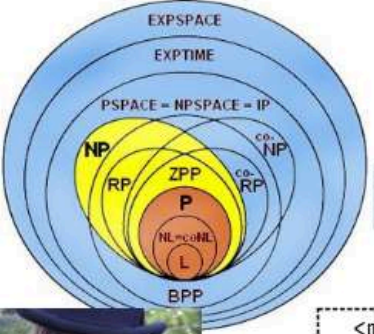
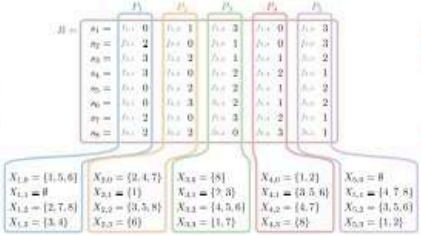
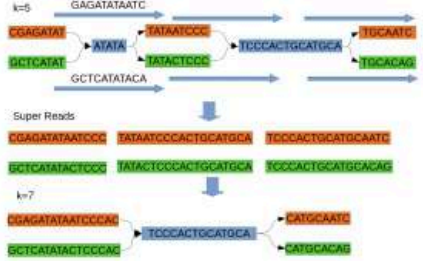
A little tour of assembly methods

Antoine Limasset & Camille Marchet
CRIStAL, Université de Lille, CNRS, France

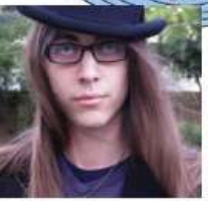
antoine.limasset@gmail.com @npmalfoy
camille.marchet@univ-lille.fr @camillemrcht

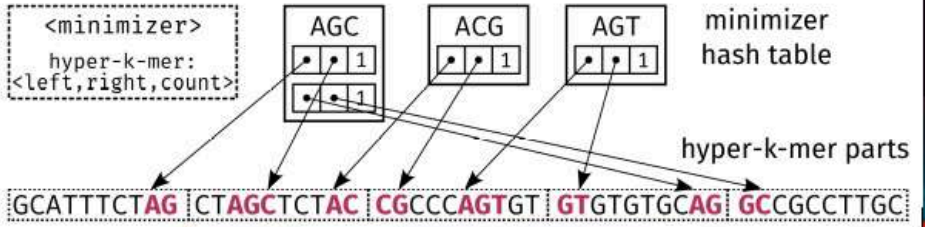

EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
co-NP
RP
ZPP
co-RP
P
NL=coNL
L
BPP




k=5
GAGATATAATC
CGAGATAT
ATATA
TATAATCCC
TATACTCCC
TCCCACTGCATGCA
TGCAATC
TGCACAG
GCTCATAT
GCTCATATACA
Super Reads
CGAGATATAATCCC
TATAATCCCACTGCATGCA
TCCCACTGCATGCAATC
GCTCATATACTCCC
TATACTCCCACTGCATGCA
TCCCACTGCATGCACAG
k=7
CGAGATATAATCCCAC
TCCCACTGCATGCA
CATGCAATC
GCTCATATACTCCCAC
CATGCACAG

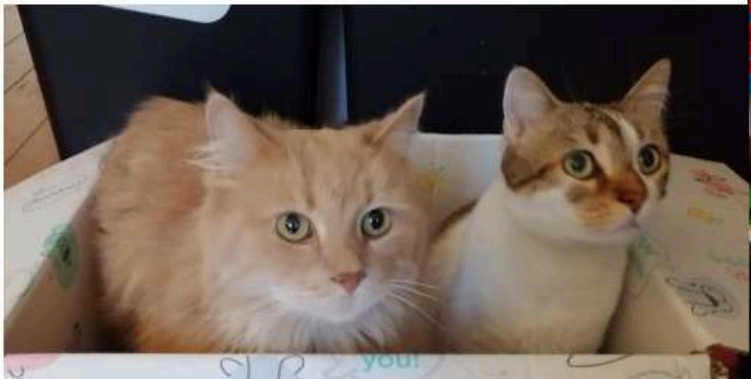

NIBANA




<minimizer>
hyper-k-mer:
<left,right,count>
AGC
ACG
AGT
minimizer
hash table
hyper-k-mer parts
GCATTTCTAG
CTAGCTCTAC
CGCCCAGTGT
GTGTGTGCAG
GCCGCCTTGC


BALATRO


The Rust
Programming
Language




typst
The new foundation
for documents

2013 (Lyon)

LBBE
LABORATOIRE DE BIOMÉTRIE ET BIOLOGIE ÉVOLUTIVE

2015 (Rennes)

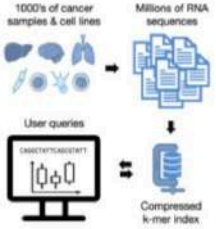
UMR IRISA

Since 2018: Lille

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

USP Universidade de São Paulo Brasil

EMBL-EBI



Transipedia:
browse RNA-seq data and abundances at scale

RECOMB-SEQ

wellcome connecting science

Genome Informatics
2-4 december 2026
Wellcome Genome Campus, UK


2020
2022
2023
2024
2025
2026

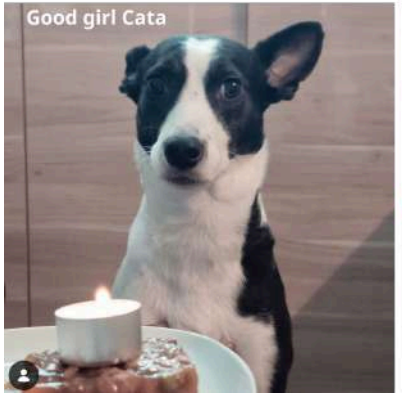
Content of this course

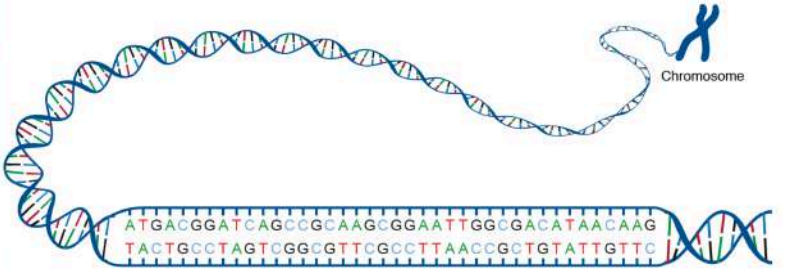
- How to reconstruct a genome with sequencing data?
- What are the main challenges?
- Which solutions have been proposed?

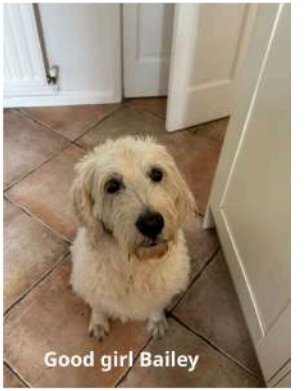
Bingo: find the French title of a (SciFi) book that we both love.

genome size: $\sim$ 40 gigabases

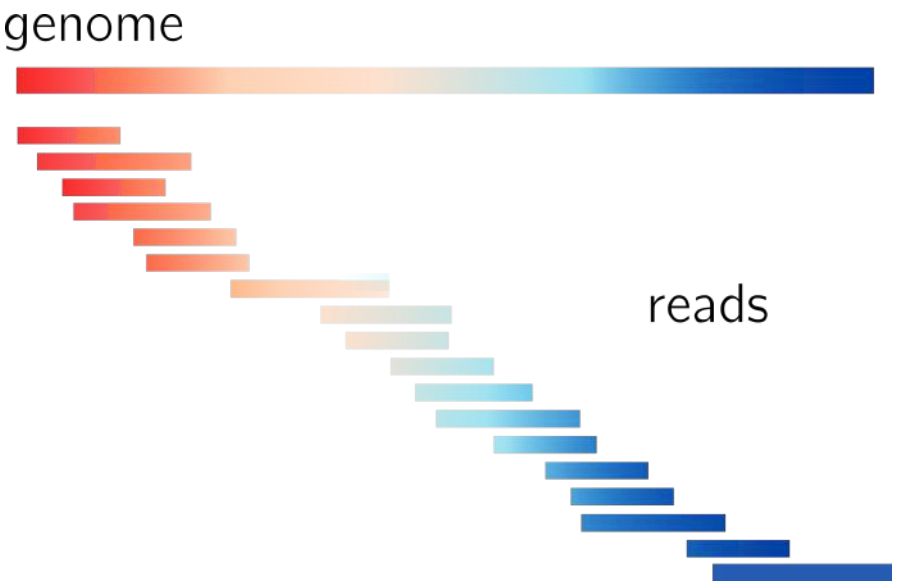
• Accessing a genome



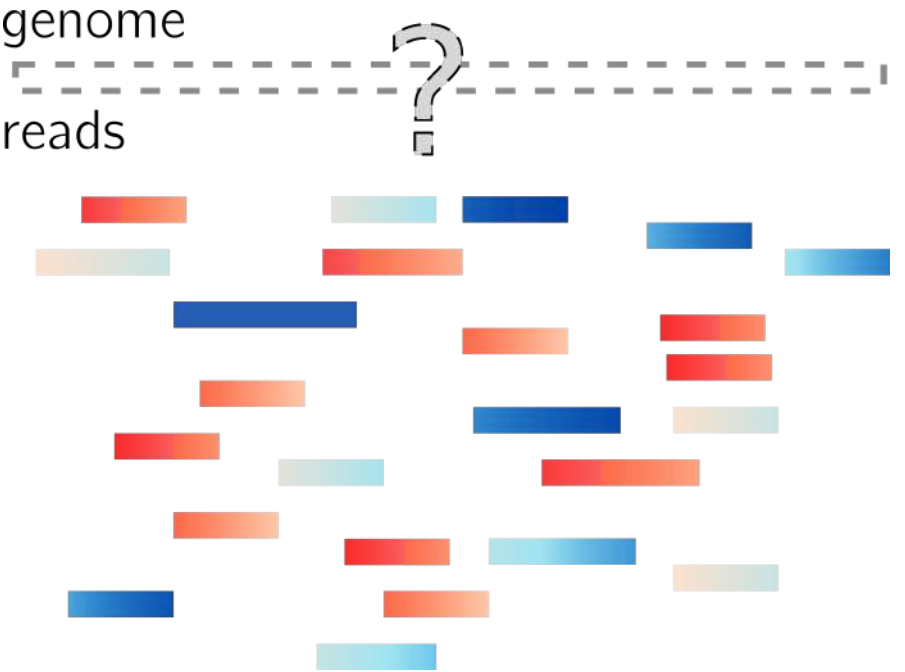




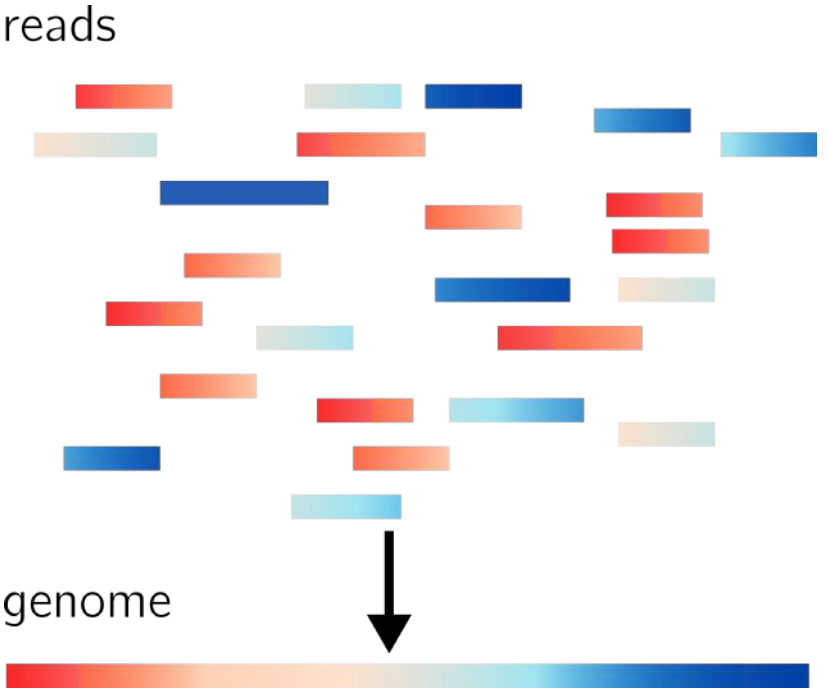
- **Reads are subsequences from the genome**



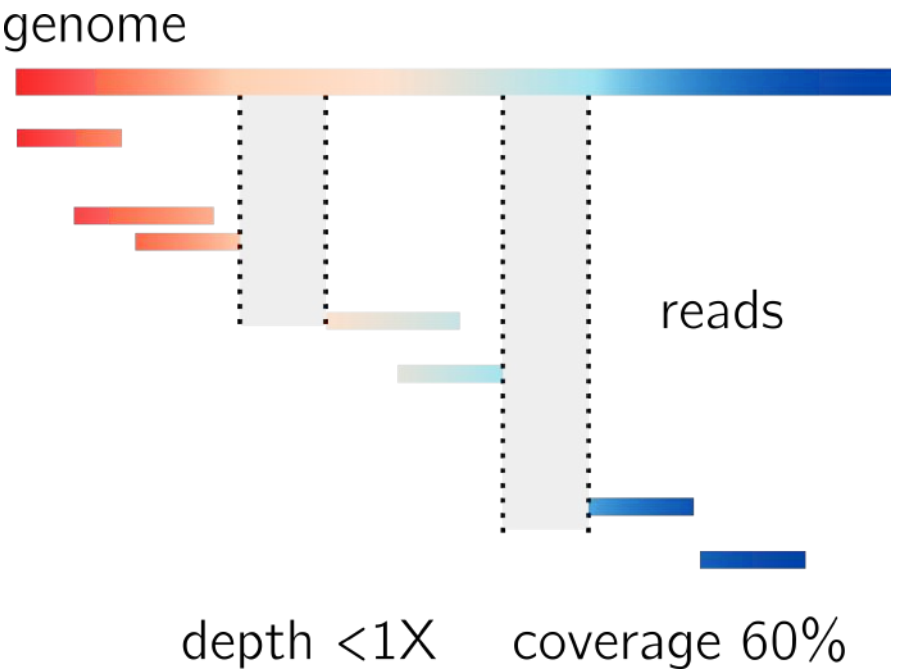
- **Reads are shuffled subsequences from the genome**



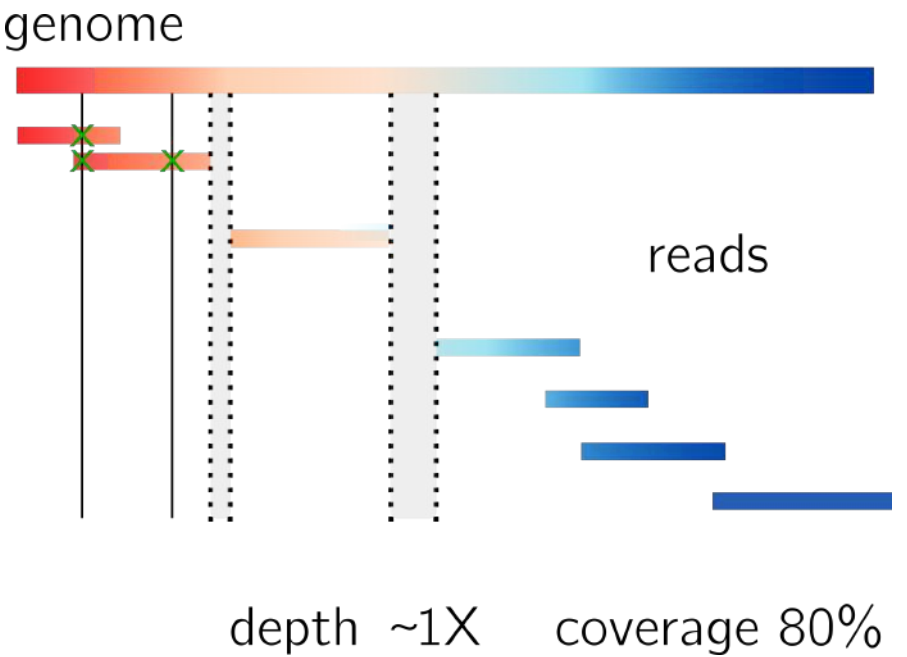
- **Genome assembly task**



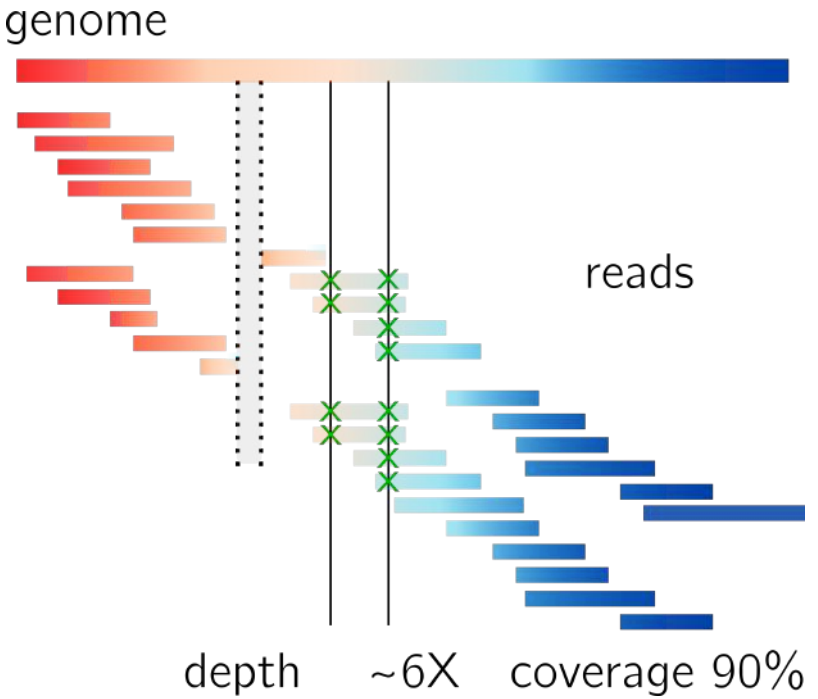
- **Genome sequencing: depth & coverage**



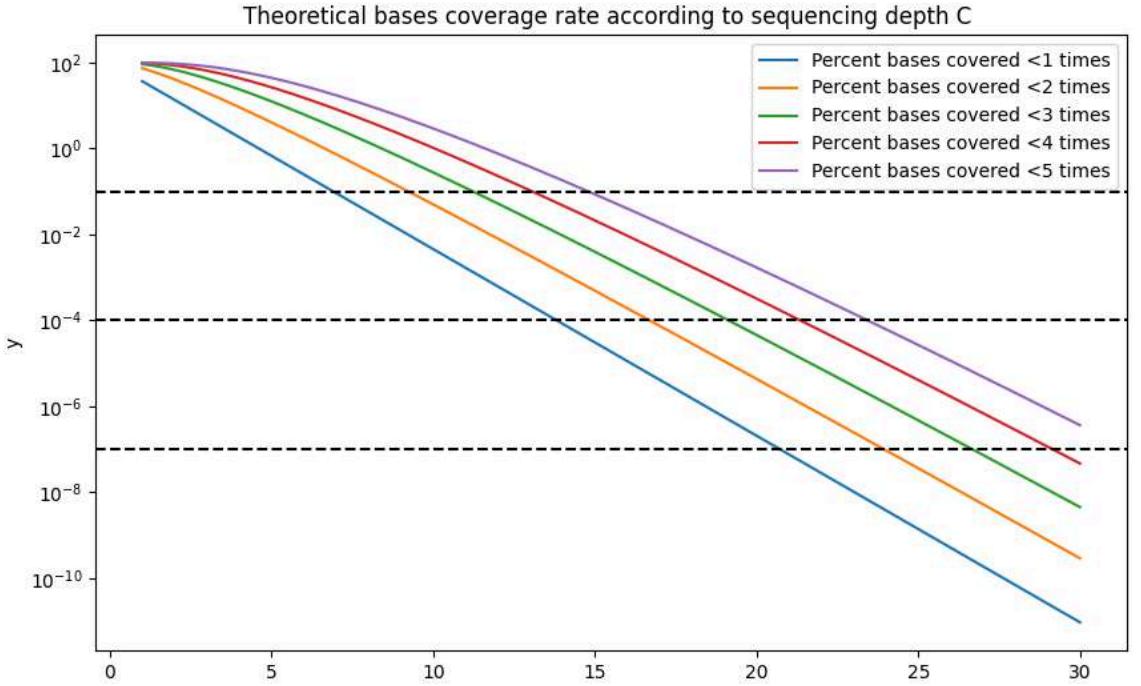
- **Genome sequencing: depth & coverage**



- **Genome sequencing: depth & coverage**

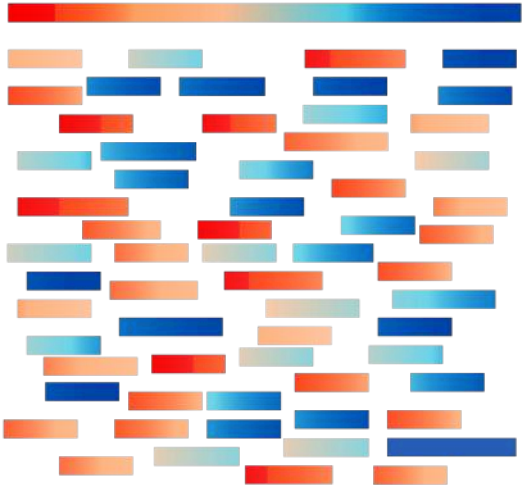


Poisson law


Theoretical bases coverage rate according to sequencing depth C
Percent bases covered <1 times
Percent bases covered <2 times
Percent bases covered <3 times
Percent bases covered <4 times
Percent bases covered <5 times
y
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
5
10
15
20
25
30

- **First experiment: theoretical, errorless reads for a *very good boy***

100kb region from the genome
(only for the record, we actually don't have it)

