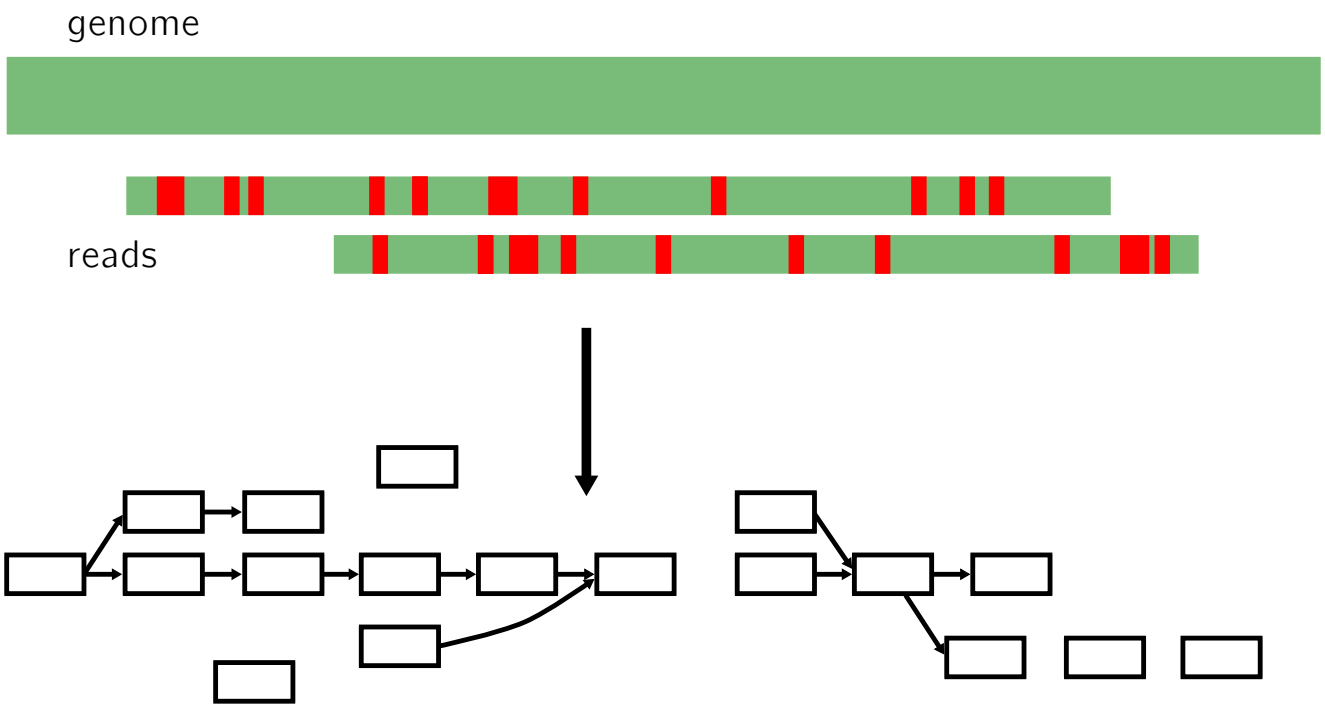
- Long reads $\approx 10kbp$
- High error rate $\approx 10\%$
- High throughput (up to millions of reads per run)

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

Vote!

- Greedy
- Overlap graph
- de Bruijn graph

Long reads for assembly: de Bruijn graph?



Most k-mers will contain at least an error and will be useless

Long reads for assembly: overlap graph?

Supposed to be super expensive!

```
TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACTTCAAGTAAAAACACCTCACGAGTTAAAACACCTAAGTTC
TAAGAAAGCT
 AAGAAAGCTC
  AGAAAGCTCT
   GAAAGCTCTG
    AAAGCTCTGA
     AAGCTCTGAA
      AGCTCTGAAT
       GCTCTGAATC
        CTCTGAATCA
         TCTGAATCAA
          CTGAATCAAC
           TGAATCAACG
            GAATCAACGG
             AATCAACGGA
              ATCAACGGAC
               TCAACGGACT
                CAACGGACTG
                 AACGGACTGC
                  ACGGACTGCG
                   CGGACTGCGA
                    GGACTGCGAC
                     GACTGCGACA
                      ACTGCGACAA
                       CTGCGACAAT
                        TGCGACAATA
                        …
```

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

```
TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACTTCAAGTAAAAACACCTCACGAGTTAAAACACCTAAGTTC
TAAGAAAGCTCTGAATCAACGGACTGCGAC
   GAAAGCTCTGAATCAACGGACTGCGACAAT
      AGCTCTGAATCAACGGACTGCGACAATAAG
         TCTGAATCAACGGACTGCGACAATAAGTGG
            GAATCAACGGACTGCGACAATAAGTGGTGG
               TCAACGGACTGCGACAATAAGTGGTGGTAT
                  ACGGACTGCGACAATAAGTGGTGGTATCCA
                     GACTGCGACAATAAGTGGTGGTATCCAG
                        TGCGACAATAAGTGGTGGTATCCAGAAT
                           GACAATAAGTGGTGGTATCCAGAATTTG
                              AATAAGTGGTGGTATCCAGAATTTGTCA
```

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

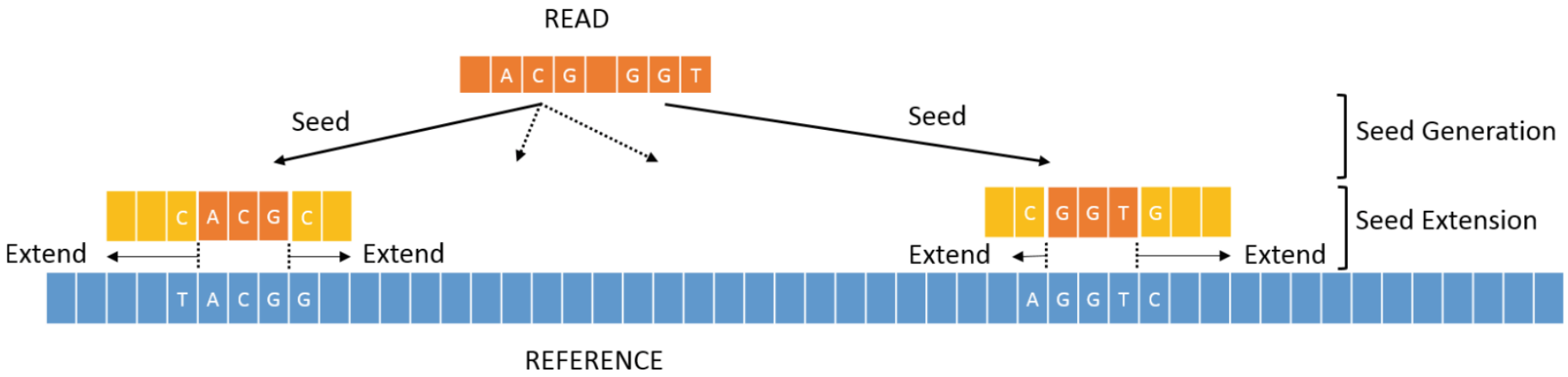
Long reads

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

Very long reads

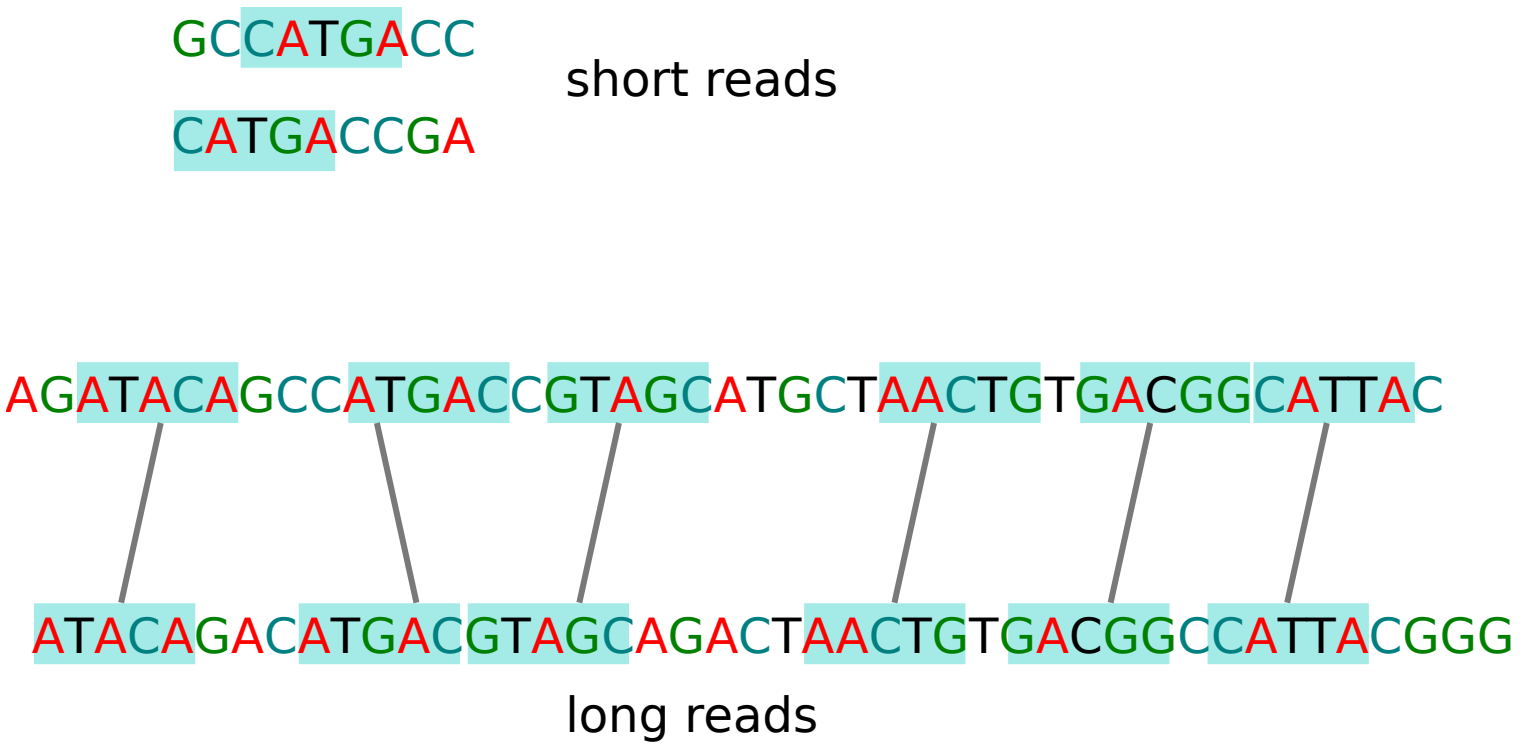
- 5,000 100kbp reads
- 99,000 bp average overlap

- Are large overlaps hard to compute?



