Where are we now?

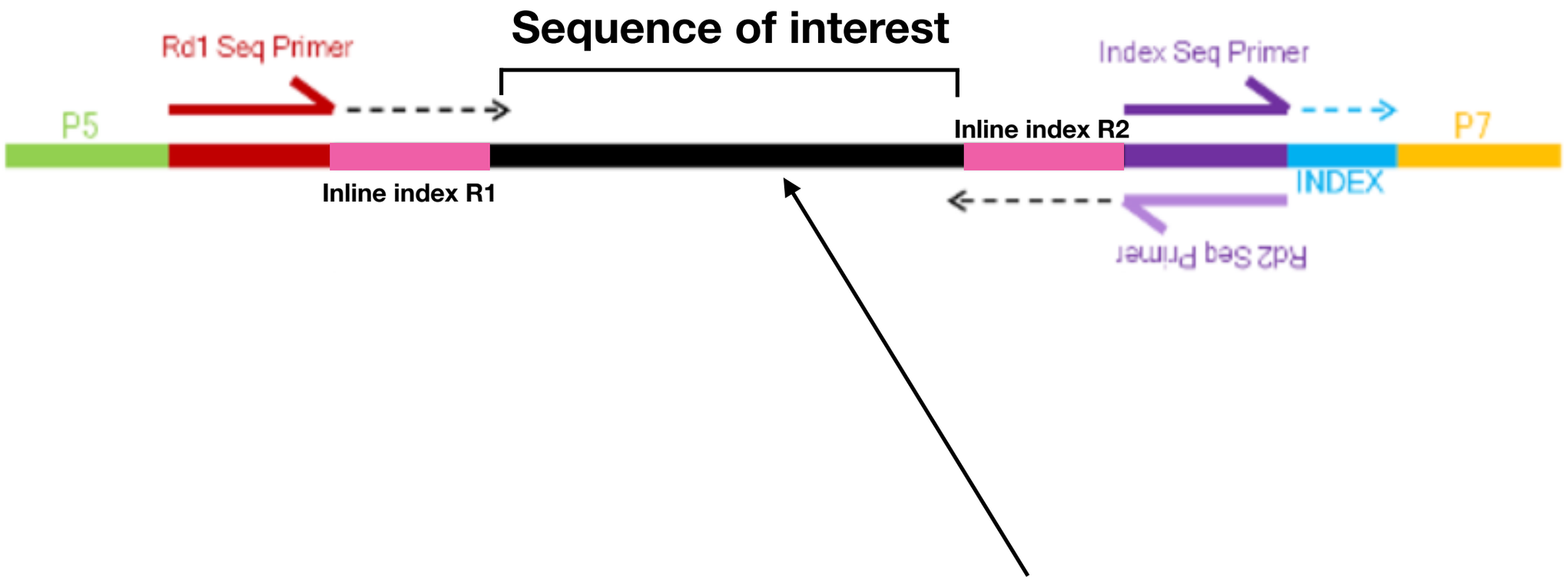
Exhausting experiment

Sequencing

ATCG…..

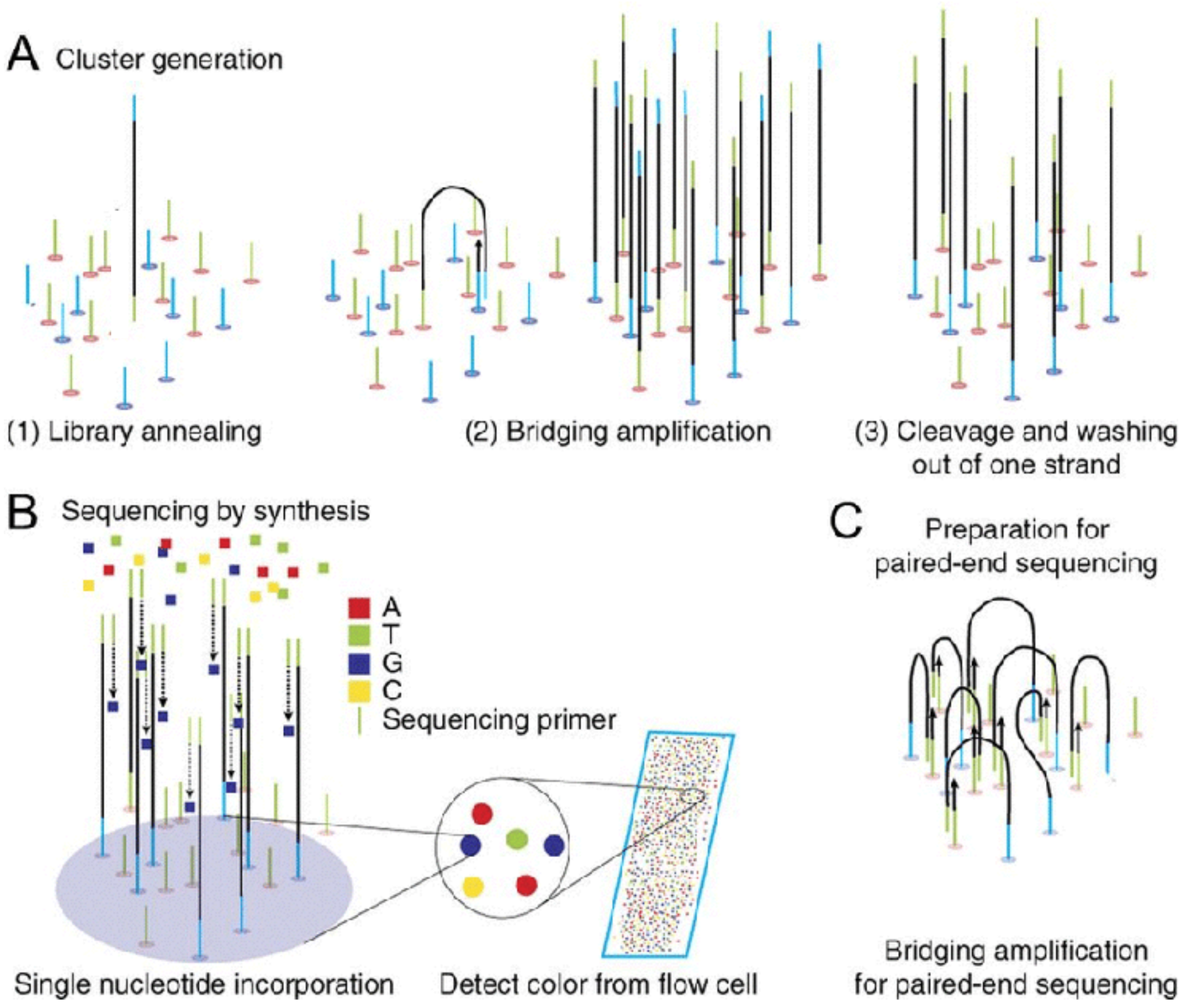
Analysis our data

Structure of prepared library

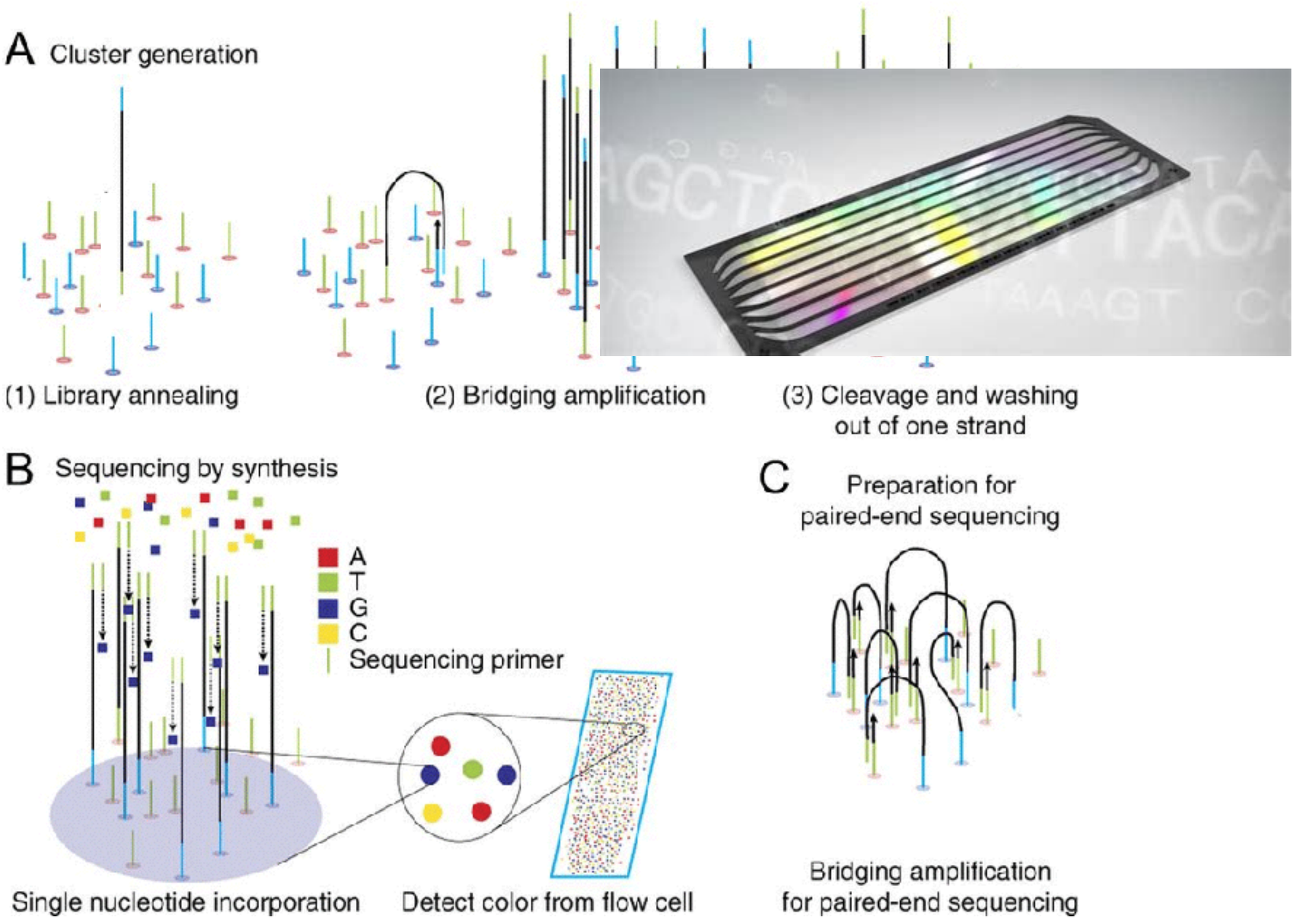


Length of this part is sometimes called “insert size”,
which totally depends on the length of DNA after shearing