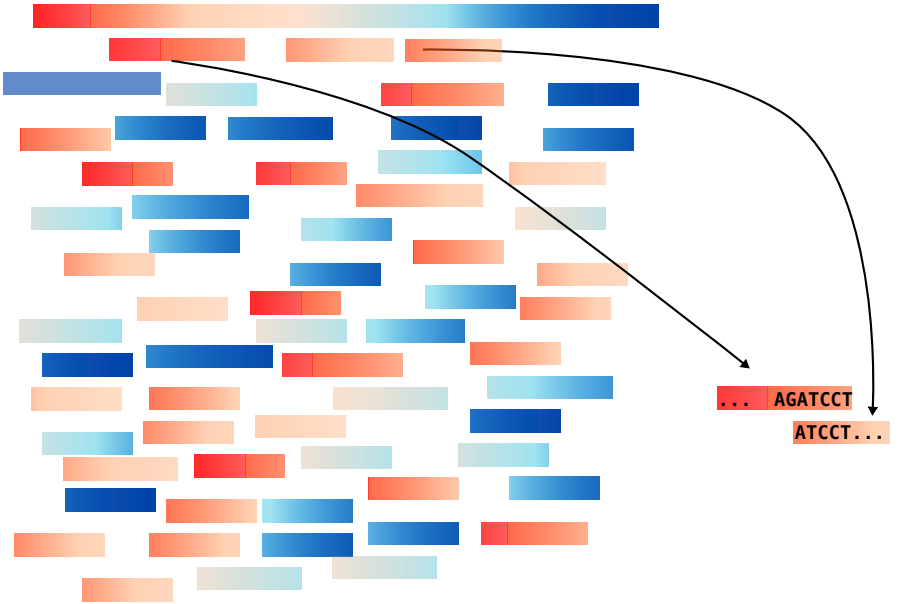


Genome size
2 billion bases

Reads
10 million
mean size 10kb

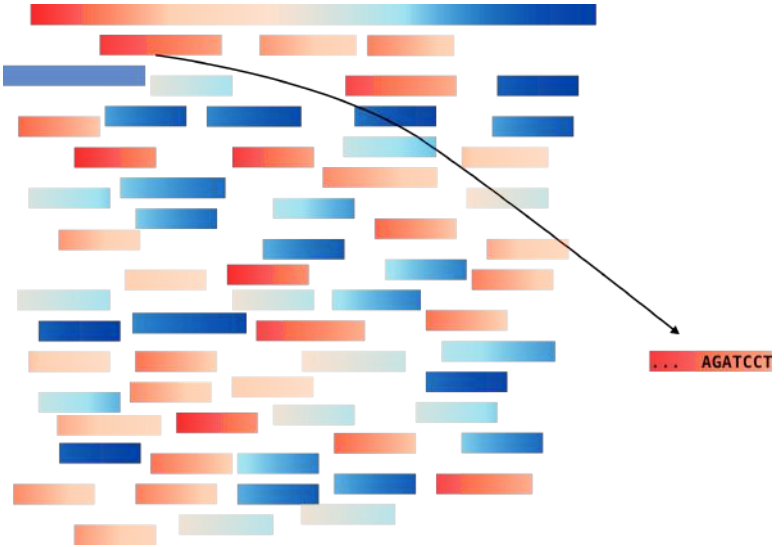
• Order according to overlaps

Overlaping reads are likely successive part of the genome



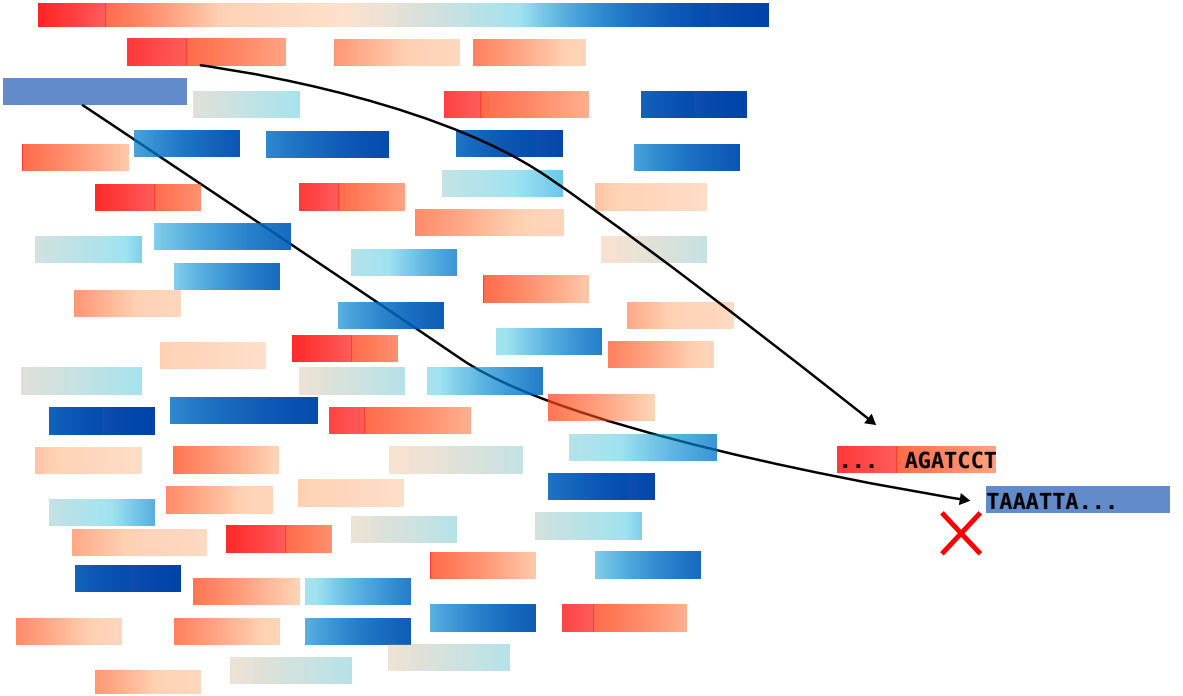
- **Check all reads for overlaps**

For a given read, scan all others

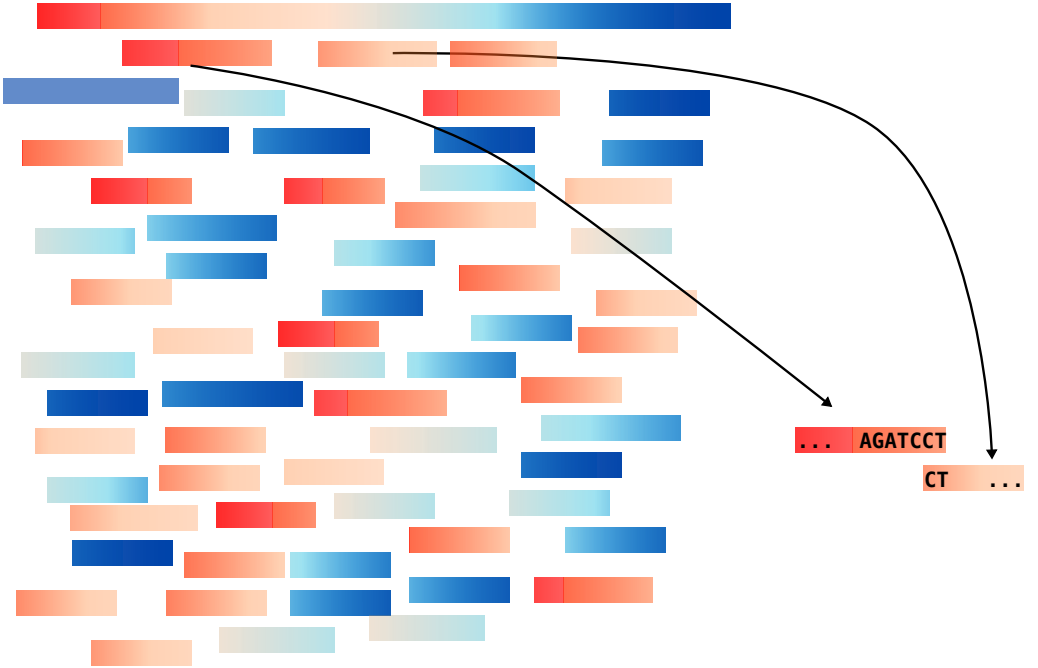


...and find **exact matches**

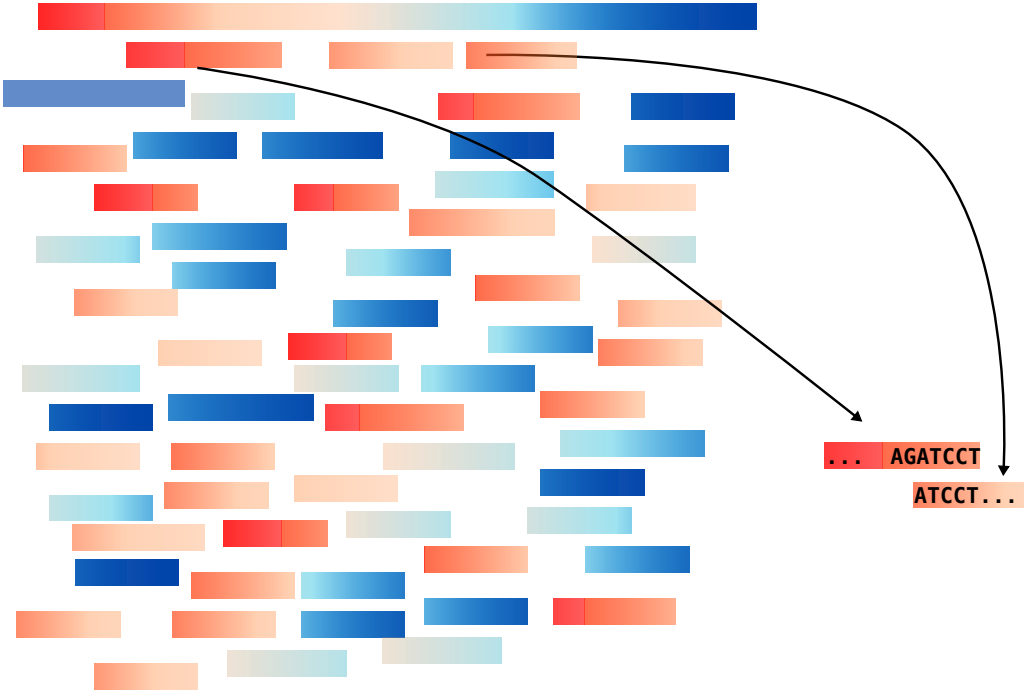
- **Most cases: no overlap**



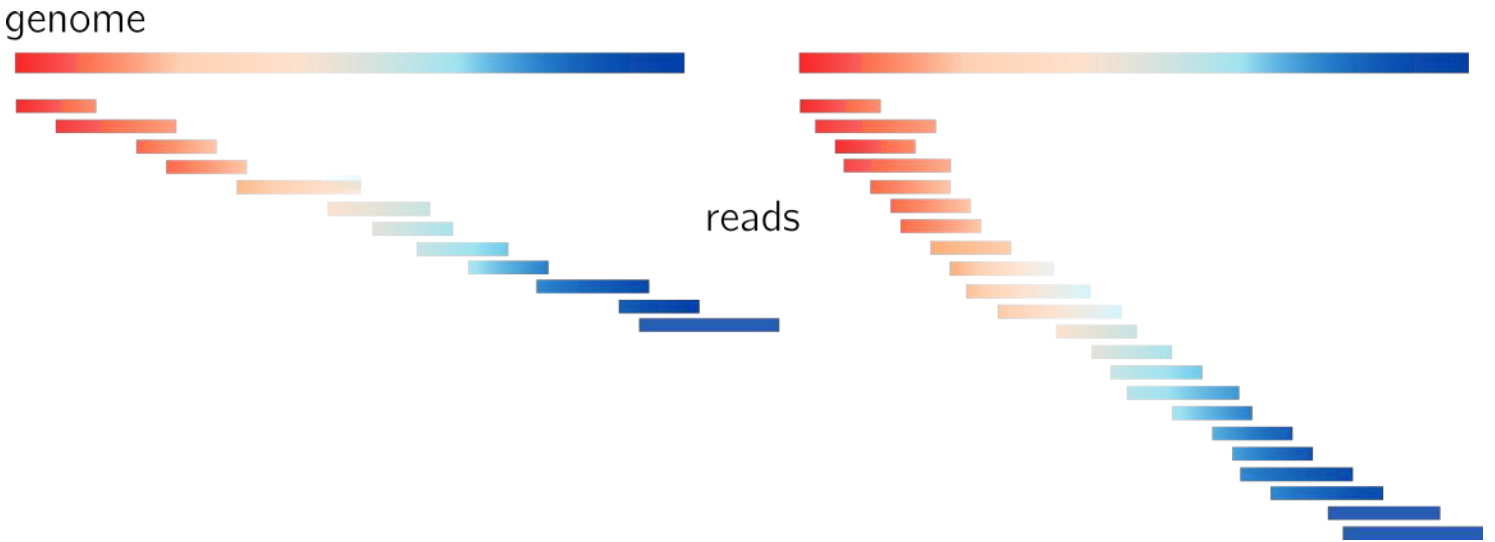
- **Small overlaps can happen "by chance"**



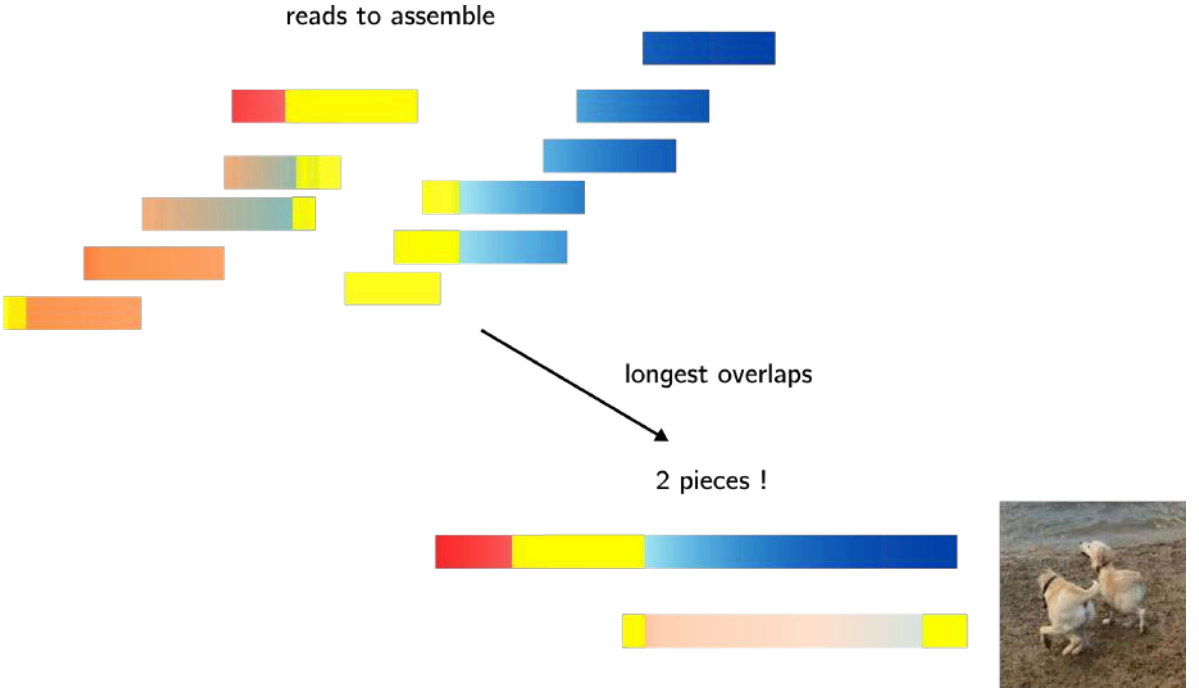
- We are more confident in longer overlaps



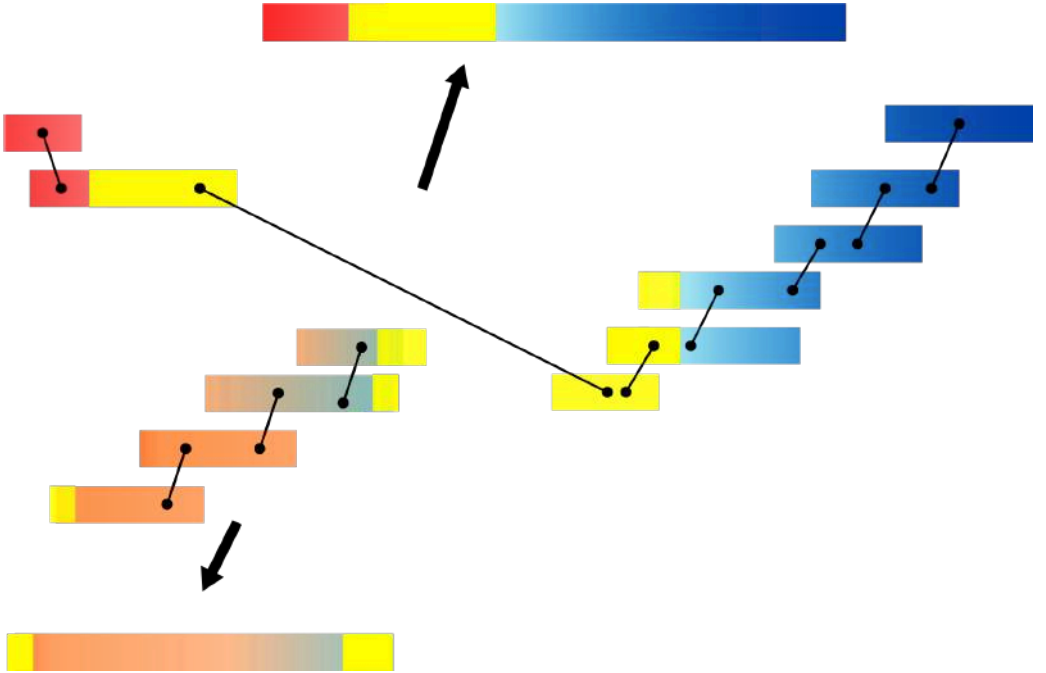
- **Higher depth, longer overlaps**



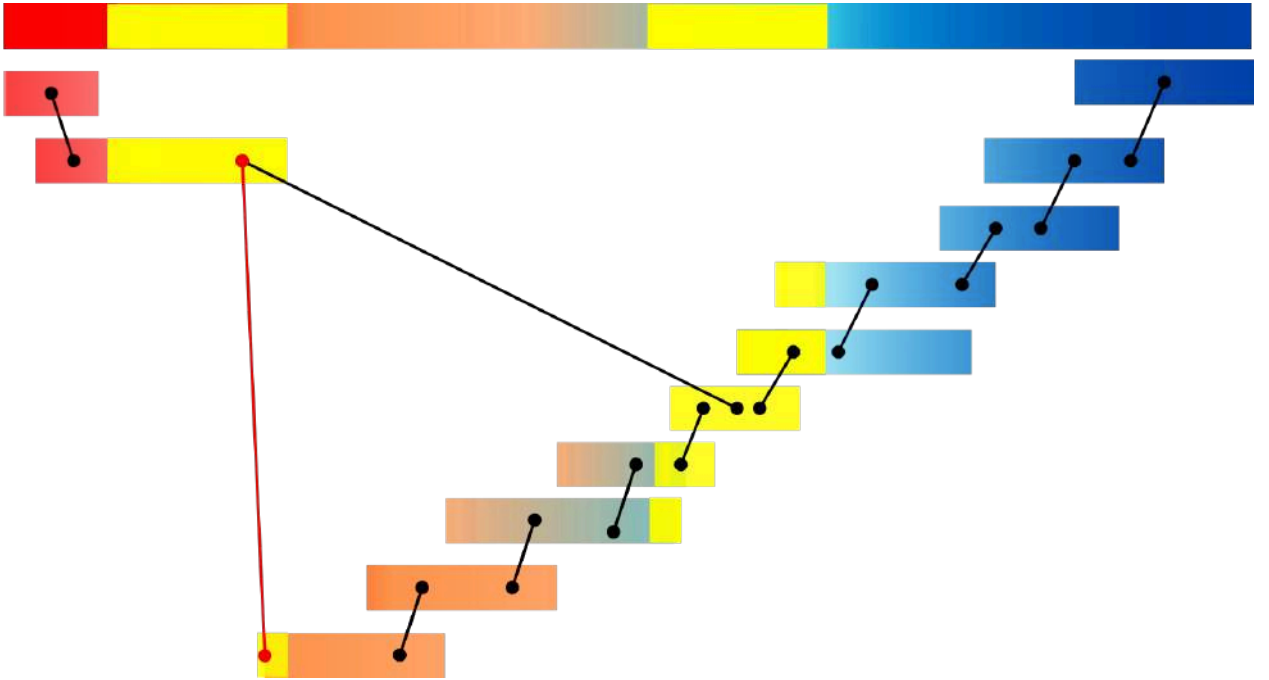
- **Something weird happened**



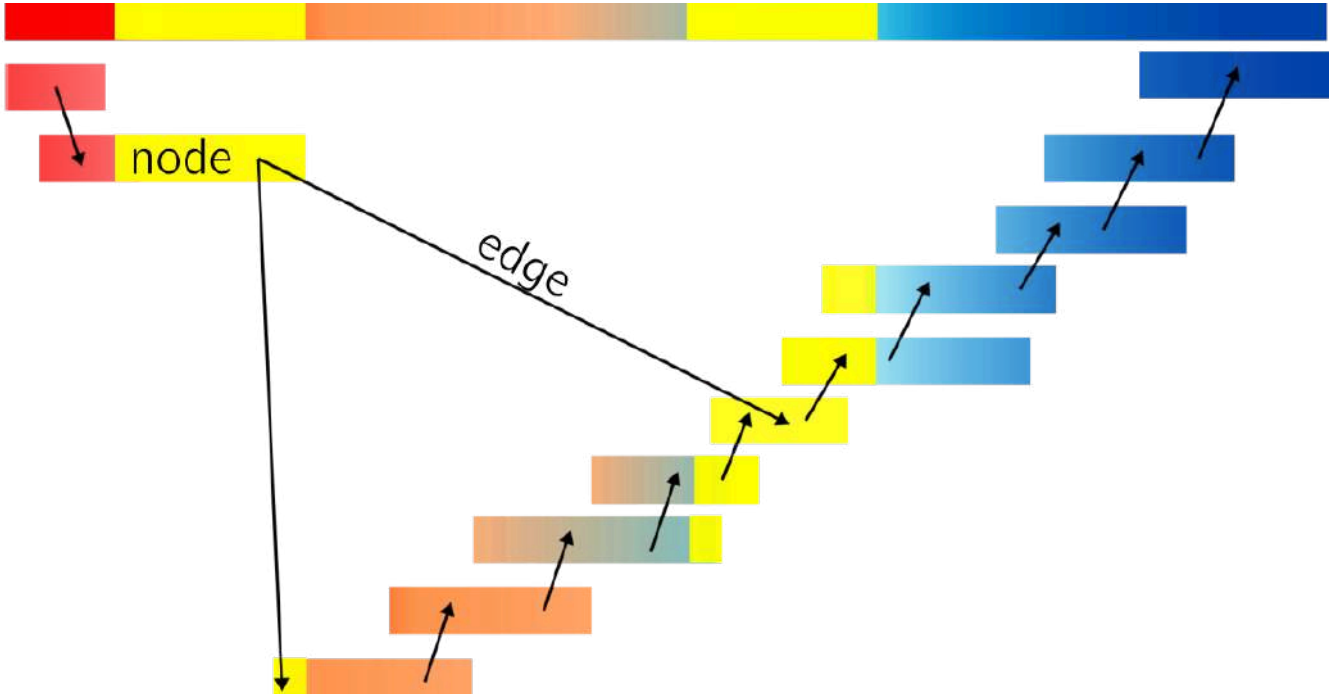
- **All longest overlaps**



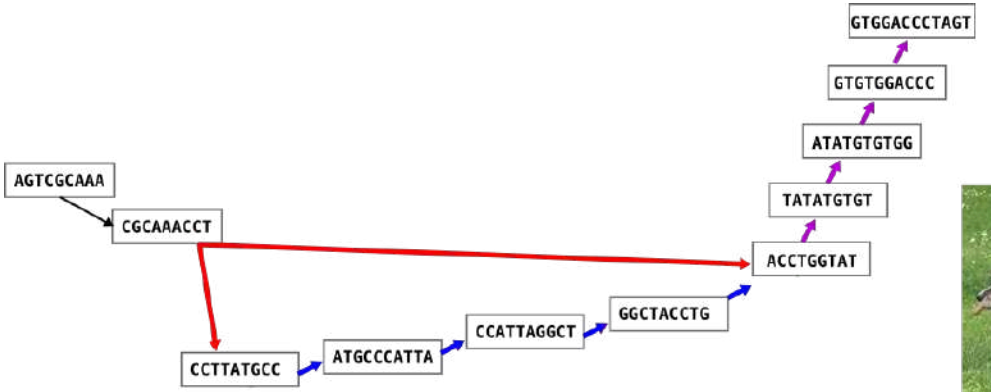
- **Take into account other overlaps?**



- **Look, a graph!**

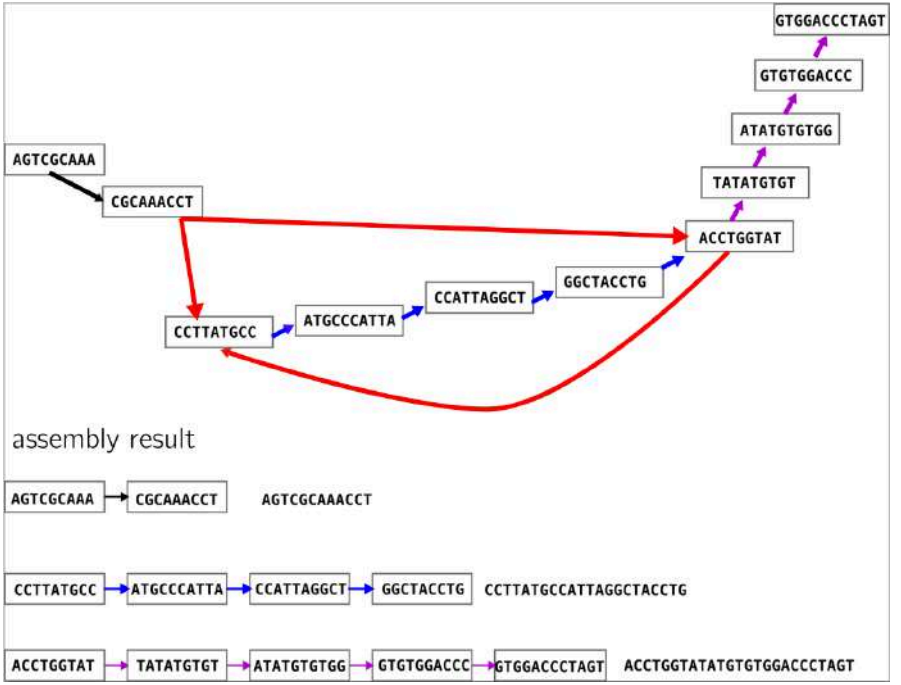


• Unsafe paths in an overlap graph

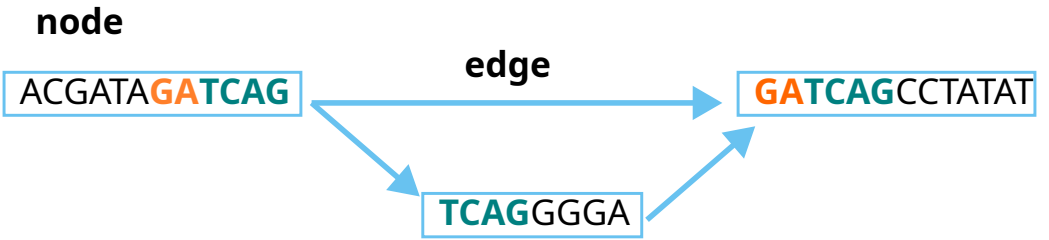


spurious assembly result

• Safe paths in an overlap graph



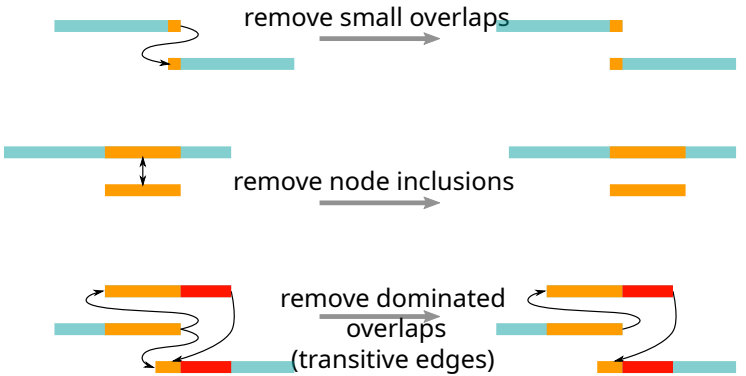
• A f*ing overlap graph (quoting Josie)

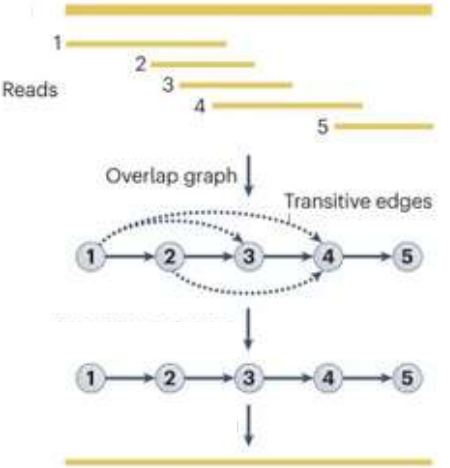


- Nodes are sequences of various sizes (reads)
- Edges are **long enough** overlaps between sequences

Overlap graph simplifications

helps graph simplification and traversal to output contigs

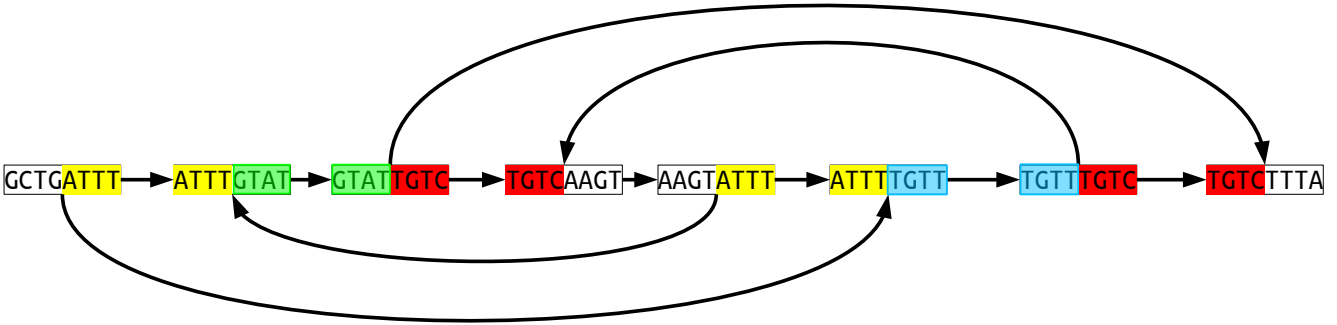




Multiple repeats

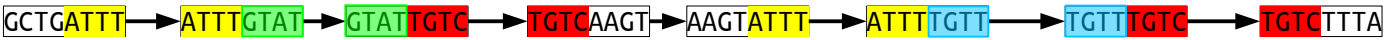
```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA
```

Overlap graph:



First solution

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

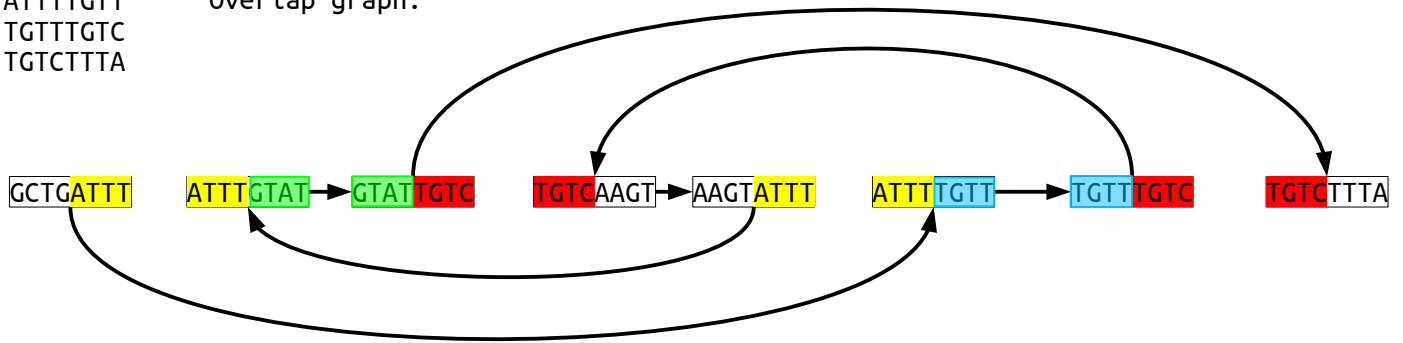


Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA

• Second solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:

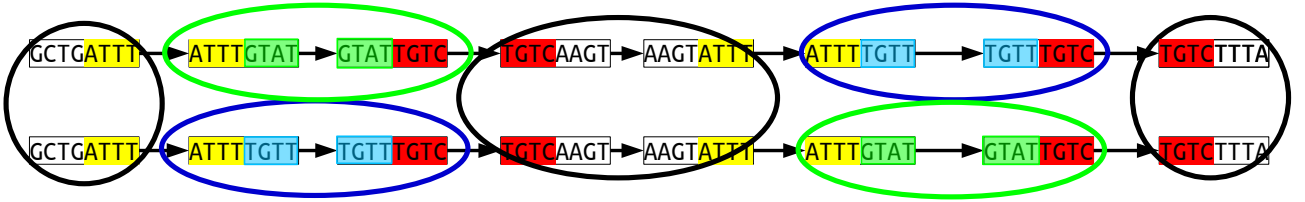


Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
GCTGATTTTGTTTGTCAAGTATTTGTATTGTCTTTA

Those two solutions are indistinguishable

• Parsimonious solution: do not assemble

Possible assemblies:



Genome pieces:

Repeats lead to the fragmentation of the assembly

Genomes pieces that make **con**-sensus across the differents solution are called **Con**-tigs

- **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:

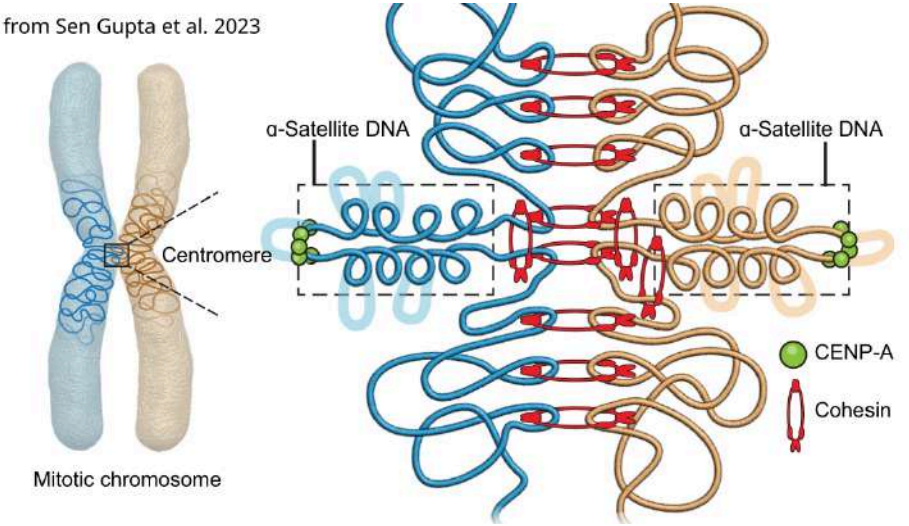
0.99999728949878430238317205542136383671202317193893202410 6...