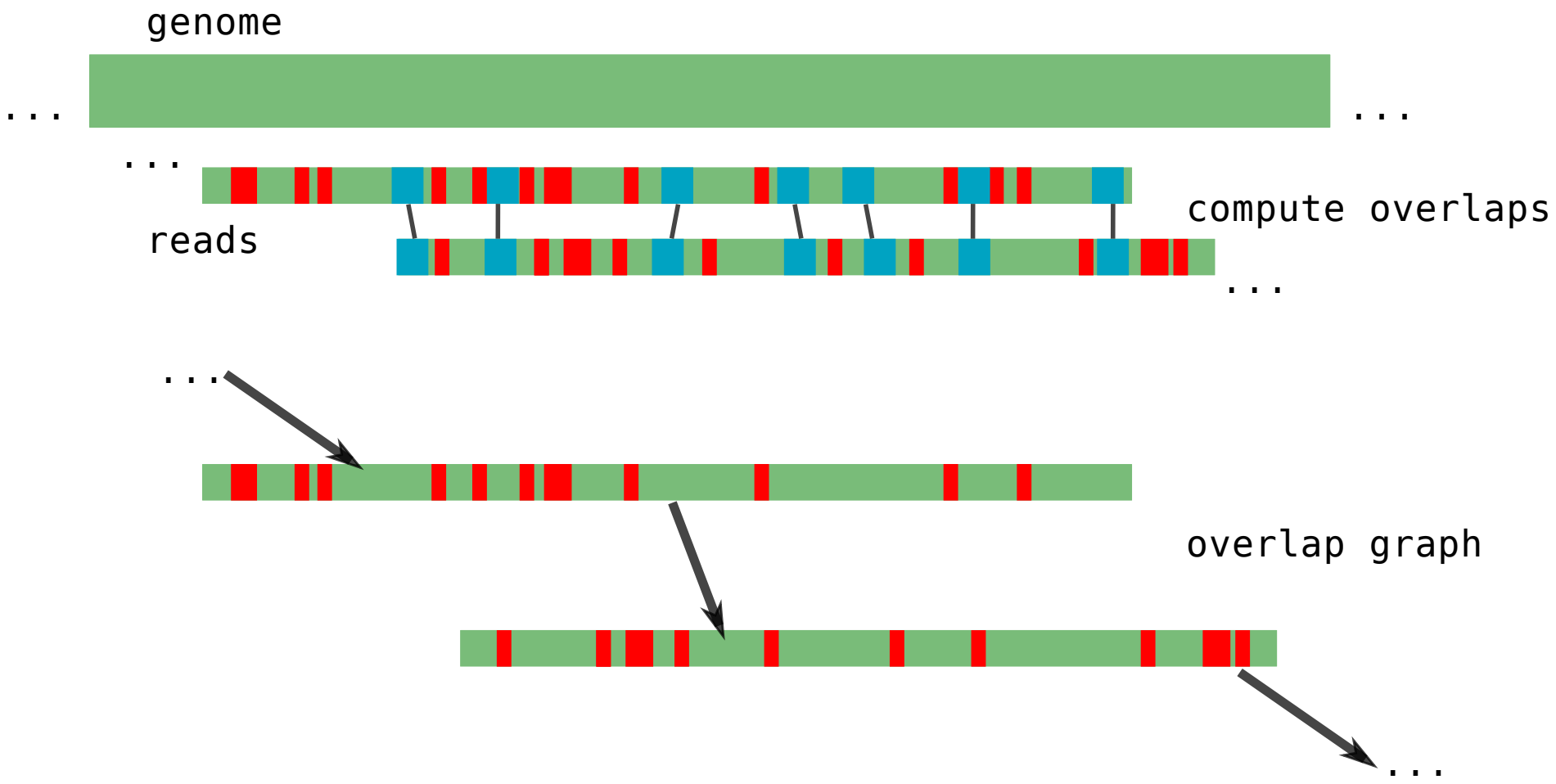
Aligning very long and highly erroneous regions is expected to be expensive, as alignment is quadratic $\approx \mathcal{O}(n^2)$!

"Anchor chaining" in overlap graph



- For long reads: typically Minimap2's [Li 2018] job
- "Anchor chaining": find common chains of anchors (k-mers) in the same order in 2 sequences (can be **linear** in practice in most cases)

Long reads for assembly: overlap graphs



- Sequencing errors

```
Genome:
 ATCGGTATCGTTACGGTATACC

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

High rate of insertions and deletions rendered genome annotation nearly impossible

Using coverage to remove noise: Consensus

```
 Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACTT

 Reads:
AAAGAAAGCACTGAATCATGGGACTTCGAG
   GAAAGCTCTCAACCAACGGACTGCGACTTT
      ACCTCTCAAGCAACGGACTGCGACAAAAAG
         TCTGAATCACCGGACTGCGTCAAAAAGTGC
            GAATCACCGGACTGCGACAGTTTGTGGTGG
               TCAACGCACTGCGACAATAAGTCCTGGTAT
                  ACGGACTGCGACAAAAAGTGTGGGTATCCA
                     GACTGCCACAAAAAGTGGTGGTATCCAG
                        TGCGACAAAAAGTGGGGGTATCCAGAAT
                           GACAATAAGGGGGGGTATCCAAAATTTG
                              AAAAAGGGGTGGTATCCAGAATTTTTCA
                                TAAGTGGGGGTATCCAAAATTTTTCAGTT
 Consensus:
AAAGATAGCTCTGAATCAACGGACTGCGACAAAAAGTGGTGGTATCCAGAATTTTTCAGTT
1/1                4/7    9/10   6/11                    3/4
```

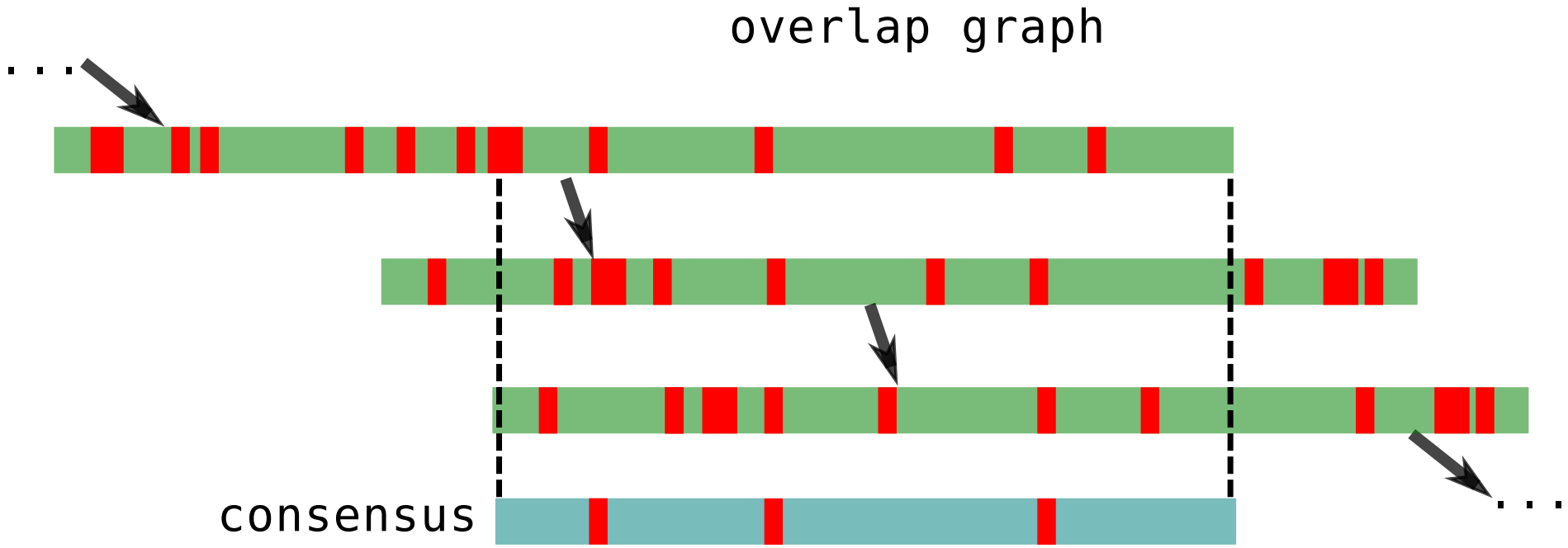
- Exercise: Perform a consensus

```
RS1:  ACTTCGAACGT
RS2:  TCGATCGTTT
RS3:  GATCAGTTTAG
RS4:  TCATTTCGTA
RS5:  GTTTCGTCCG
REF:  ACTCGAATGTTTTCCTACG
```

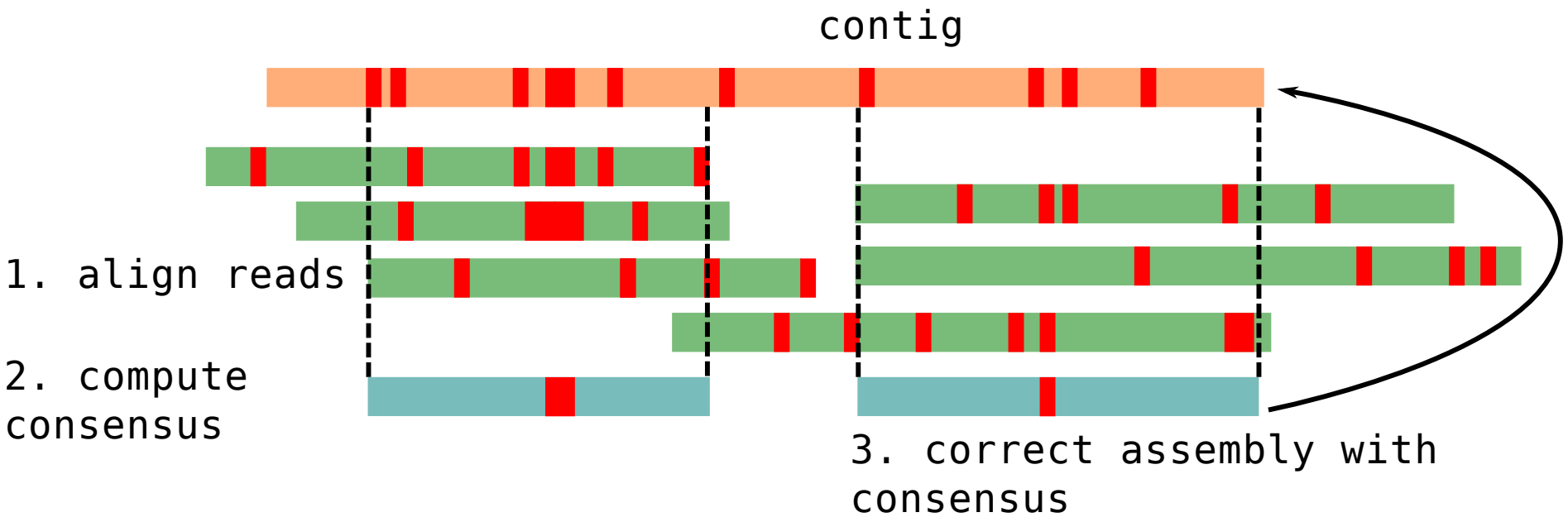
- Exercise: Perform a consensus - solution