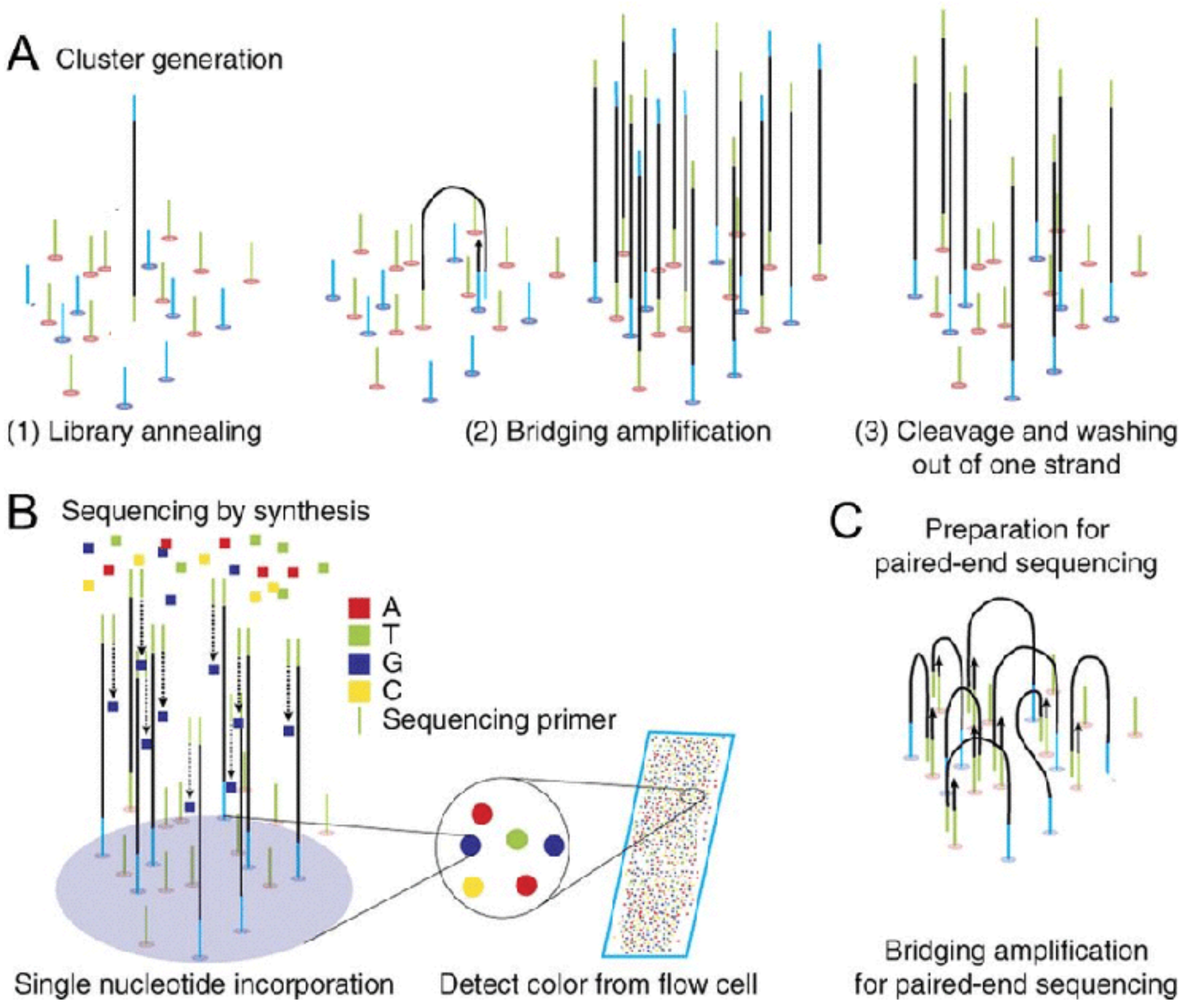
Pair-end Sequencing



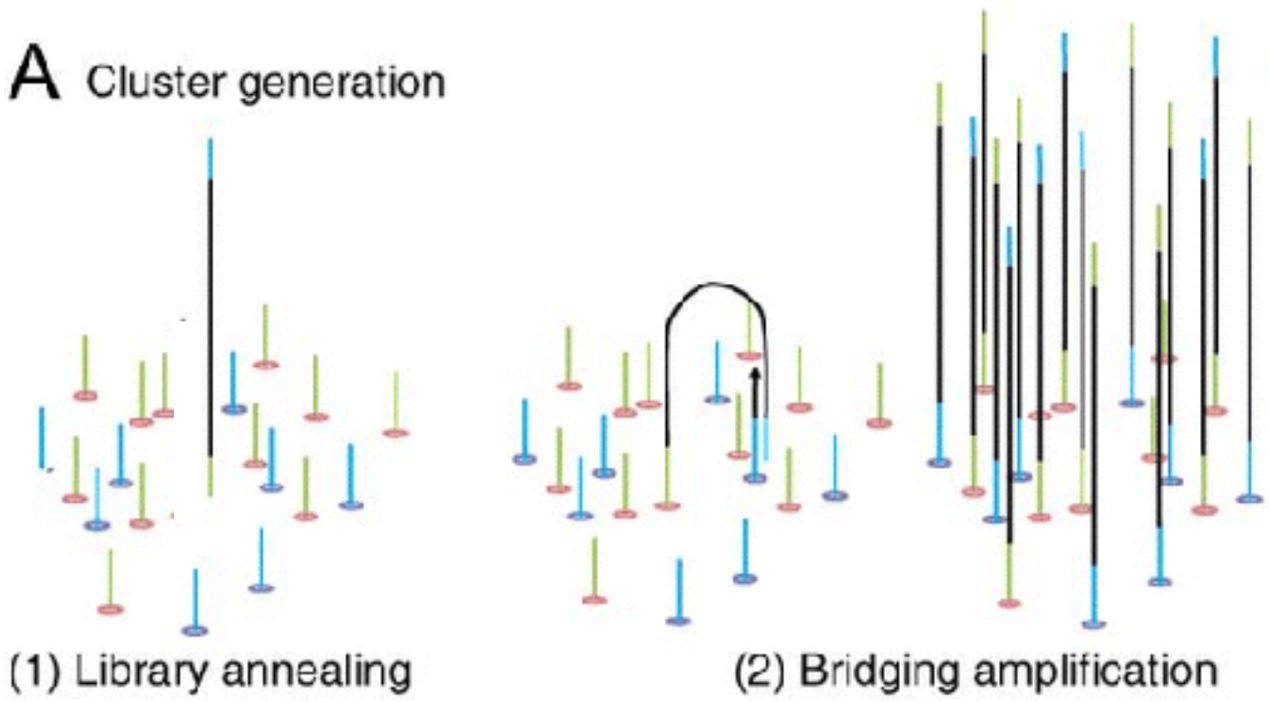
Pair-end Sequencing

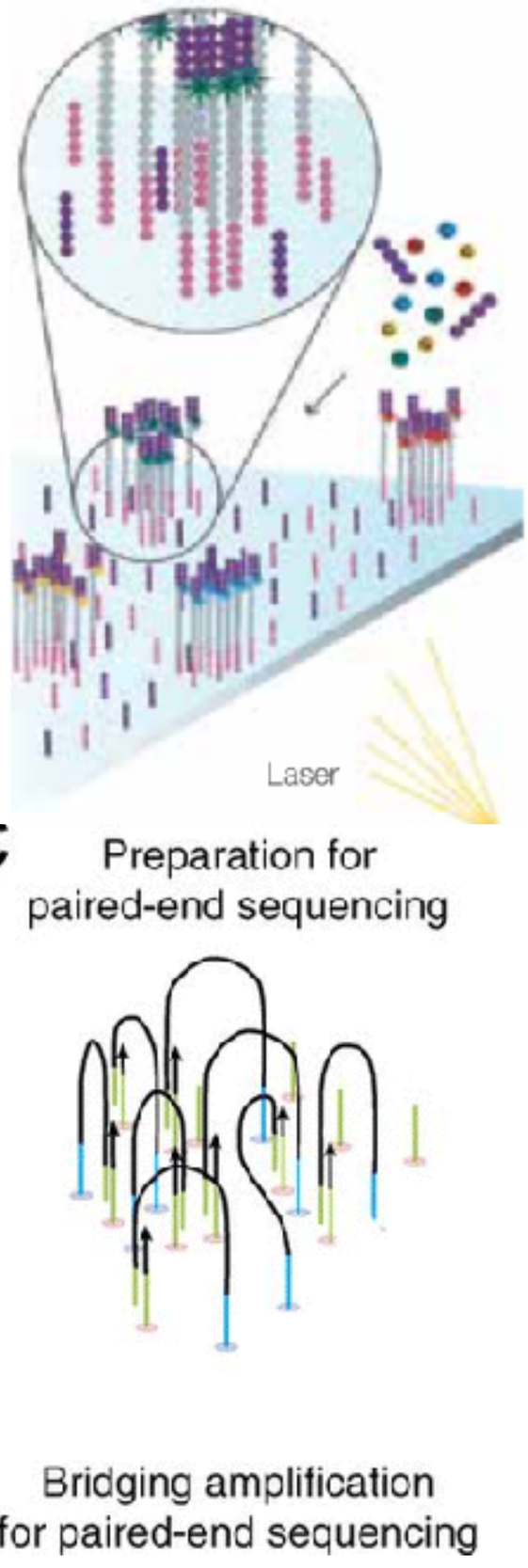


Pair-end Sequencing

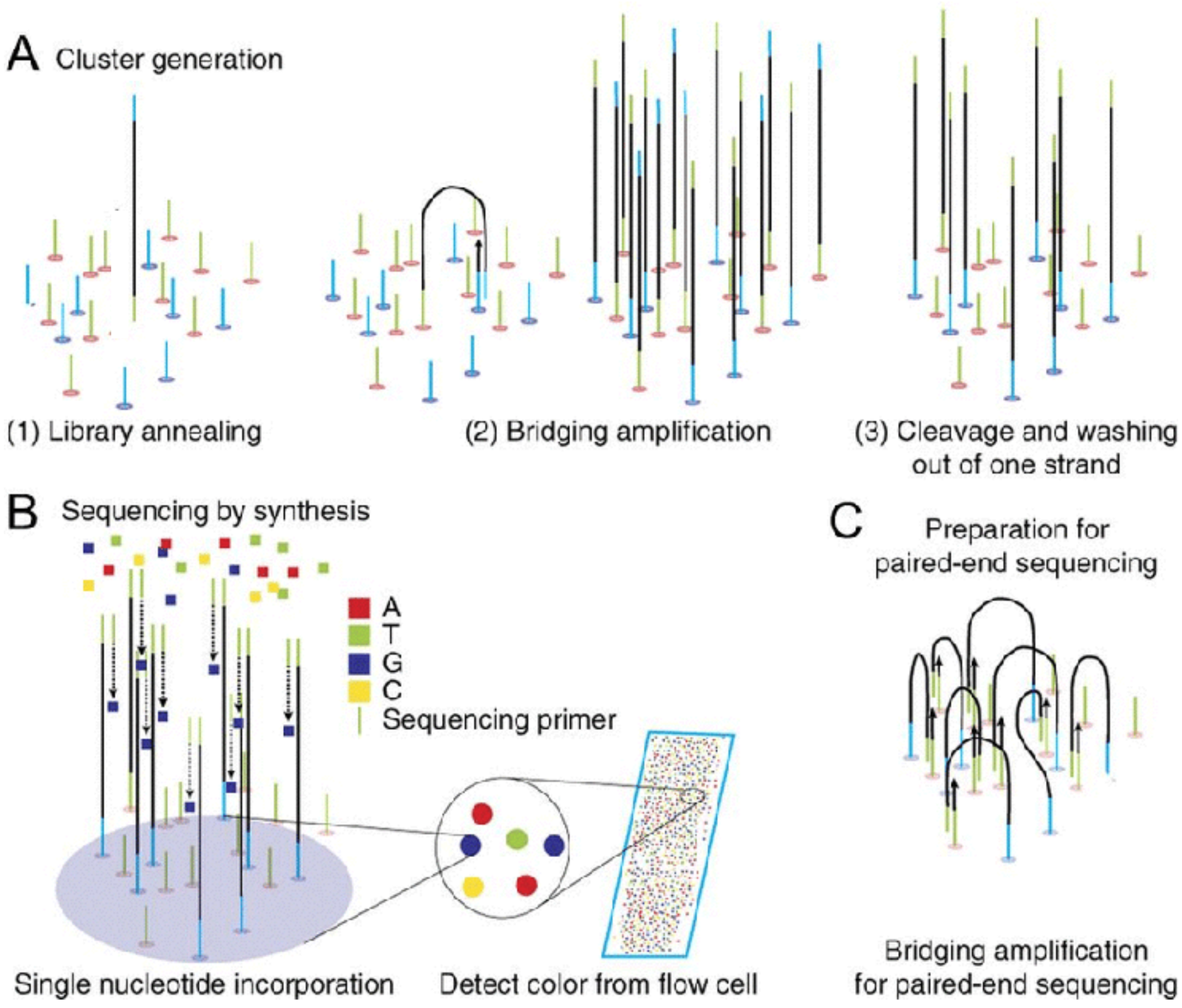


Pair-end Sequencing

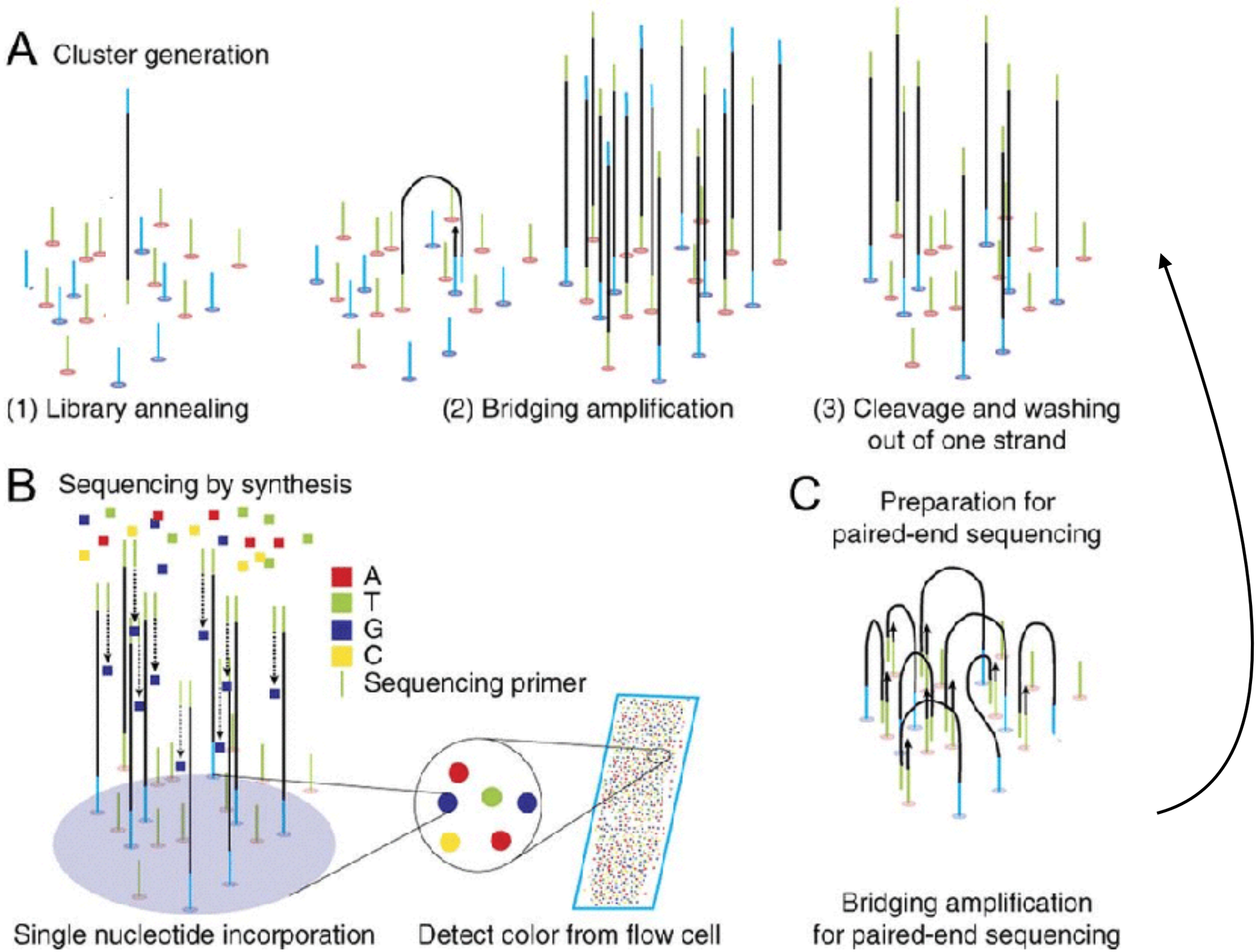




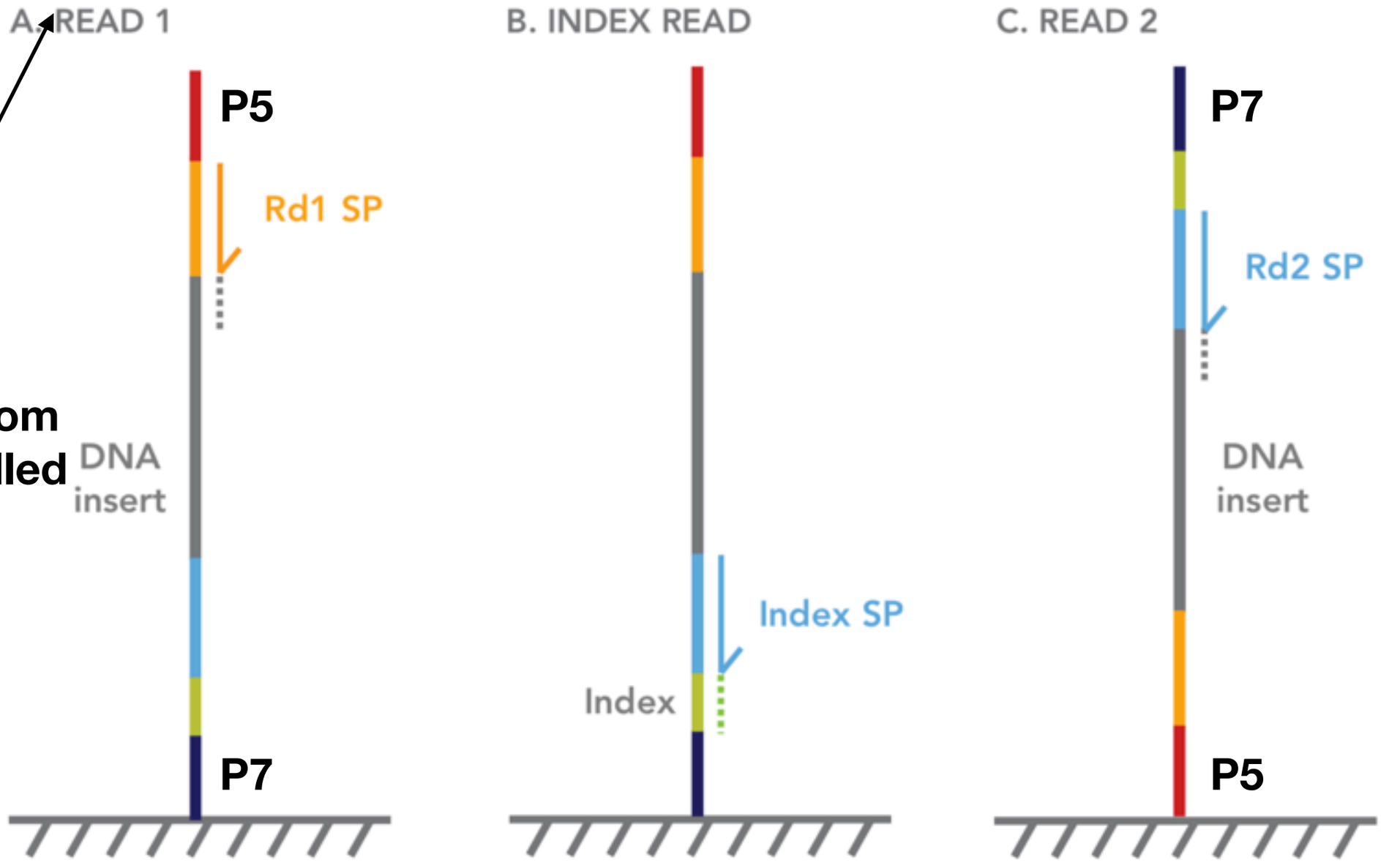
Pair-end Sequencing



Pair-end Sequencing



Pair-end Sequencing

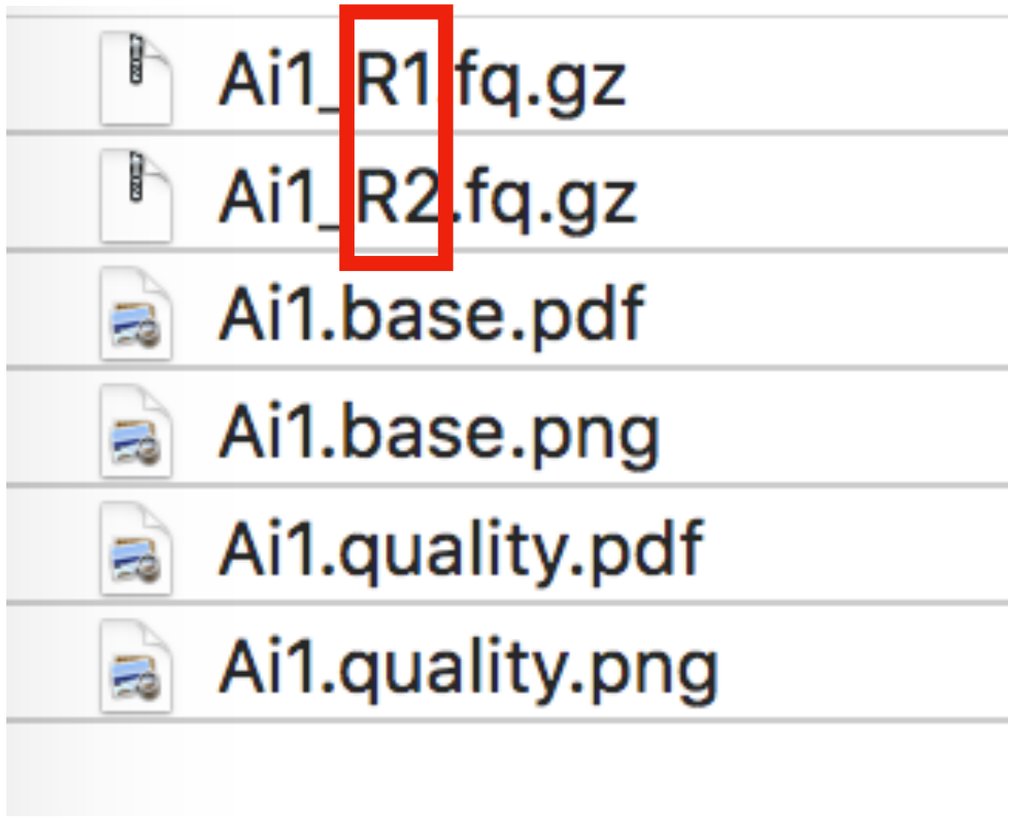


Sequence transferred from sequencer called "reads"

forward direction

reverse direction

What comes from the sequencer



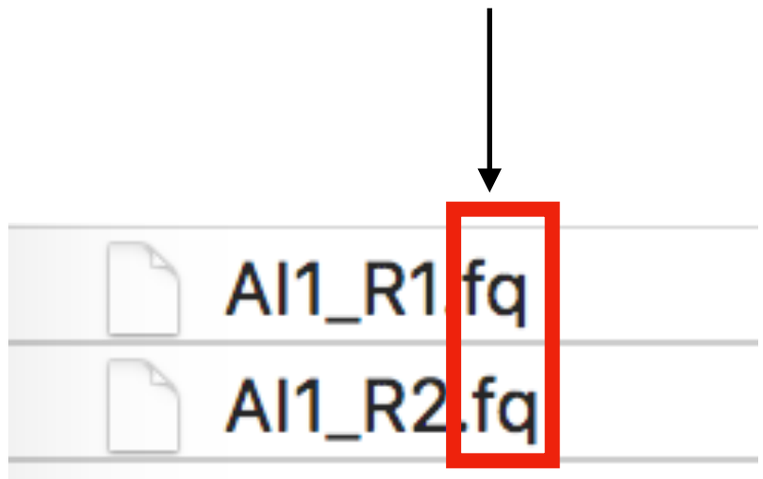
Assigned (more professionally called “demultiplexed ”) reads according to index close to P7

Reads

Bases composition

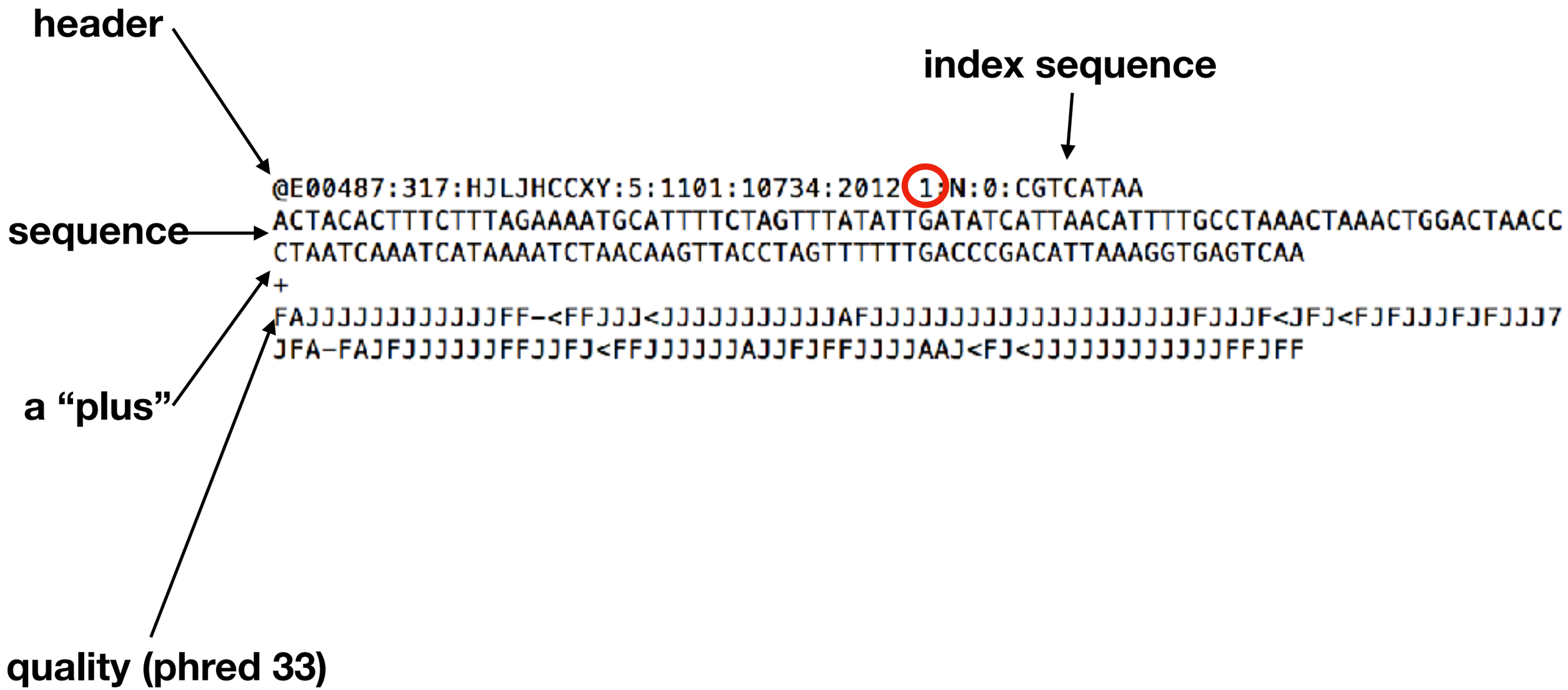
Quality of Bases

File suffix is “fq” or “fastq”