The burden of assembly: genomic repeats

Amount of repeats larger than a given size in the human genome
21: 34,060,114
31: 12,857,884
51: 5,094,786
101: 973,550
1,001: 53,698
10,001: 833
100,001: 9
500,001: 8

Genomic repeats are NOT random events

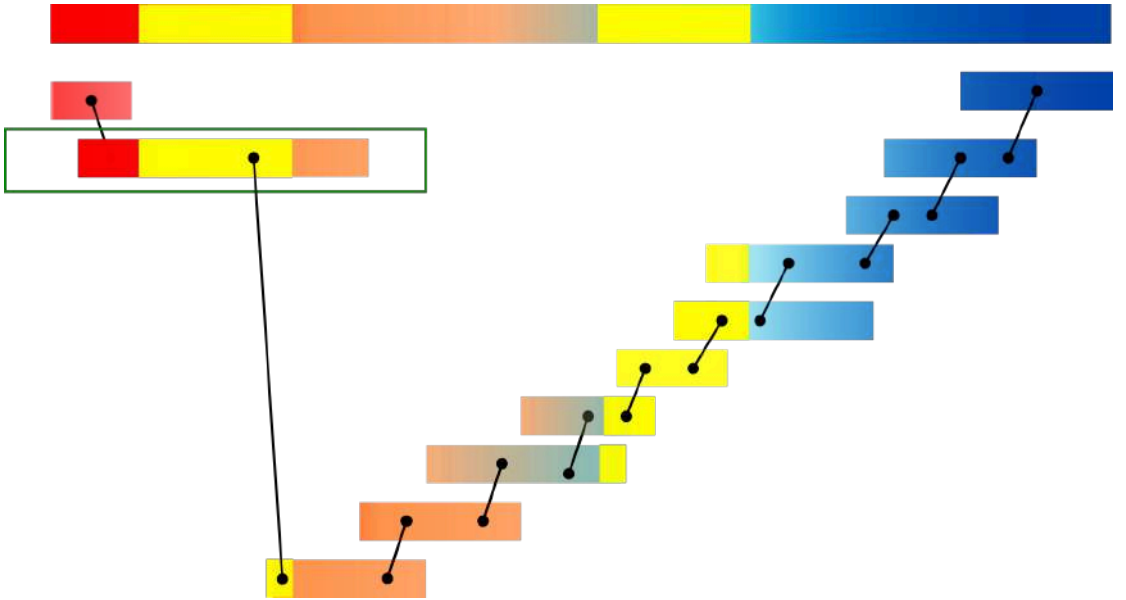
• The burden of assembly: genomic repeats



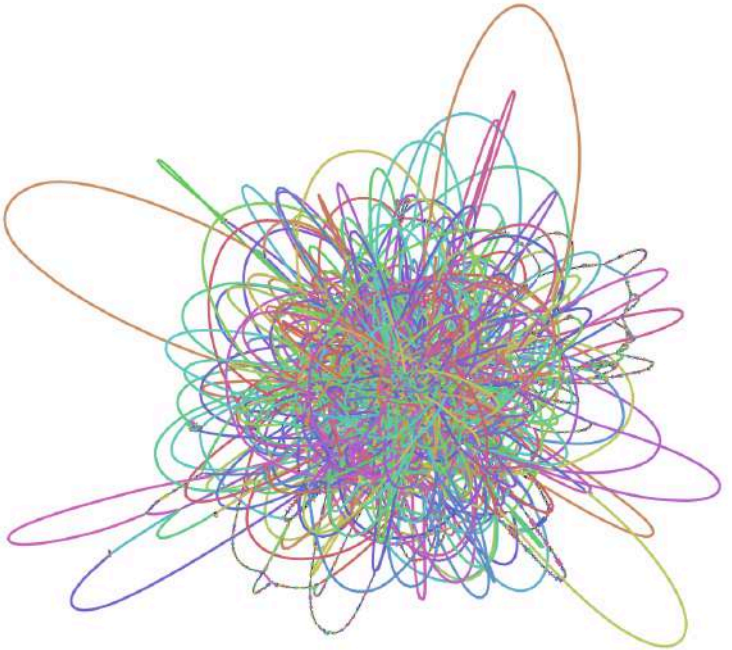
Example: pericentromeric region of human chr I :
20Mb of satellite repeats, identical repeats up to 10kb

• With longer reads

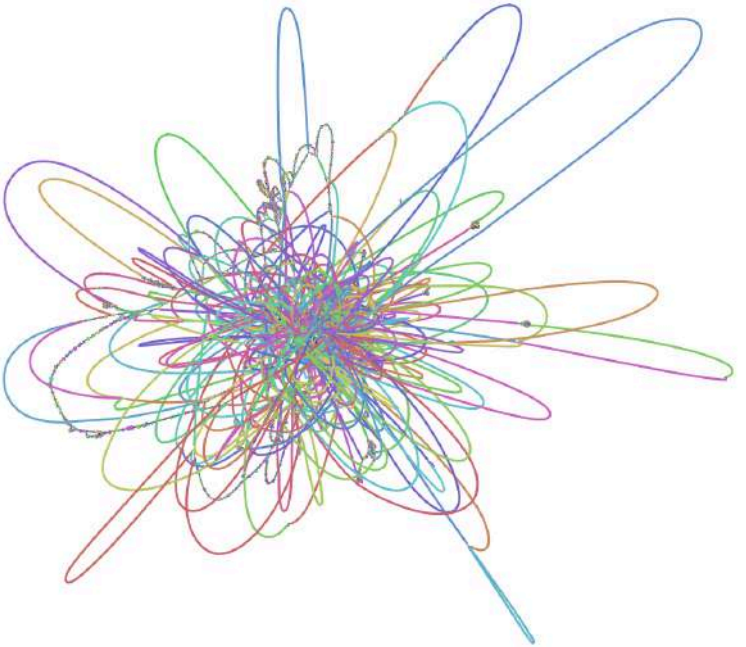
Reads longer than the repeat “solve” it



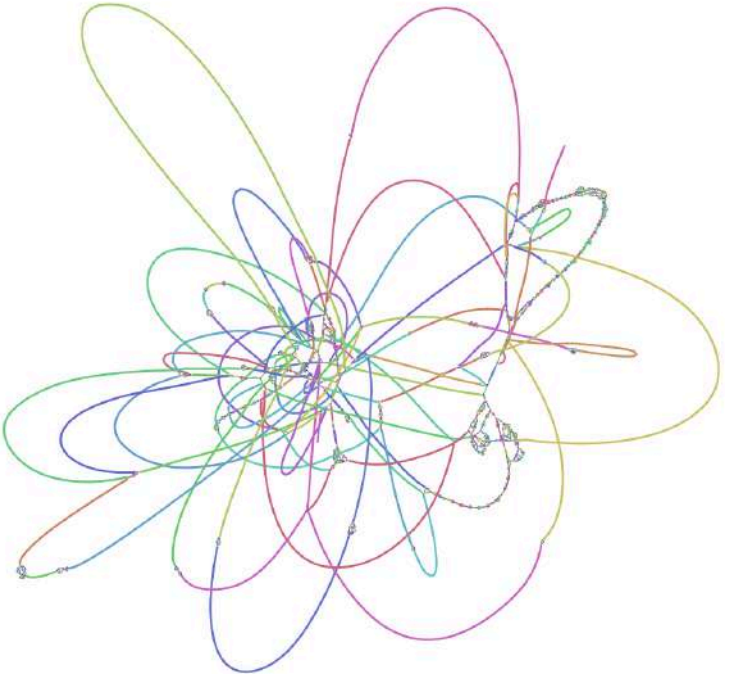
- **Read length matters (read size=21)**



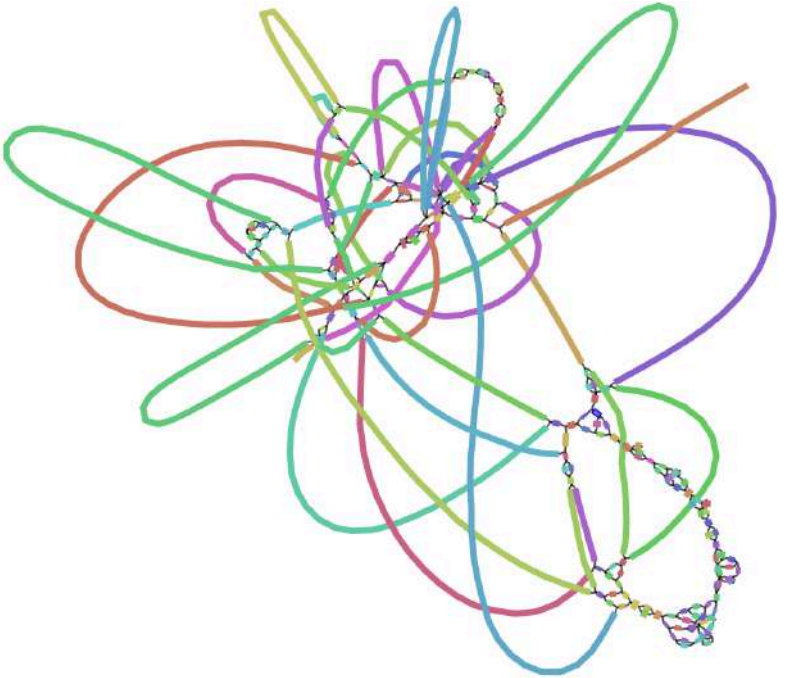
- **Read length matters (read size=31)**



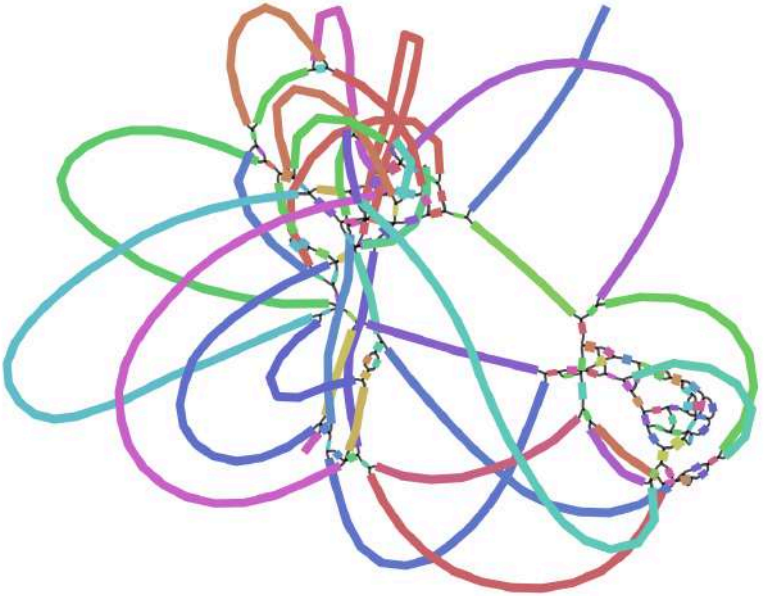
- **Read length matters (read size=63)**



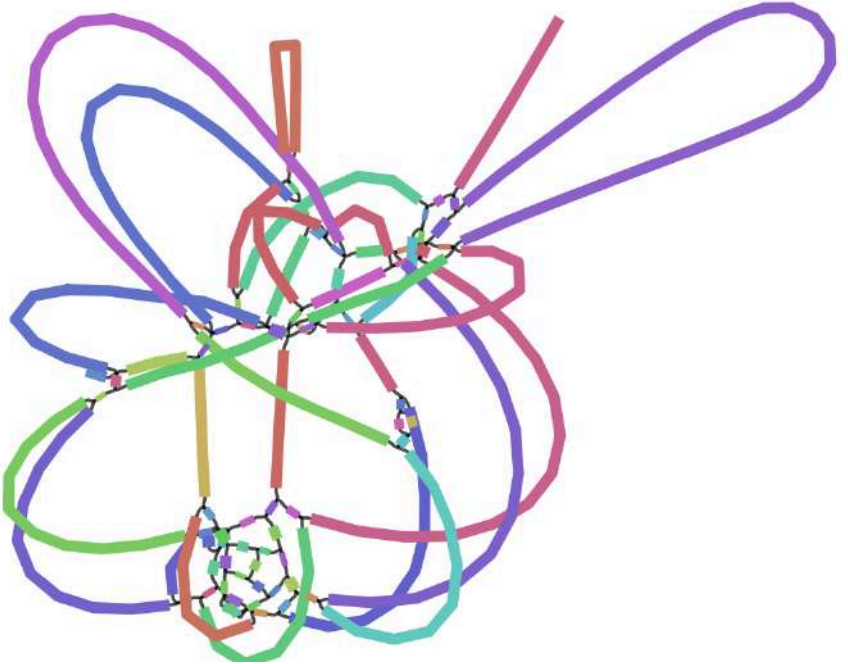
- **Read length matters (read size=255)**



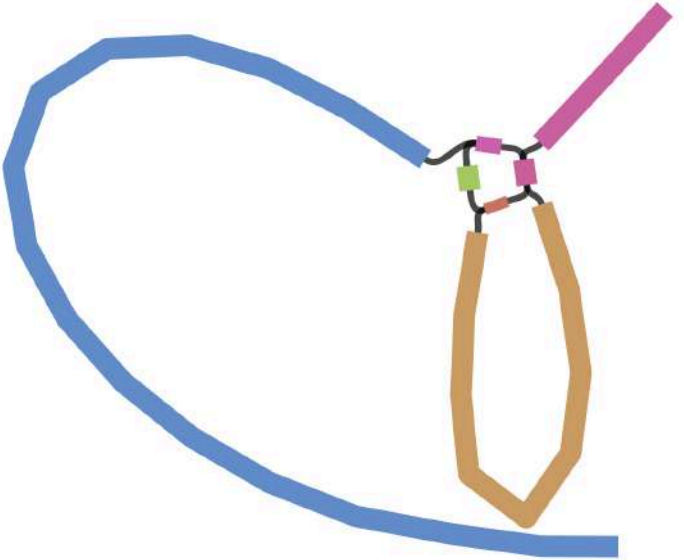
- **Read length matters (read size=500)**



- **Read length matters (read size=1000)**



- **Read length matters (read size=2000)**



First (and most important) checkpoint

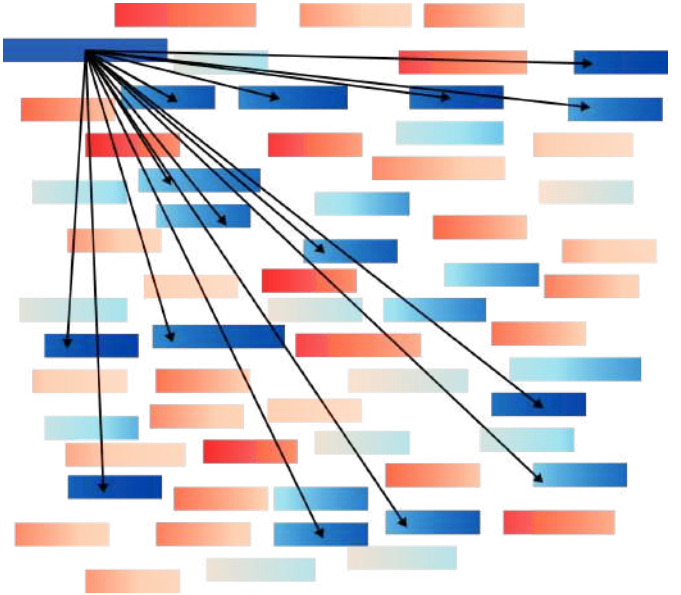
To remember

- Assembly orders reads using **overlaps**
- Longer overlaps are **generally better**
- Multiple possible overlaps necessitate **graphs** for structuring information
- **Repeats** longer than reads result in fragmented assembly (contigs).

- **Ok but...**

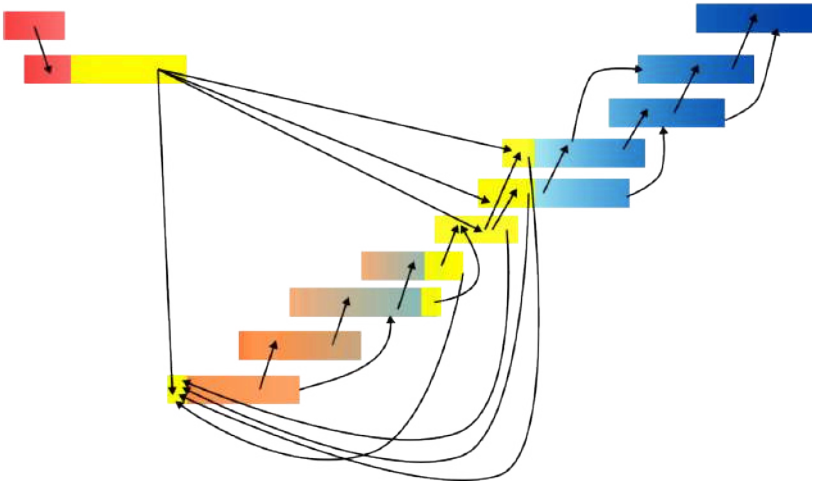
• Computing overlaps

Detecting overlaps means a lot of comparisons



• Computing long overlaps

Even considering only long overlaps means a lot of comparisons



• Overlap graph burden: number of reads

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

We have to be efficient and focus on **relevant** overlaps

• Overlap graph burden: number of overlaps

For each base of the genome:

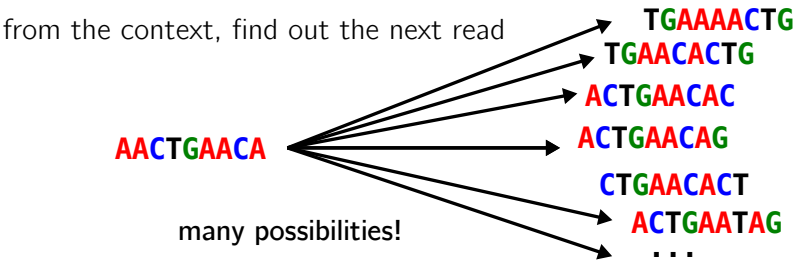
Read depth	Overlaps depth
10	100
20	400
50	2,500
100	10,000

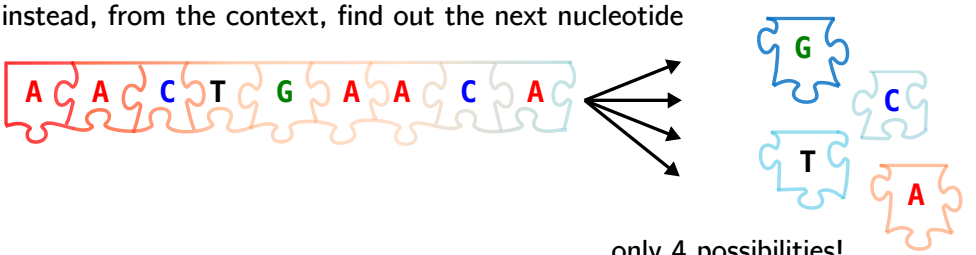
The amount of overlaps is not linear

Linear: 2X data 2X time

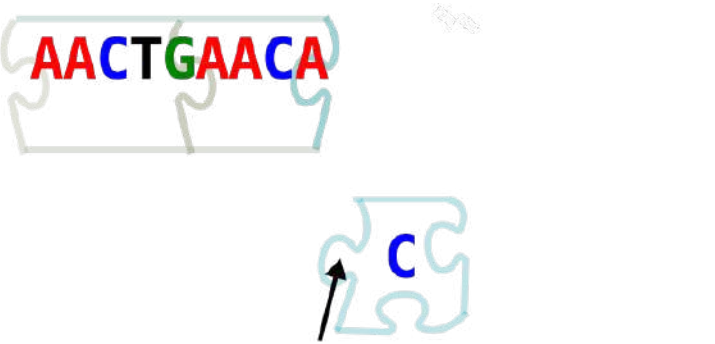
Quadratic: 2X data 4X time

• Another idea for genome assembly

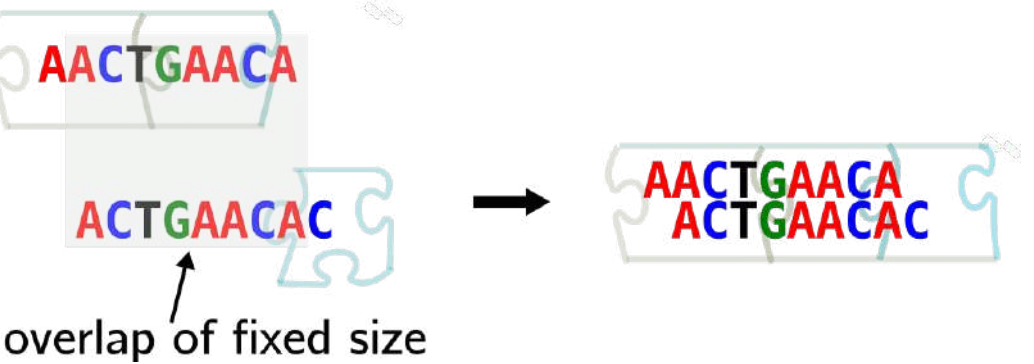




- **Context**



- **Context**


AACTGAACA
ACTGAACAC
overlap of fixed size
AACTGAACA
ACTGAACAC

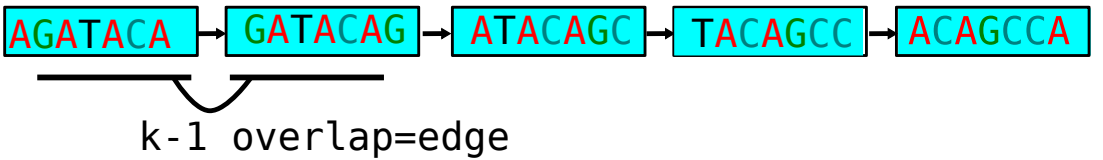
The de Bruijn graph

Read

AGATACAGCCA

De Bruijn graph

Kmer=node



AGATACA + G + C + C + A

=AGATACAGCCA

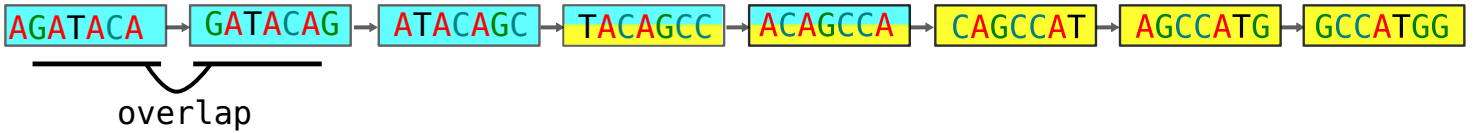
de Bruijn graph assembly

Overlapping reads

AGATACAGCCA

TACAGCCATGG

De Bruijn graph



Resulting sequence

AGATACAGCCATGG

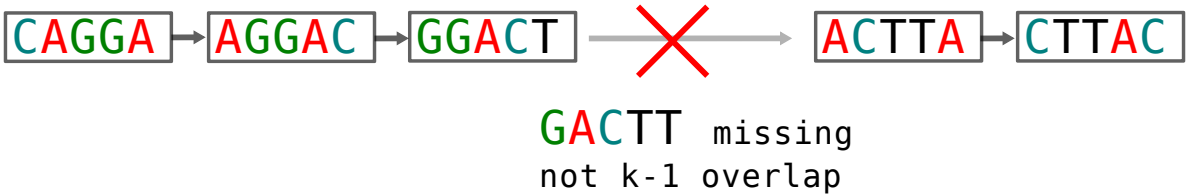
- **Why bother with k-mers?**

I liked reads.

• Why bother with k-mers?

In my graph, k-mer size = read size

• de Bruijn graphs limitation 1: Fixed overlaps



GGACT and ACTTA overlap is only of size 3 !