```
RS1: ACTTCGA-AC-GT---------
RS2: --T-CGA-TC-GTTT-------
RS3: -----GA-TCAGTTT-AG----
RS4: ---------TC-ATTT-CGTA--
RS5: ------------GTTT-CGTCCG
REF: ACT-CGAAT--GTTTTCCTACG
CON: ACT-CGAATC-GTTT-CGTACG
```

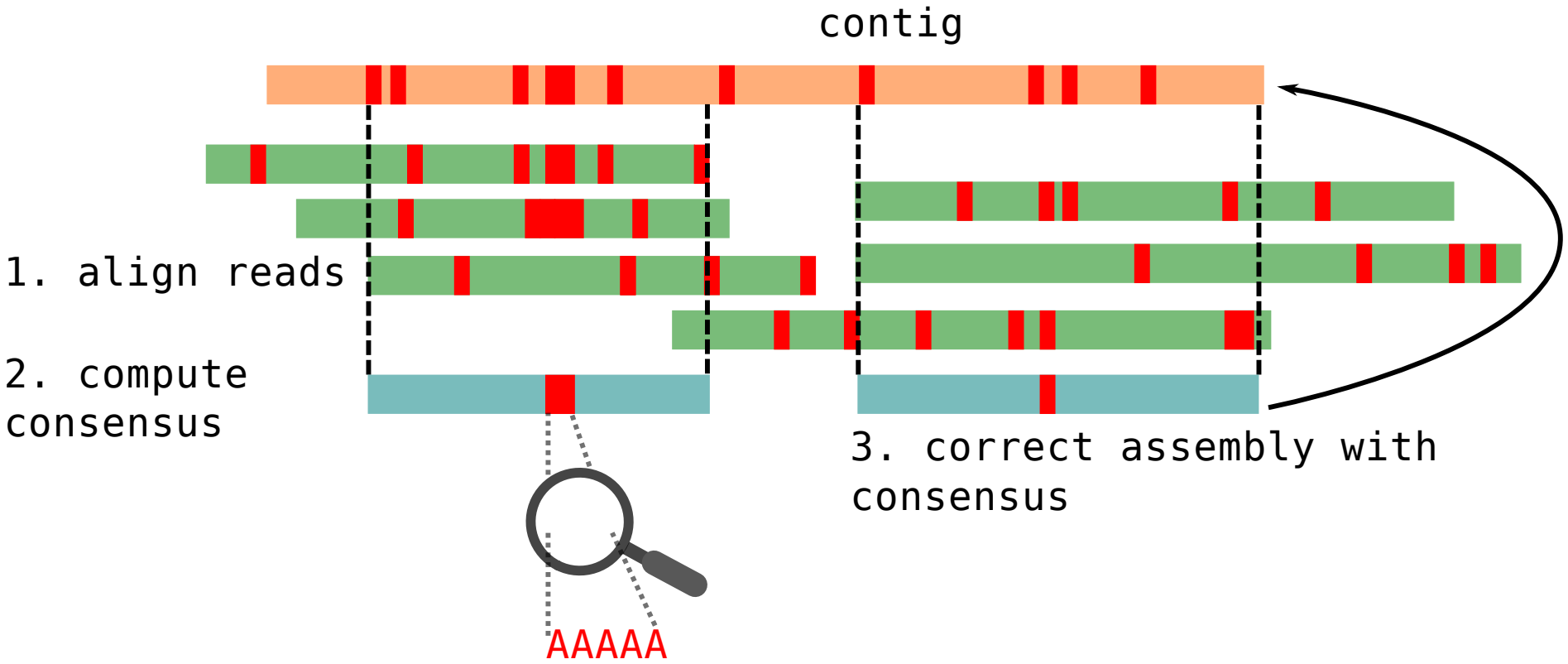
- Consensus during assembly



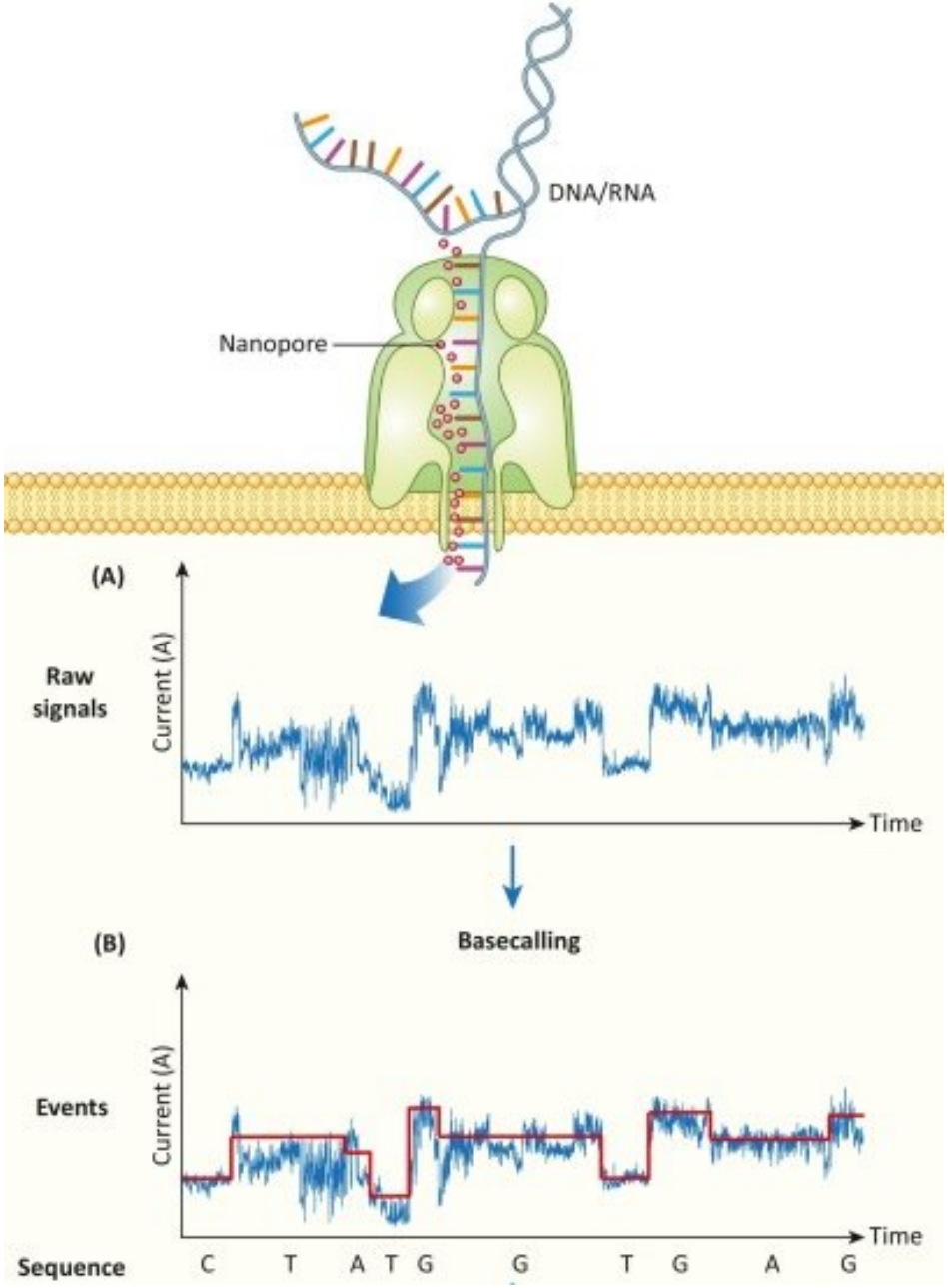
Consensus after assembly: polishing



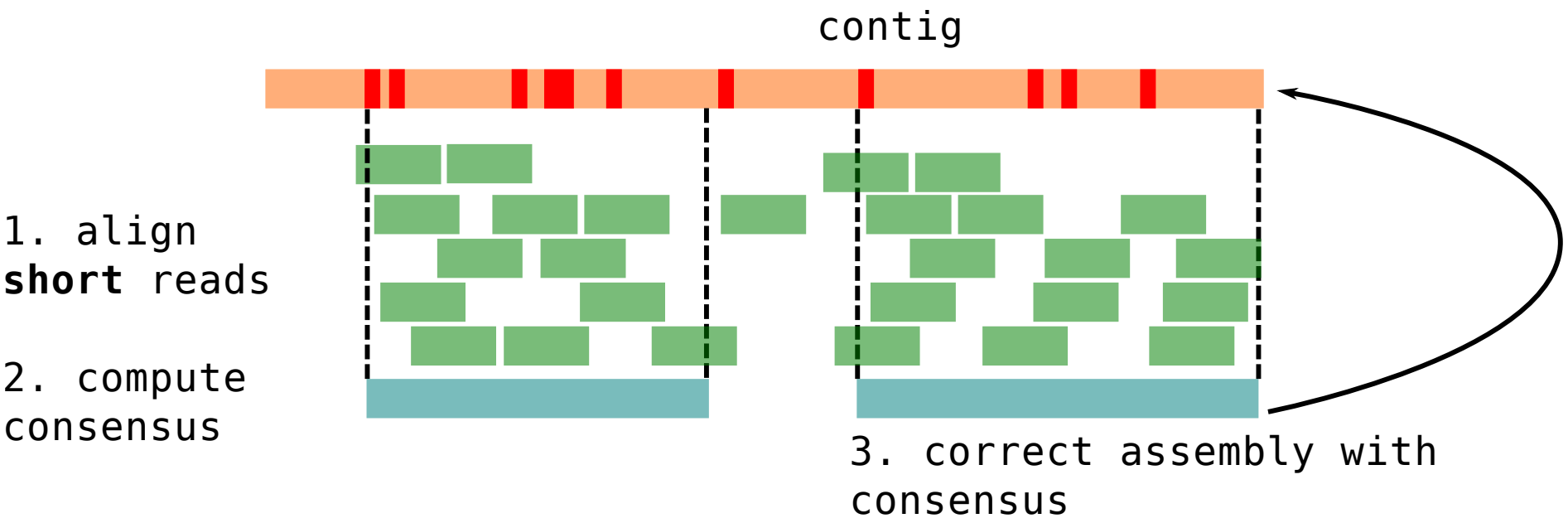
- Consensus after assembly: polishing



- Homopolymers are hard to read

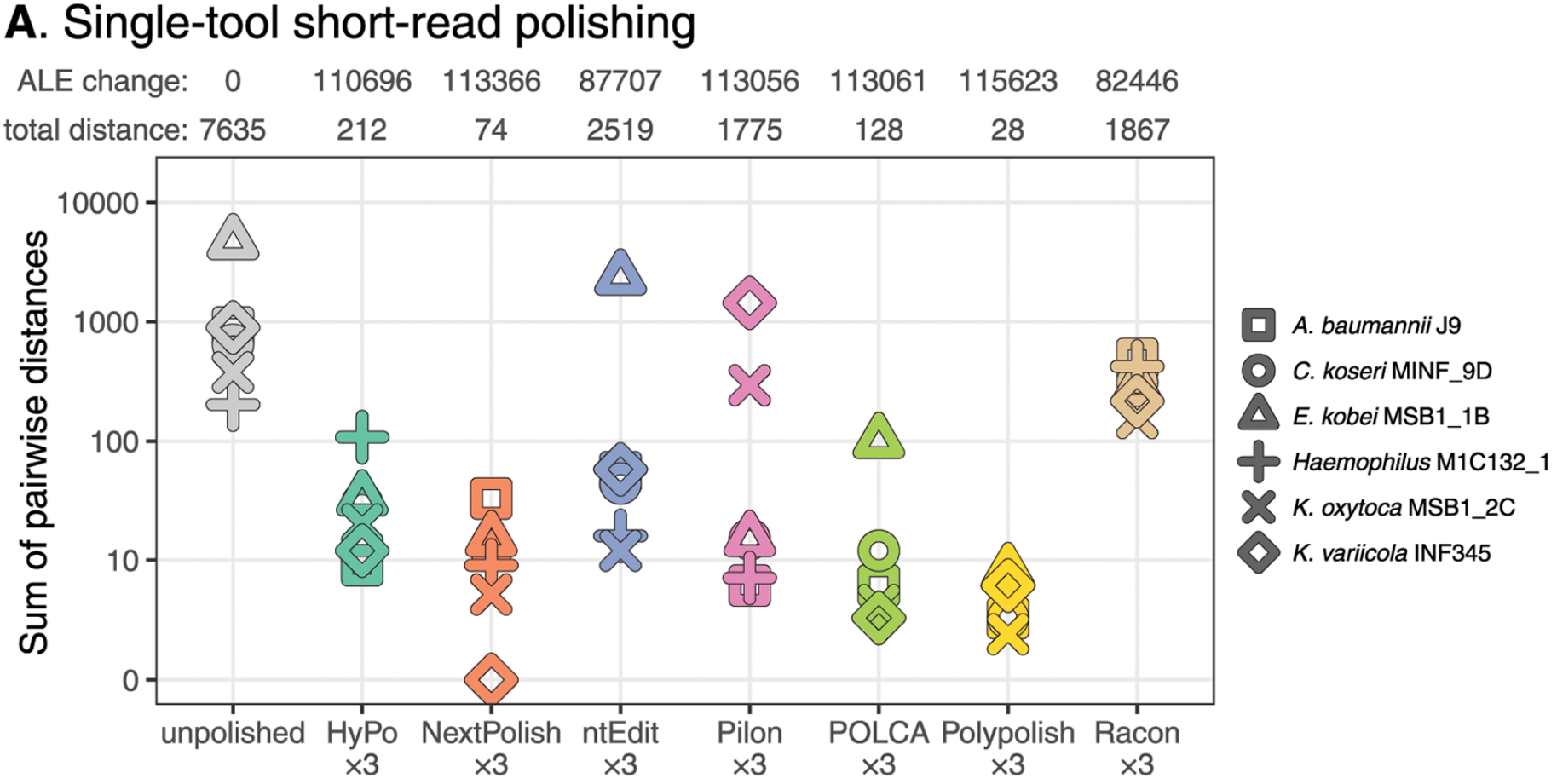


Polishing using accurate reads



Systematics errors

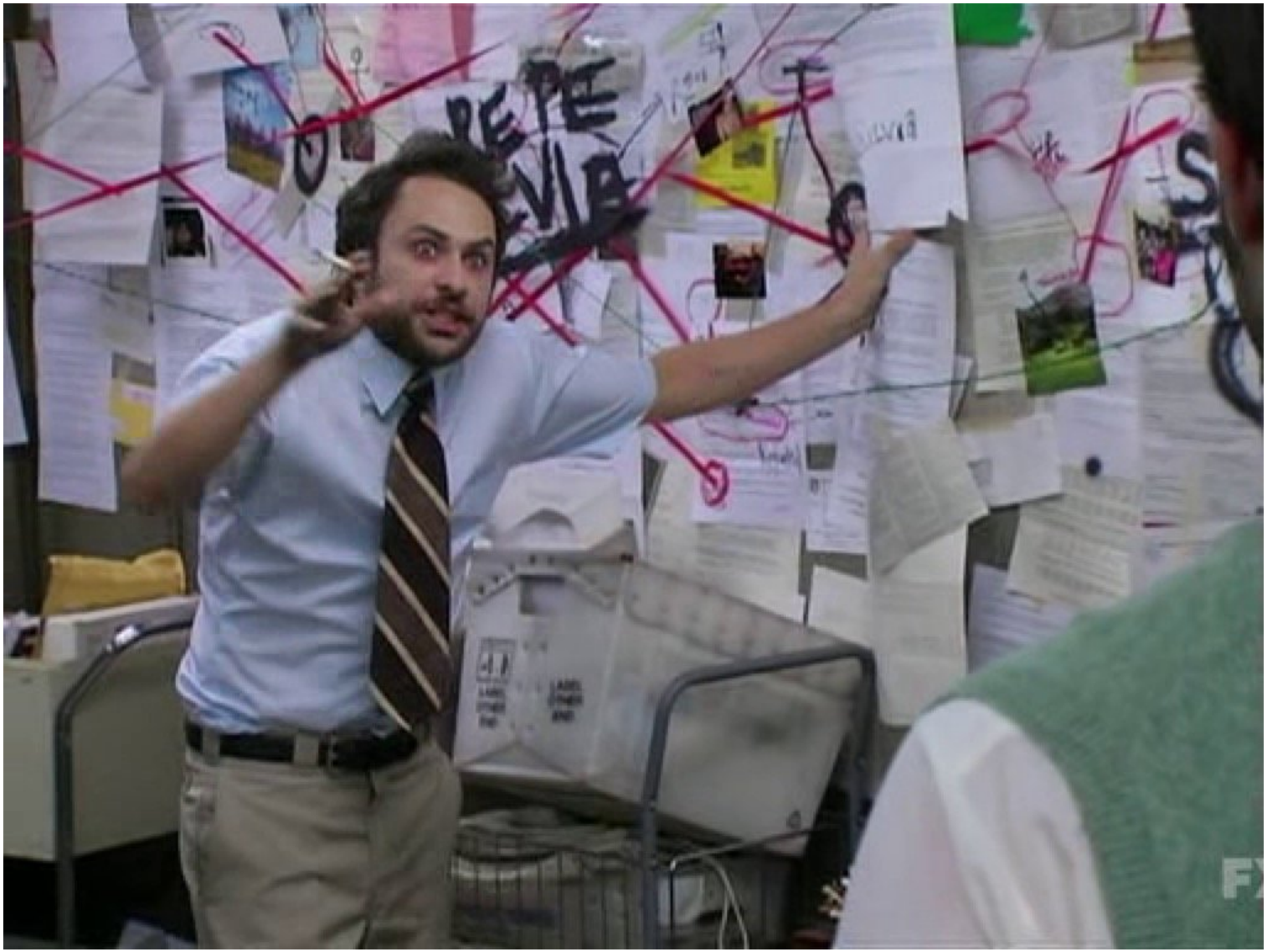
Polishing with Illumina data can improve the final error rate



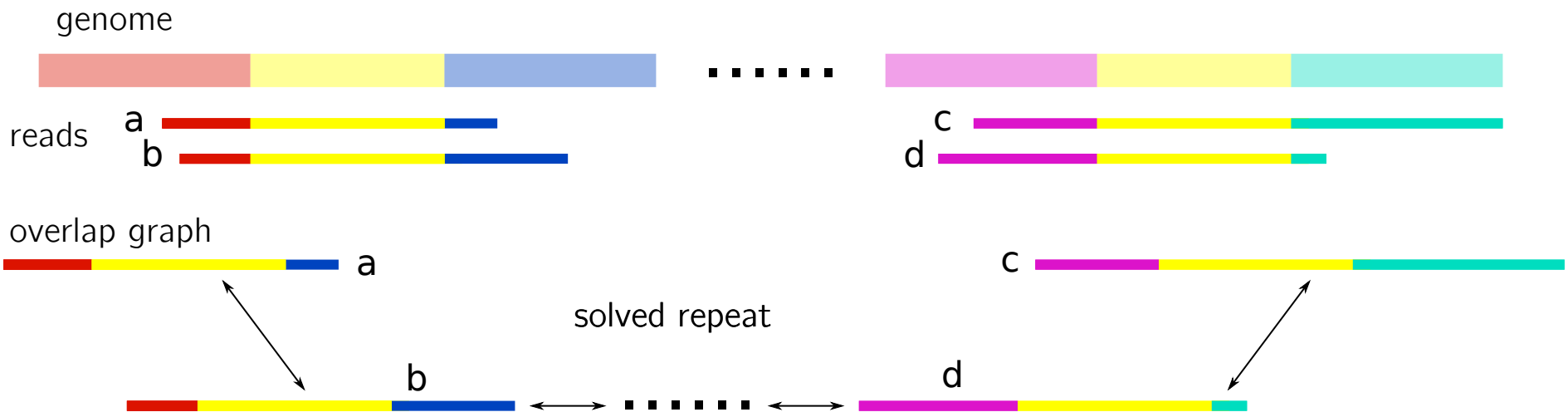
From Polypolish: Short-read polishing of long-read bacterial genome assemblies

I know we said it was the end

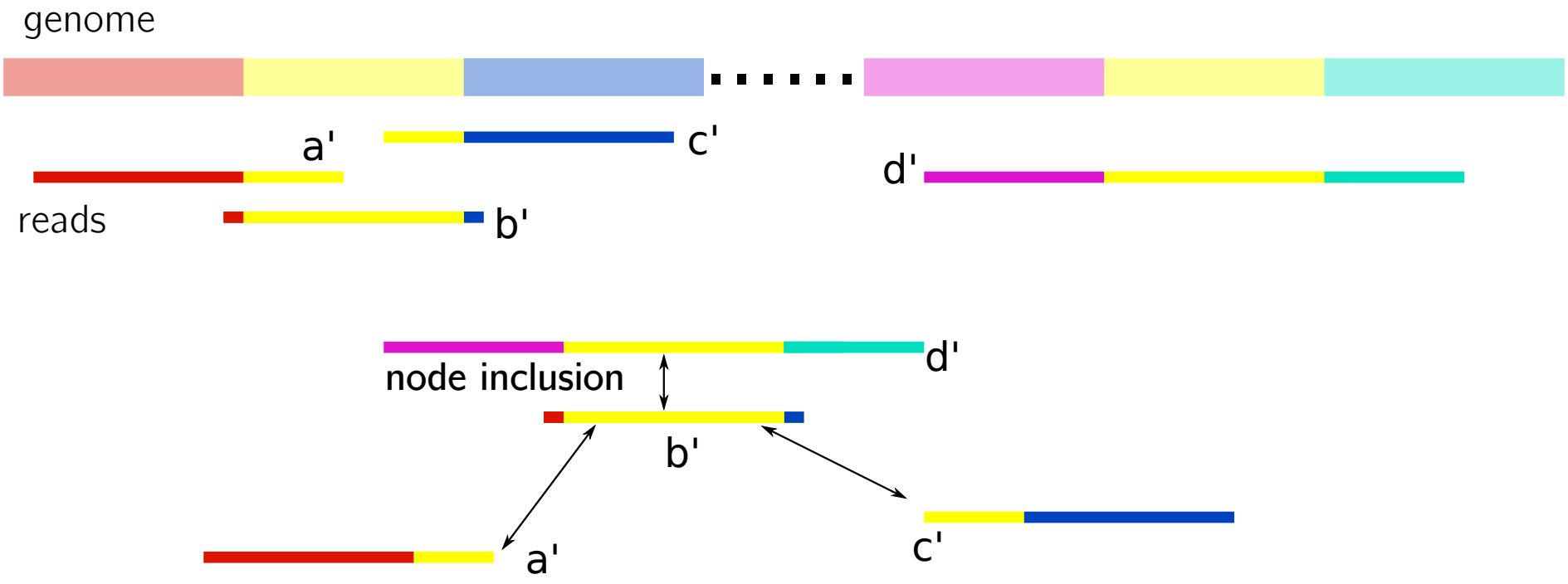
Just one or two more graphs