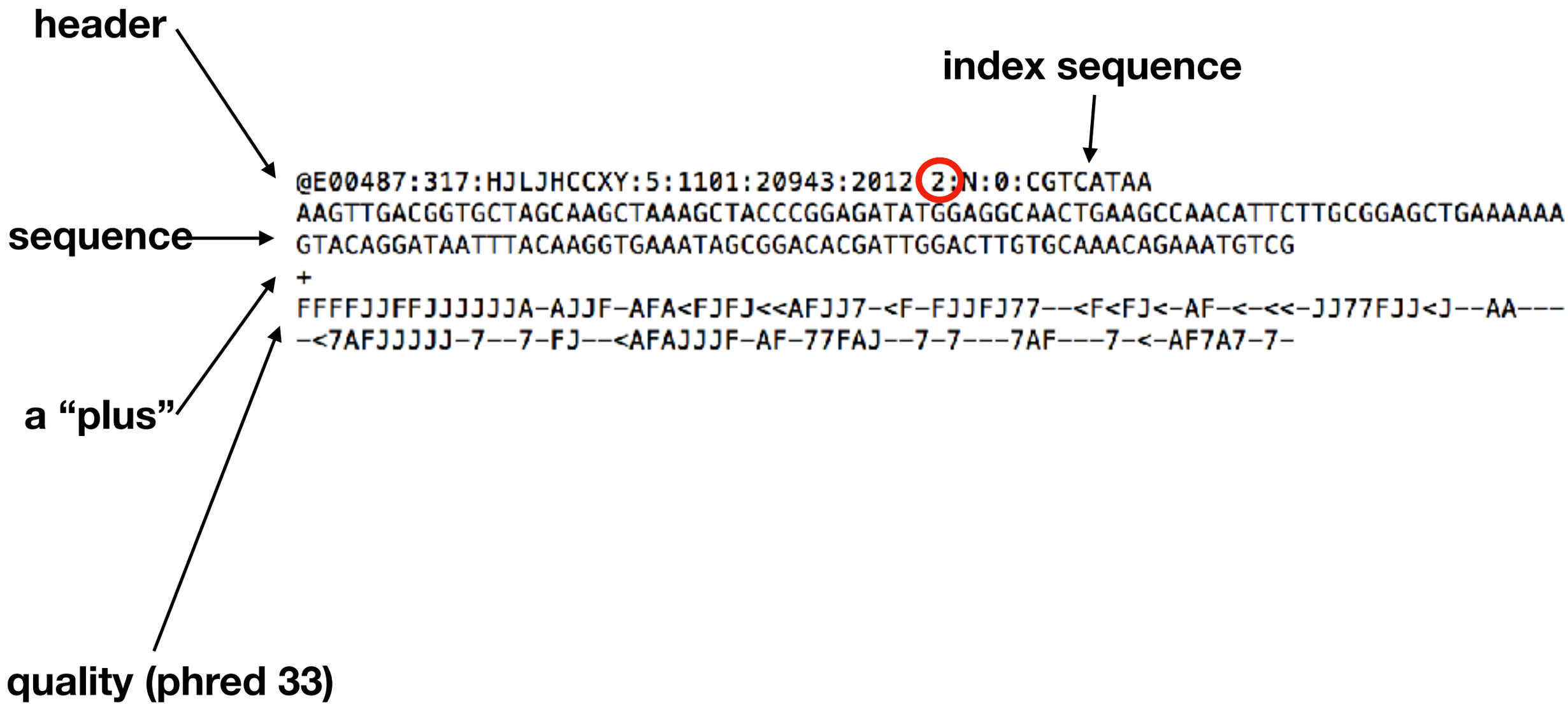


Expanded file

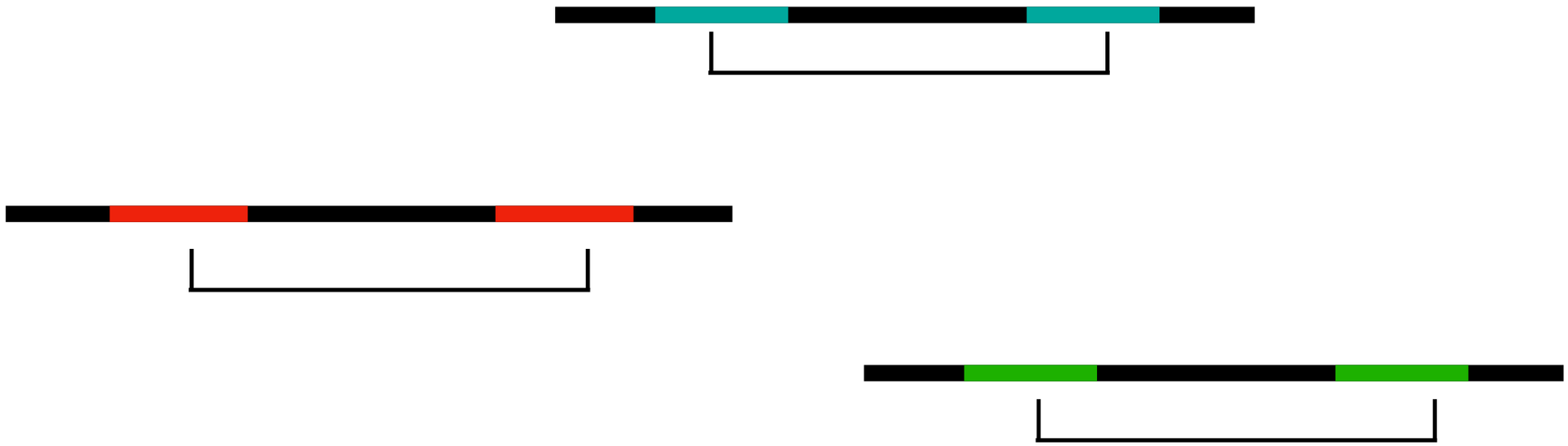
How reads in xxx_R1.fq looks like



How reads in xxx_R2.fq looks like



What’s paired reads



Reads originated from the same molecular before bridge amplification called paired reads

Paired reads in xxx_R1.fq and xxx_R2.fq

xxx_R1.fq

```
@E00487:317:HJLJHCCXY:5:1101:10734:2012 1:N:0:CGTCATAA
ACTACACTTTCTTTAGAAAATGCATTTTCTAGTTTATATTGATATCATTAACATTTTGCCTAAACTAAACTGGACTAACC
CTAATCAAATCATAAAATCTAACAAGTTACCTAGTTTTTTGACCCGACATTAAAGGTGAGTCAA
+
FAJJJJJJJJJJJJFF-<FFJJJ<JJJJJJJJJJJAFJJJJJJJJJJJJJJJJJJJJFJJJF<JFJ<FJFJJJFJFJJJ7
JFA-FAJFJJJJJJFFJJFJ<FFJJJJJJAJJFJFFJJJJAAJ<FJ<JJJJJJJJJJJJFFJFF
```

xxx_R2.fq

```
@E00487:317:HJLJHCCXY:5:1101:20943:2012 2:N:0:CGTCATAA
AAGTTGACGGTGCTAGCAAGCTAAAGCTACCCGGAGATATGGAGGCAACTGAAGCCAACATTCTTGCGGAGCTGAAAAAA
GTACAGGATAATTTACAAGGTGAAATAGCGGACACGATTGGACTTGTGCAAACAGAAATGTCG
+
FFFFJJFFJJJJJJA-AJJF-AFA<FJFJ<<AFJJ7-<F-FJJFJ77--<F<FJ<-AF-<-<<-JJ77FJJ<J--AA---
-<7AFJJJJJ-7--7-FJ--<AFAJJJF-AF-77FAJ--7-7---7AF---7-<-AF7A7-7-
```

Paired reads have unique id in their corresponding file and exactly the same name

Compare reads in xxx_R1.fq and xxx_R2.fq

xxx_R1.fq

```
@E00492:247:HFMH3CCXY:8:1101:18436:2909 1:N:0:CTGCAATG
CTCAAAAACAACAATAATCTATTATCACAACCAAAGCAACAGCATACATATAATATCAATGCTTCTTTATCTCGATGTGAATAATAAGAAGTCAATGGGAACTTACCTTT
+
FJJJJAJJJJJFJJ<-FJJJFJJJJJJFJJJJFJJFJJJJJJJAJJJJJJJJJJJJFJFJJ<JJJJJJFFAFJF77-A<AJJJJJAJJJJJFFJJJ7<F-7FJFJ-AJF-F
@E00492:247:HFMH3CCXY:8:1101:6725:3401 1:N:0:CTGCAATG
AATTCAGCGACTTGGTTCACACAATCCTCCATCCCTCCTTCCTTTACCACGATGTTTGATGATGCTGCAGGTACAGCTGCTGAAGAGACACGAGCTGTGACCAGTAACTA
+
7JJJAJJJ-FJJJJJFJJJJAJFJFJA-FFJJFJFJJJAAJJF7AJJJJJJJF7A7JJJJFJJJJFJJJAFJJAJFFFJJJ7JFFJJFJJJJJF<-----77---7--77
@E00492:247:HFMH3CCXY:8:1101:8745:3841 1:N:0:CTGCAATG
AAATGTAAGGTCAAACTACAACCTTGTCTATGAAGCCGGTGTGCCATTTGCGTAGCTTCCTGTTTTCGGTGGAGAGTTTCTCAGCAGCAGACTGCAGCTTGGCATTGTCA
+
JJJJJJJJJJJJJJFJJJFFJJJJFJJJJJJJJJJFAFJJJJJ<FJJJJFJJJJFA<FJ-<AAJFJJJF-7A-AJJ-7JA7FJJFJJ7JFJJJJJ7J7777--7-<A<7A
```

xxx_R2.fq

```
@E00492:247:HFMH3CCXY:8:1101:18436:2909 2:N:0:CTGCAATG
ATAATGGTTTTTGACAAGTTTAGCTTAAAGGAGGAAAATTGCATGTGTTGTGATCTTGTCACTCTCTGTGTCTCCTAGACTCCTGCAGAATTGCAAGGAGGACAGGTTAGA
+
AJJ--7F<-<F---<F--FFJ7FJ-77AJ-AF-77-7F<-<<---F-7-AF--7---<----7AAA7-7-----7AJ-A-AA-A-AJF7A7--AAF7J<A7FJAJ--7A-F
@E00492:247:HFMH3CCXY:8:1101:6725:3401 2:N:0:CTGCAATG
GGTATCTTTAAAACACATTTAACATGCGATACATAACTTATAATTGGTCATGTTTGTCATACAAATTGAATTAGACAGTGAATCAAAGAAAATGTGCTTACACTGACAAGT
+
JAA7A--<FJJ--<JFJ7FF<JFJJ<-F-<JJ-FJF77F<FJJJFJJA-FJJJFJ<<JJ7JJFFF<F-FFA-A--7<-F-AFAJJA<FFJFJFFFF-JJFJF--A7--77-
@E00492:247:HFMH3CCXY:8:1101:8745:3841 2:N:0:CTGCAATG
AATGGTTGCCAAGCAACACAGAGACGAAGTGGCAAGTATGATTGAAGATTGGAGTGATATGAGTGATATTAACATTCAGCTGGGGGTGATTATTAACATTTAGCAGGATAT
+
FAF<-7FA<7-<7<<7FFJ7F<<--A--<AFF-7F-A<FF-<A<-FF7FF7AJF77<JF-<AF-J-F-AFAFJJJAF-7<AAAAAJAFJ<FFFJFJJJF<J<<A<<<-A-7
```

Paired reads are placed at the same line of its corresponding file

Let’s start analysis now!!

What's the goal of analysis?

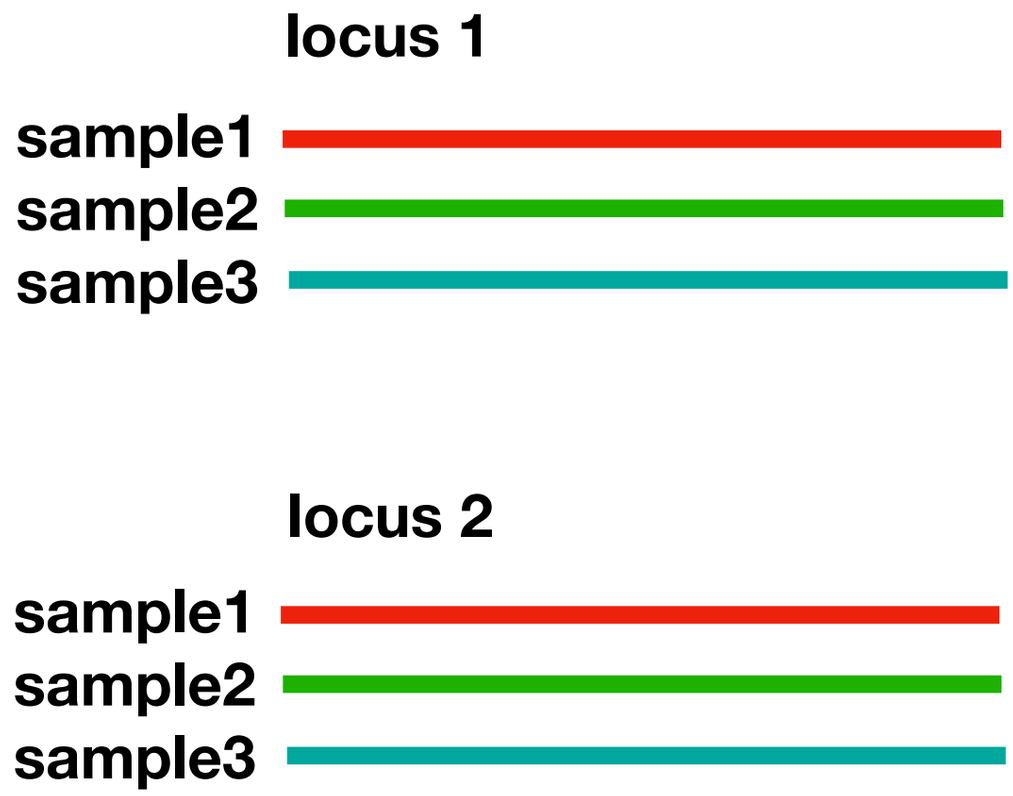
The sequence of loci used to design the baits, we call it "reference"

locus 1

locus 2

locus 3

Intact sequences of corresponding locus of each sample



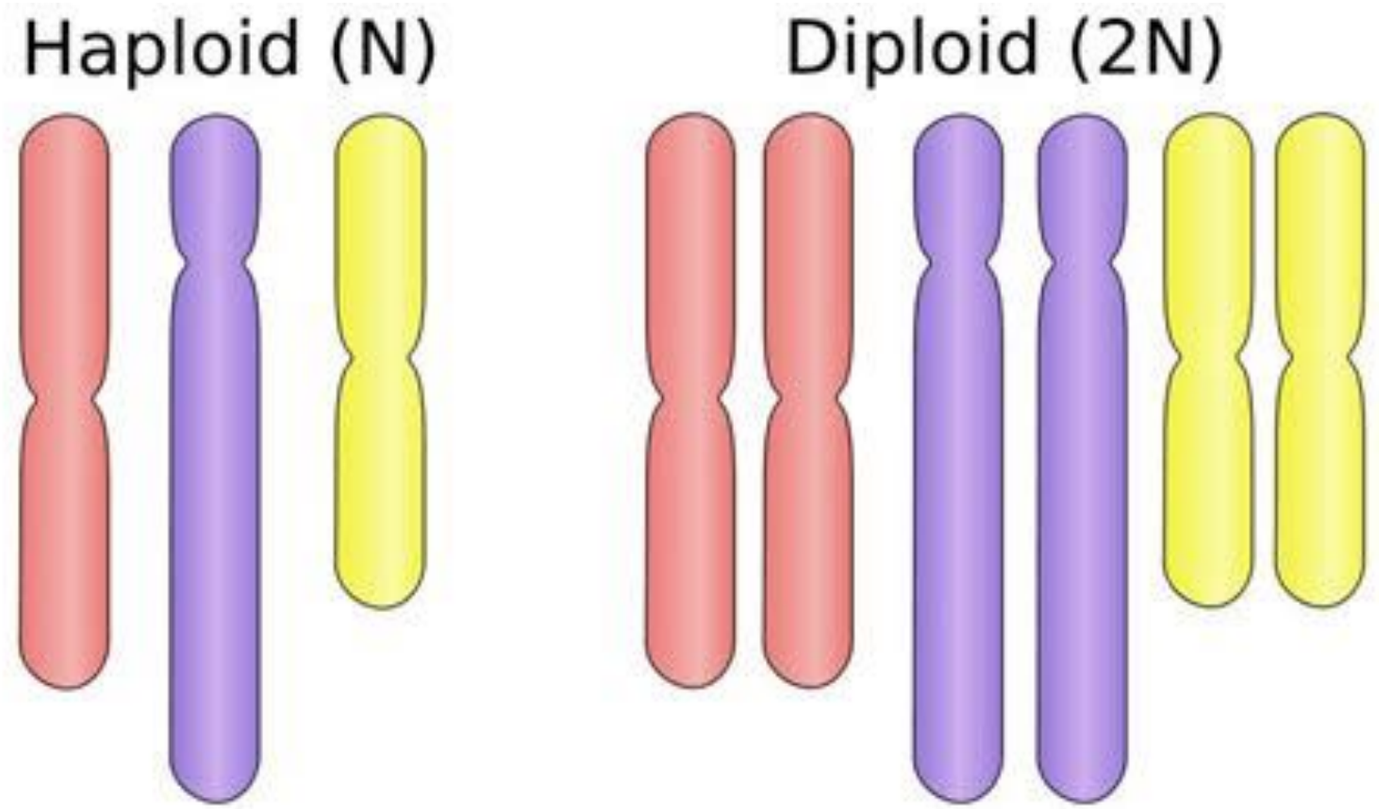
A locus is a fixed position on chromosome. In our lab, a locus is always an exon flanked by intron or UTR region