We need a smaller k value than the read size to detect overlaps

- **Exercise 1: de Bruijn graph time!**

Reads

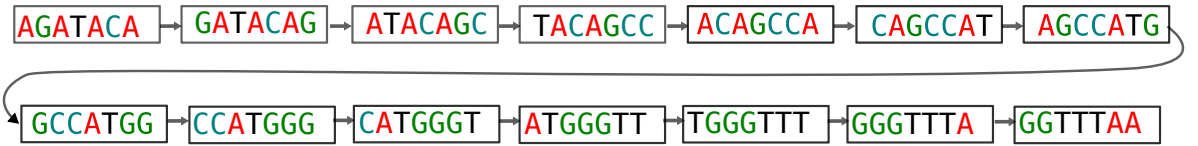
GCCATGGGTTT
TACAGCCATGG
AGCCATGGGTT
GCCATGGGTTT
AGCCATGGGTT
ACAGCCATGGG
GATACAGCCAT
ATACAGCCATG
CATGGGTTTAA
CAGCCATGGGT
GATACAGCCAT

Hint: Use 7-mers

• Exercise 1: Solution

```
GATACAGCCAT
 ATACAGCCATG
  TACAGCCATGG
   ACAGCCATGGG
   ACAGCCATGGG
    CAGCCATGGGT
     AGCCATGGGTT
      GCCATGGGTTT
      GCCATGGGTTT
       CCATGGGTTTA
        CATGGGTTTAA
```

de Bruijn graph



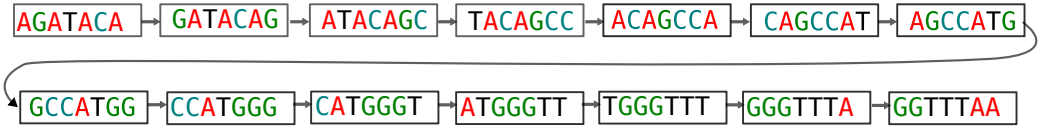
• de Bruijn graphs abstract redundancy

read overlaps

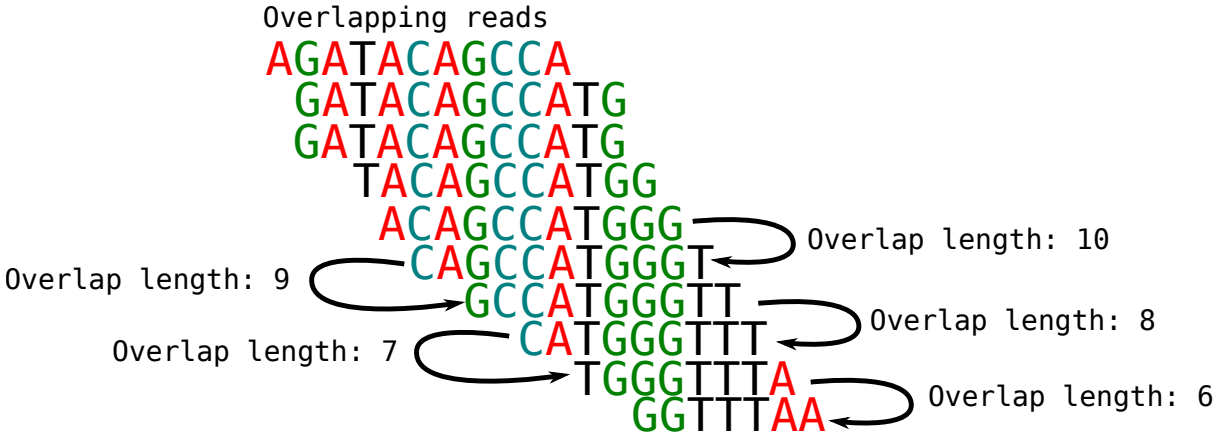
```
AGATACAGCCA
 GATACAGCCAT
 GATACAGCCAT
  ATACAGCCATG
   TACAGCCATGG
    ACAGCCATGGG
    ACAGCCATGGG
     CAGCCATGGGT
      AGCCATGGGTT
       GCCATGGGTTT
       GCCATGGGTTT
        CCATGGGTTTA
         CATGGGTTTAA
```

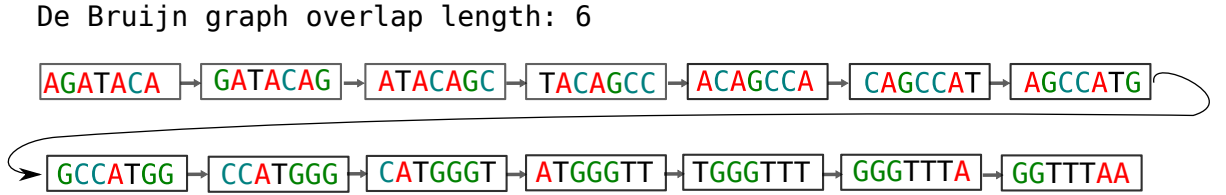
65 non distinct 7-mers in reads

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



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

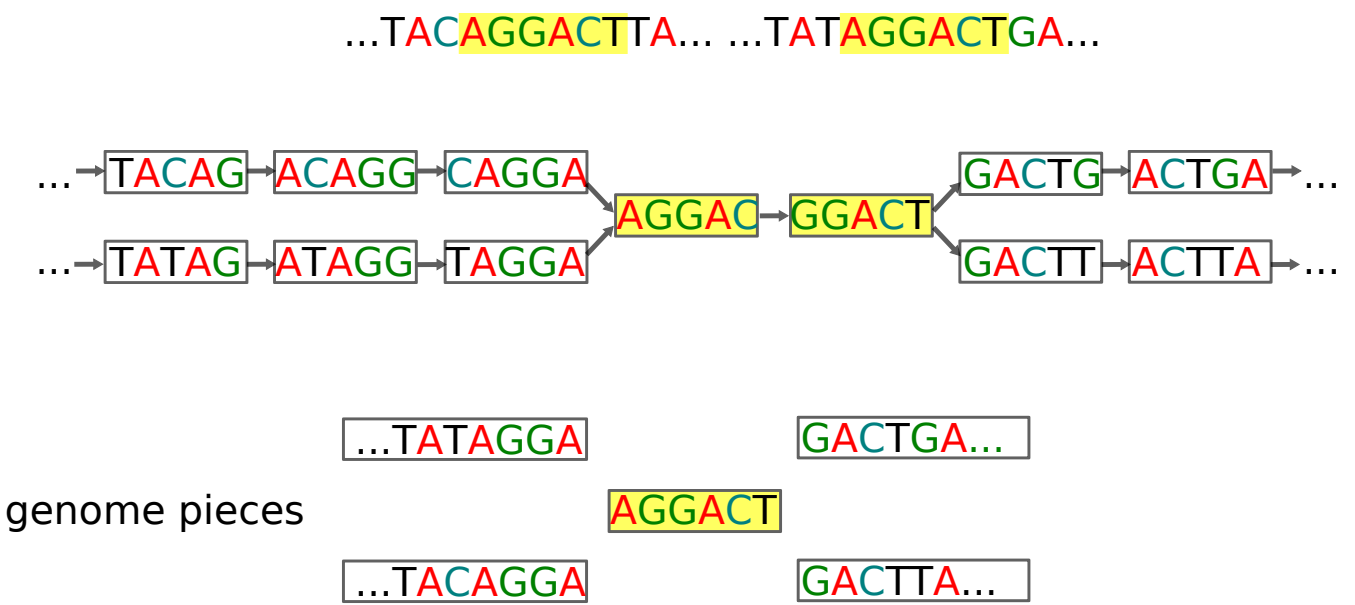




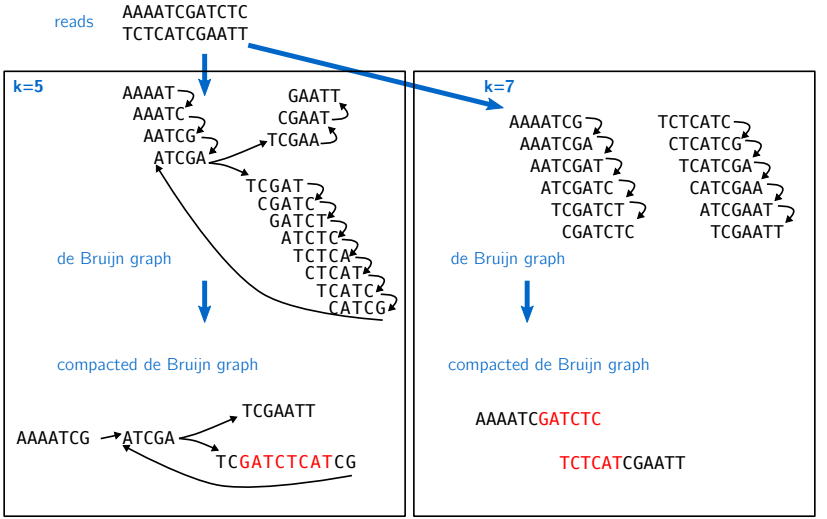
• Repeats in a de Bruijn graphs

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

• de Bruijn graphs limitation 2: Repeats



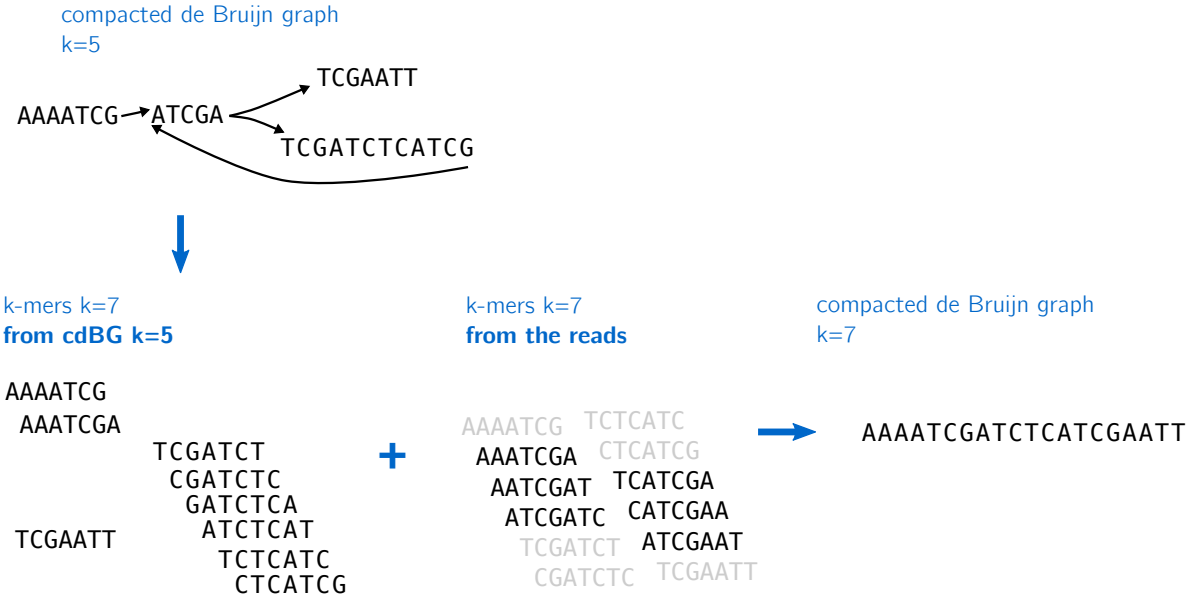
• Multiple k assembly



We are missing GATCTCA and ATCTCAT in the second graph.

But they are present in the first graph!

• Multiple k assembly



Multiple k assembly is done under the hood in modern DBG assemblers

• Method checkpoint: de Bruijn graph versus overlap graph

Overlap graph

Quadratic growth with coverage

Issue with "repeats"[1] **longer than the reads**

De Bruijn graph

Abstracts coverage

Issues with "repeats"[1] longer than k

Use multiple k values in practice

[1] genomic repeats like satellites, long homozygous regions

- **Something is wrong with my assembly**

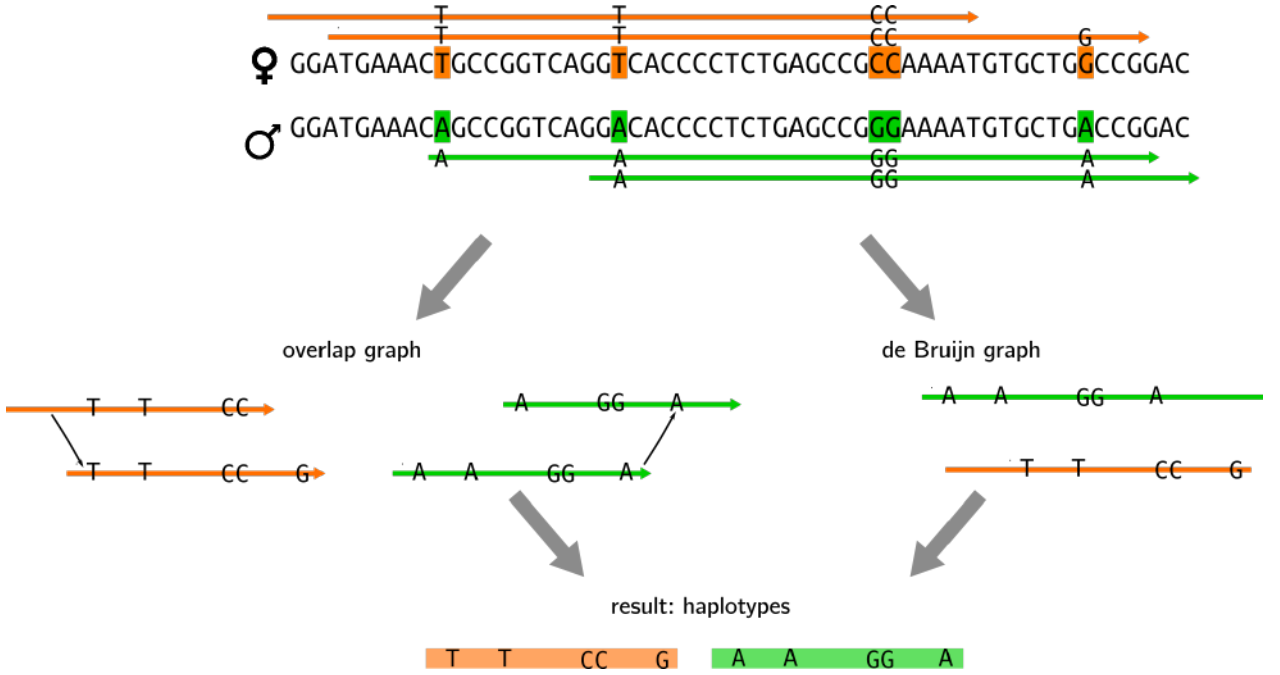
I cannot recover the haplotypes!

• The good boy is diploid!

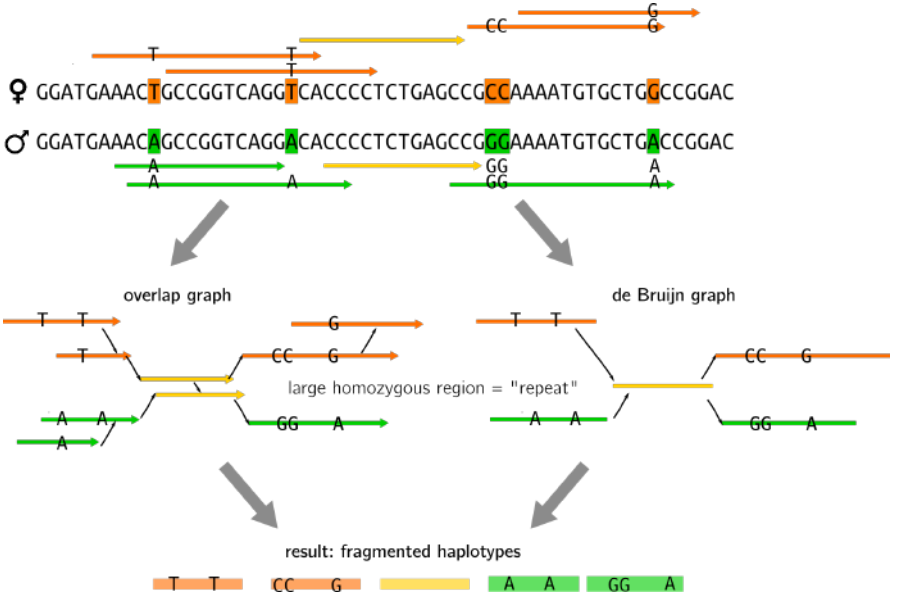
♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

• Ploidy and very long reads

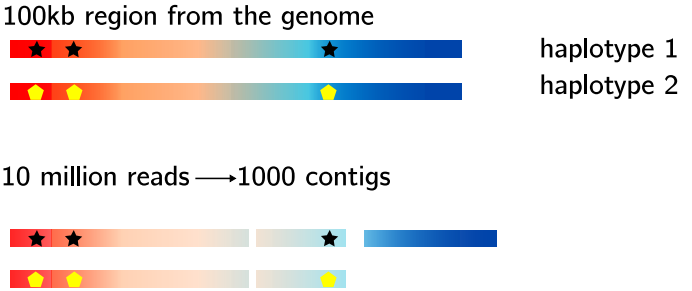


• Homozygous vs heterozygous regions



Assembly concession: assembly can be fragmented due to ploidy

• Data checkpoint: results with the long, errorless reads for the *very good boy*



Long and accurate reads

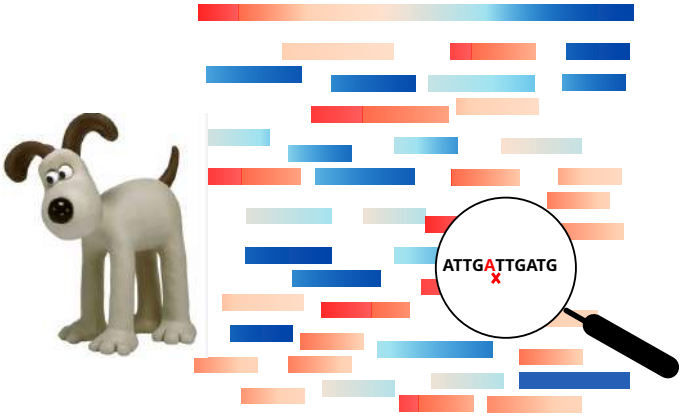
Contigs can reach the chromosome's order of magnitude in length (megabases)

Breaks due to large repeats

Haplotypes can be partially reconstructed

• Second experiment: noisy long reads for a *clever boy*

100kb region from the genome
(only for the record, we actually don't have it)



Genome size
2 billion bases

Reads
1 million
mean size 100kb
sequencing errors: 1%

Oxford Nanopore (simplex)
or PacBio hifi reads

- **de Bruijn graph or overlap graph?**

Image credit: Aardman/ BBC/ Netflix

- **de Bruijn graph or overlap graph?**

Image credit: Aardman/ BBC/ Netflix

Both after reads correction

• Overlap graph: inexact matches

GATTACA

compute overlap?

GCATGCG

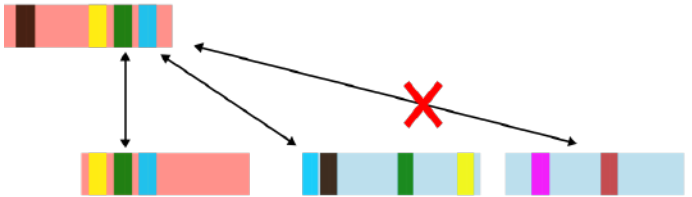
match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

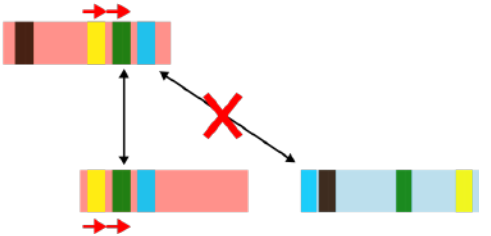
Pairwise read alignment requires a **quadratic** number of **costly** alignments!

• Overlap graph: drop alignment

1. find common seeds

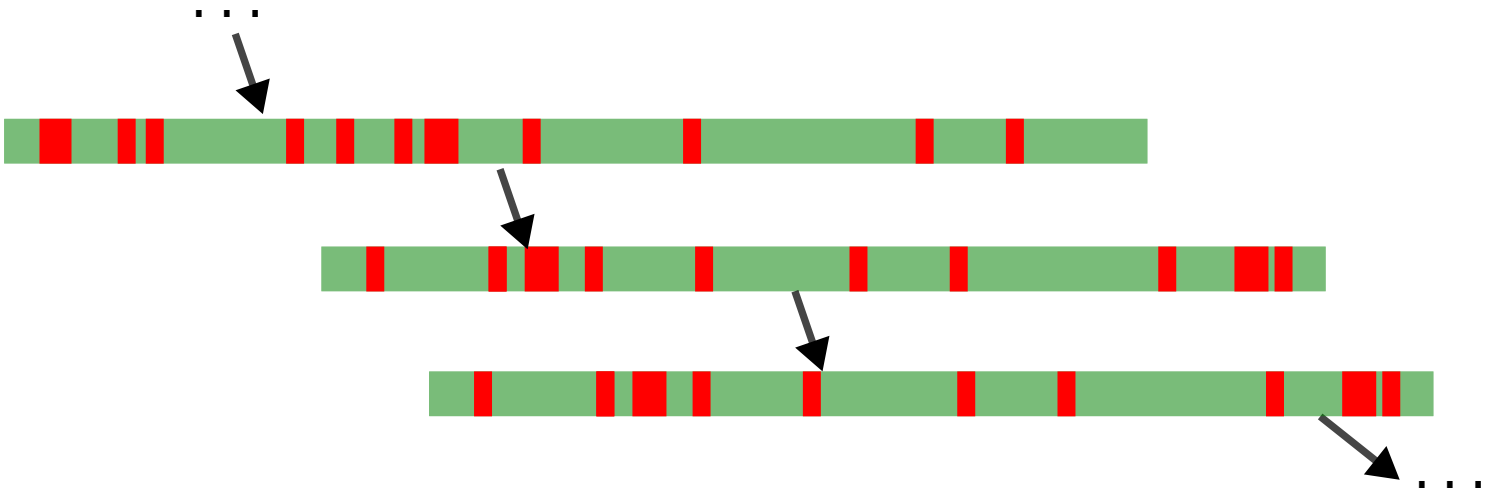


2. find if long chains of common seeds are in same order



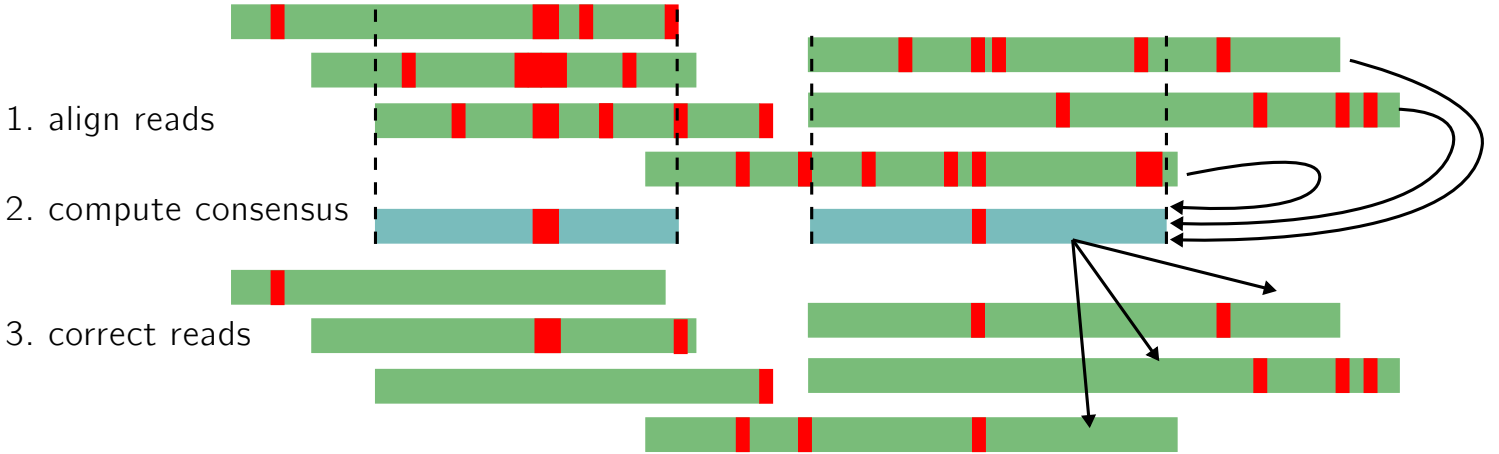
Procedure called *seed and chain* or anchor chaining.