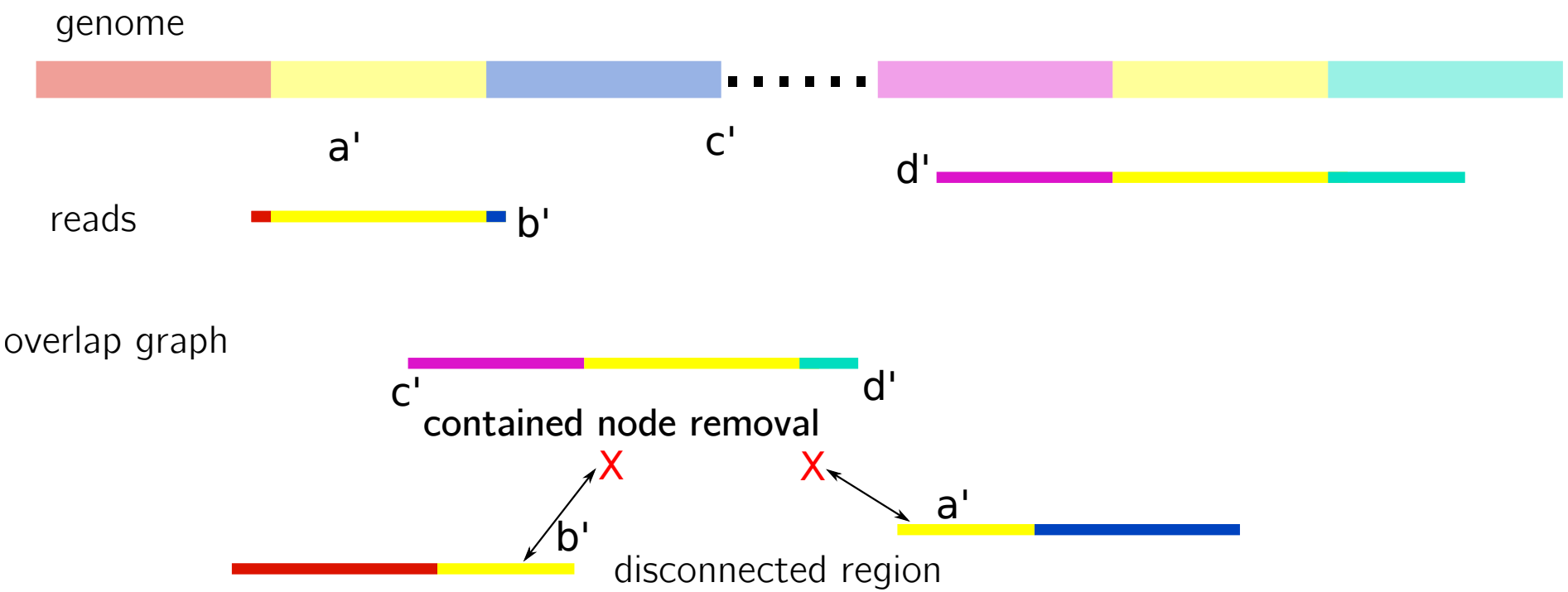
- An overlap graph limitation. Swept under the carpet?



- An olverlap graph limitation. Swept under the carpet?



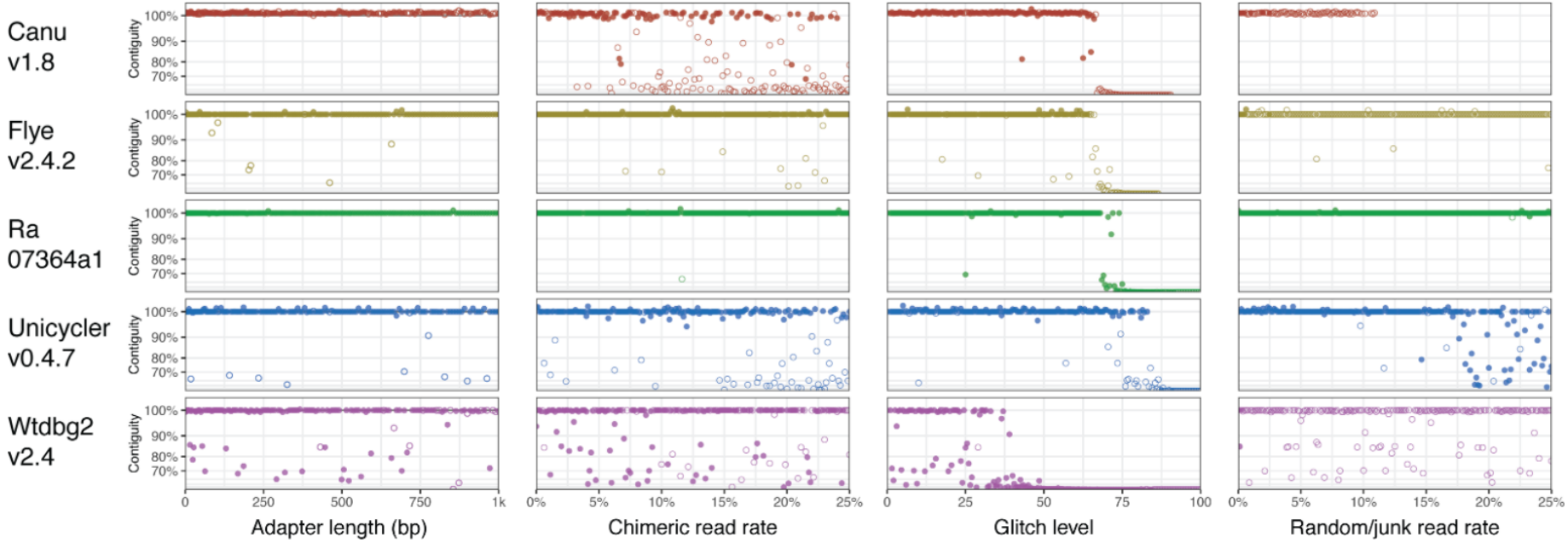
- An overlap graph limitation. Swept under the carpet?



Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the software fails



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

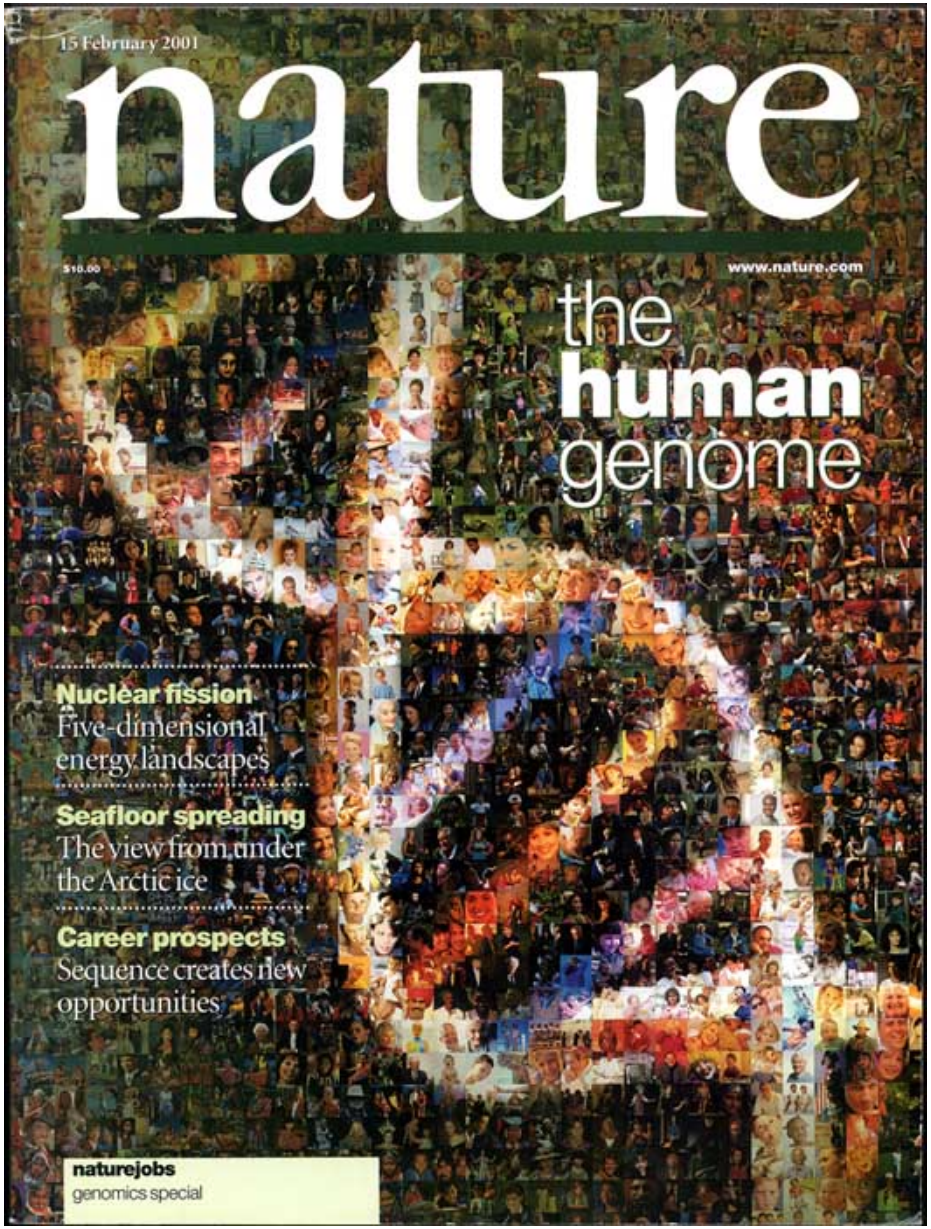
- Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the data cannot solve the problem

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

- 20 years later



• Telomere-to-Telomere consortium

Has produced in 2021 a complete human genome with one contig per chromosomes !

- 30x PacBio HiFi
- 120x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 10X Genomics, BioNano DLS and Arima Genomics HiC
- 100 authors from 50 labs

Long reads assemblers

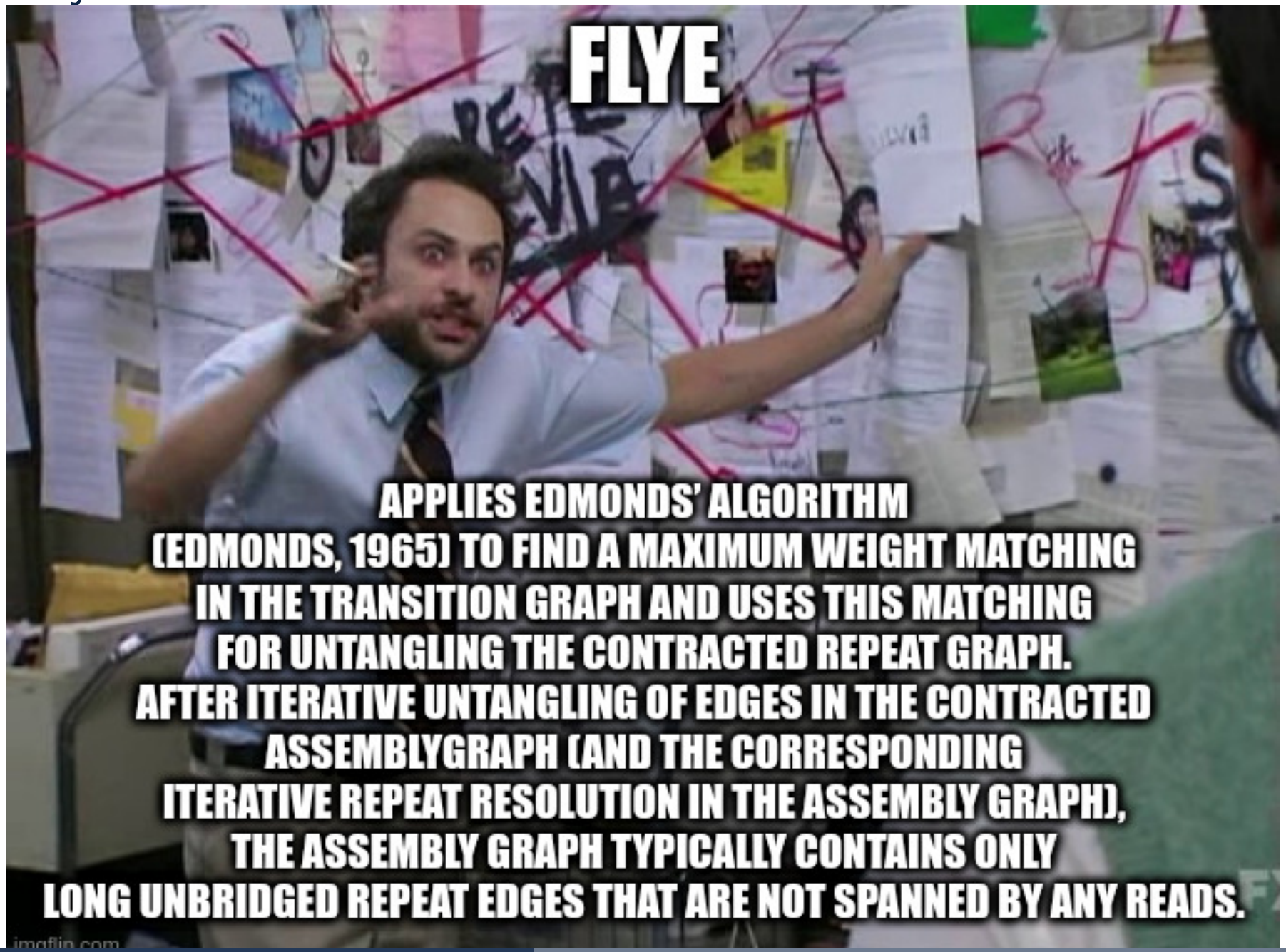