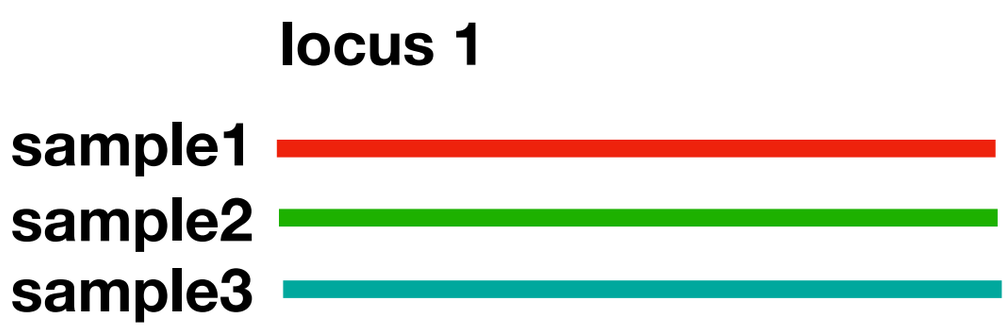
What’s the goal of analysis?

```
@E00492:247:HFMH3CCXY:8:1101:18436:2909 1:N:0:CTGCAATG
CTCAAAAACAACAATAATCTATTATCACAACCAAAGCAACAGCATACATATAATATCAATGCTTCTTTATCTCGATGTGAATAATAAGAAGTCAATGGGAACTTACCTTT
+
FJJJJAJJJJJFJJ<-FJJJFJJJJJJFJJJJFJJFJJJJJJJAJJJJJJJJJJJJFJFJJ<JJJJJJFFAFJF77-A<AJJJJJAJJJJJFFJJJ7<F-7FJFJ-AJF-F
@E00492:247:HFMH3CCXY:8:1101:6725:3401 1:N:0:CTGCAATG
AATTCAGCGACTTGGTTCACACAATCCTCCATCCCTCCTTCCTTTACCACGATGTTTGATGATGCTGCAGGTACAGCTGCTGAAGAGACACGAGCTGTGACCAGTAACTA
+
7JJJAJJJ-FJJJJJFJJJJAJFJFJA-FFJJFJFJJJAAJJF7AJJJJJJJF7A7JJJJFJJJJFJJJAFJJAJFFFJJJ7JFFJJFJJJJJF<-----77---7--77
@E00492:247:HFMH3CCXY:8:1101:8745:3841 1:N:0:CTGCAATG
AAATGTAAGGTCAAACTACAACCTTGTCTATGAAGCCGGTGTGCCATTTGCGTAGCTTCCTGTTTTCGGTGGAGAGTTTCTCAGCAGCAGACTGCAGCTTGGCATTGTCA
+
JJJJJJJJJJJJJJFJJJFFJJJJFJJJJJJJJJJFAFJJJJJ<FJJJJFJJJJFA<FJ-<AAJFJJJF-7A-AJJ-7JA7FJJFJJ7JFJJJJJ7J7777--7-<A<7A
```

Short raw data

There’s only one sequence for each sample





3 steps to recover qualified assemblies from raw data

Data preparation

Assembling

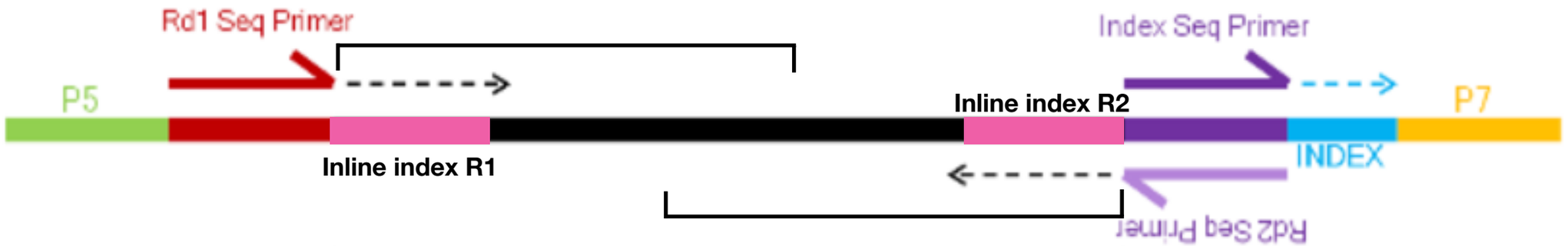
Further processing

Data preparation

Demultiplex reads according to inline index

Trim low quality bases and adaptor

Demultiplex reads according to inline index



Reads we got still includes inline index

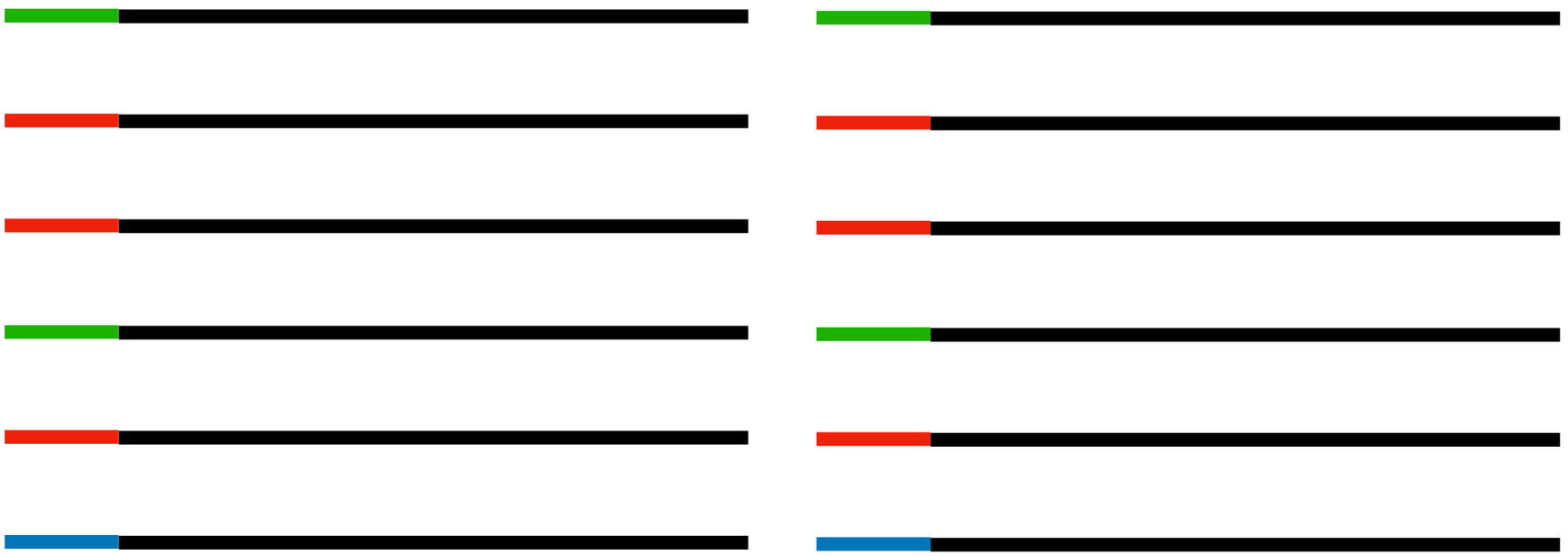
We need to demultiplex reads according to them, then cut them out

assign reads to its sample

How to demultiplex

Reads before demultiplexing

Paired Reads



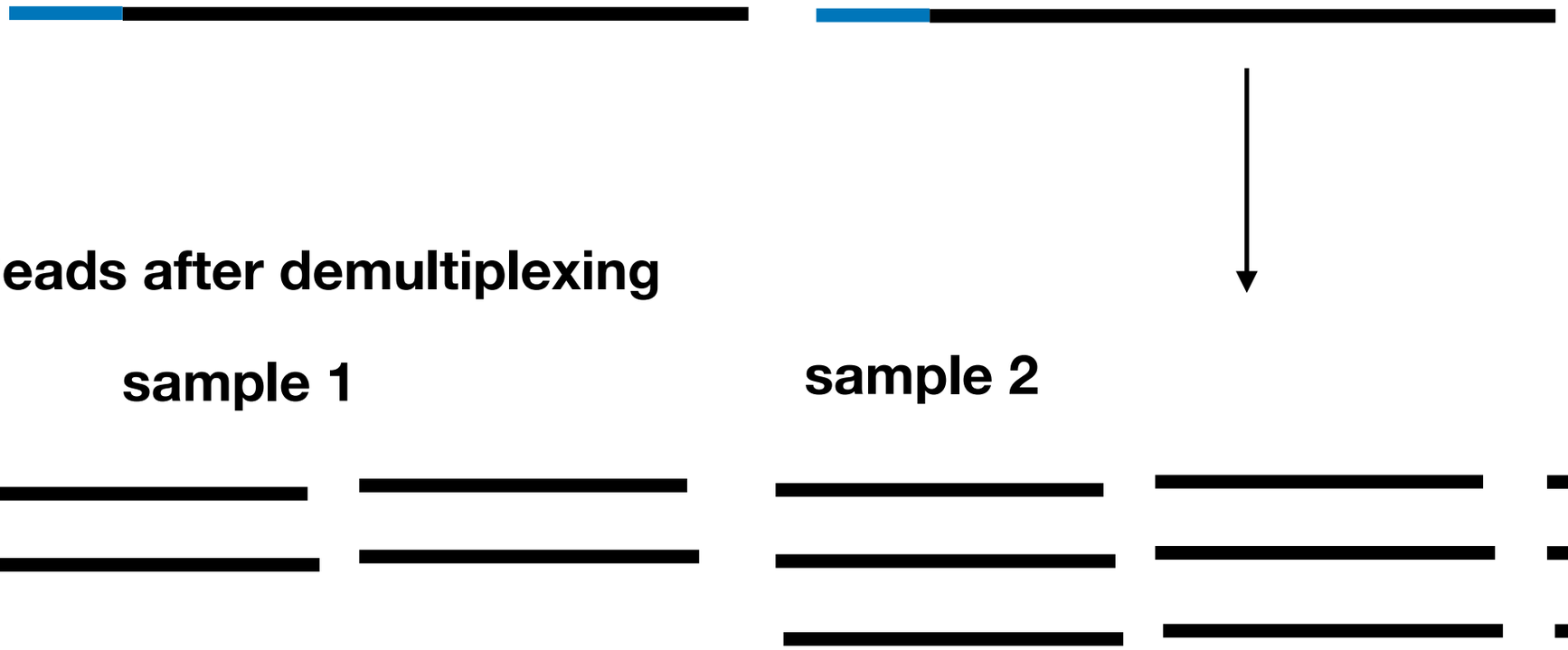
Table

Index “green”: sample 1

Index “red”: sample 2

Index “blue”: sample 3

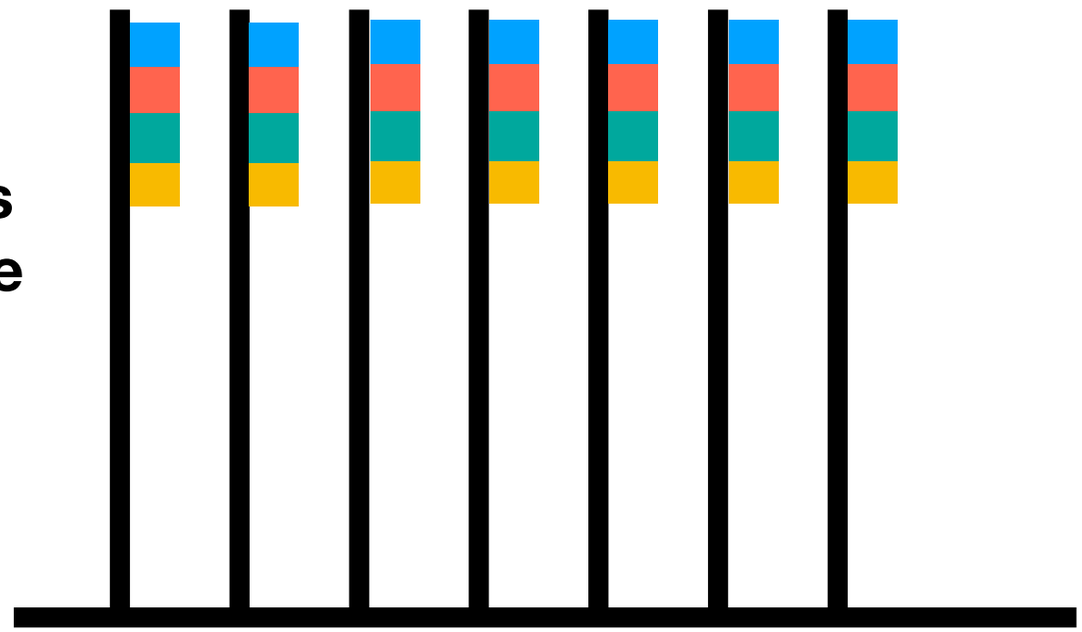
Reads after demultiplexing



Trim low quality bases and adaptor

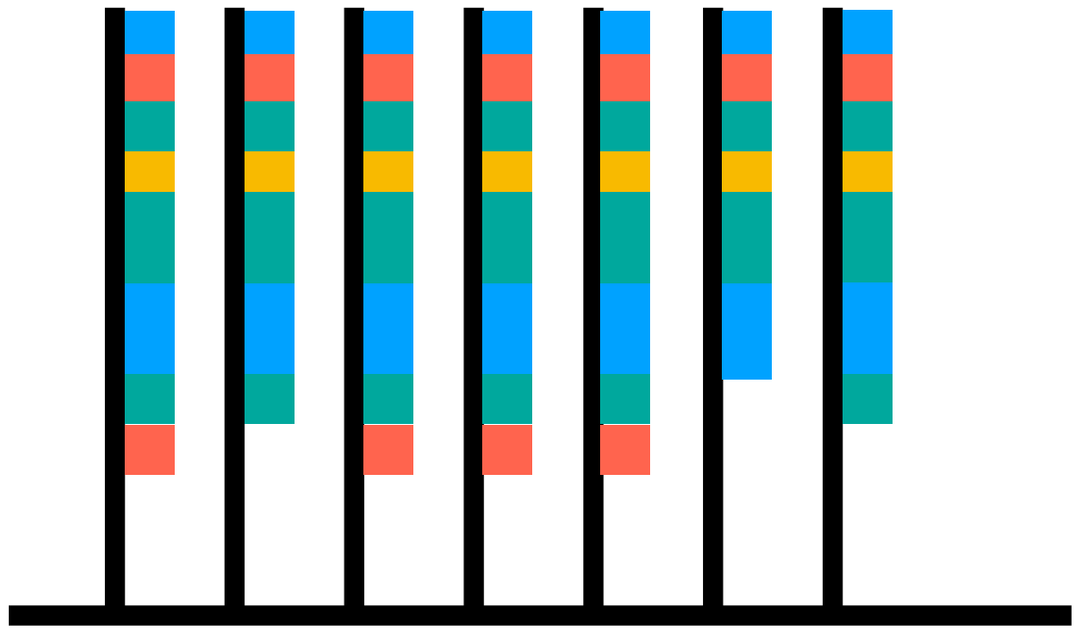
Why we need to trim low quality bases

A cluster of reads amplified from the same molecular



Last base is

Several rounds of extension later

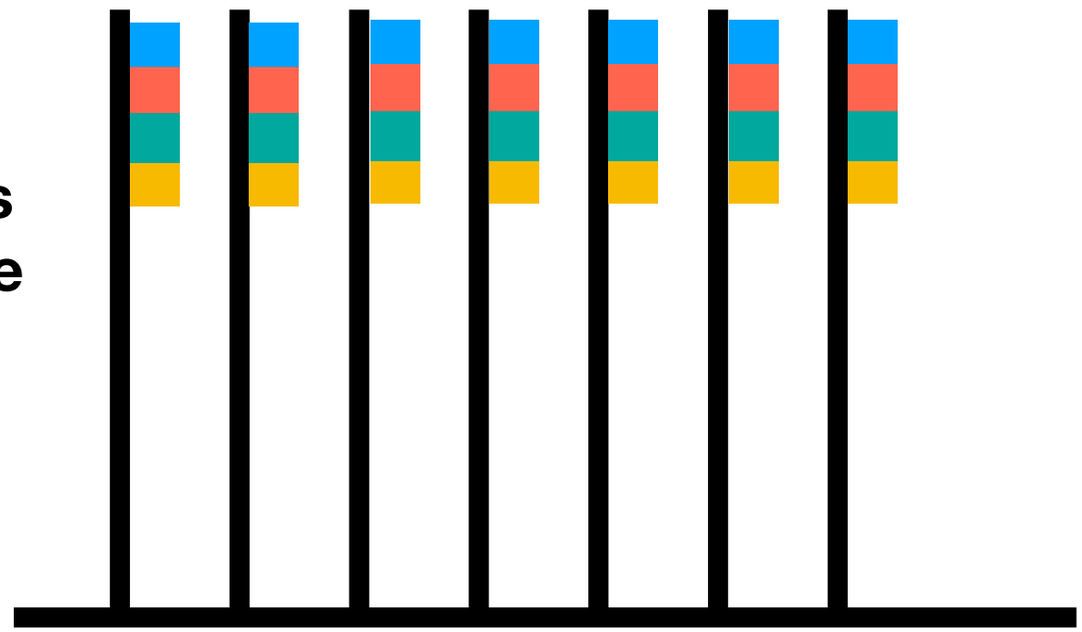


Last base is or ?

Trim low quality bases and adaptor

Why we need to trim low quality bases

A cluster of reads amplified from the same molecular

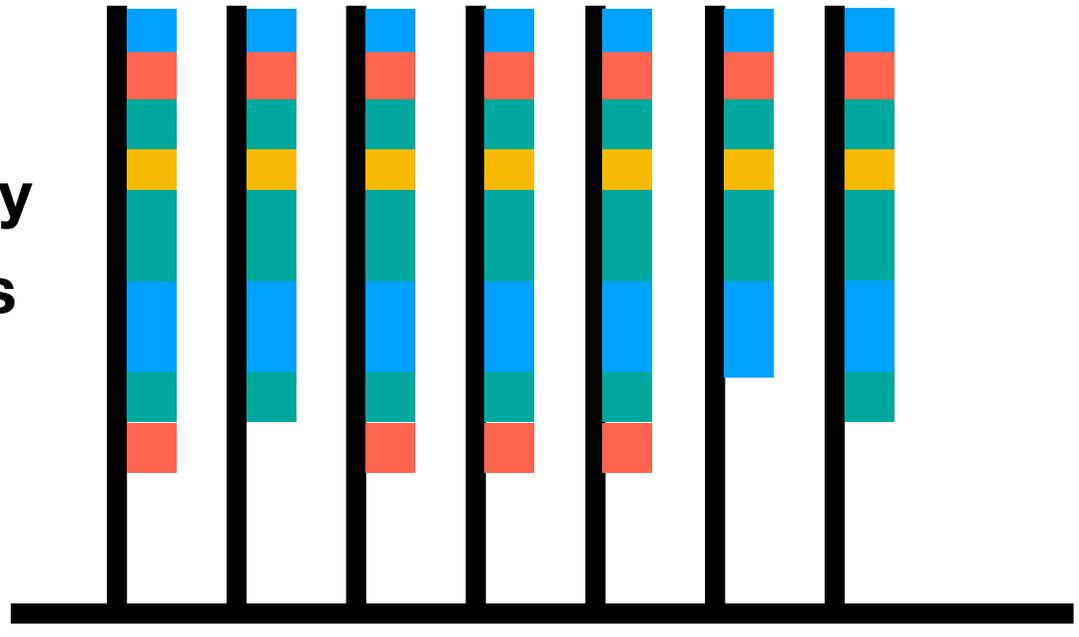


Last base is

High quality bases

Several rounds of extension later

longer the time of sequencing quality of bases becomes lower



Last base is or ?