- **How to get accurate contigs from noisy reads?**

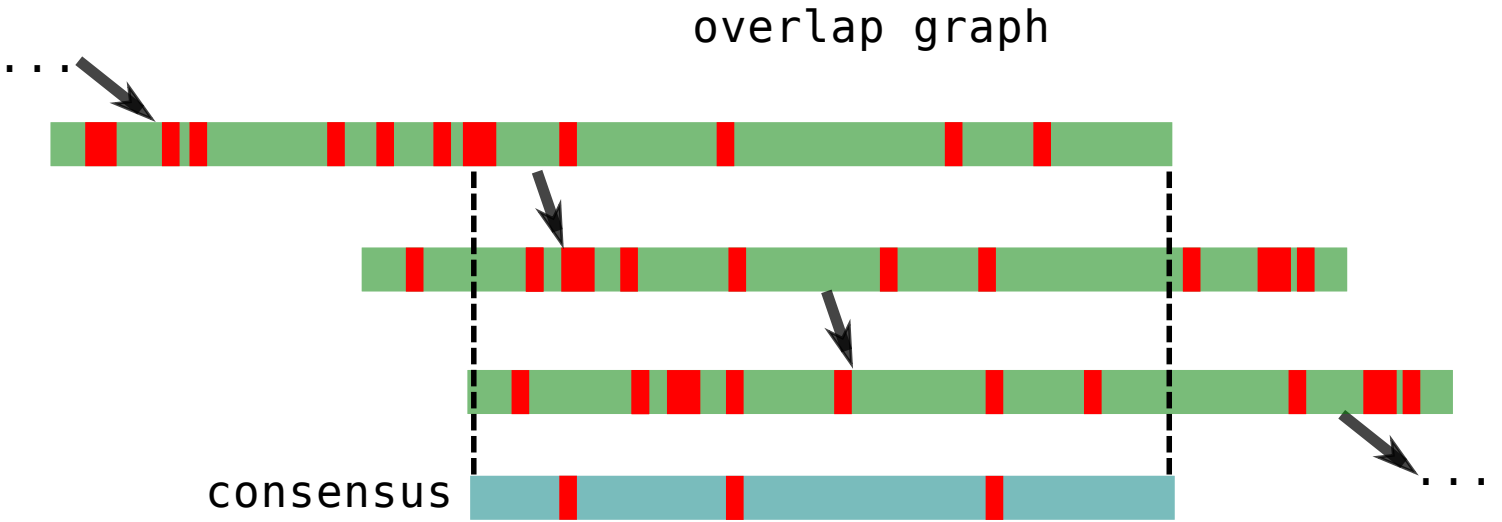


• Using coverage to remove noise: consensus

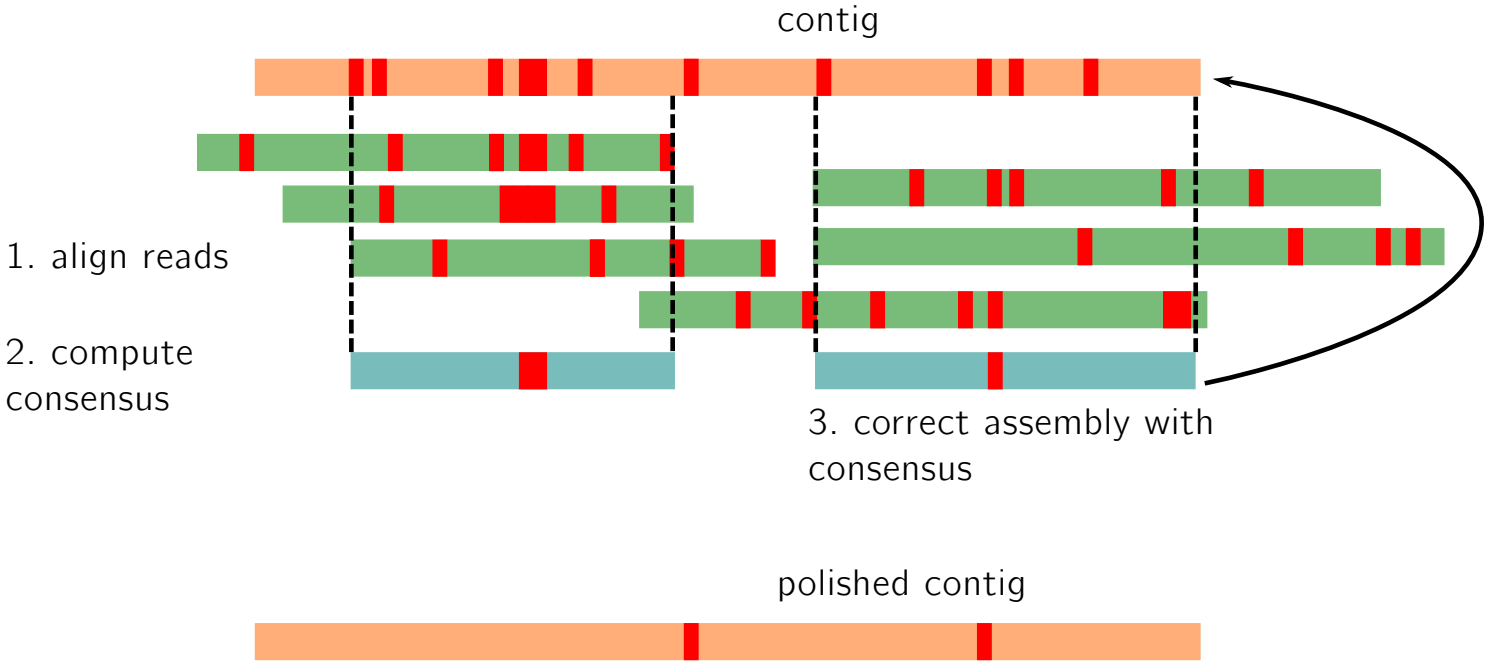
• Consensus before assembly: correction



- **Consensus during assembly (hence the OLC)**



• Consensus after assembly, polishing

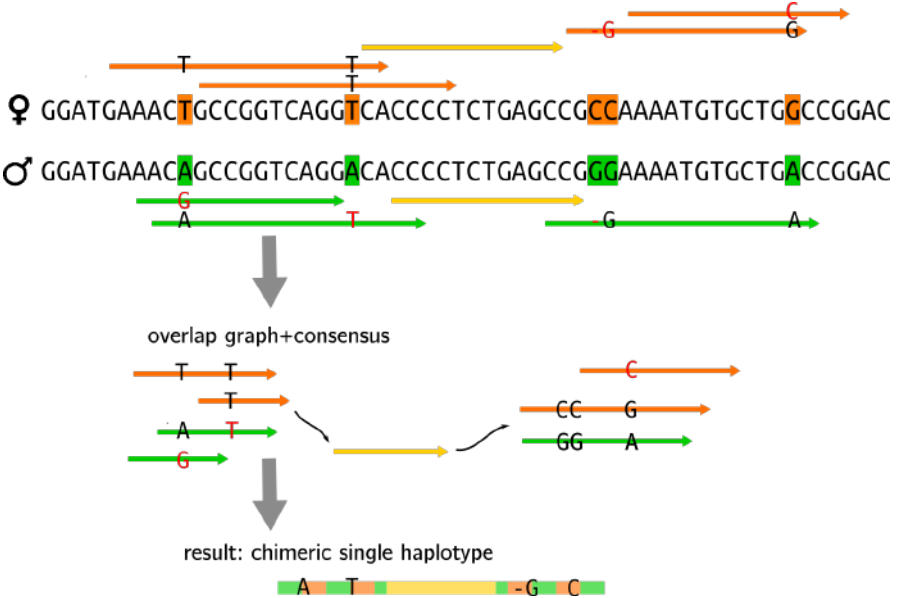


- **Beware: consensus destroys heterozygosity**

```
                AATTGATCCGATACCC-GTAA-A
                AATTGGCCGATACCC-GTAA-AG
                -ATTGATCCGA-ACCCCGTAA-A
                AATTGATCCGATACCC-GTAA-A
reads               GCTCCGAGACCA-GTCA-ATTG
                    GCTCC-AGACCA-GTCA-ATTT
                       CCGAGACCA-GTCG-ATTGCAAA-
                       CCGAGACCA-GT-A-ATTGCGAAC
                       CCGACACCA-GTGAATTGCAAAC
                ---------------------------------
consensus       AATTGATCCGAGACCA-GTCA-ATTGCAAAC
```

→ a mix between the two alleles

• Consensus destroys heterozygosity



Assembly concession: “haploid” assembly due to errors

• Some assemblers handle haplotype-aware consensus

two different
parts of the graph

```
reads      AATTGATCCGATACCC-GTAA-A
           AATTGGGCCGATACCC-GTAA-AG
           -ATTGATCCGA-ACCCCGTAA-A
           AATTGATCCGATACCC-GTAA-A
consensus  AATTGATCCGATACCC-GTAA-A
```

```
           GCTCCGAGACCA-GTCA-ATTG
reads          GCTCC-AGACCA-GTCA-ATTT
                   CCGAGACCA-GTCG-ATTGCAAA-
                   CCGAGACCA-GT-A-ATTGCGAAC
                   CCGACACCA-GTGAATTGCAAAC
consensus      GCTCCGAGACCA-GTCA-ATTGCAAAC
```

Calling two different haplotypes is called *phasing*

High quality reads are needed

Method checkpoint: two strategies

Correct then assemble

- Time expensive
- Assembly step becomes easy
- Overlap graph or DBG (**Assemblers:** Canu, FALCON, MECAT, hifiasm, LJA...)
- **IF** correction is haplotype-aware → phased assembly (**Assemblers:** hiCanu, hifiasm)

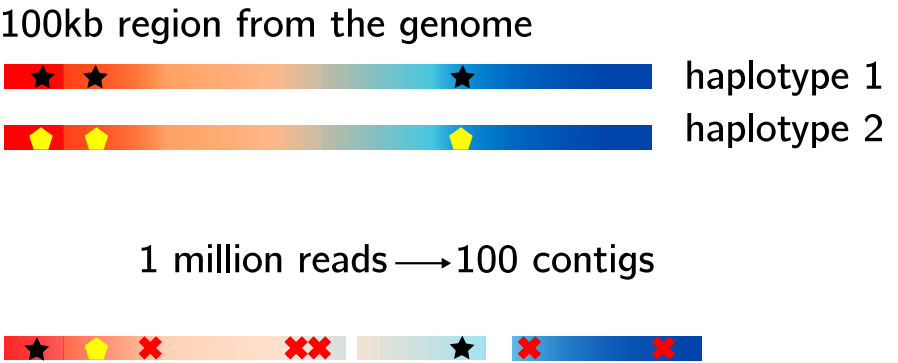
Assemble raw reads

- "Fast"
- Assembly step has to handle errors
- Overlap graphs
- Hard to separate haplotypes

(**Assemblers:** Flye, Raven, Shasta, SMARTdenovo ...)

Both can improve their final assemblies with polishing afterward!

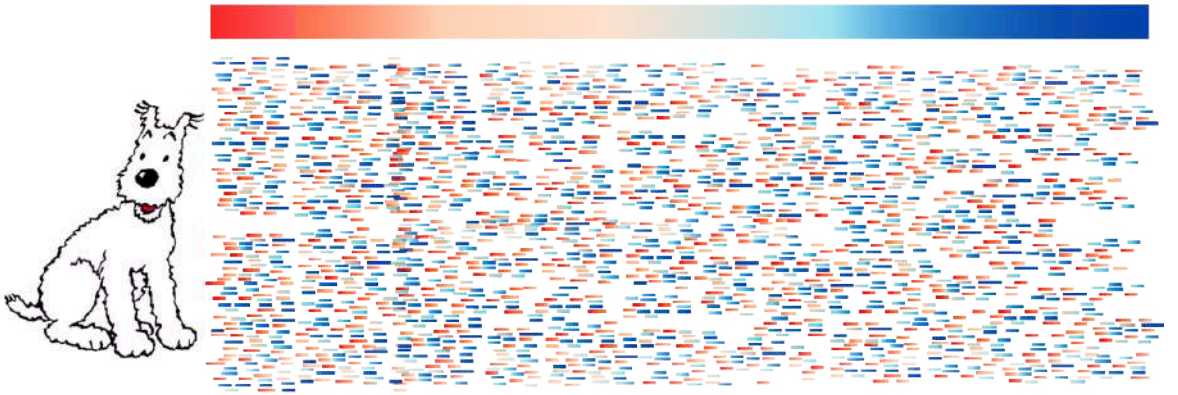
• Data checkpoint: results for ONT/PacBio long reads



- Contigs can reach the chromosome's order of magnitude in length (megabases)
- Breaks due to very large repeats
- Contigs are chimeras of haplotypes **unless** you have very good quality reads + a diploid assembler

- **Third experiment: Illumina short reads for a *short boy***

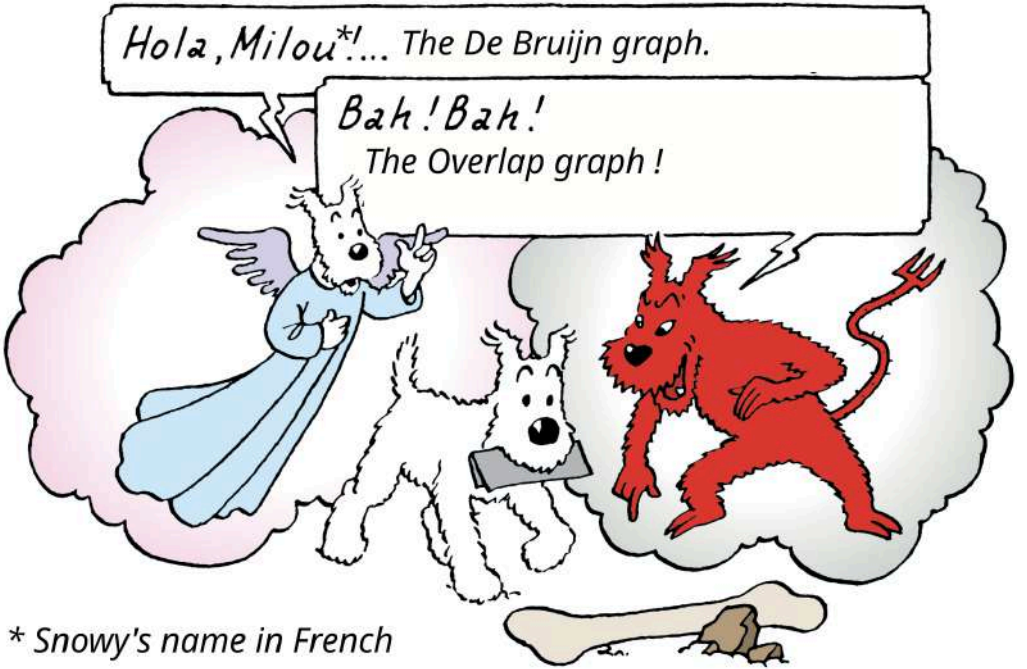
100kb region from the genome
(only for the record, we actually don't have it)



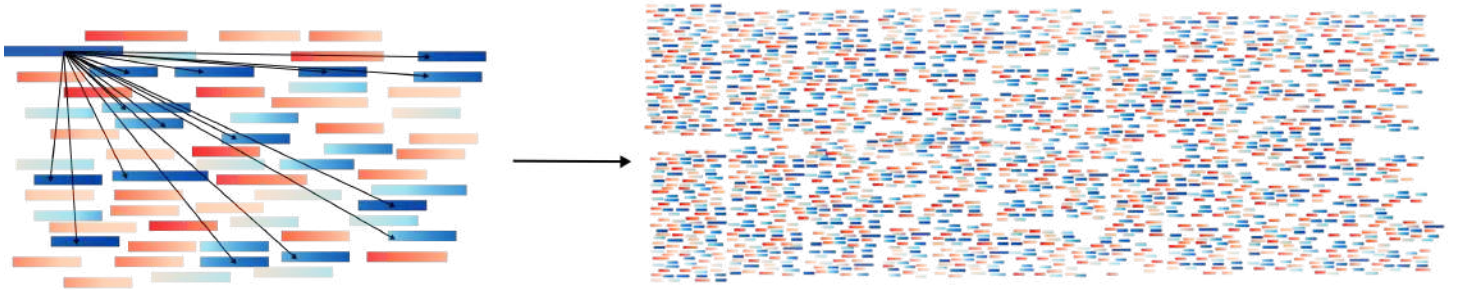
Genome size
2 billion bases

reads: Illumina
10 or 100 billions
size 100-300 bp
<1% error rate

- **de Bruijn graph or overlap graph?**



• Scalability issue for the overlap graph

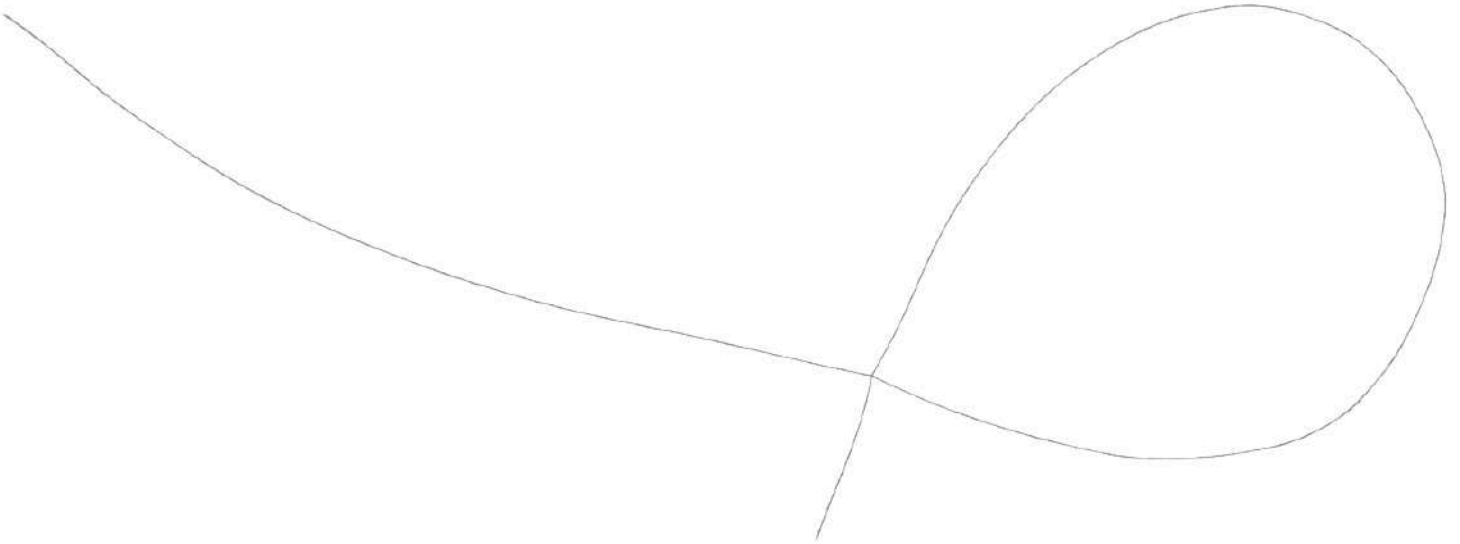


At equal coverage we got

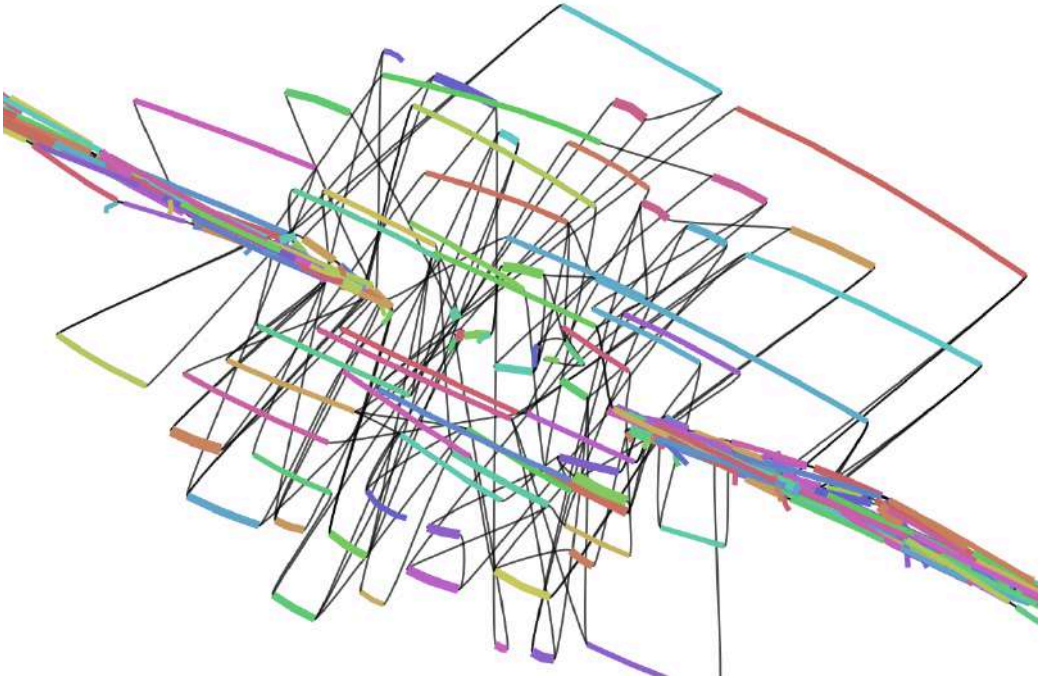
1000 x more reads $\rightarrow$ 1 million x more overlaps to check!

Overlap graph out!

- **de Bruijn graph on a real dataset**



- **de Bruijn graph on a real dataset ZOOMED IN**



• Erroneous k-mers vs genomic k-mers

Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACA

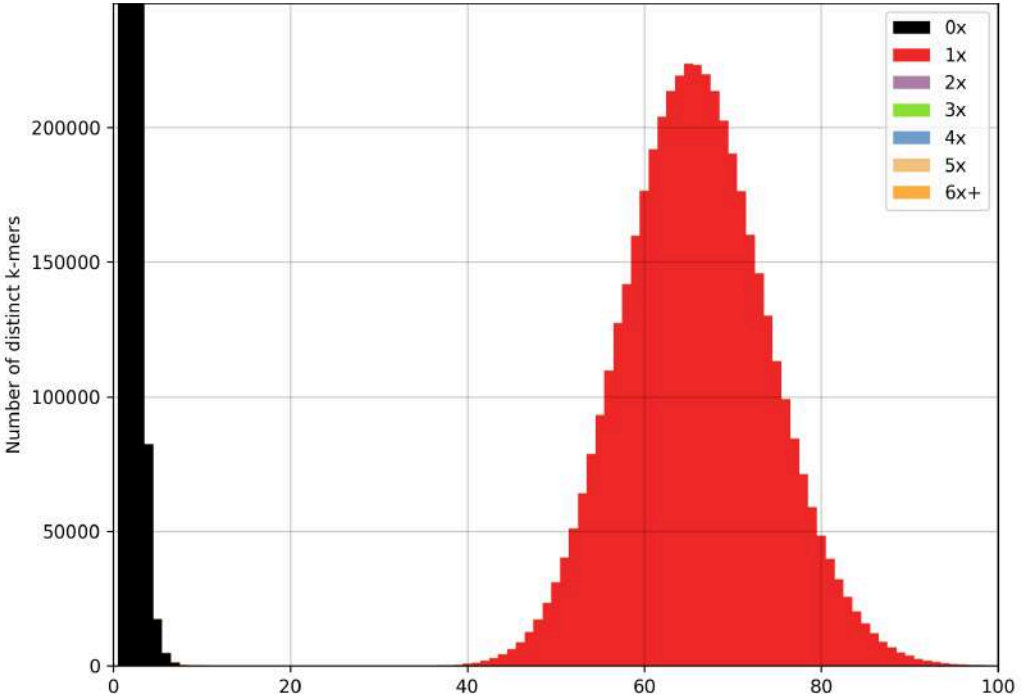
Reads:

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

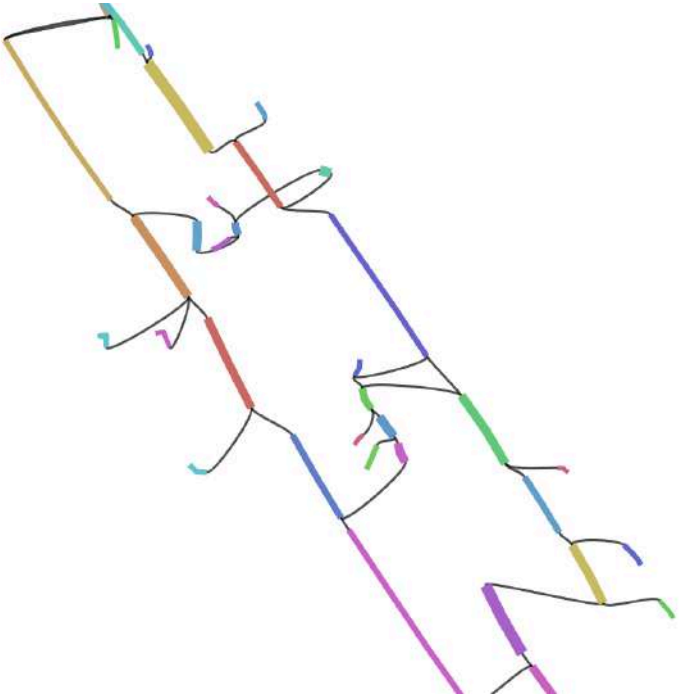
```
9 times  TCTGAAT
1 time   TCTAAAT

6 times          CAACGGA
1 time           CAACGGT
```

Erroneous k-mers are seen less than genomic ones

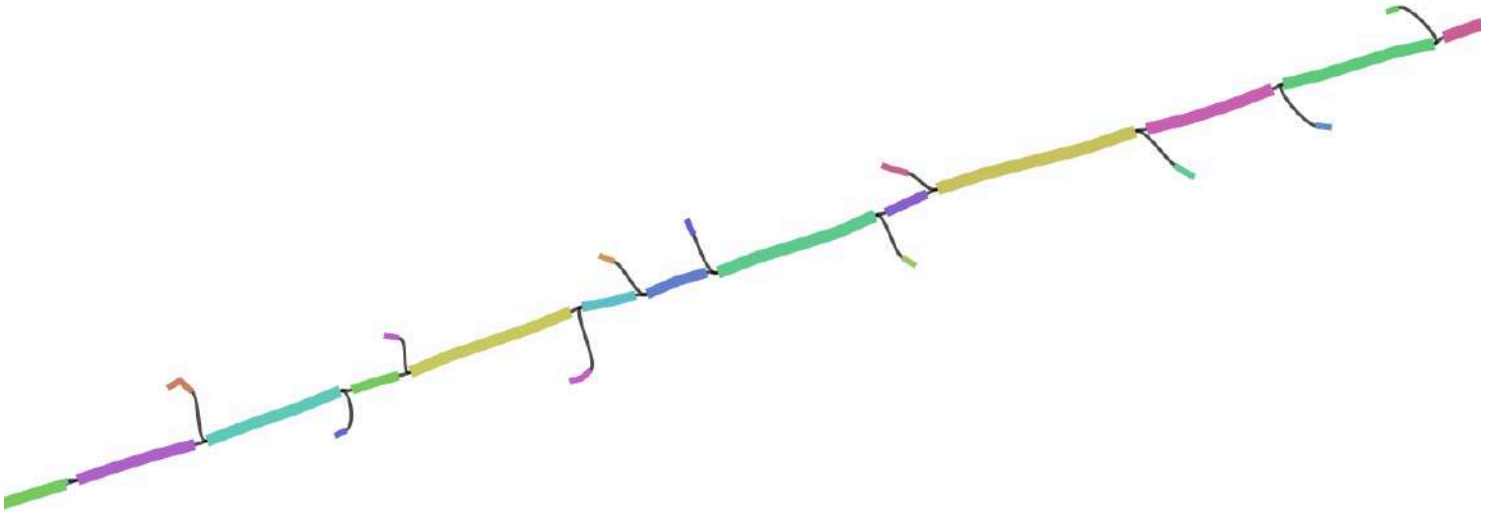
- K-mer histogram



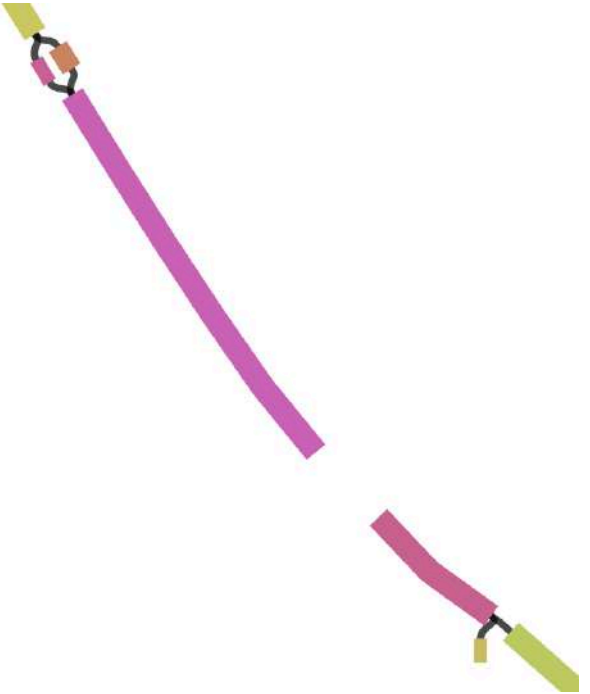
- **Removing unique k-mers**



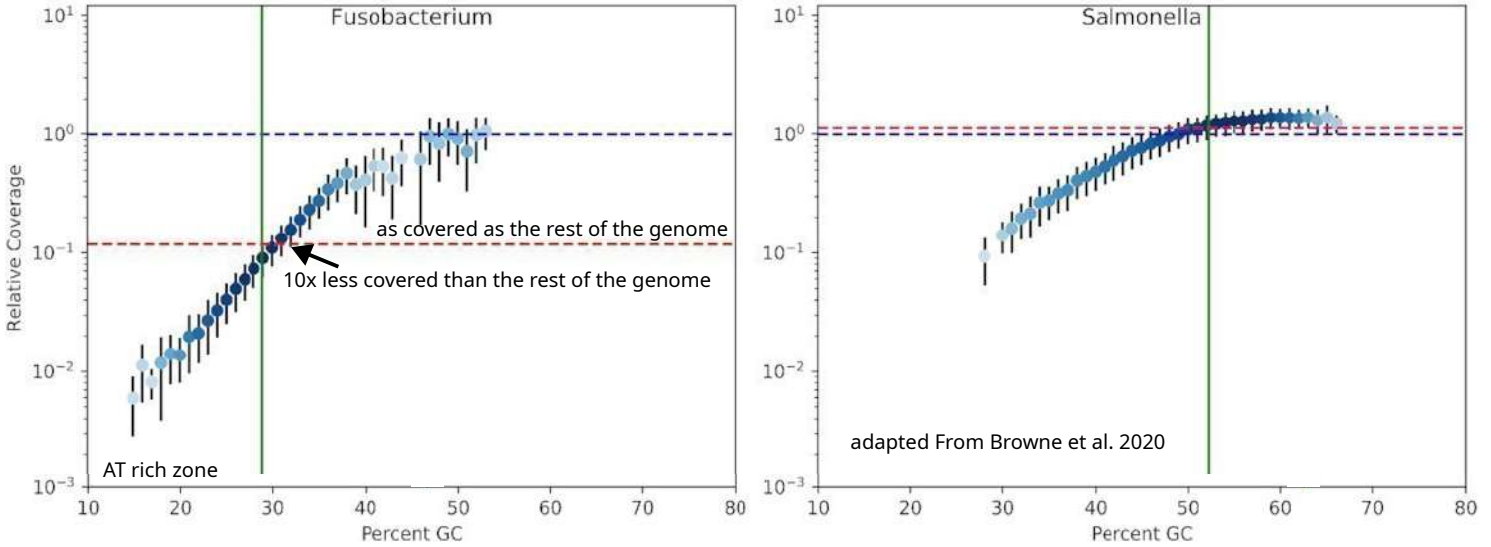
- **Removing k-mers seen less than 3 times**