Best performing assemblers

- Flye (Repeat graph) [Kolmogorov et al 2019]
- Raven (OLC) [Vaser et al 2021]
- NECAT/MECAT(OLC) [Xiao et al 2017]

Other notable assemblers

- Canu (Greedy) [Koren et al 2017]
- Shasta (OLC) [Saffin et al 2020]
- Redbean (fuzzy de Bruijn graph) [Ruan 2019]
- ...

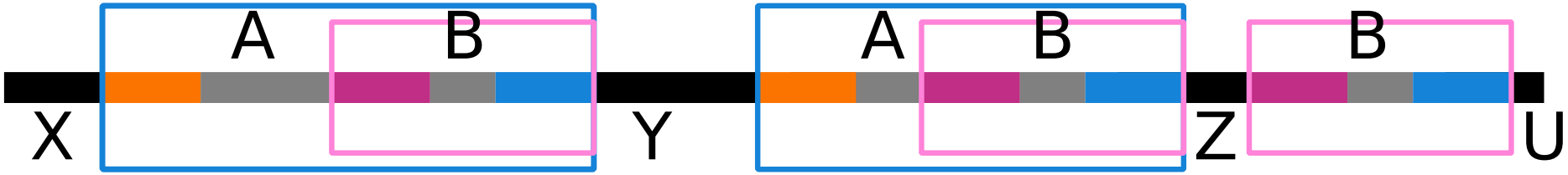
Flye


FLYE
APPLIES EDMONDS' ALGORITHM
(EDMONDS, 1965) TO FIND A MAXIMUM WEIGHT MATCHING
IN THE TRANSITION GRAPH AND USES THIS MATCHING
FOR UNTANGLING THE CONTRACTED REPEAT GRAPH.
AFTER ITERATIVE UNTANGLING OF EDGES IN THE CONTRACTED
ASSEMBLYGRAPH (AND THE CORRESPONDING
ITERATIVE REPEAT RESOLUTION IN THE ASSEMBLY GRAPH),
THE ASSEMBLY GRAPH TYPICALLY CONTAINS ONLY
LONG UNBRIDGED REPEAT EDGES THAT ARE NOT SPANNED BY ANY READS.

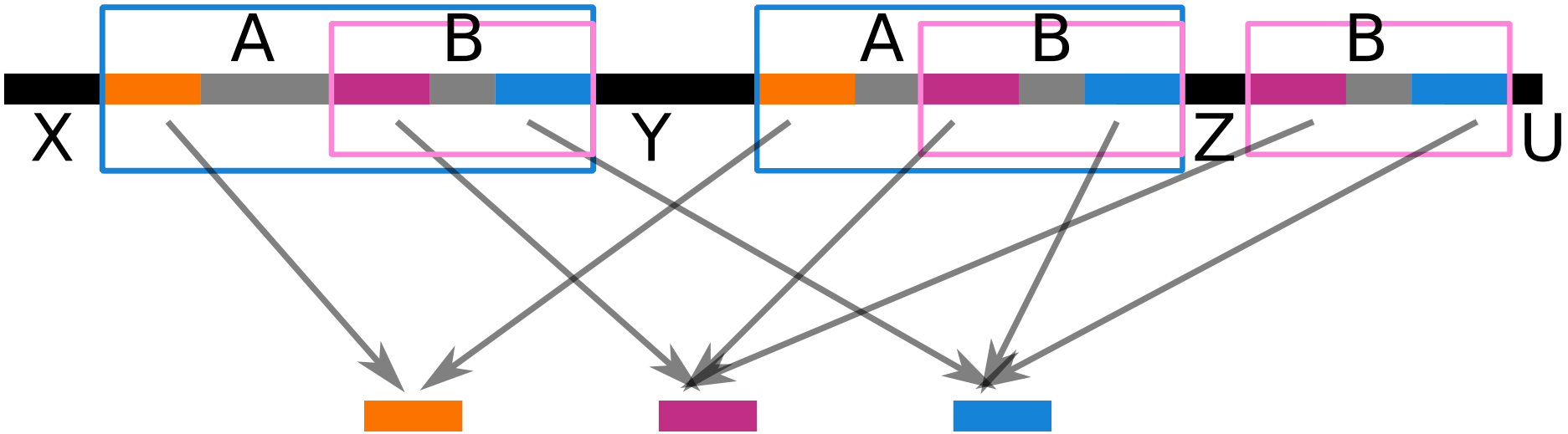
Repeat graph

a genome

highlighted repeated regions

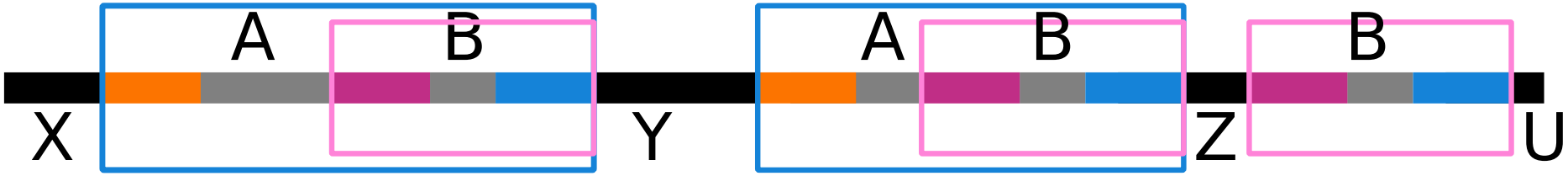