Low quality bases

How to trim low quality bases

E(quality)>=15

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

E(quality)=30 GOOD

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

E(quality)=29 GOOD

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

E(quality)=25 GOOD

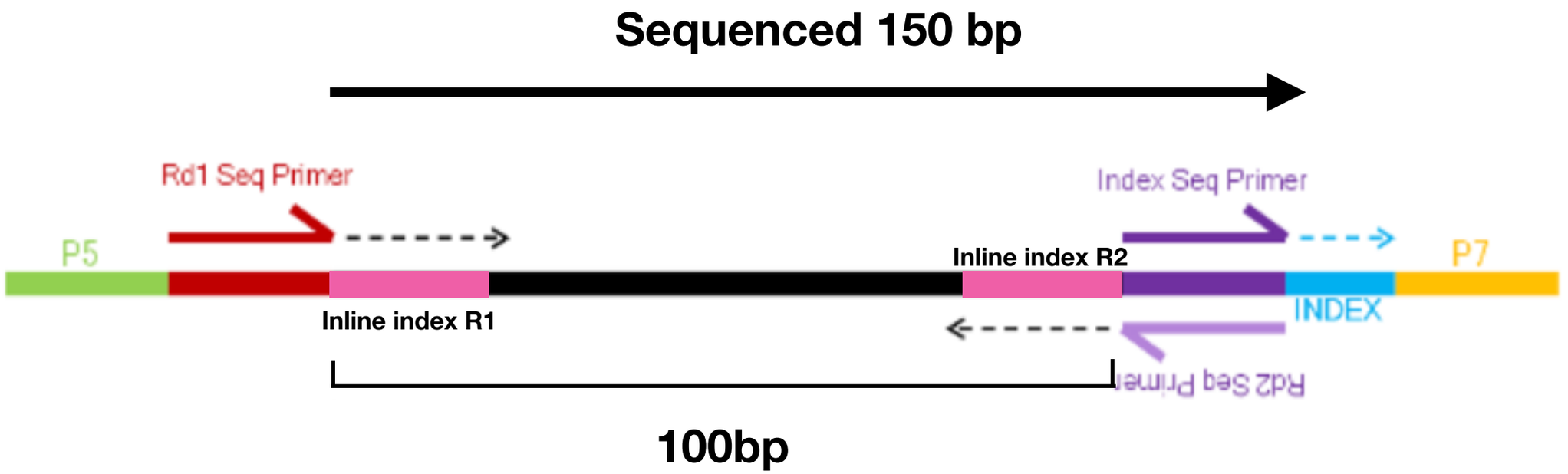
...

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

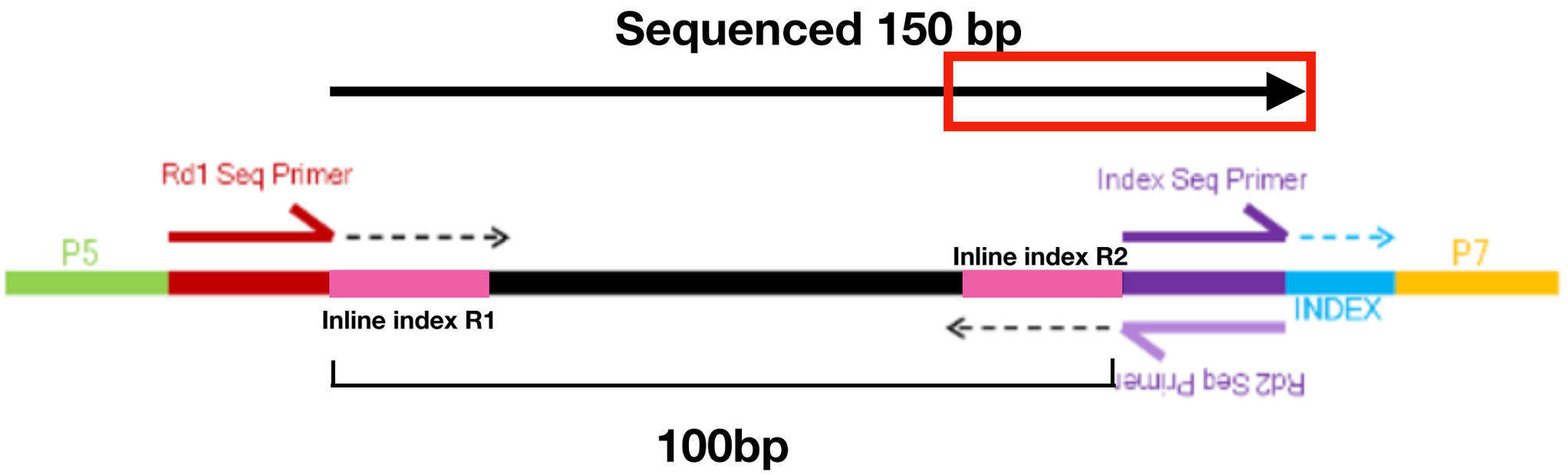
E(quality)=10 BAD!!

ACGGCGTAGGCTGATGATCG

Why we need to trim adaptor?

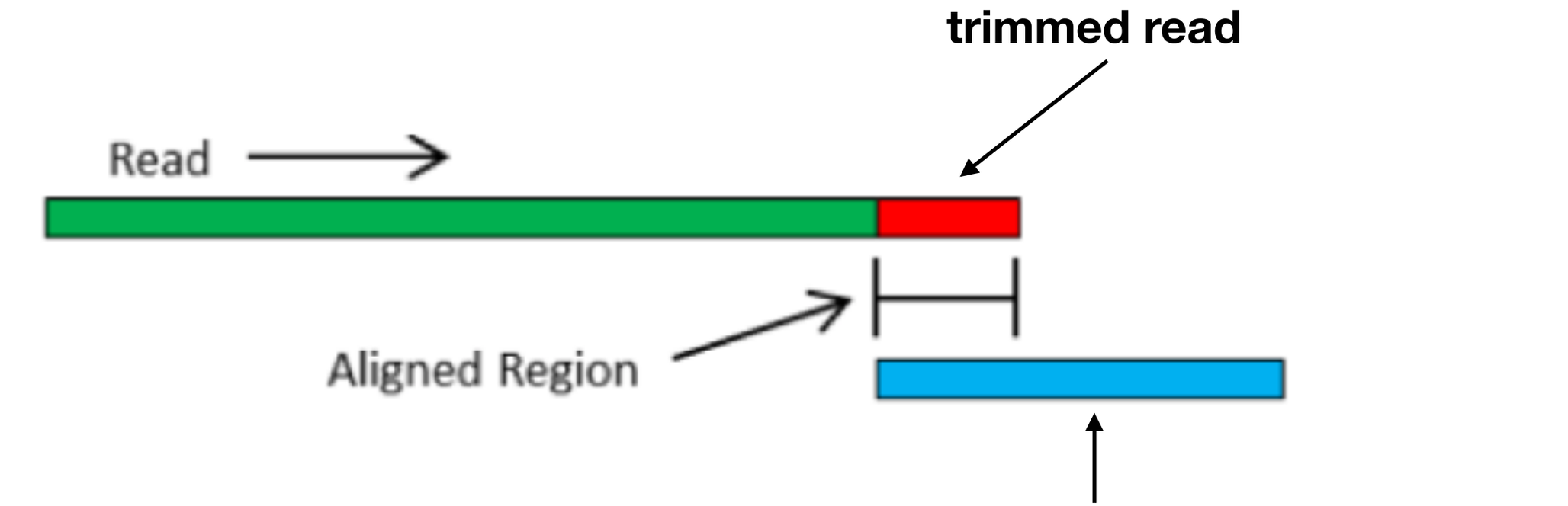


Why we need to trim adaptor?



100bp

How to trim adaptor?



Reads have been cleaned. Let's start assemble !

What’s assemble

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Reads 3: ATCTGGGCATGGCGTTCGATGTCGCTAT

3 reads in a sample

What’s assemble

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Reads 3: ATCTGGGCATGGCGTTCGATGTCGCTAT

Resulting contig:

Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

“Contig” is the sequence assembled from the reads

What’s assemble

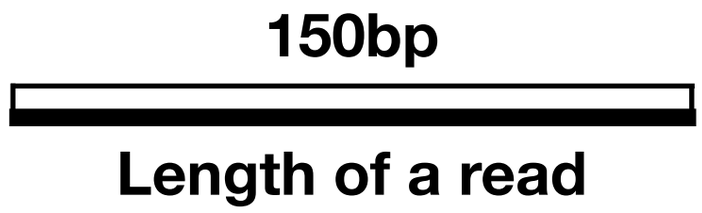
Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

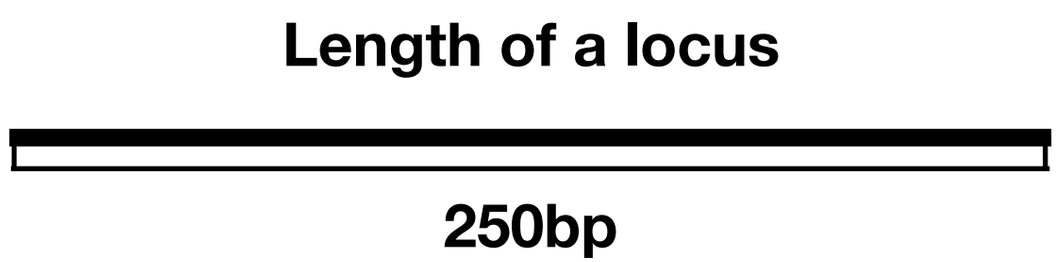
Reads 3: ATCTGGGCATGGCGTTCGATGTCGCTAT

Resulting contig:

CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCATGGCGTTCGATGTCGCTAT

Why raw data need to be assembled before various analysis?





Reads are too short to reach the length of the locus

How raw reads magically become sequences of loci of each sample?

Remove PCR duplicates

Parse reads to loci

Assemble parsed reads

Further assemble

Get orthologue assemblies

Remove PCR duplicates

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

PCR duplicate of Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

PCR duplicate of Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Resulting contig:

Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

Remove PCR duplicates

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

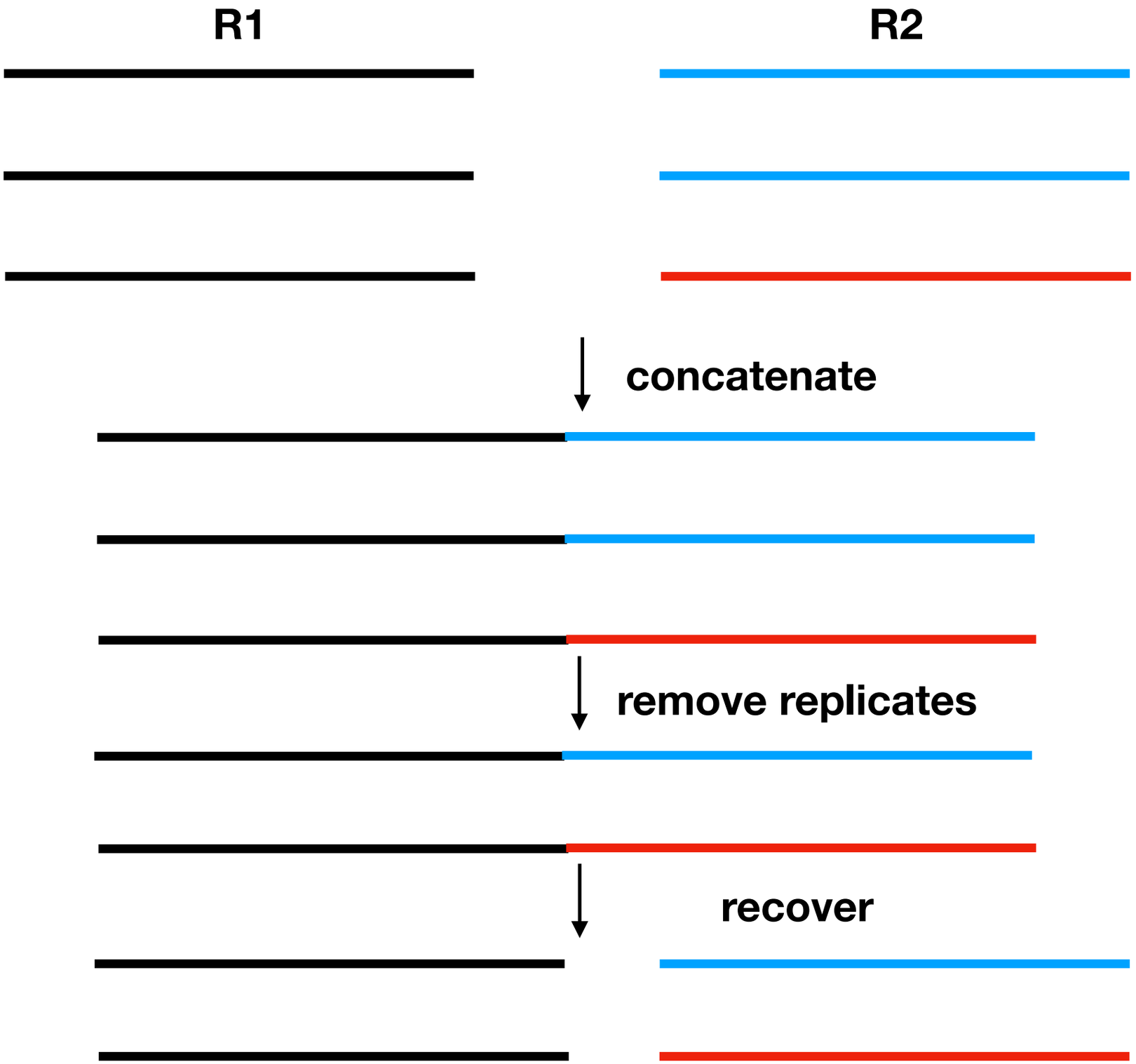
Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Resulting contig:

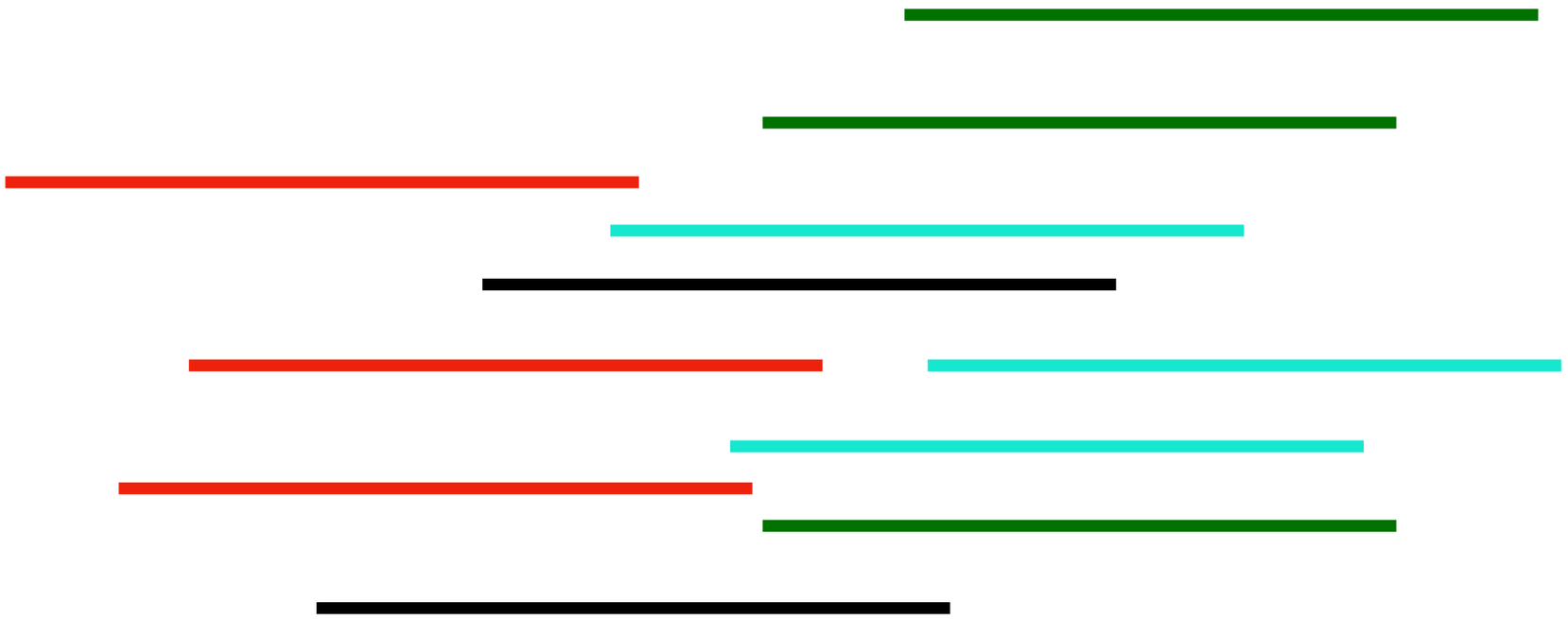
Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