- **Removing k-mers seen less than 4 times**

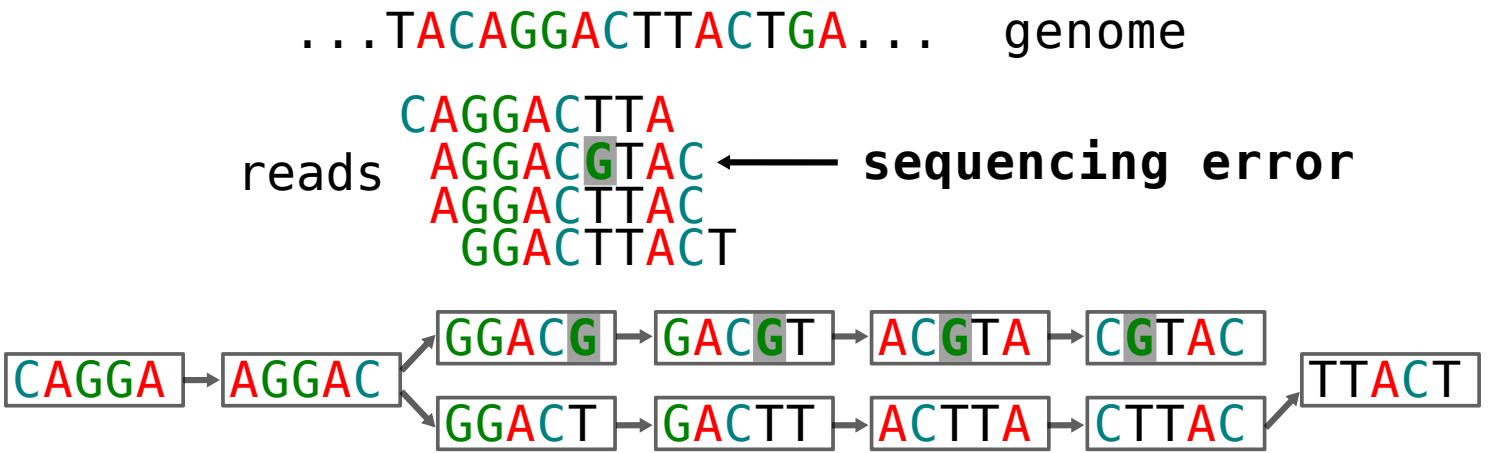


GC bias

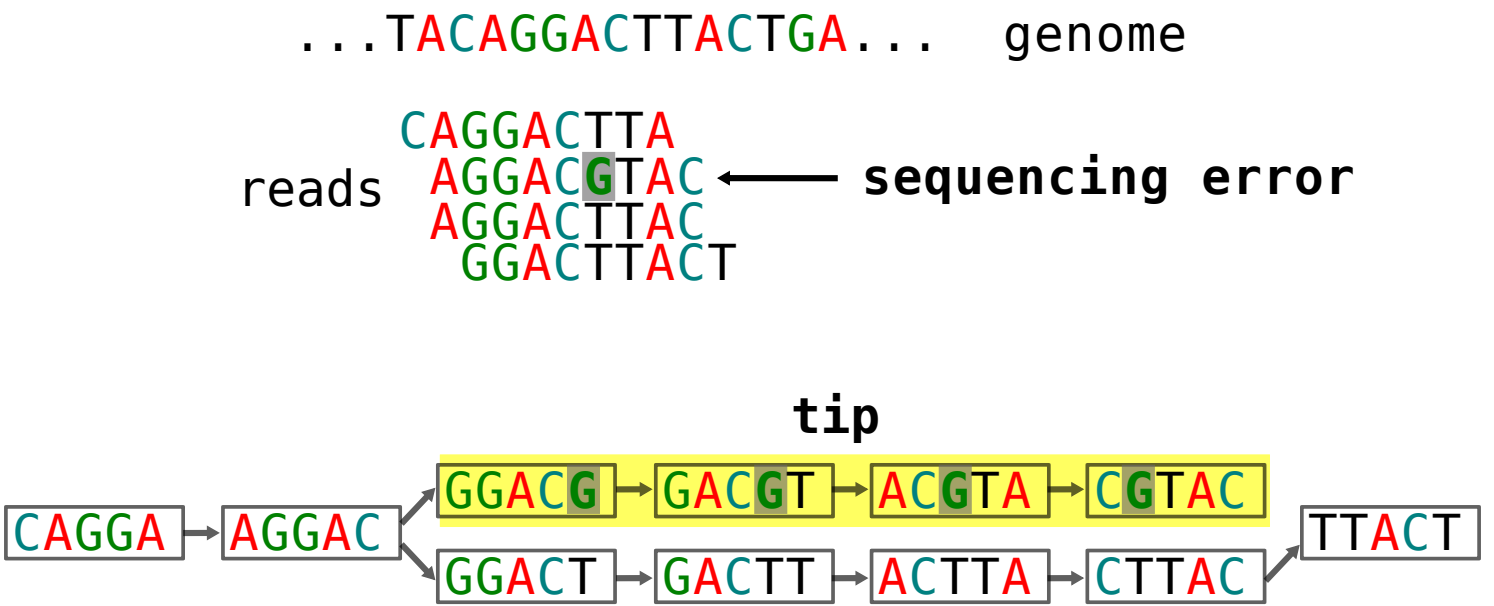


GC-low regions can be way less sequenced

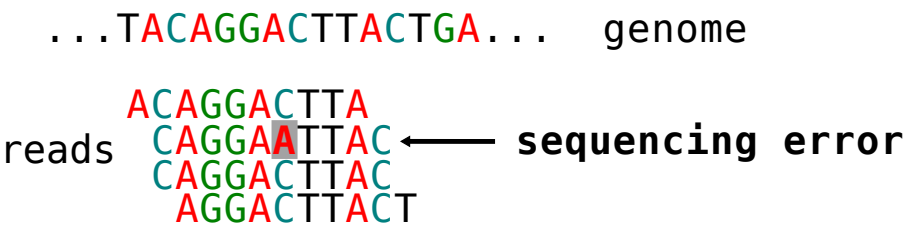
• Errors in de Bruijn graphs

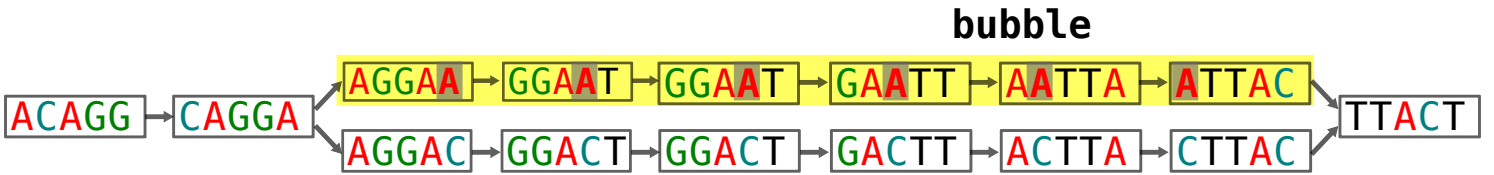


• Errors in de Bruijn graphs

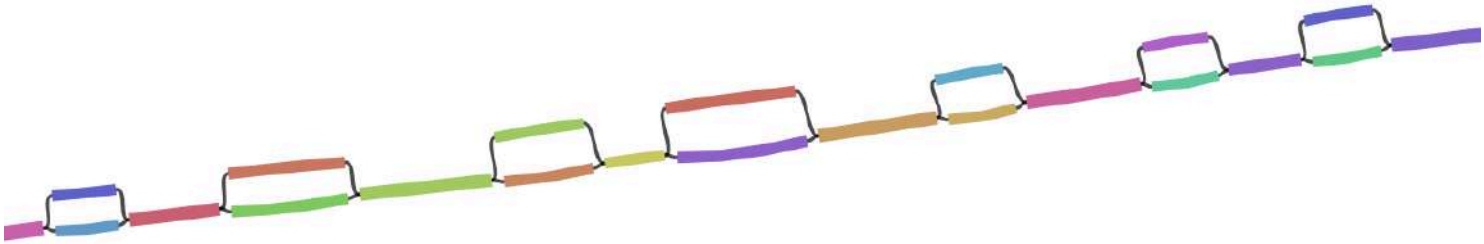


• Errors in de Bruijn graphs



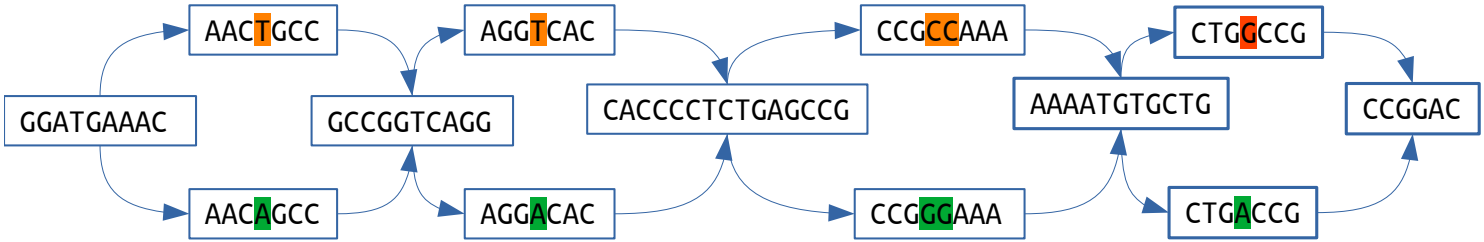


- **de Bruijn graph on my diploid genome**



• Ploidy and de Bruijn graph

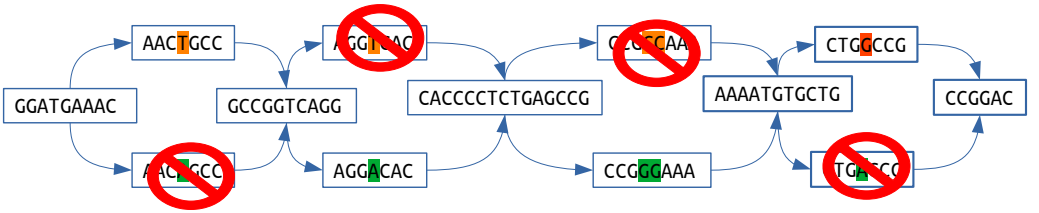
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC



• Bubble crushing

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

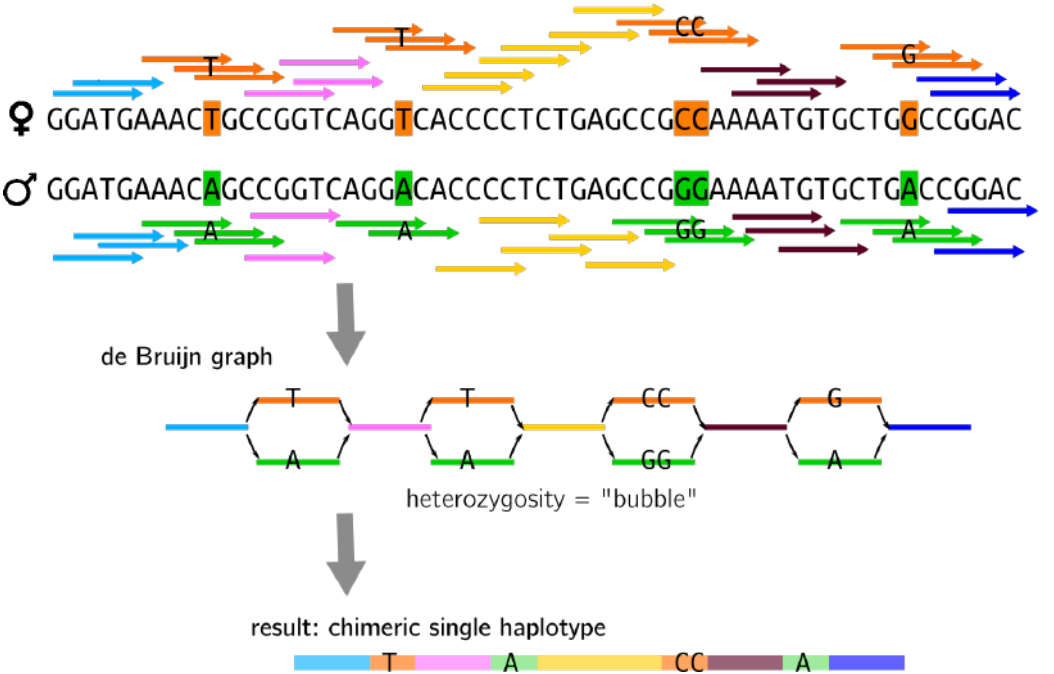


Assembly:

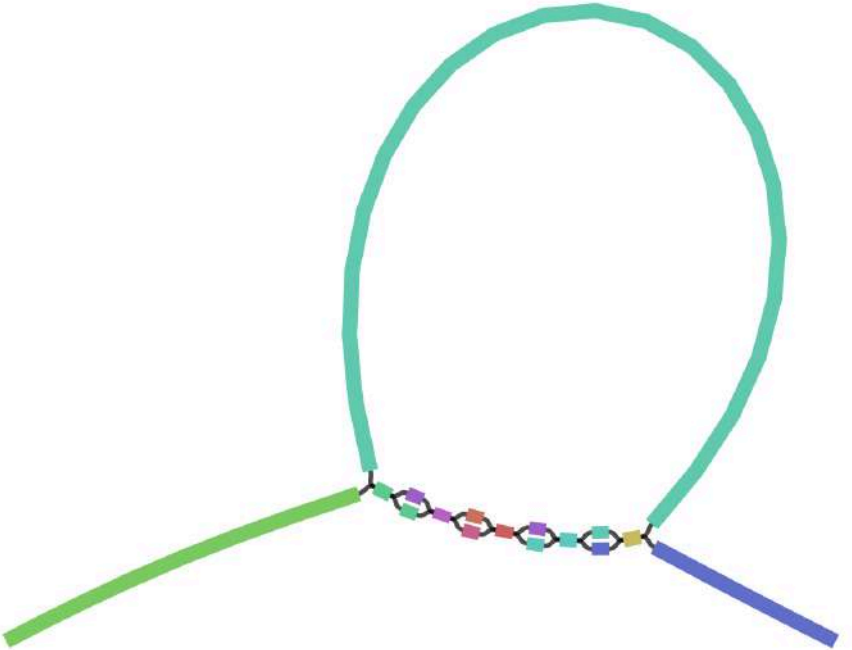
GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

Algorithms leverage graph topology to apply correction

• Haploid assembly

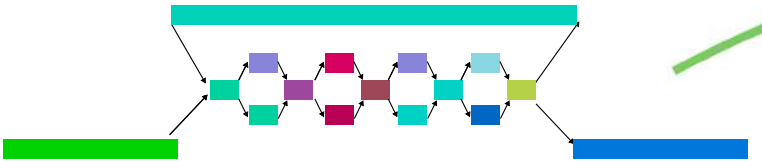


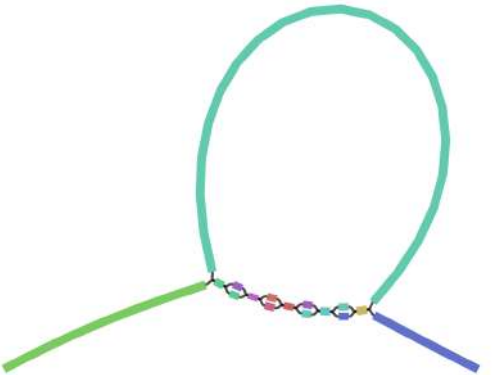
- **Paralog genes/repeats**



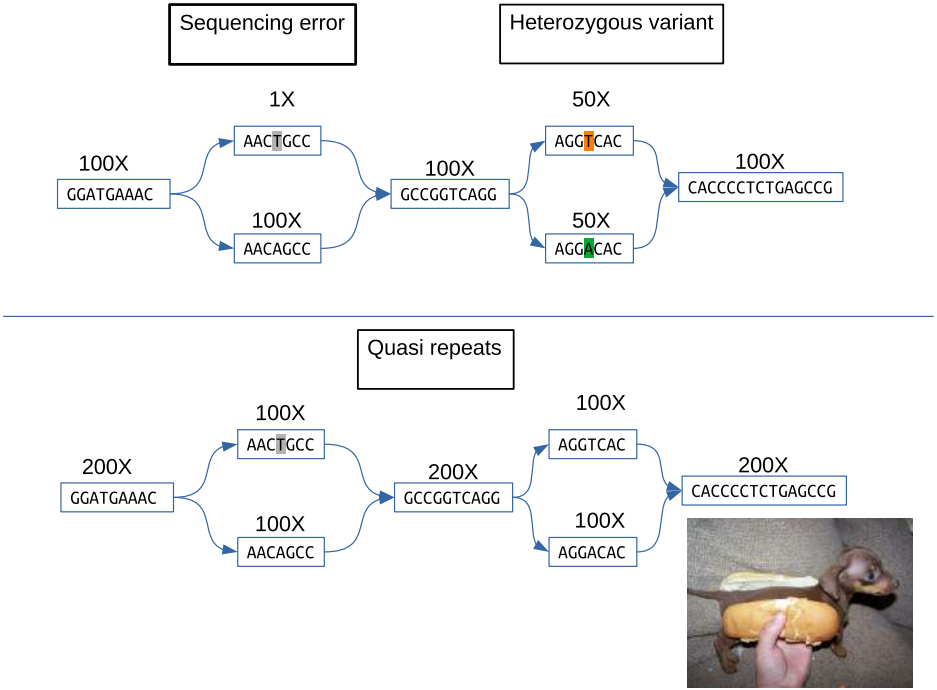
• Paralog genes/repeats

compacted de Bruijn graph





• An assembly is a sausage: don't ask how it's made



• Method checkpoint: de Bruijn graph versus overlap graph

Overlap graph

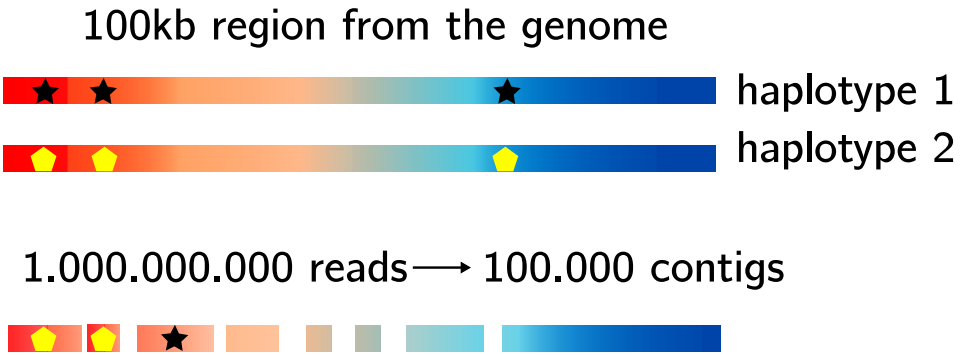
All v all comparisons are not scalable in this setting

De Bruijn graph

- *k*-mers abstract coverage
- Errors can be dealt with

Assemblers: SPAdes, ABySS, IDBA...

- **Data checkpoint:** *short boy* **results**

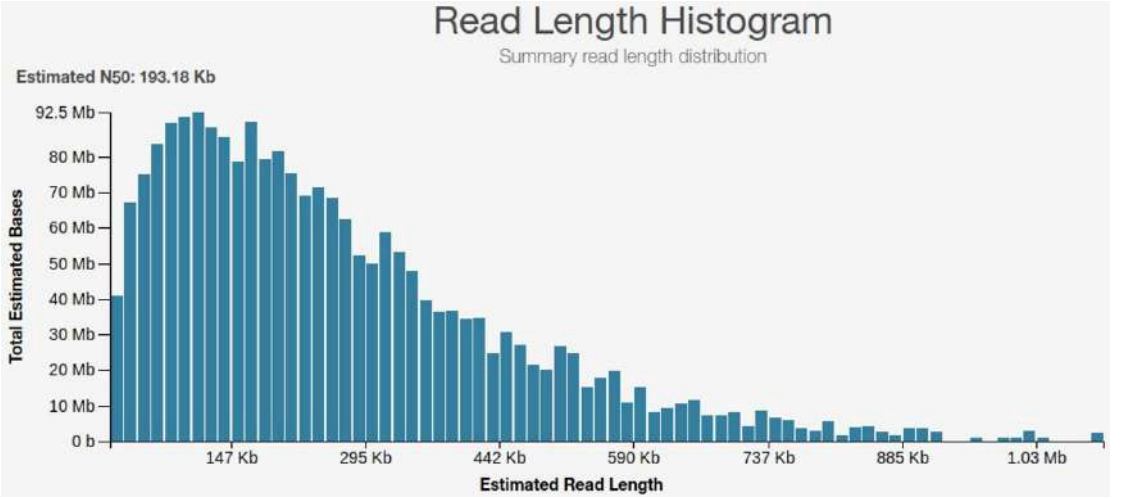


- Very **fragmented** assembly of short contigs (mostly below 100kb)
- Very high base **accuracy**
- Contigs are **chimeras** of haplotypes
- Can miss **low GC** content

- **Fourth experiment:** ***golden boy*****'s genome with long range information**

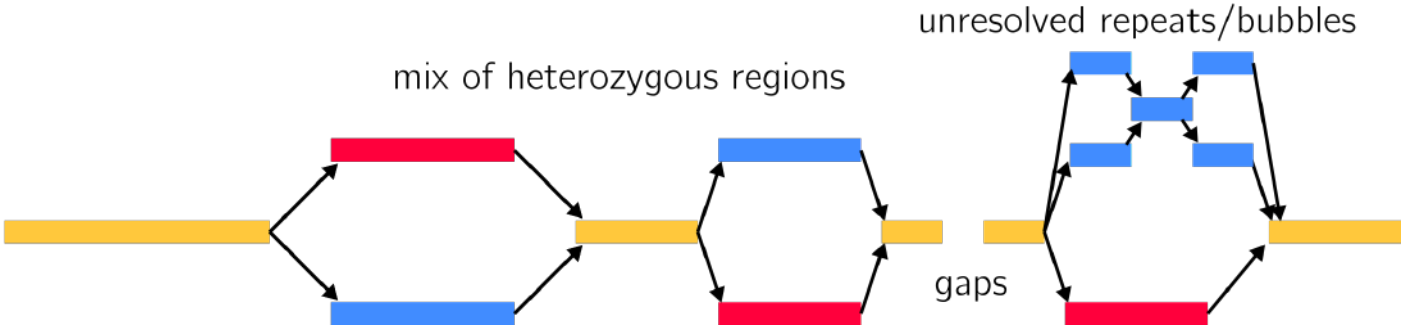
- **Fourth experiment:** *golden boy*'**s genome with long range information**

Oxford Nanopore (ONT) ultra long reads

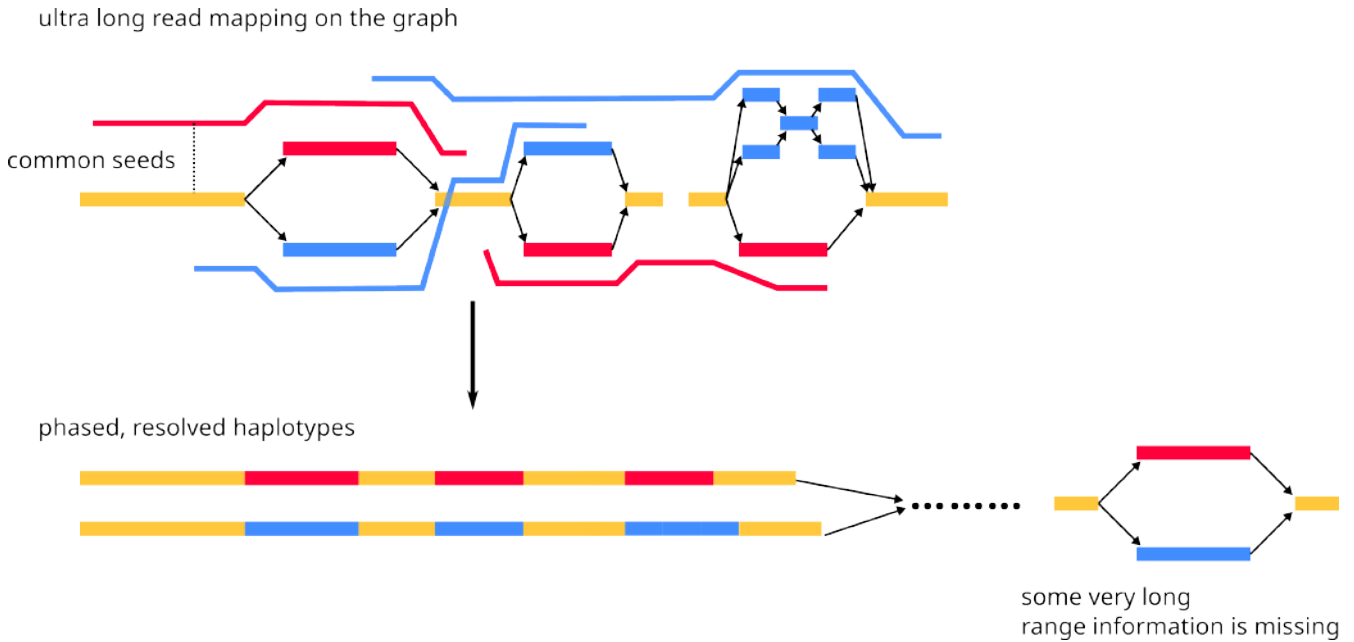


• Output of OLC/DBG after long reads assembly

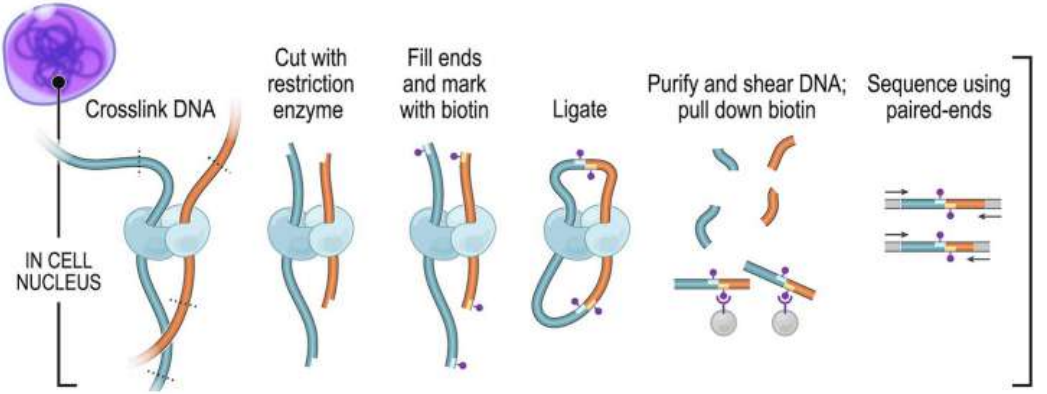
final OLC/de Bruijn graph (after cleaning/graph simplification)