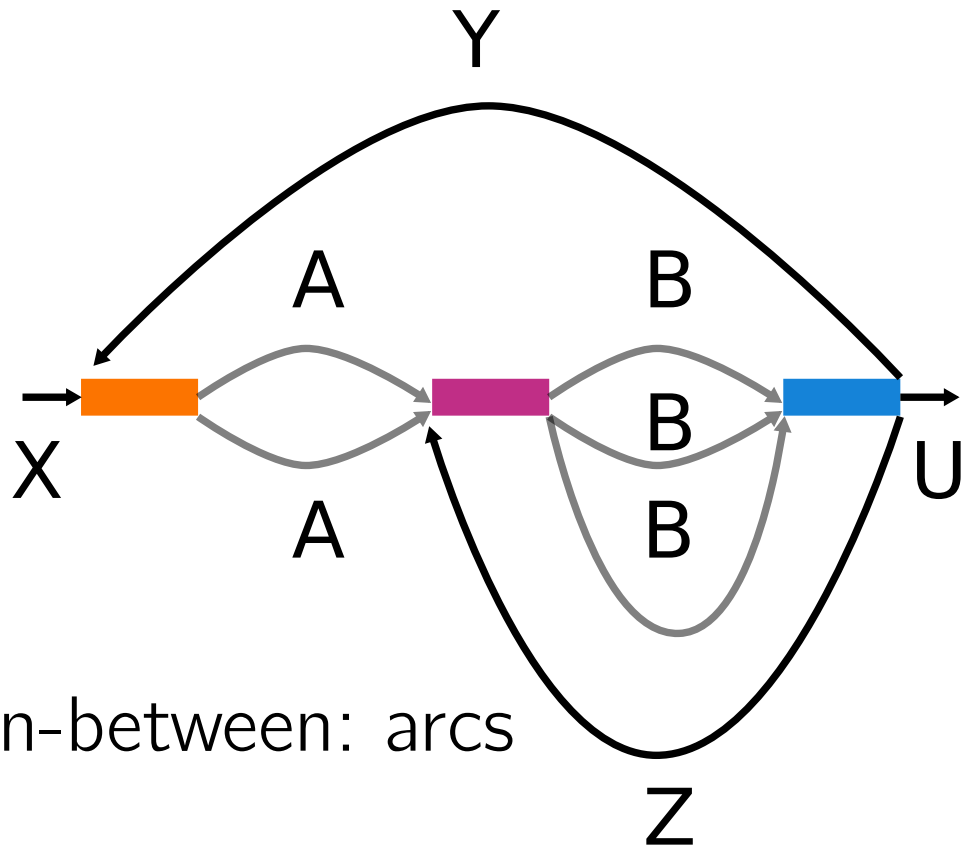
Repeat graph



repeats extremities: graph's nodes

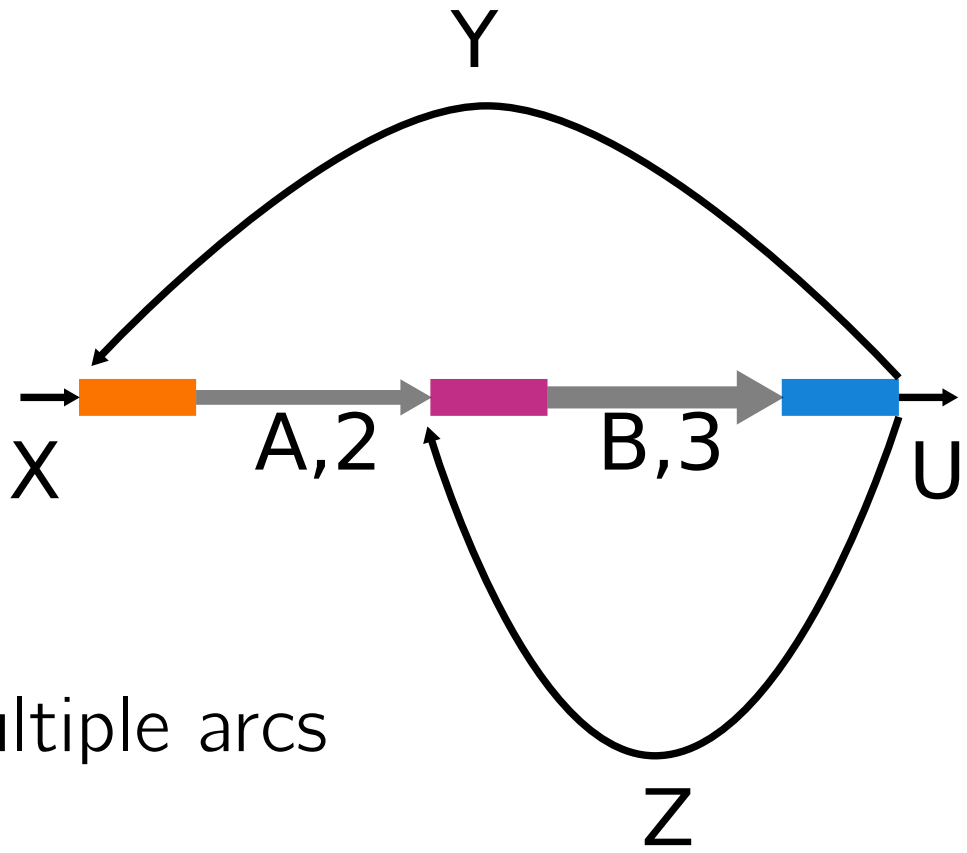
Repeat graph





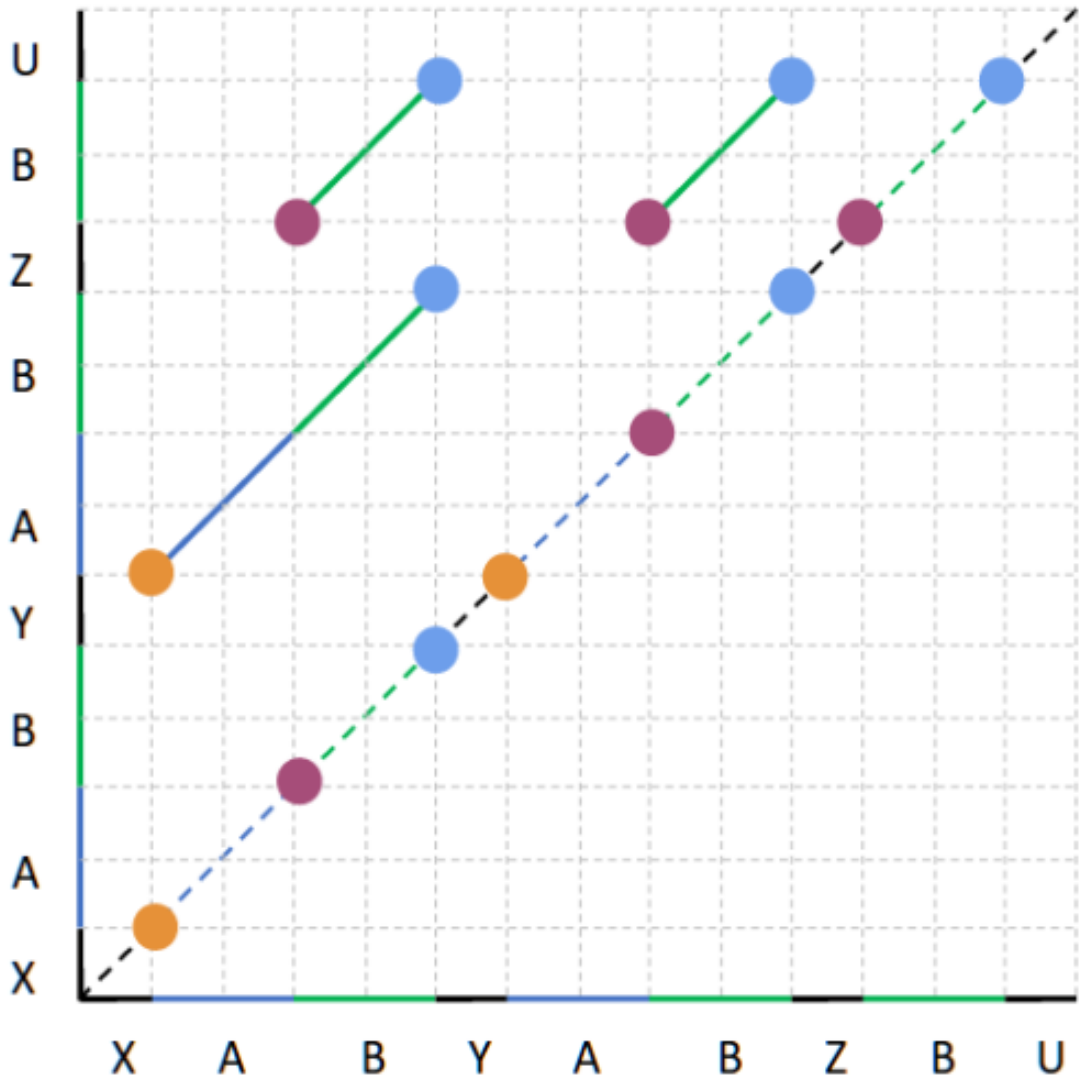
sequences in-between: arcs

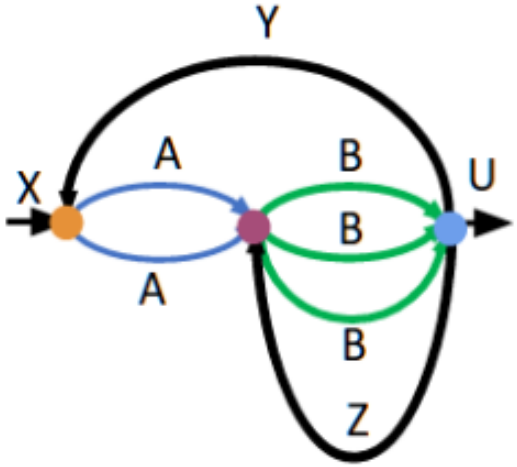
Repeat graph

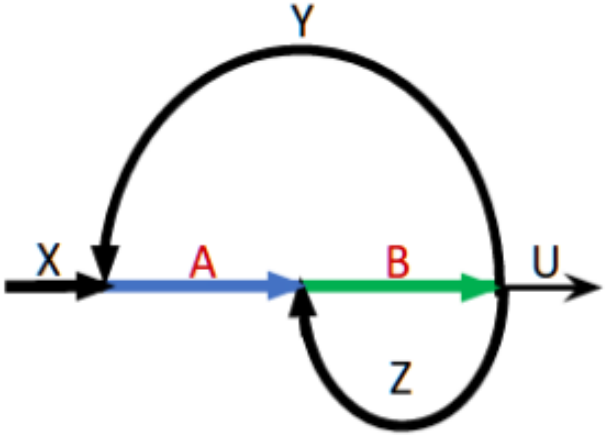


collapse multiple arcs

Repeat graph



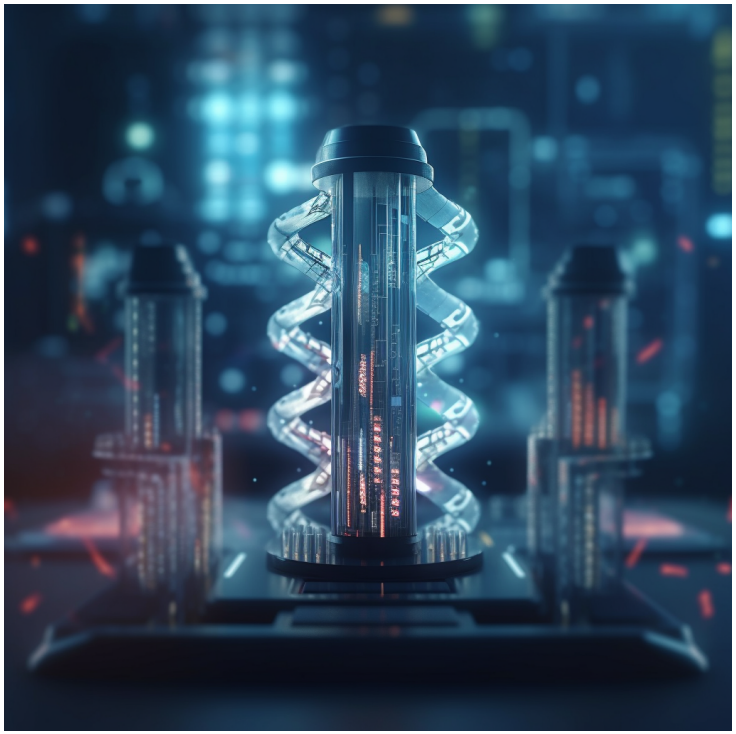




Long read assembly summary

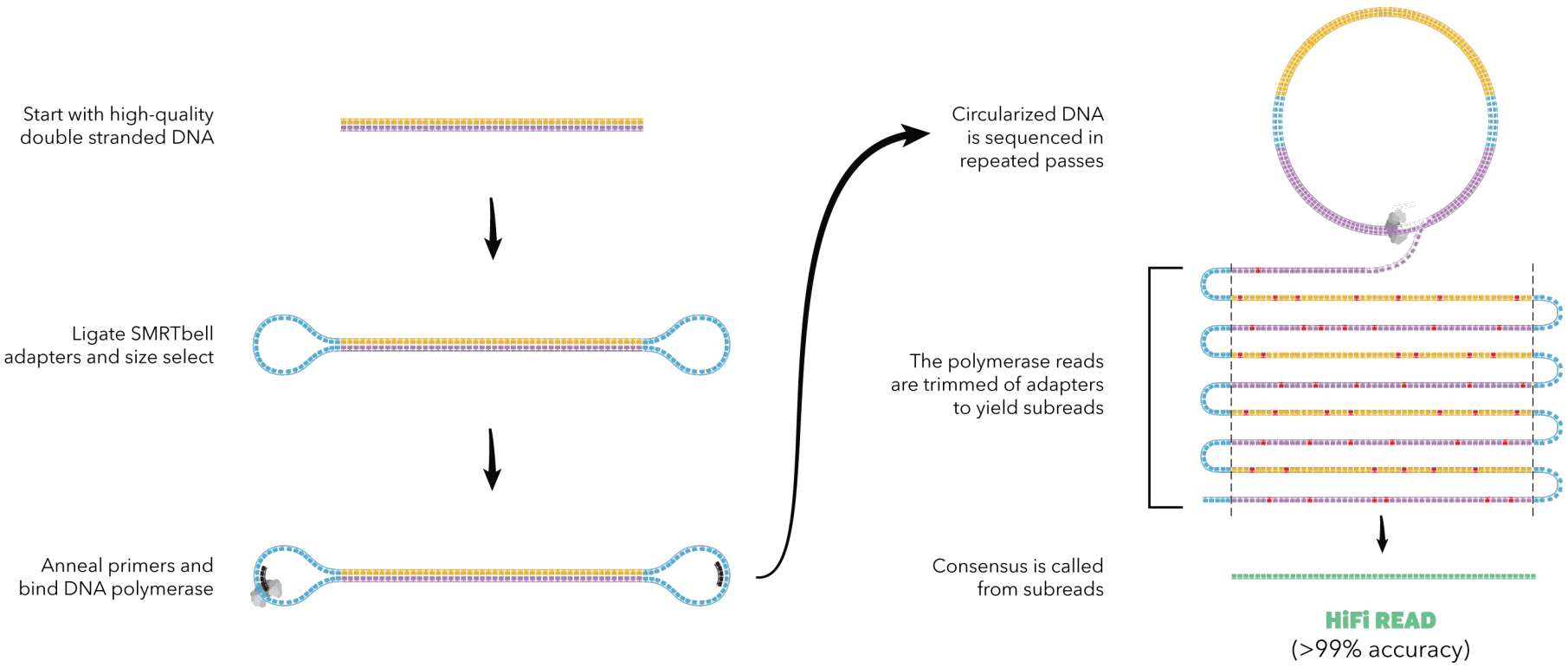
- Overlap graphs with quick overlap computation
- Long reads can span repeats and improve assemblies
- Methods to polish contigs

Future of assembly



The future of genome sequencing according to MidJourney.
Left: yes, but kinda boring. Right: hmmm.

Consensus during sequencing



HiFi data

Stands for "High Fidelity"

Very low error rates $\approx$ 0.1% 0.01%

Almost only homopolymer errors remain