PCR duplicates are redundant for assembly

Remove PCR duplicates



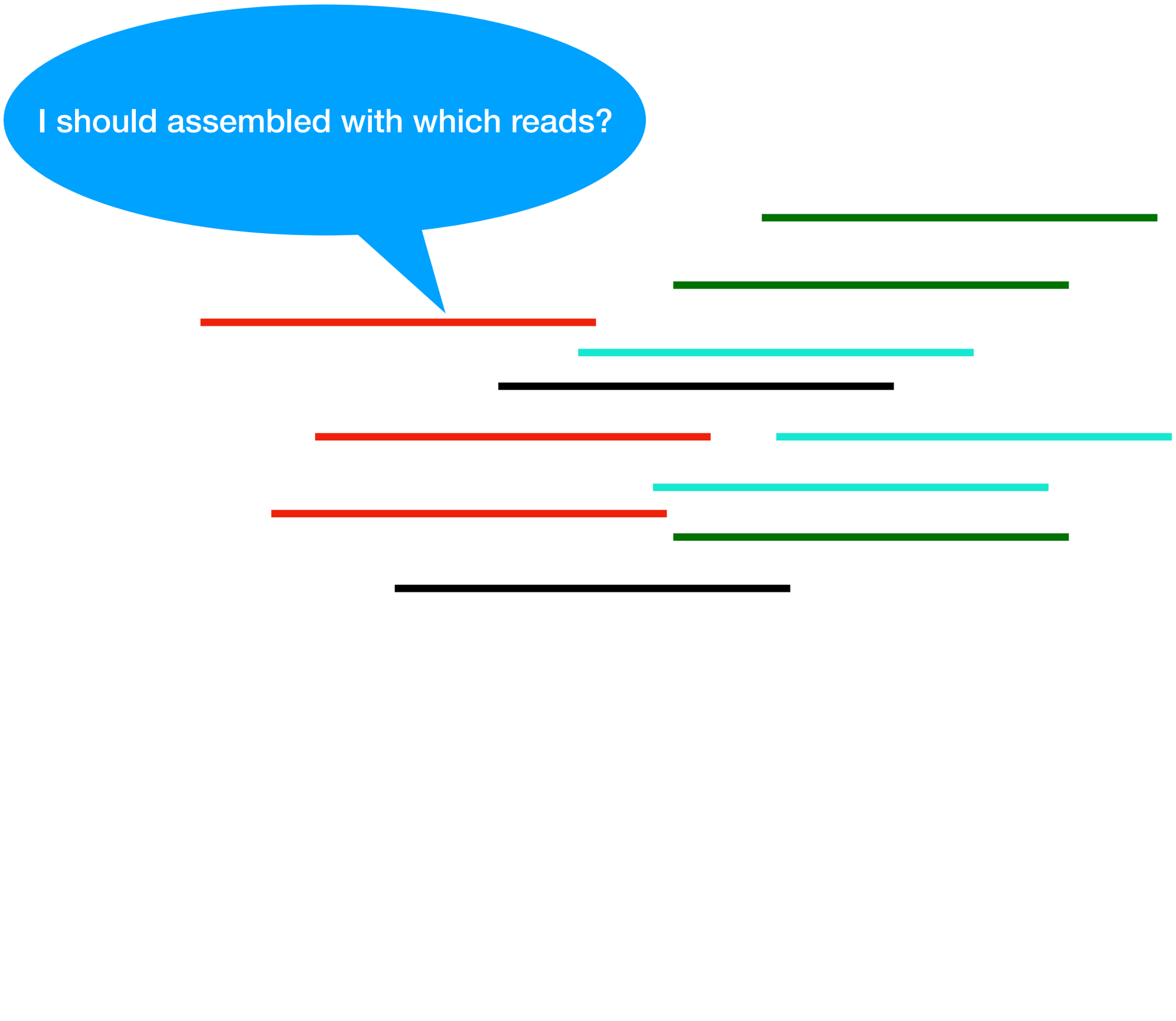
Parse reads to loci



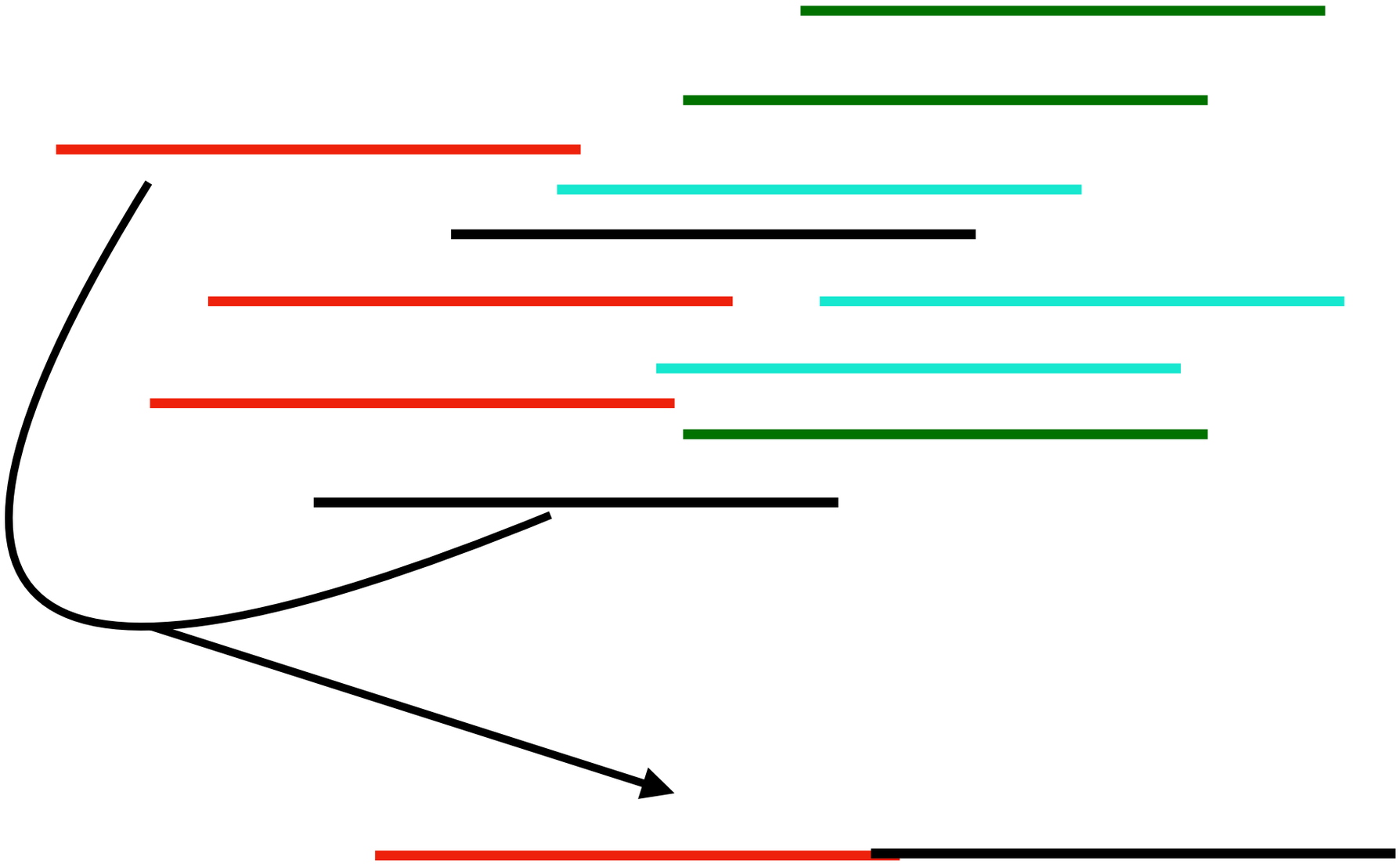
Mixed short reads from lots of loci

Reads of the same color indicate they come from the same loci

Parse reads to loci

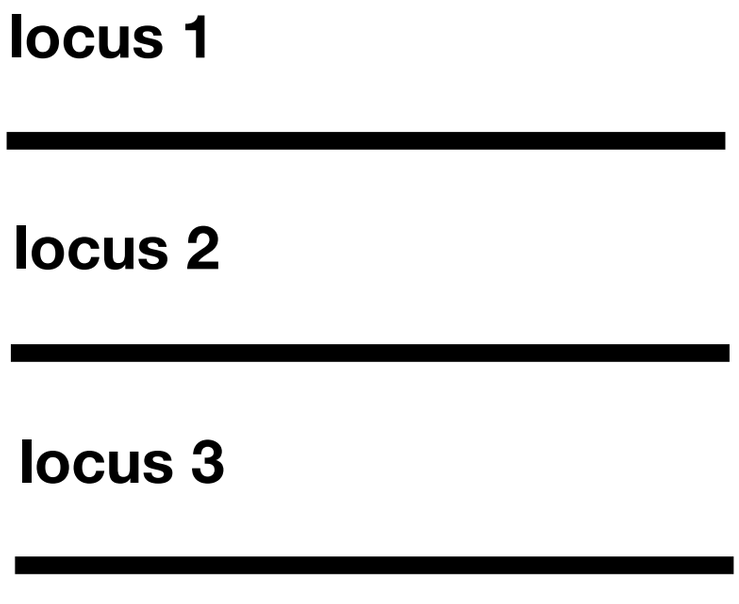


Parse reads to loci



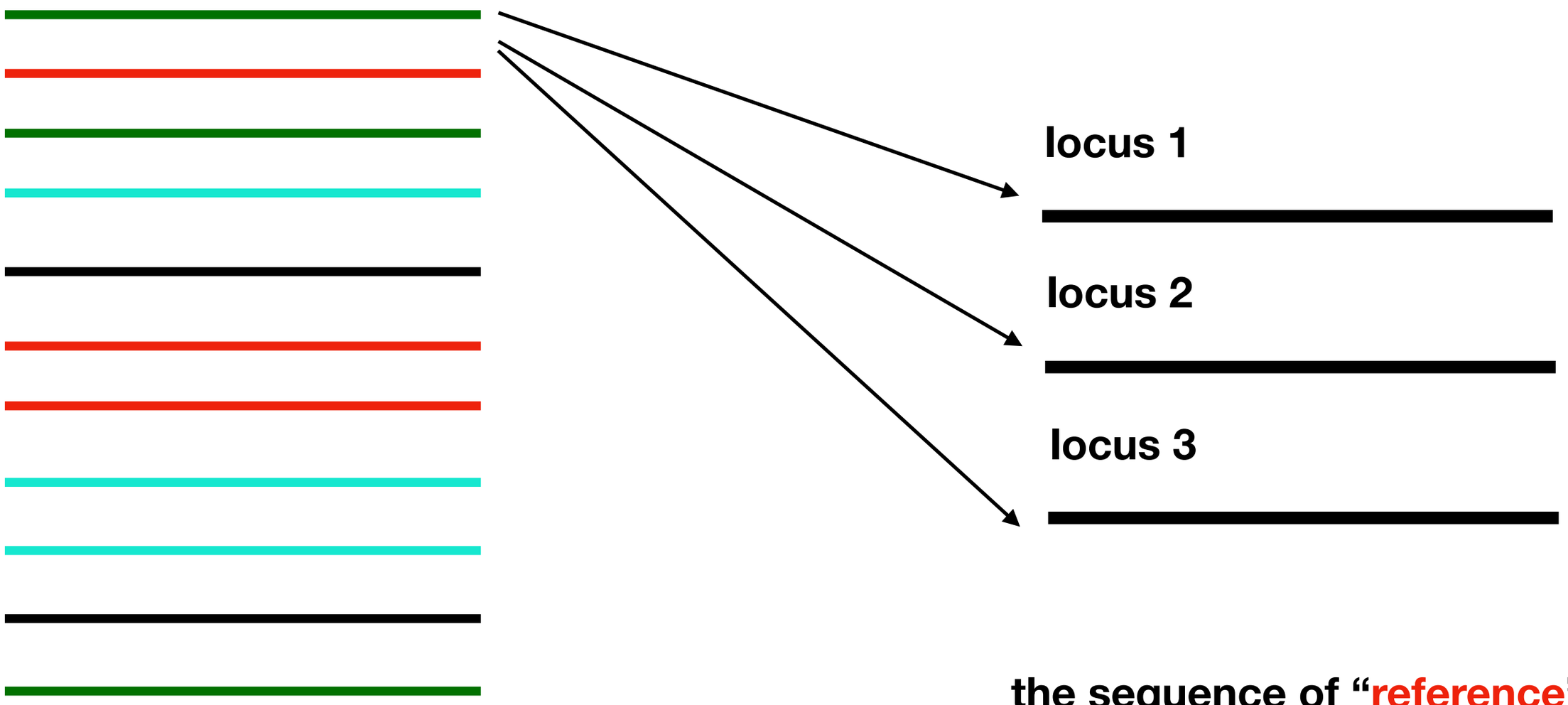
If reads from different loci assembled together,
the resulting contig will be “chimera”

Parse reads to loci



Remember me? I’m the sequence of “reference”, used to design the baits

Parse reads to loci

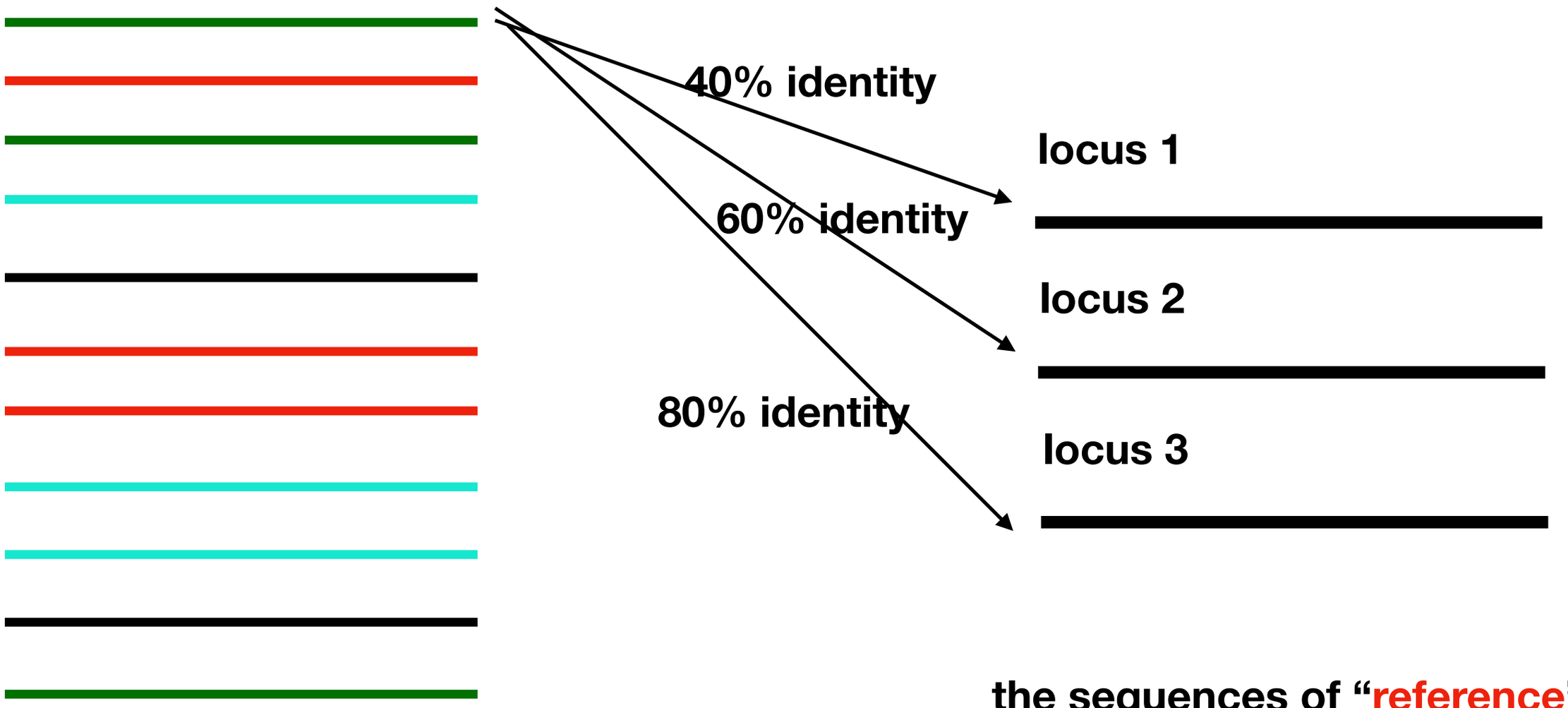


the sequence of “reference”

Mixed short reads from several loci

Compare reads with each locus

Parse reads to loci

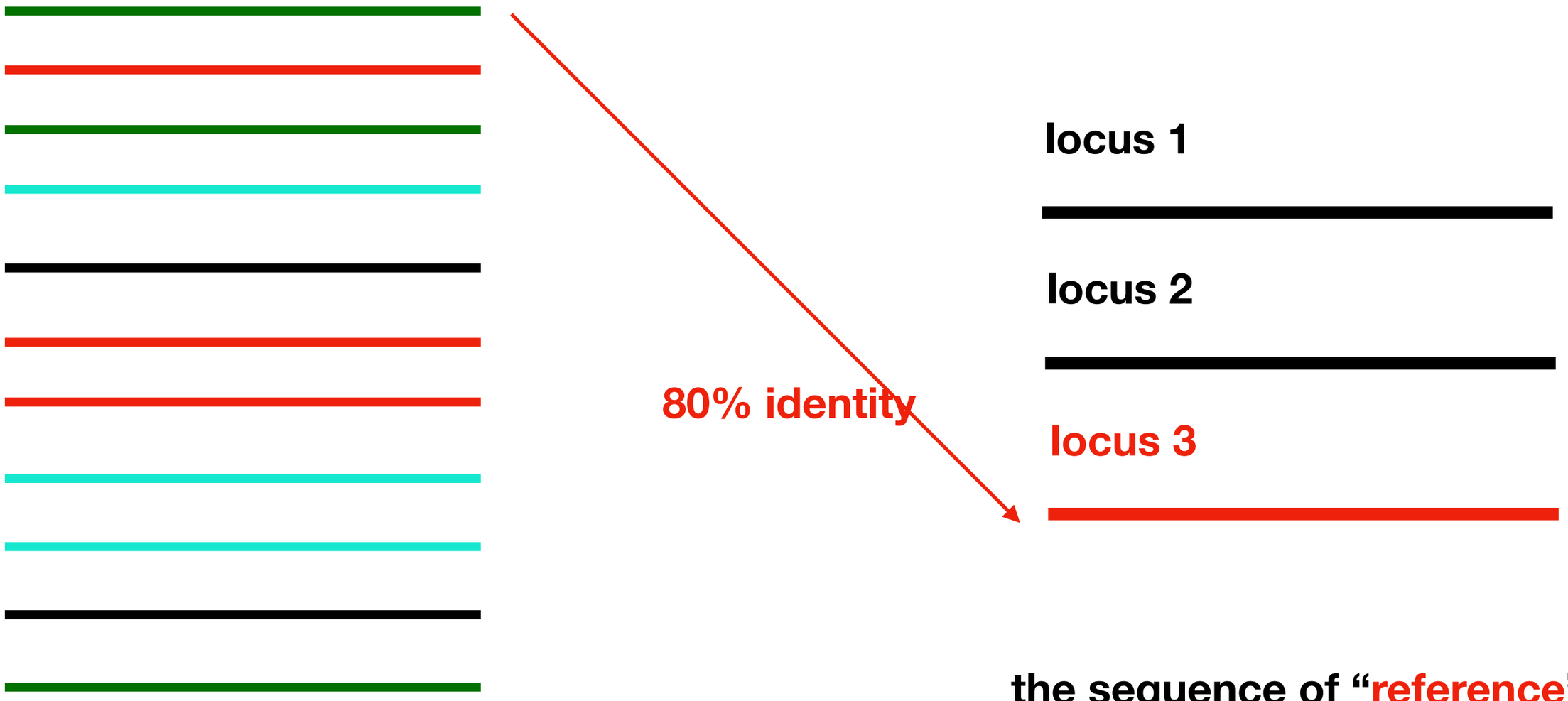


the sequences of “reference”