• Phase and fill with ultra long reads



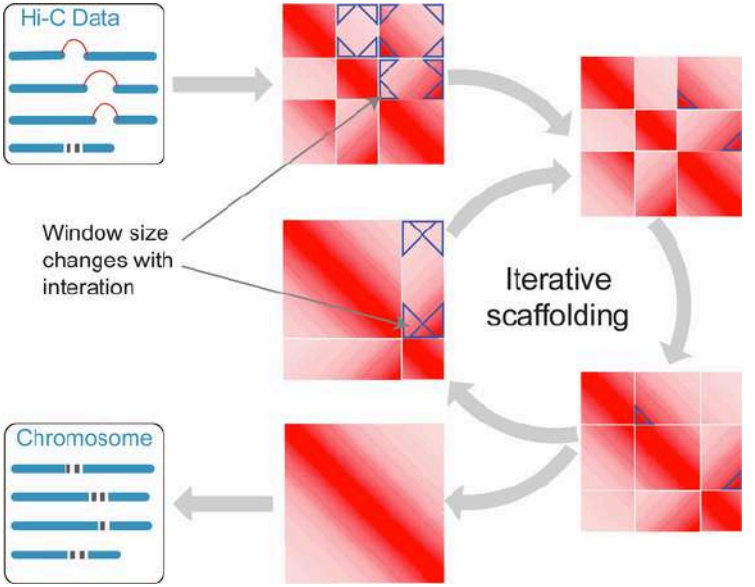
- **Long range information (e.g., Hi-C)**



from Rao et al. 2015

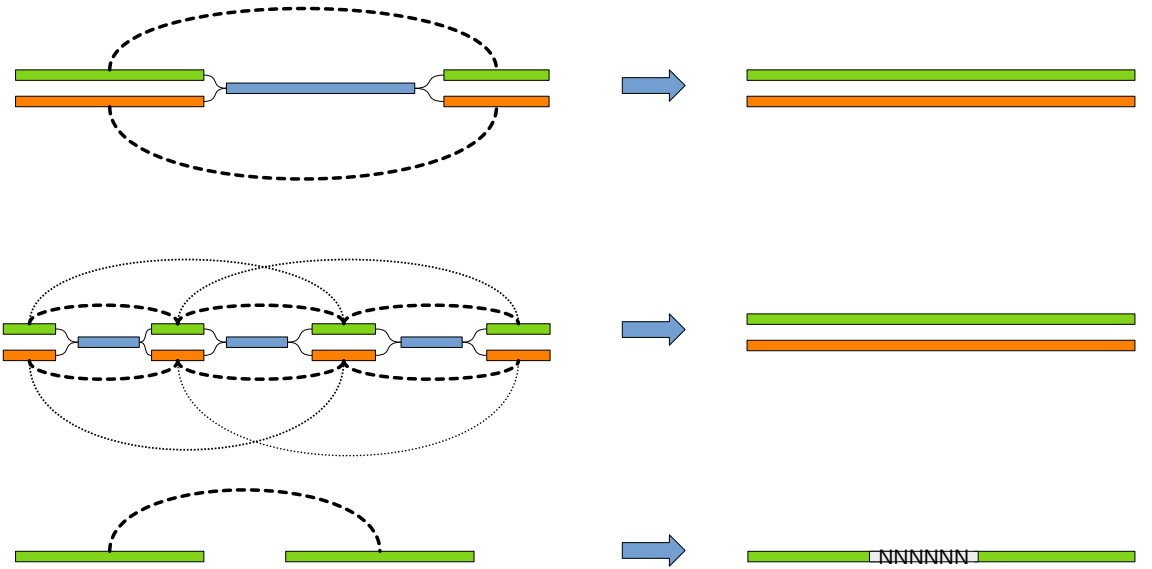
Sequences on consecutive contigs are likely to be crosslinked and appear as pairs

• Using Hi-C

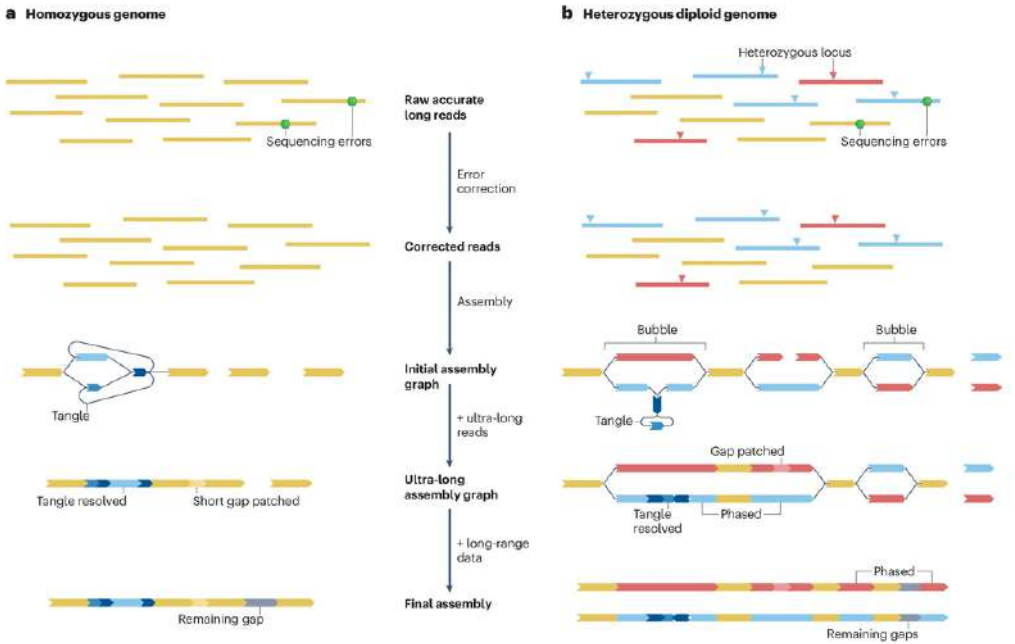

Hi-C Data
Window size
changes with
interation
Iterative
scaffolding
Chromosome

• After assembly: scaffolding

Use long range information to order contigs into *scaffolds*



• Mainstream “T2T” workflow



From Li and Durbin 2024, *Genome assembly in the telomere-to-telomere era* **A highly recommended read!**

Summary

Short reads (Illumina)

De Bruijn graph assembly → **Fragmented haploid** assembly

Long reads (Oxford Nanopore or PacBio)

Overlap graph assembly (+ polishing) → **Contiguous haploid** assembly

Accurate long reads (HiFi or latest ONT)

Overlap graph or de Bruijn graph assembly → **Contiguous diploid** assembly

Long range information (Ultra-long reads, HiC)

Scaffolding → **Large scale contiguous diploid** assembly

• Challenges in assembly

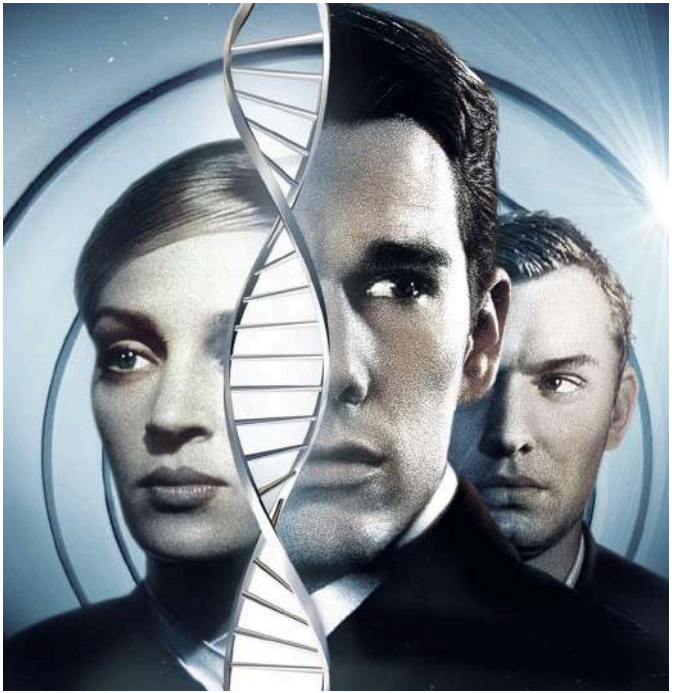
• Challenge 1: Scalability

Human

Human Genome project (2001)
1000 Genomes project (2015)
10k Genomes project (2016)
100k Genomes project (2018)
500K UK genomes (2023)

Many projects beyond human

Earth biogenome project
Vertebrate genome project
Darwin Tree of Life
Tara Ocean ...



History

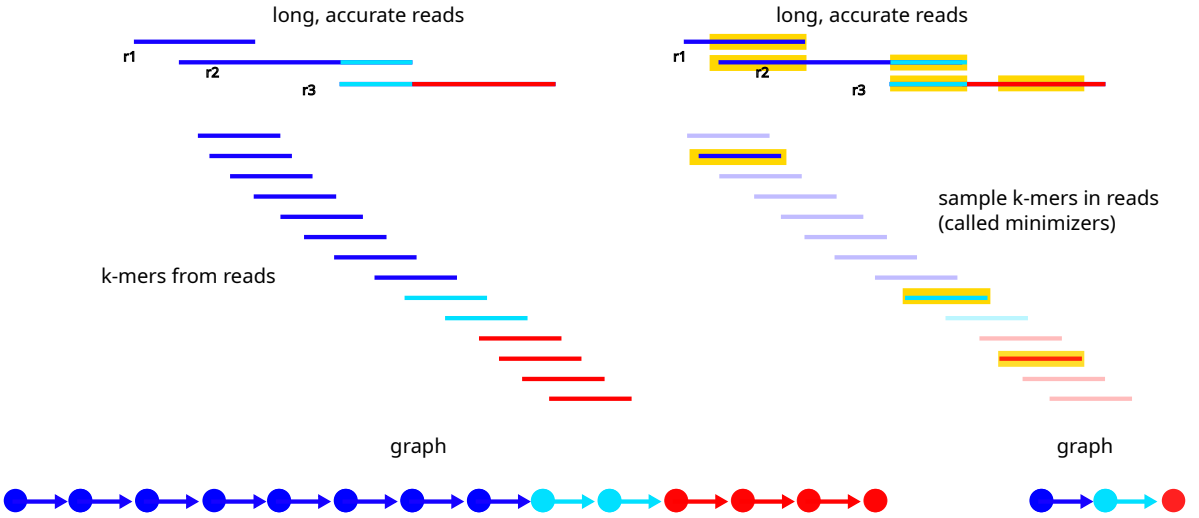
How long to assemble a human genome

Sanger: MANY CPU **years**
Illumina (Overlap graph): 2 CPU **months**
Illumina (De Bruijn graph): A CPU **day**
Long reads (Alignment): 2 CPU **years**
Long reads (Anchors chaining): 20 CPU **days**
HiFi (Anchors chaining): 2 CPU **days**
HiFi (De Bruijn graph): A CPU **hour**

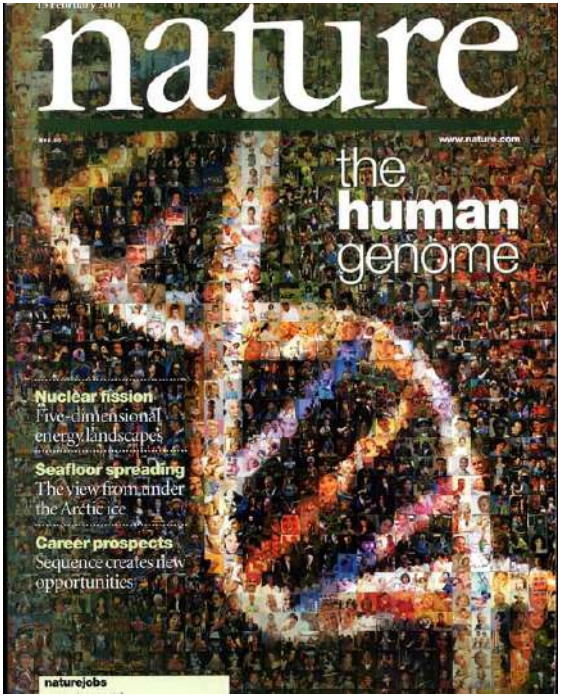
Algorithms and data structures matter!

• Very fast genome assembly with minimizers

Human genome assembled within 2 hours (Peregrine assembler) and 10 minutes (RMBG assembler), lightweight meta-genomics assembly with metaMDGB



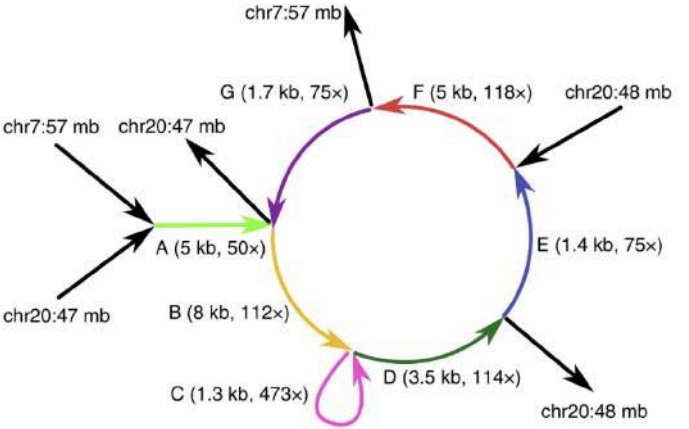
- **Telomere to telomere assembly?**



• Challenge 2: Telomere to telomere chromosomes

Main problems

Very large exact repeats
Very similar sequences
Low complexity regions
Mosaic repeats



Telomere-to-Telomere consortium

Has produced in 2022 a complete human genome with one contig per chromosome !

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

• Telomere-to-Telomere diploid human reference

T2T-YAO released in 2023 a complete human genome with one contig per chromosome !

92x PacBio HiFi
336x coverage of Oxford Nanopore (ultra long reads)
70x PacBio CLR
584x Arima Genomics HiC
BioNano DLS
Illumina HiSeq 150bp for the son and parents (278x and 116x coverage).

Don’t be desperate if your assembly is not T2T!!!

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)
- Axolotl (32Gb)

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

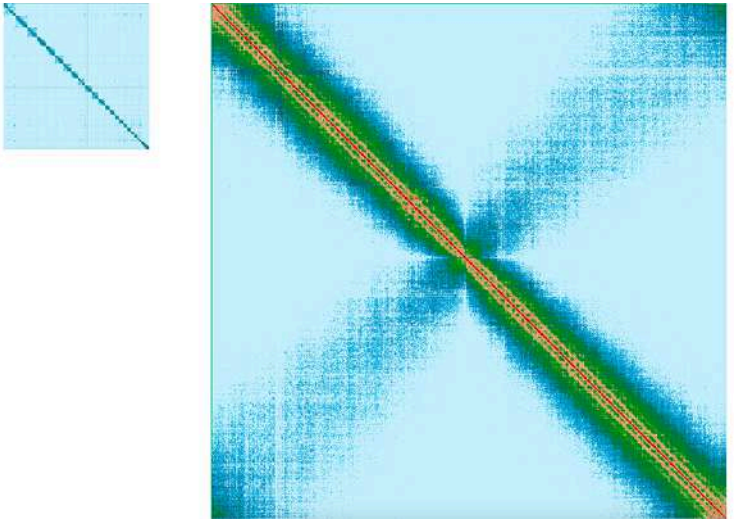
- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

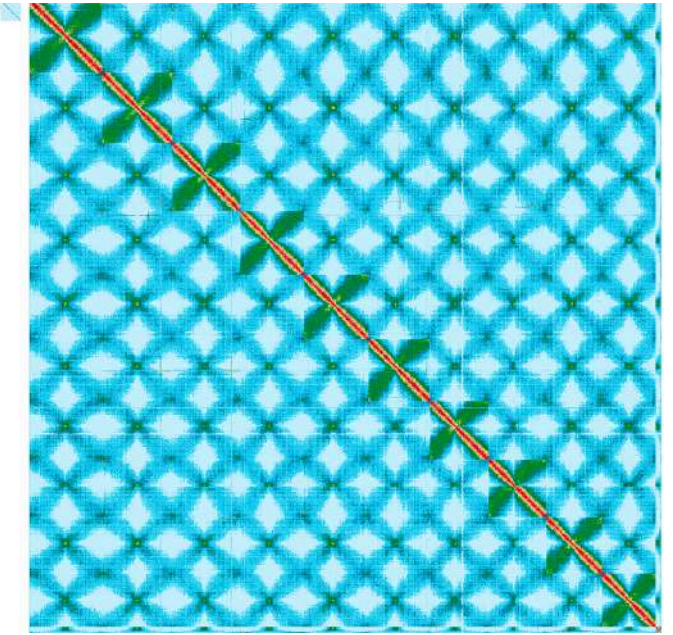
- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)
- Mistletoe (90Gb)
- Metagenomes ...

- **The human genome seems small**

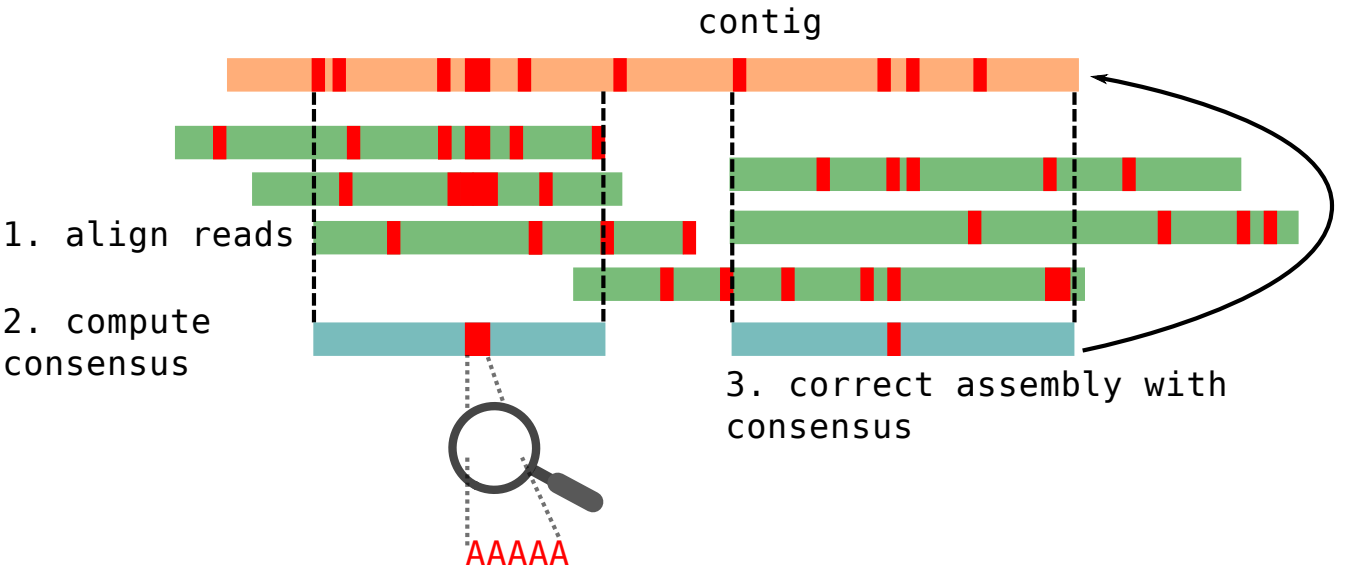


A genome contact map: main diagonal = nearby regions along the chromosome

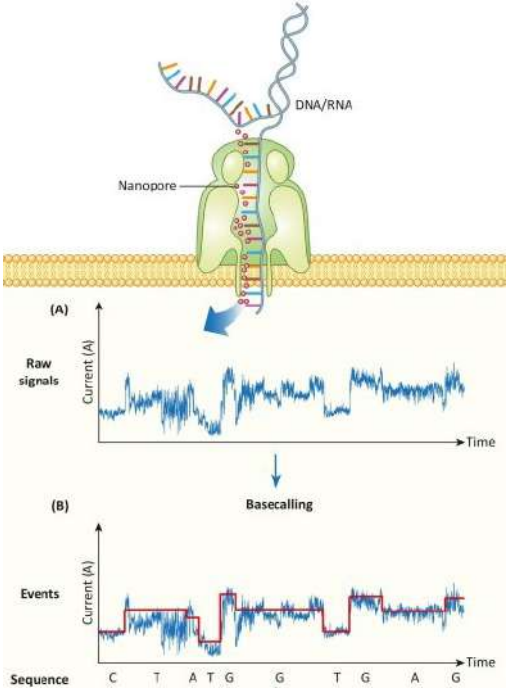
- **The human genome seems really small**



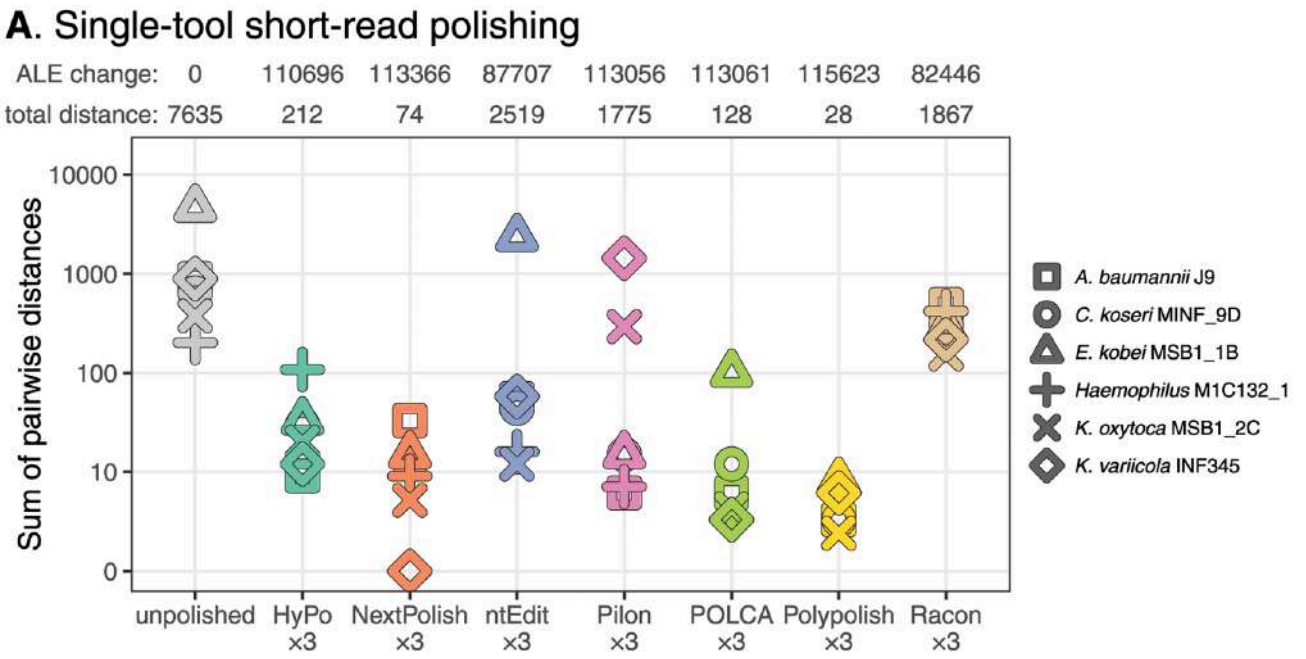
• Challenge 3: Base level accuracy



- **Homopolymers are hard to read**



• Systematic errors



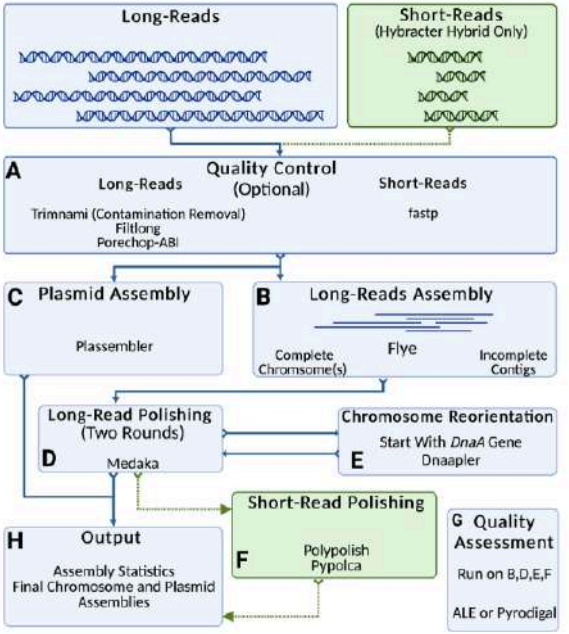
• HiFi VS (simplex) ONT

HiFi	ONT
10/20Kb long	10-100kb long
~0.05% error rate	~0.1% error rate

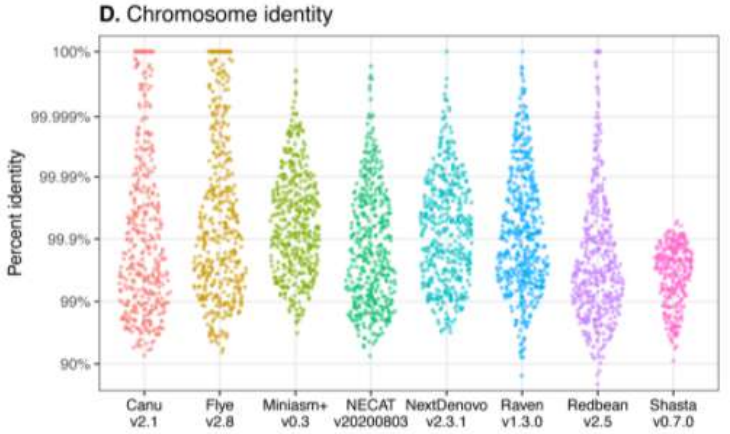
Read platform	Sub errors (per Mb)	Indel errors (per Mb)
ONT Q28	684	750
HiFi Revio	14	381
Element	287	1.3
Illumina	155	7

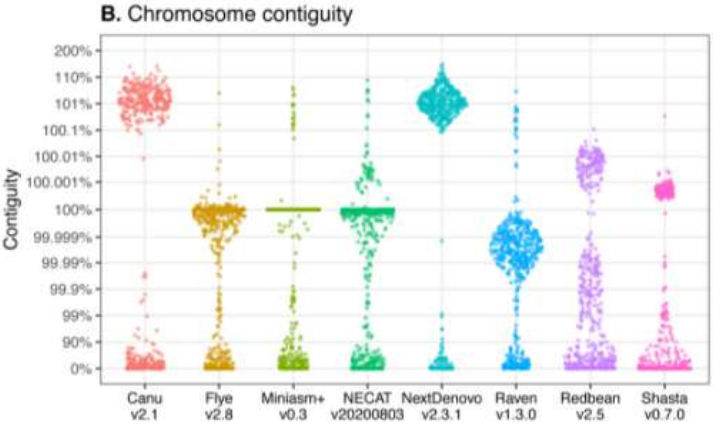
Some tools developped for HiFi can handle “Q20+” ONT (HiFiasm)

• Challenge 4 : Assembly as a software



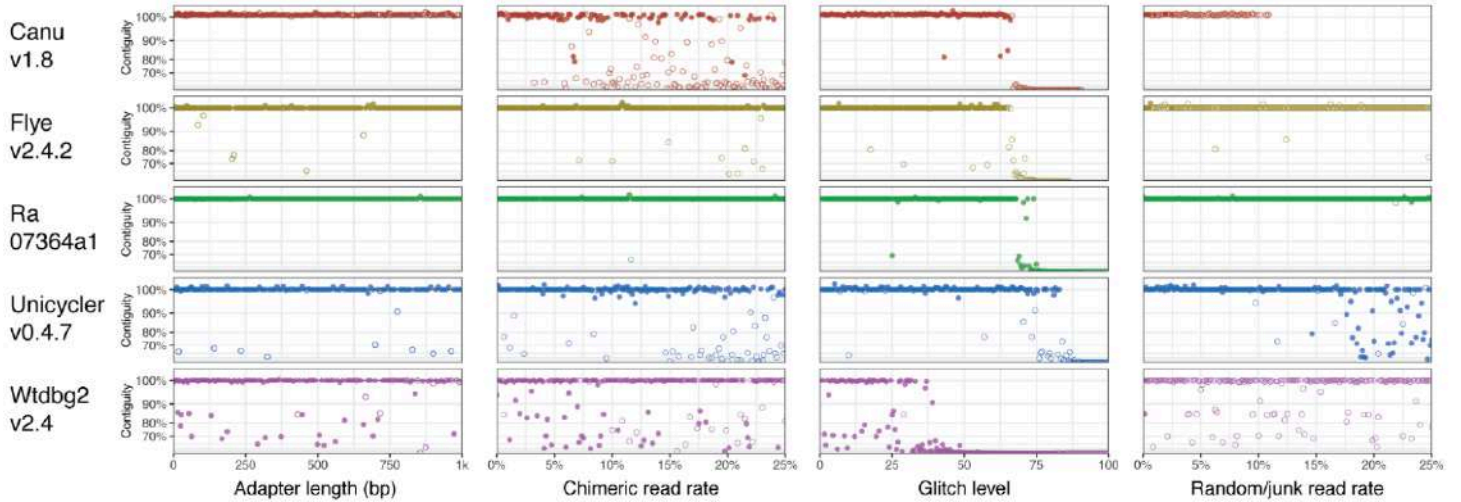
• Assemblers behave differently





From https://github.com/rrwick/Long-read-assembler-comparison

• Software robustness



From https://github.com/rrwick/Long-read-assembler-comparison

• An assembly is a model(!)