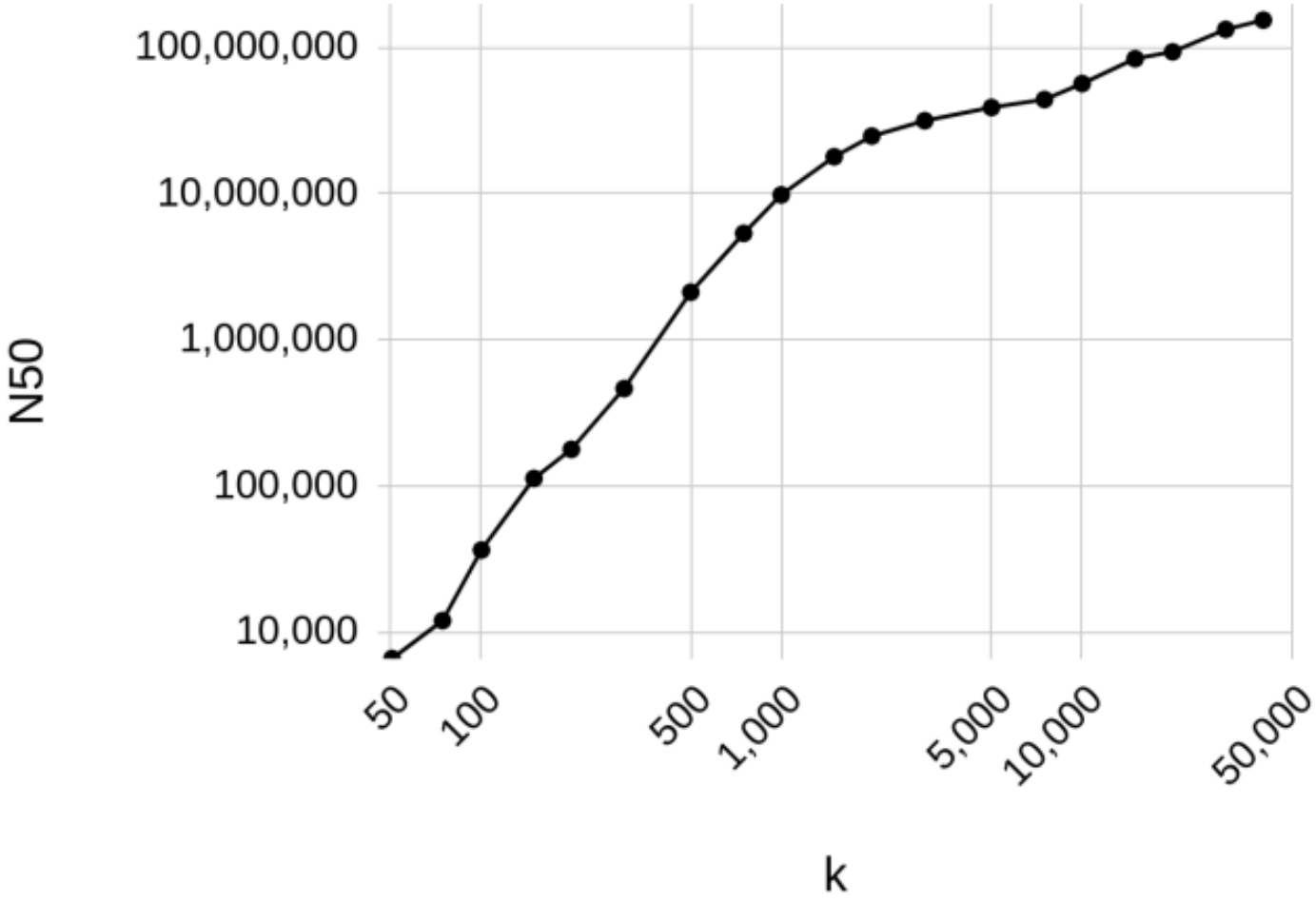
HiFi Assembly

With almost error-less long reads we have several promising improvements ahead:

- Use de Bruijn graph (more efficient data structures)
- Assemble large genomes very fast
- Perform diploid assembly

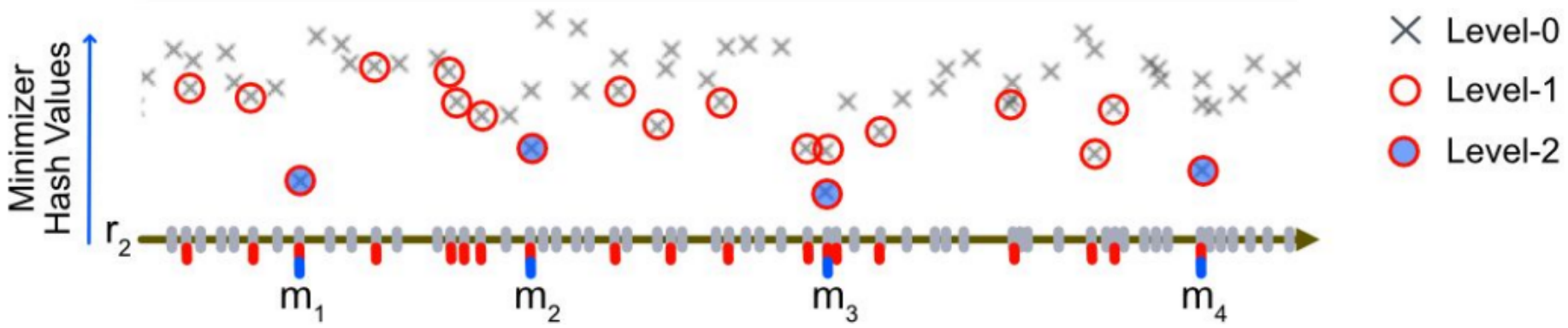
de Bruijn graph Assembly

Using K=500 and K=5000 de Bruijn graphs to assemble

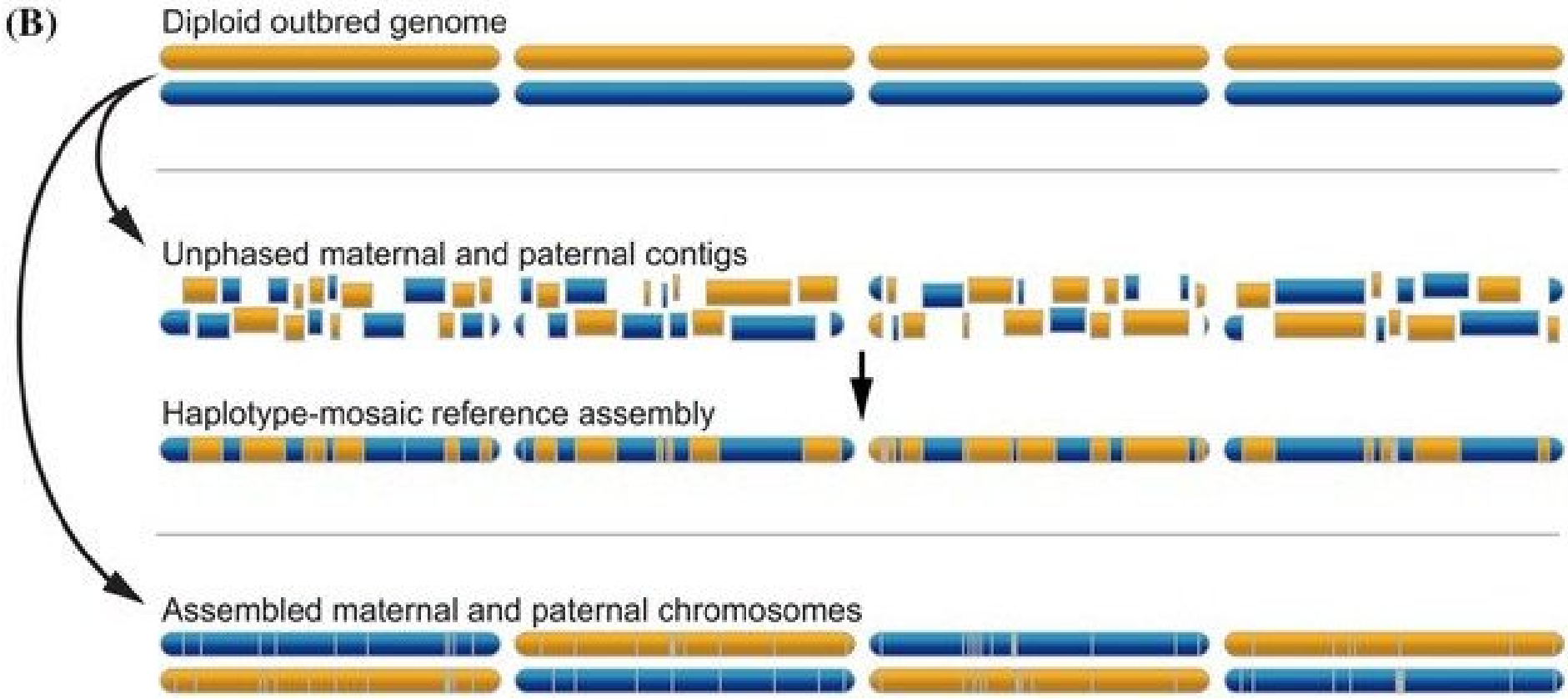


Very fast genome assembly

Human genome assembled within 2 hours (Peregrine assembler) and 10 minutes (RMBG assembler)



• Diploid assembly

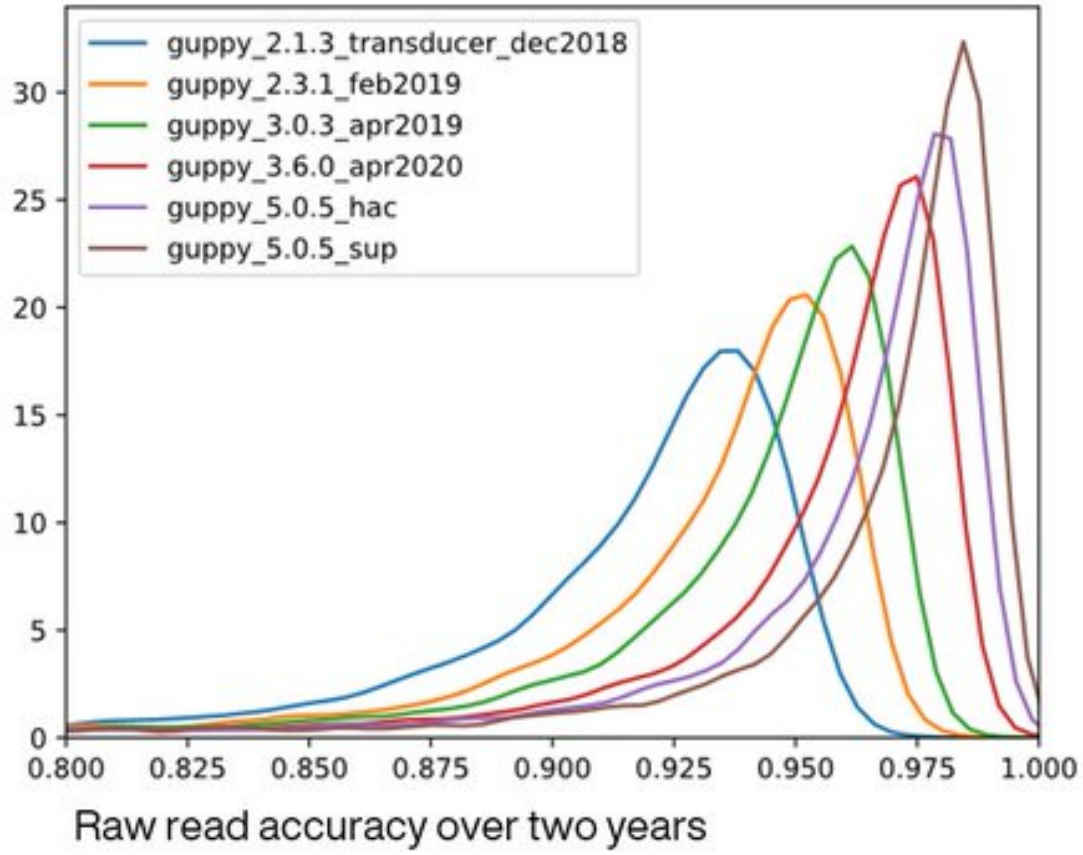


- Human diploid assembly (20/46 chr)

Asm	Contig NG50 (Mb)	Scaffold NG50 (Mb)	Hamming error	QV	#Errors Chr X & Y
Downsampled (35× HiFi / 60× ONT-UL)					
Verkko	12.90		0.13%	52.18	10
Verkko + trio	**80.77**	**102.55**	0.13%	52.40	10
Verkko + Hi-C	58.24	82.42	0.16%	52.48	10
LJA	0.39		0.14%	55.75	7
Hifiasm (unitigs)	0.35		0.23%	**61.05**	**2**