Mixed short reads from several loci

Reads got different identity with each locus

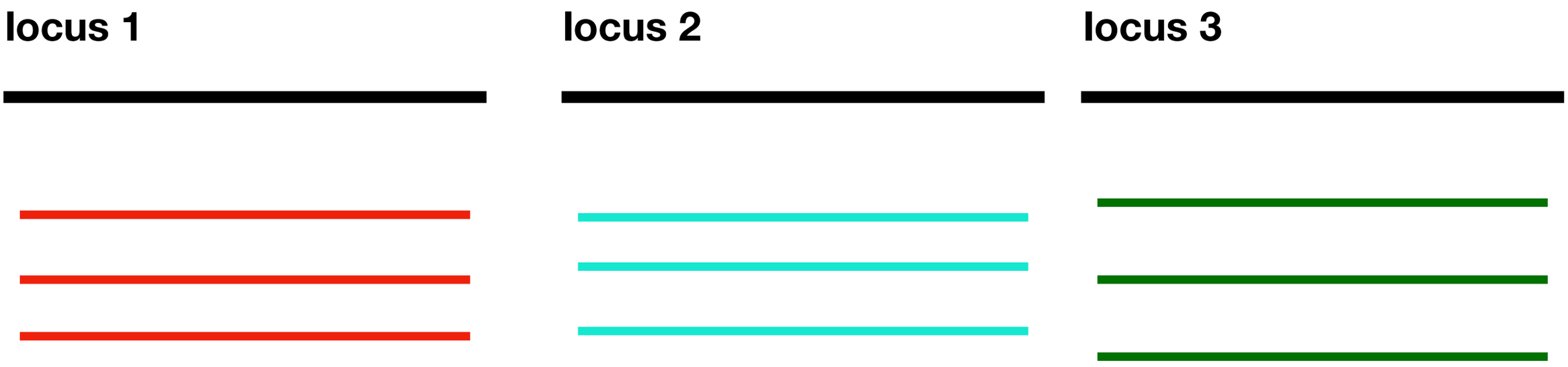
Parse reads to loci



Mixed short reads from several loci

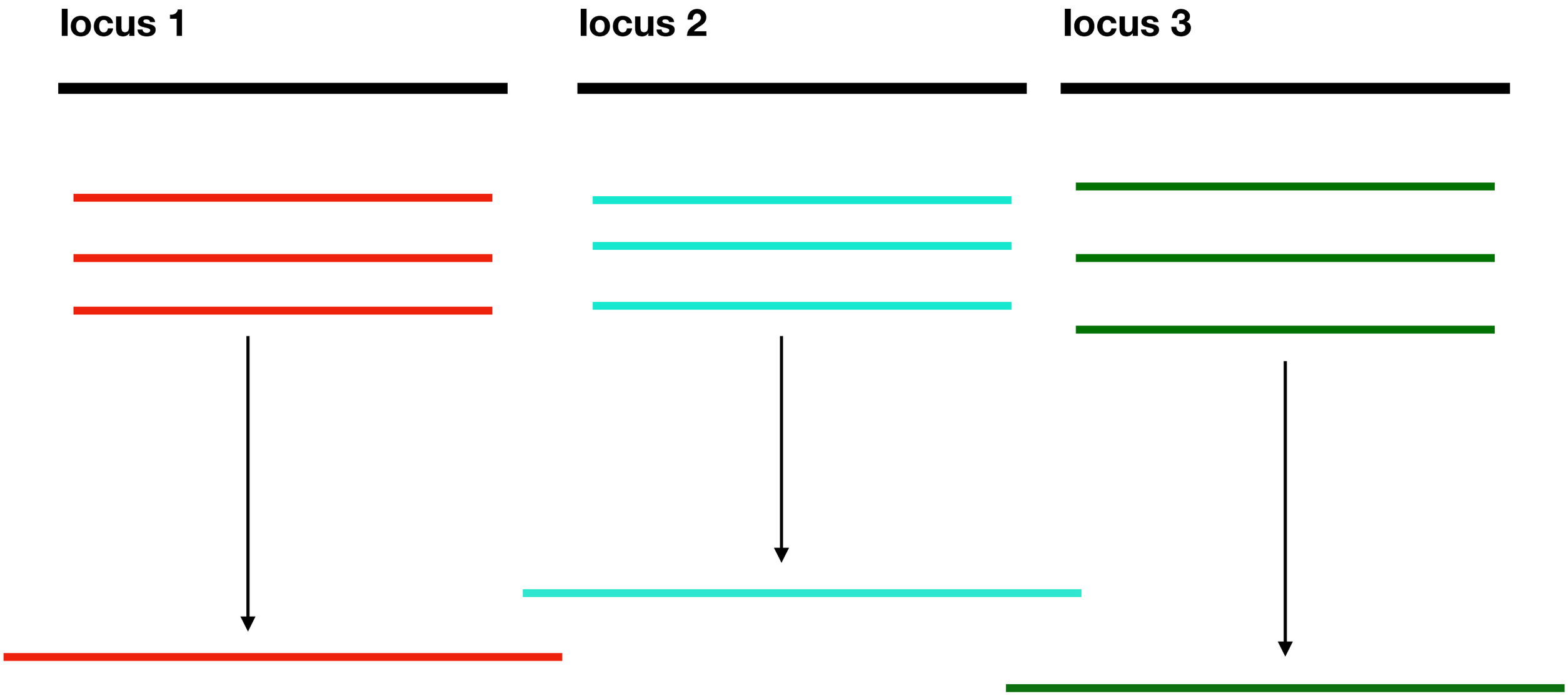
Select reads with highest identity

Parse reads to loci



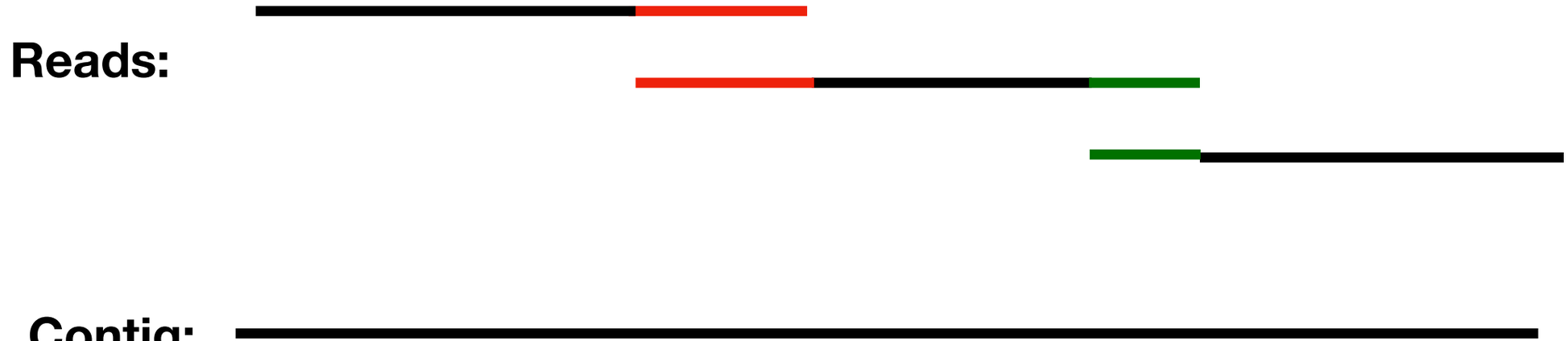
Most of reads are assigned to different loci.
Some reads from nowhere are still unassigned

Assemble parsed reads

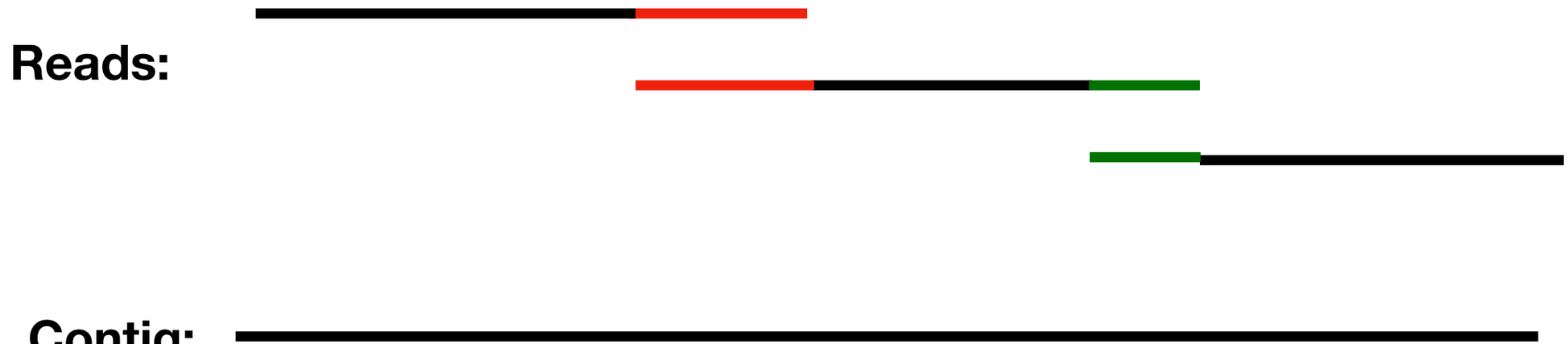


Assemble parsed reads into longer contigs

Assemble parsed reads



Assemble parsed reads



In real case, question is not that easy. We always have loci assigned with more than 2,000 reads

Assemble parsed reads

Find overlaps among all reads

Build a graph recording overlaps among all reads

Traverse through the graph to get contigs

Assemble parsed reads

Find overlaps among all reads

Align the reads

K-mer

FM-index

Assemble parsed reads

Find overlaps among all reads

Align the reads

K-mer

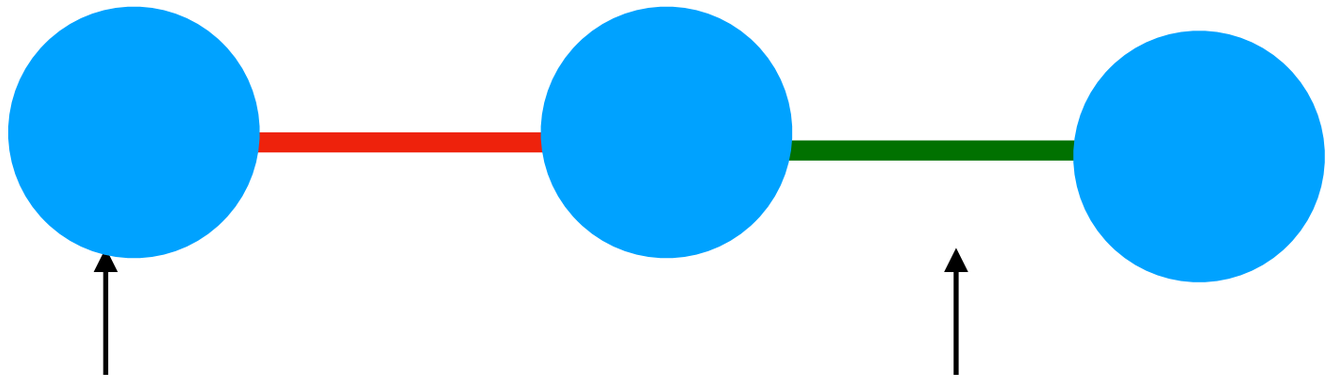
FM-index

Only considerable length of overlap between reads will be kept (25 bp), to guarantee the low probability of accidentally overlap occurs

Assemble parsed reads

Build a graph recording overlaps among all reads

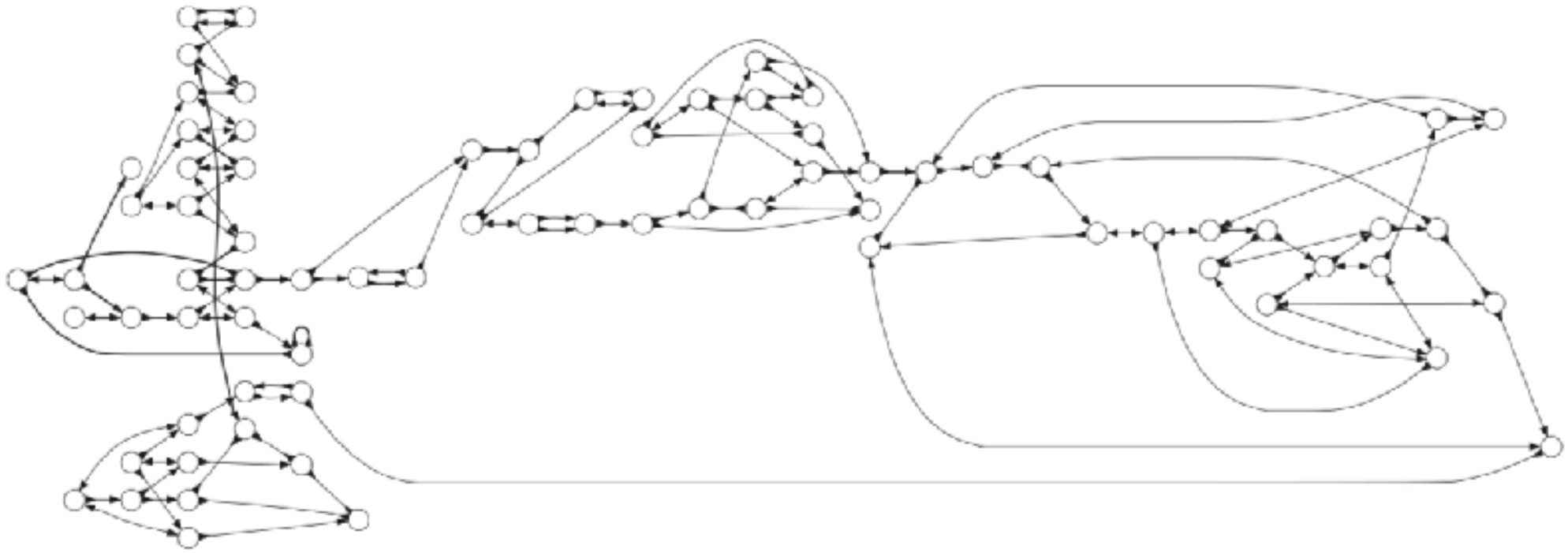
Reads:



Circle represent the read we call it “node” or “vertex”

Line represents the connection between reads called “edge”

Assemble parsed reads



*G*raph of *Campylobacter jejuni*

Assemble parsed reads

Traverse through the graph to get contigs

For each locus, there's only one sequence

Traverse through the all nodes in the graph and each node only pass once

Assemble parsed reads

Traverse through the graph to get contigs

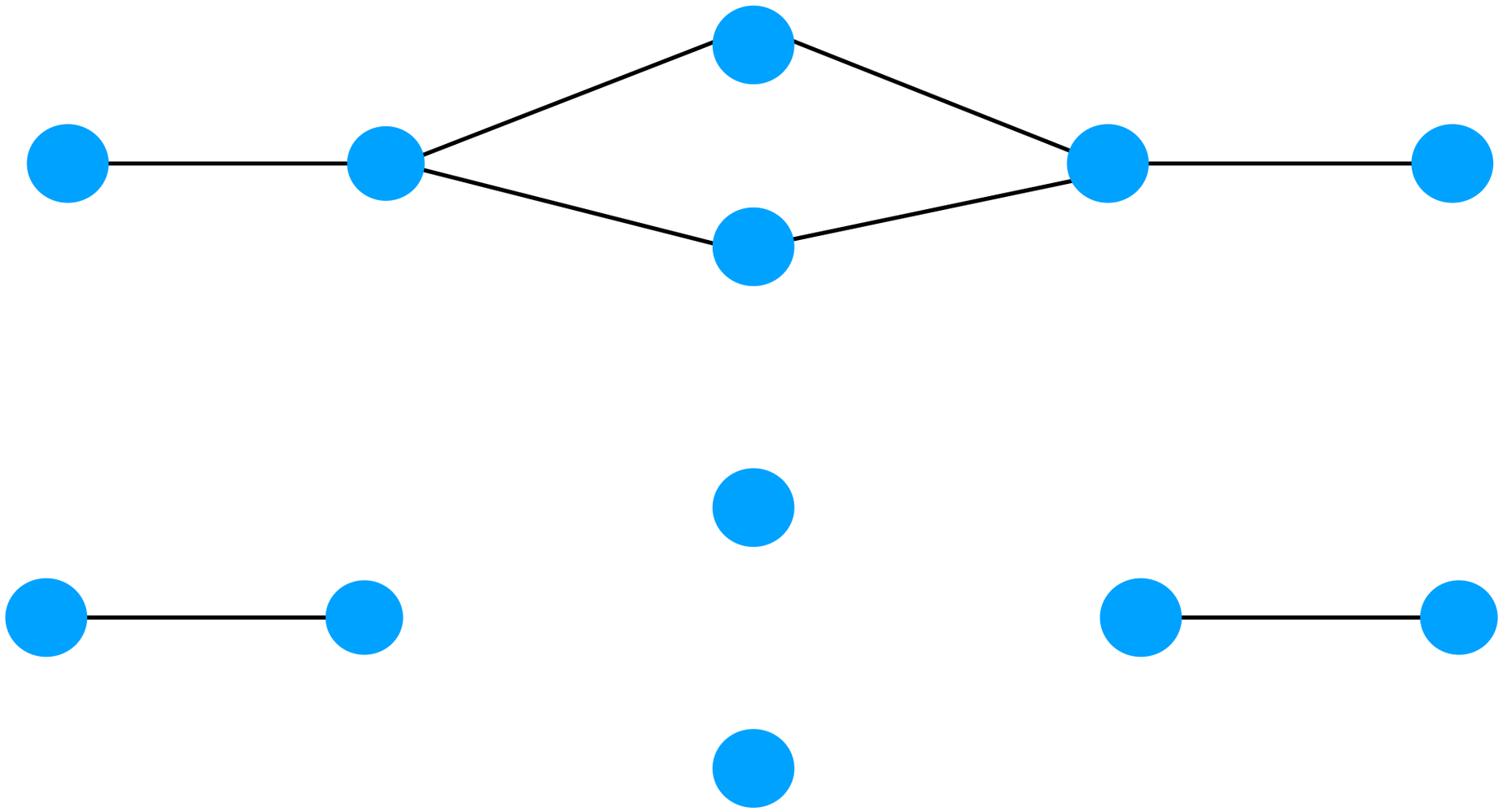
For each locus, there's only one sequence