Please remember

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

Do not trust your assembly, challenge it !

Who is a good boy ?

Is my assembly any good ?

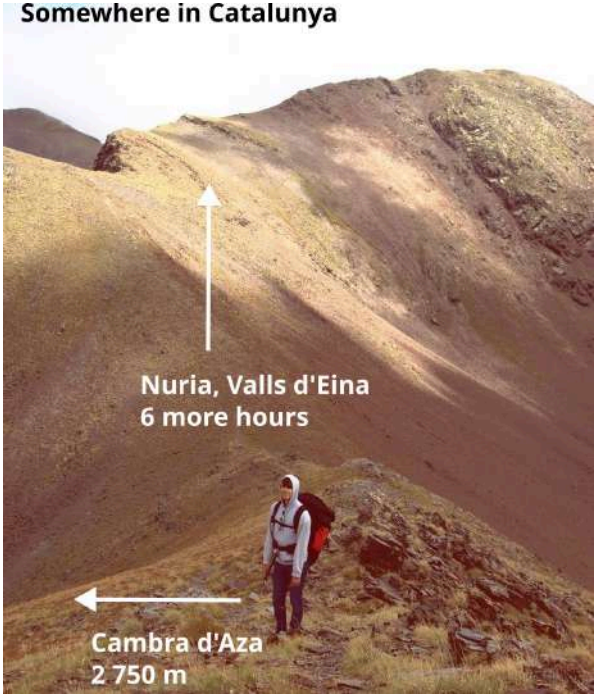
How to access my assembly quality?

How to select my best assembly?

How can I improve my current assembly?

See you at the practical !

The (first) end



Reference-based scaffolding/assembly

Pros

Do not need high coverage/long distance information to get contigous assemblies

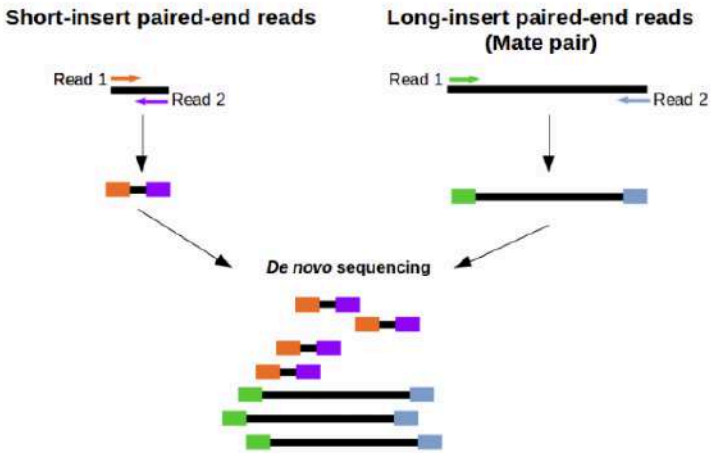
Cons

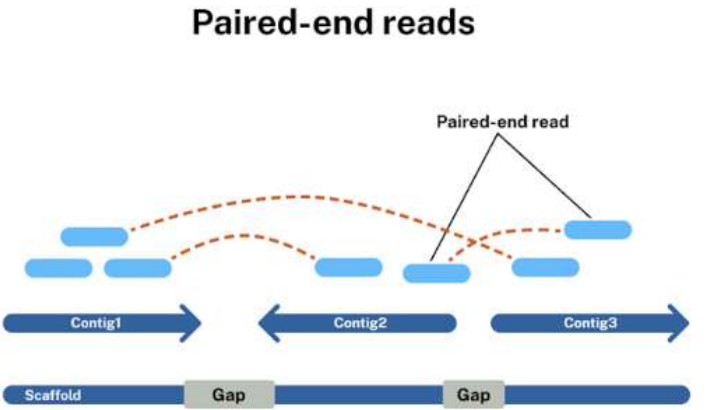
-Need a related good quality reference

-Bias toward reference sequence, for local and structural variants

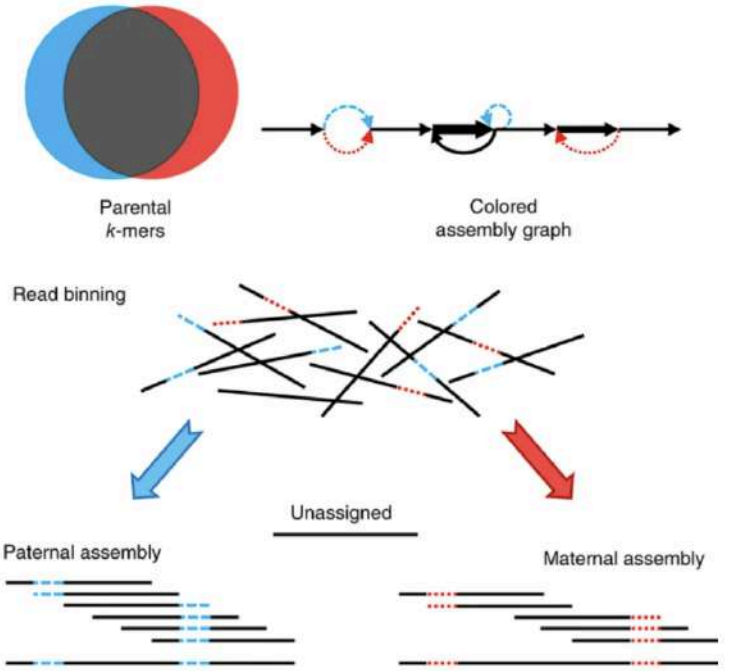
- Map the reads on a reference and compute a consensus (Medaka)
- Use a reference assembly as existing contigs (SPAdes)
- Use one (or several) related references genomes to order contigs (Ragout2)

• Paired end and mate pair

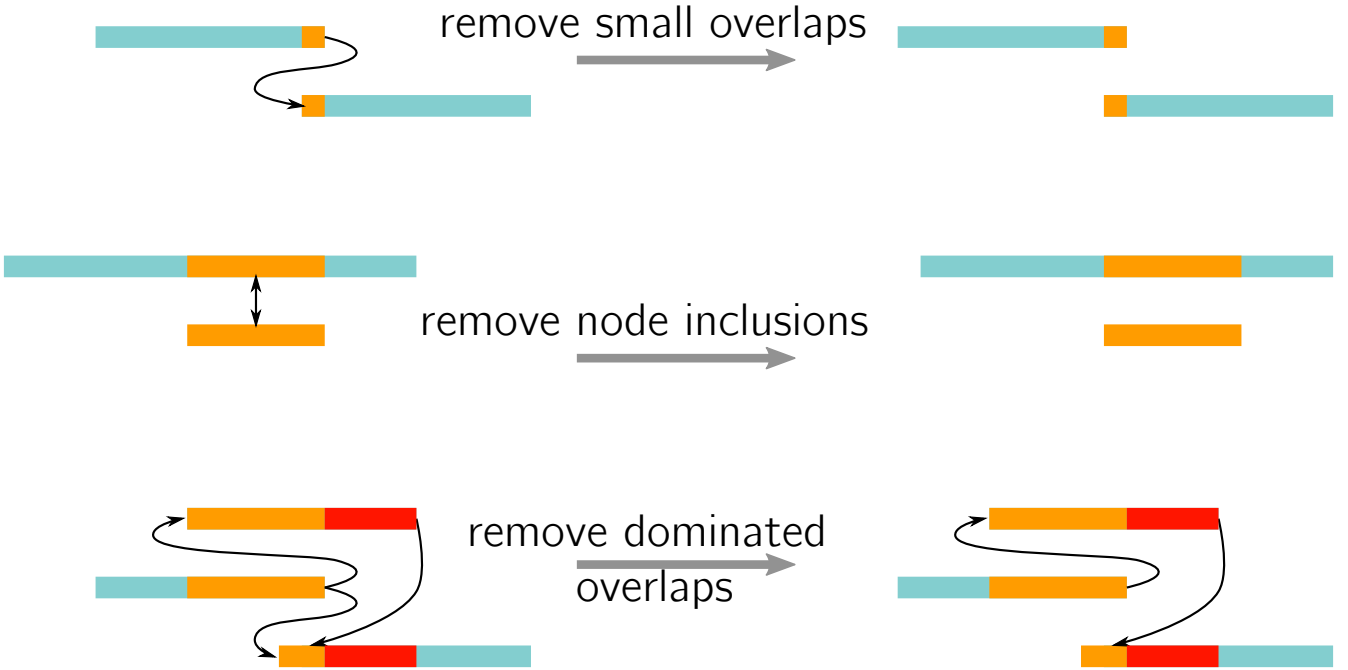




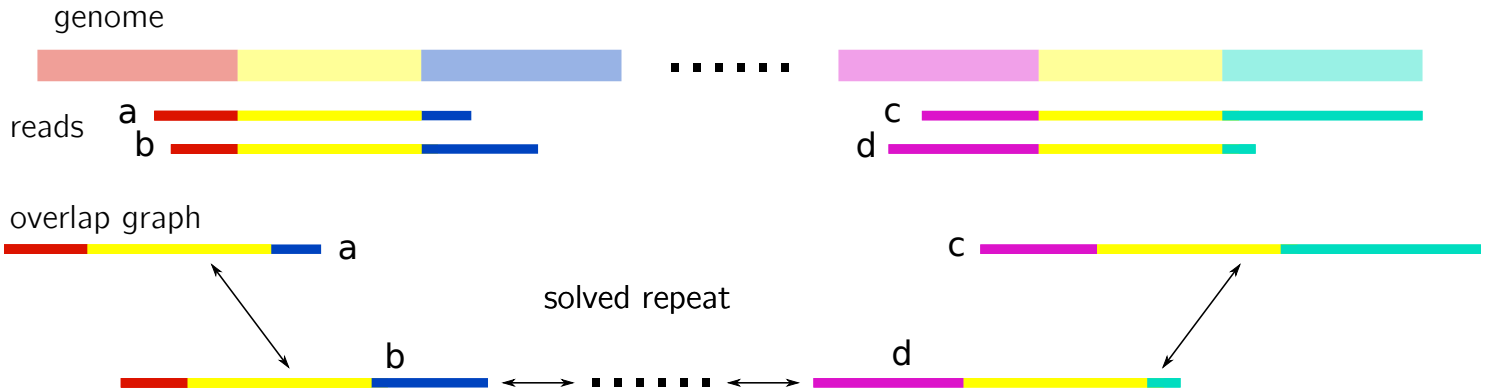
• Trio sequencing



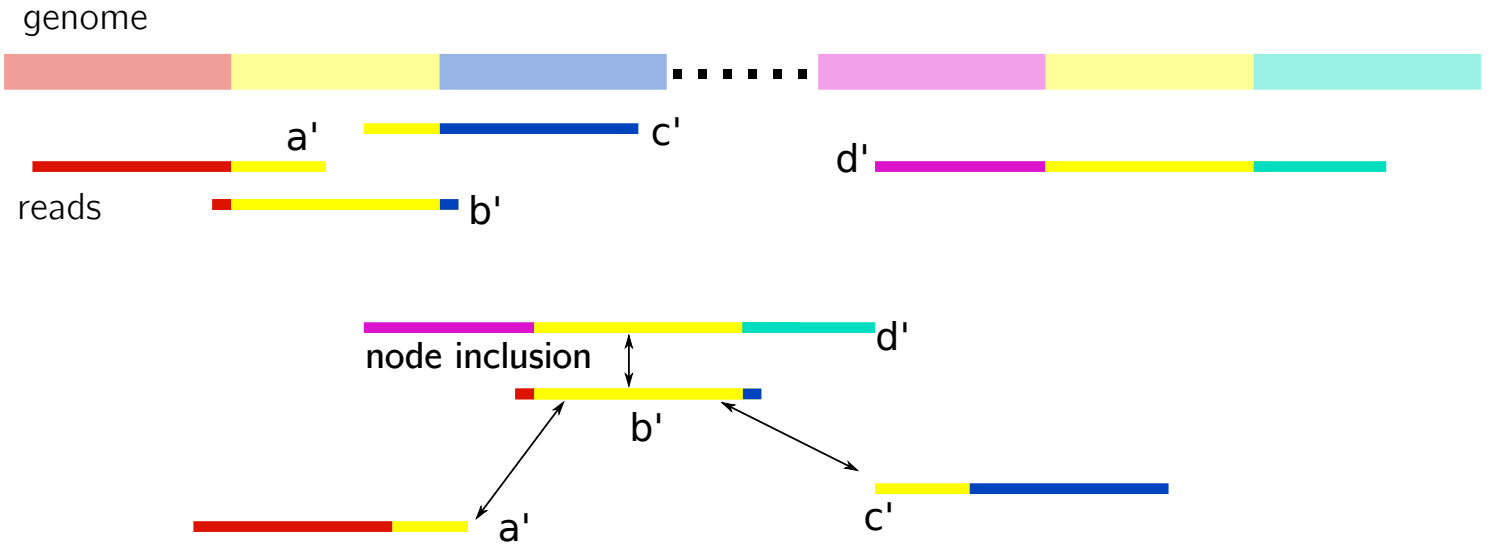
• Coming back to the overlap graph simplifications



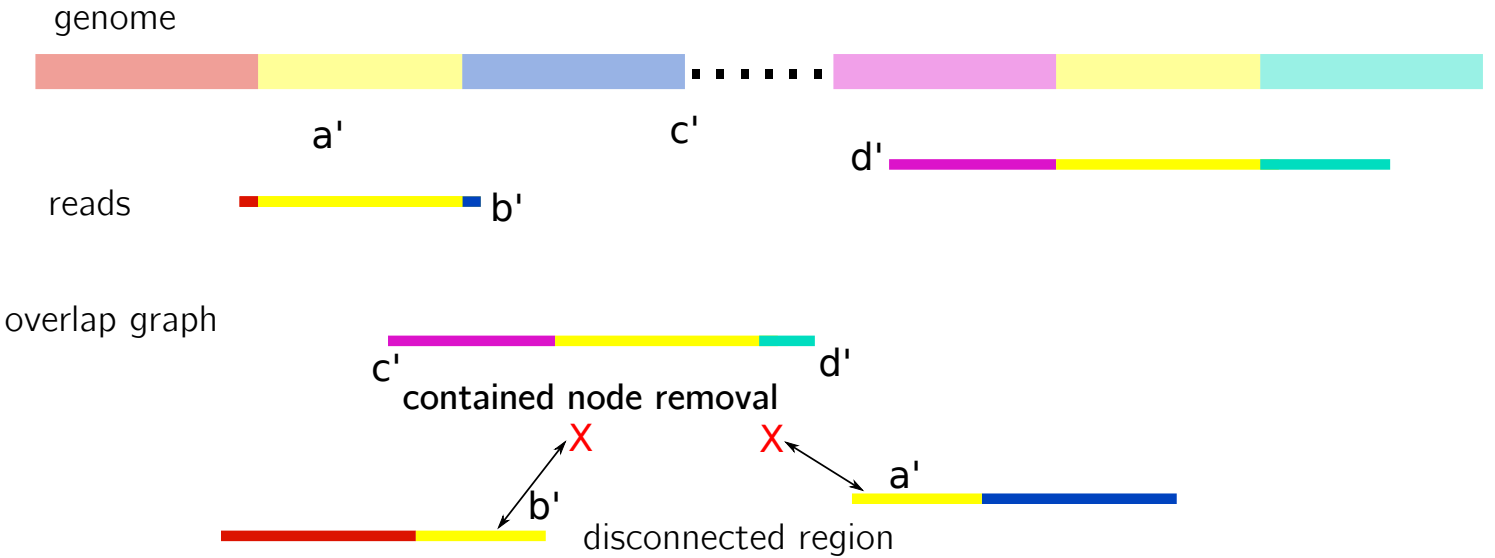
• An overlap graph limitation when using noisy reads



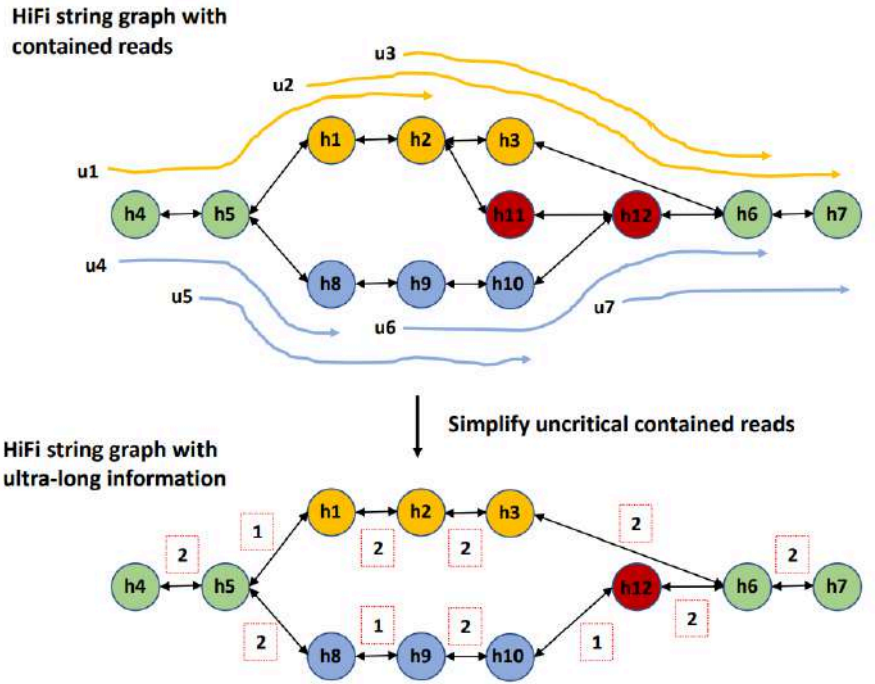
• An overlap graph limitation when using noisy reads



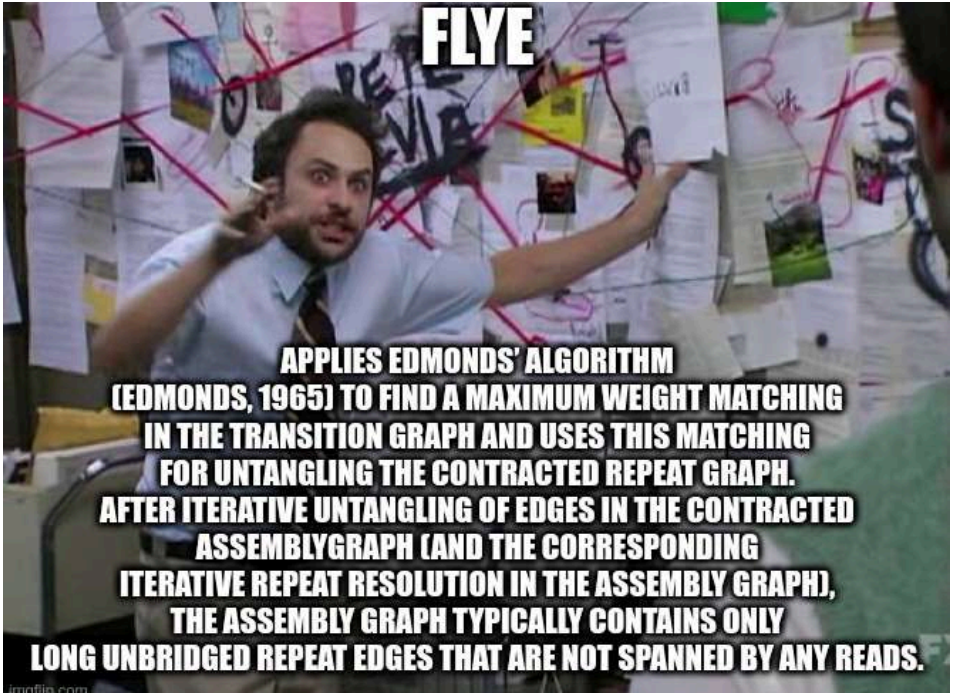
• An overlap graph limitation when using noisy reads



• Read threading alternative



- **Flye (Fly you fools!)**

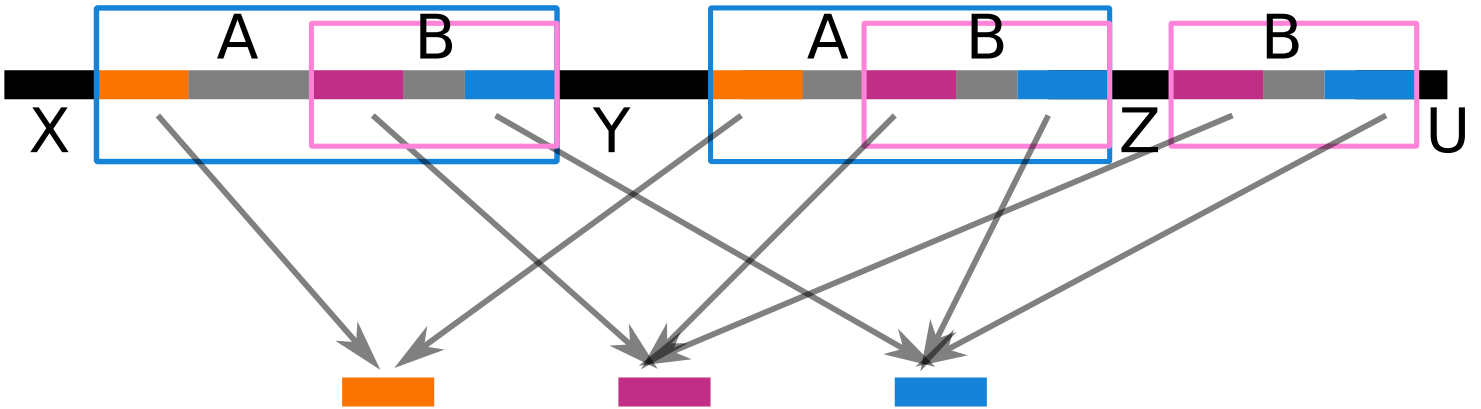


- **Repeat graph**

a genome

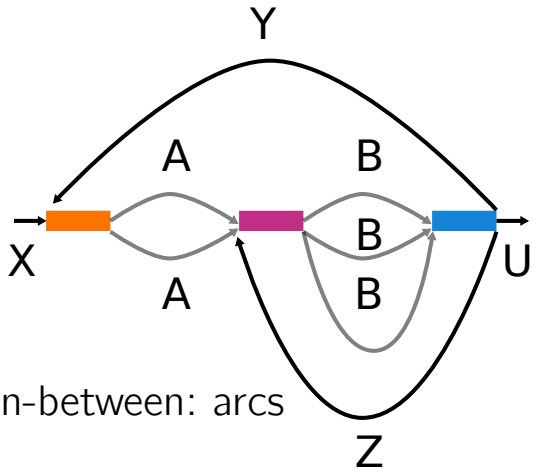
highlighted repeated regions

- **Repeat graph**



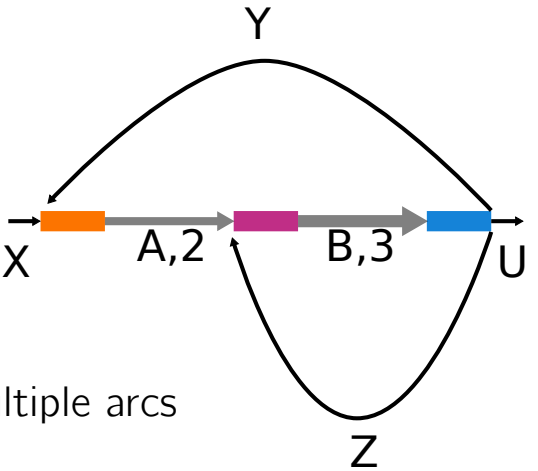
repeats extremities: graph's nodes

• Repeat graph



sequences in-between: arcs

- **Repeat graph**



collapse multiple arcs