Hifiasm + trio	64.50		**0.06%**	60.86	26
Hifiasm + Hi-C	66.34		0.79%	60.57	37
Full-coverage (>100× HiFi / 85× ONT-UL)					
Verkko	17.35		**0.05%**	54.91	**5**
Verkko + trio	**134.00**	135.80	**0.05%**	55.77	**5**
Verkko + Hi-C	68.32	85.97	**0.05%**	55.57	**5**
HPRC curated	72.70	**146.75**	0.13%	**61.35**	26

• Ongoing progress

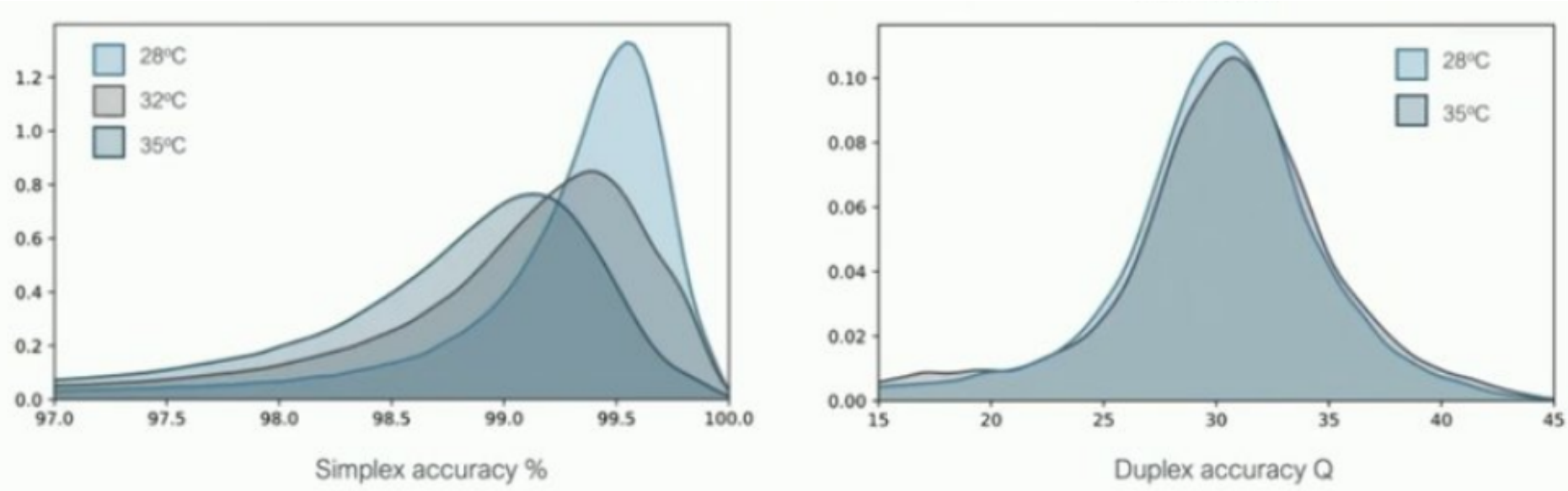
Errors in Nanopore sequencing data are rapidly diminishing



Q20 chemistry achieved modal accuracy $\approx 99\%$

- High fidelity Nanopore incoming?

Nanopore duplex reads could deliver long and precise reads in the future



Take home messages

Ultra fast summary

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

Ongoing work

Assembly Challenges

- Reconstruct haplotypes
- Scaling on large genomes
- Robustness to noisy data
- Repetitive regions

- 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