Traverse through the all nodes in the graph and each node only pass once

But this assumption is hard to fulfill

Assemble parsed reads

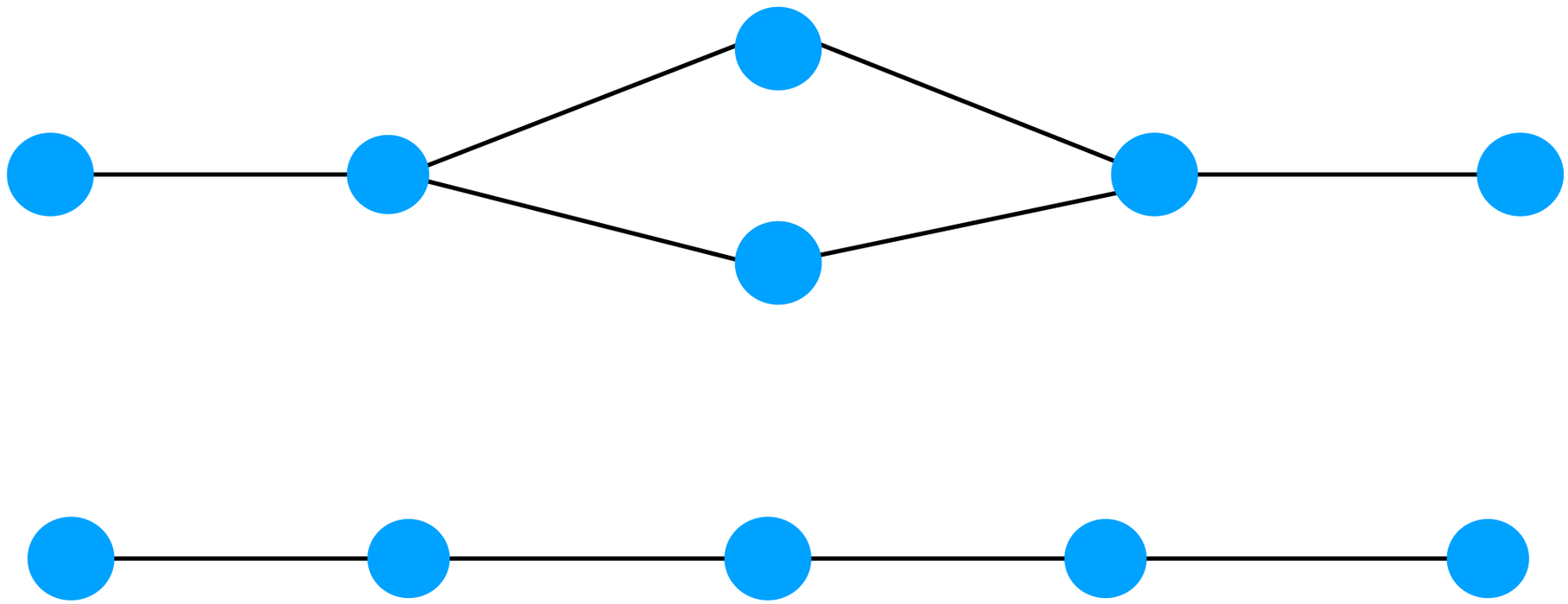
Bubble:



Break the graph into several sub-graph

Assemble parsed reads

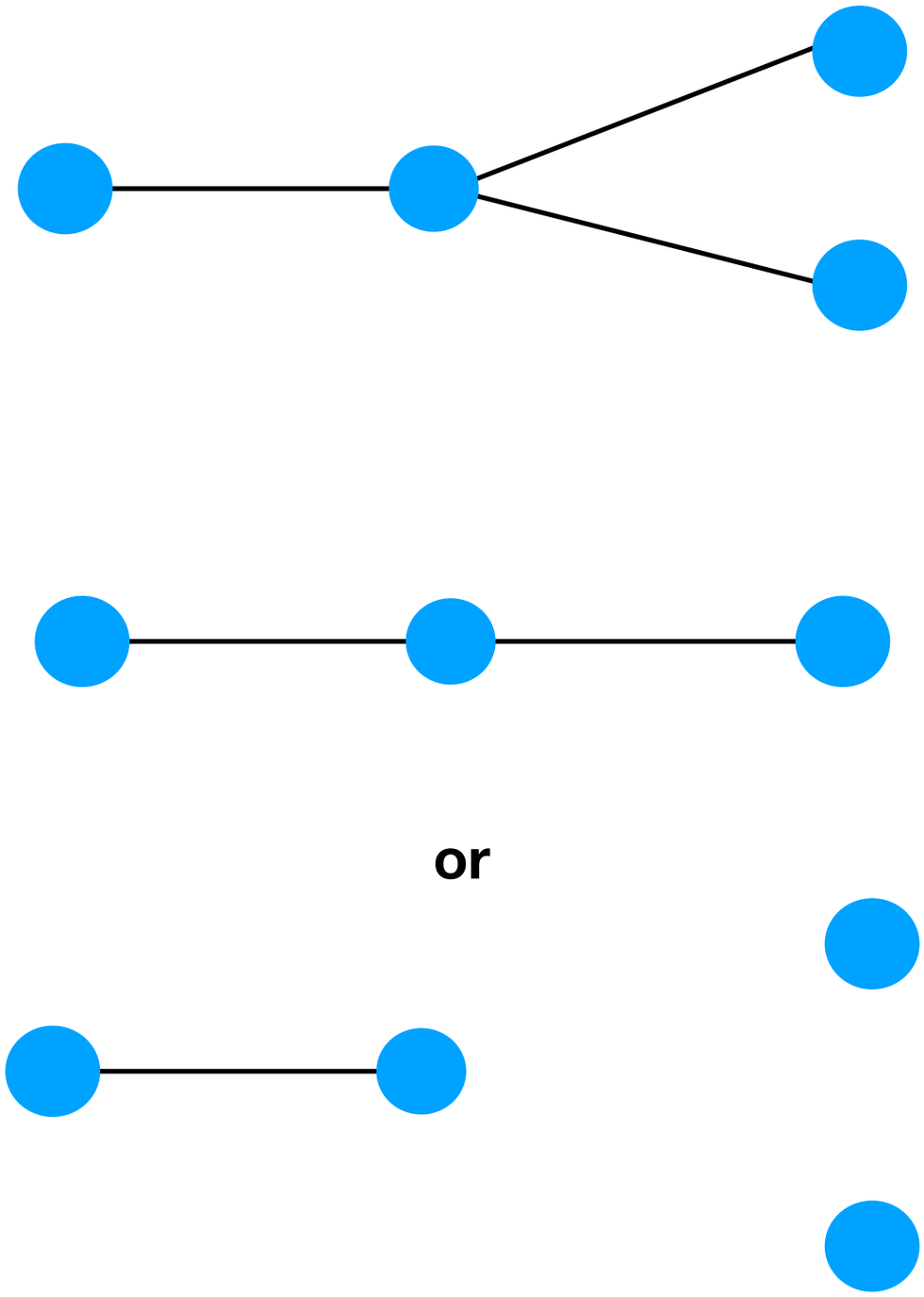
Bubble:



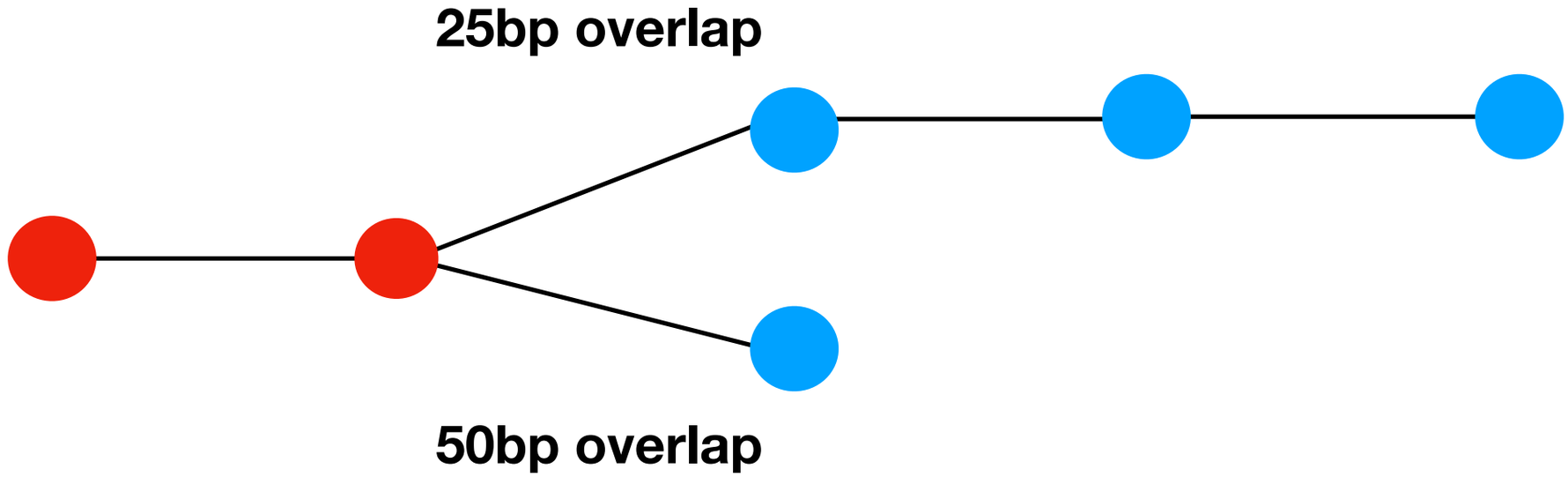
Discard one of the path

Assemble parsed reads

Tip:



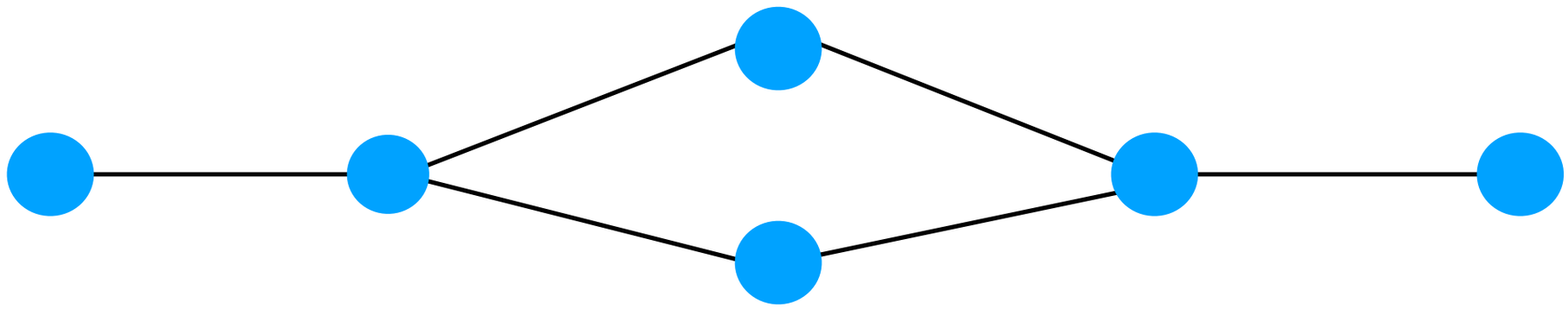
Why we need graph



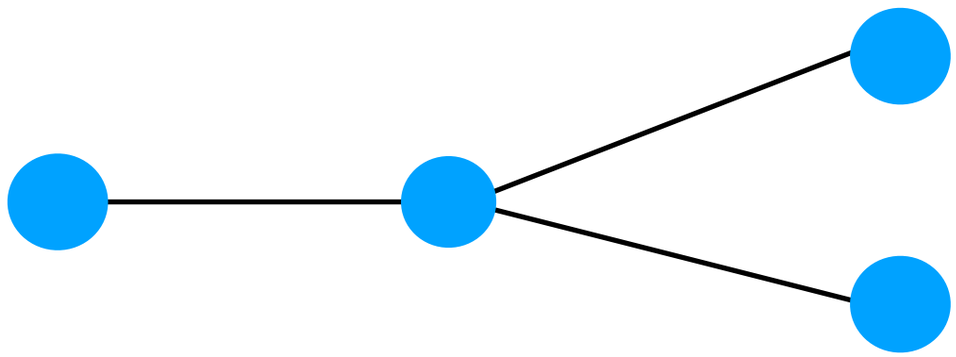
Which way you should choose

Further assemble

Bubble:



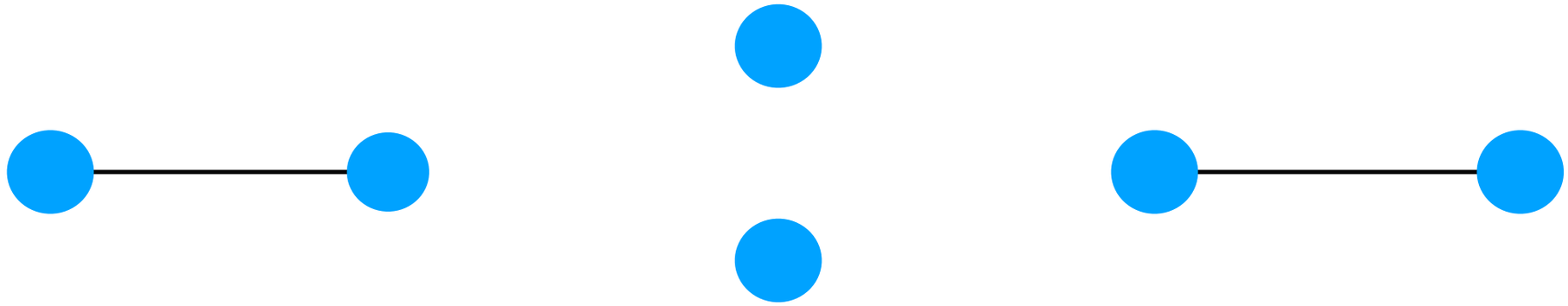
Tip:



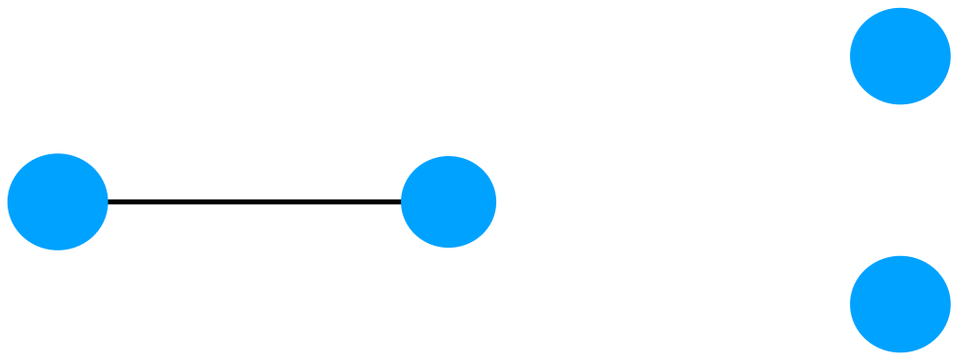
If two path is too diverged (>= 95% identity). The path will be split into several contigs

Further assemble

Bubble:

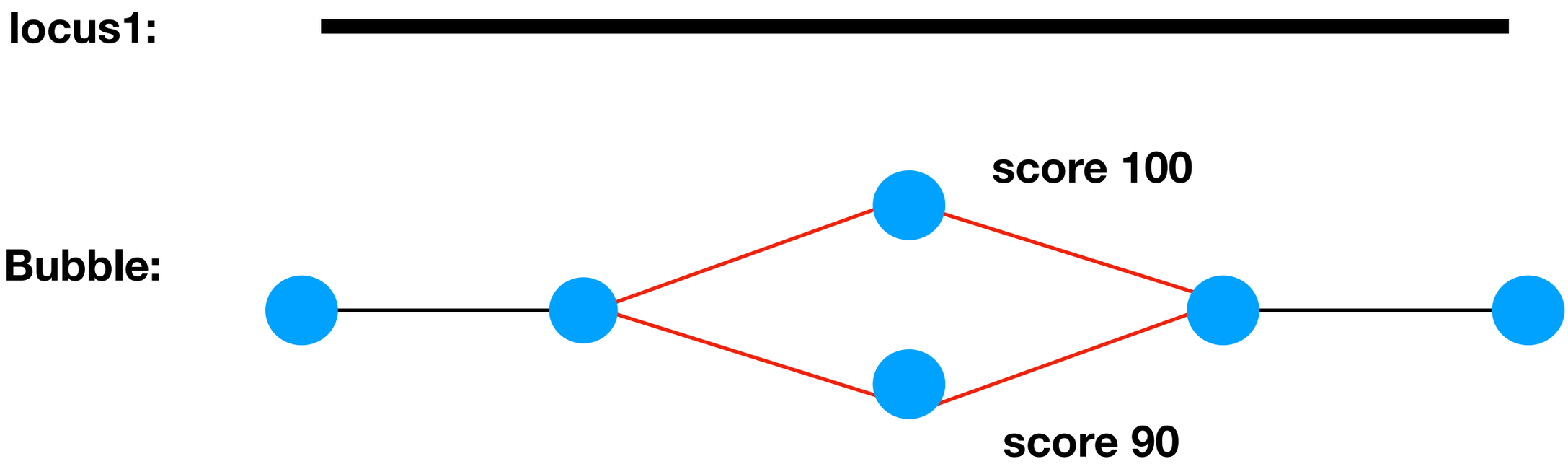


Tip:



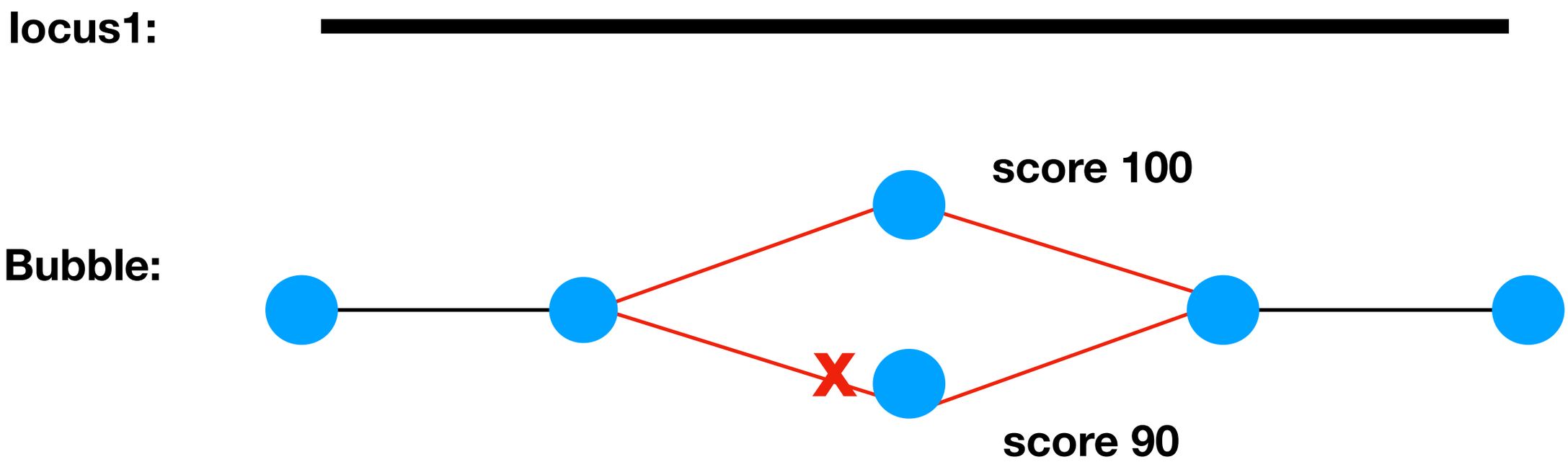
If two path is too diverged (>= 95% identity). The path will be split into several contigs

Further assemble



Each contig will be aligned to reference. Graph will be reconstructed. The alignment score of each path will be calculated.

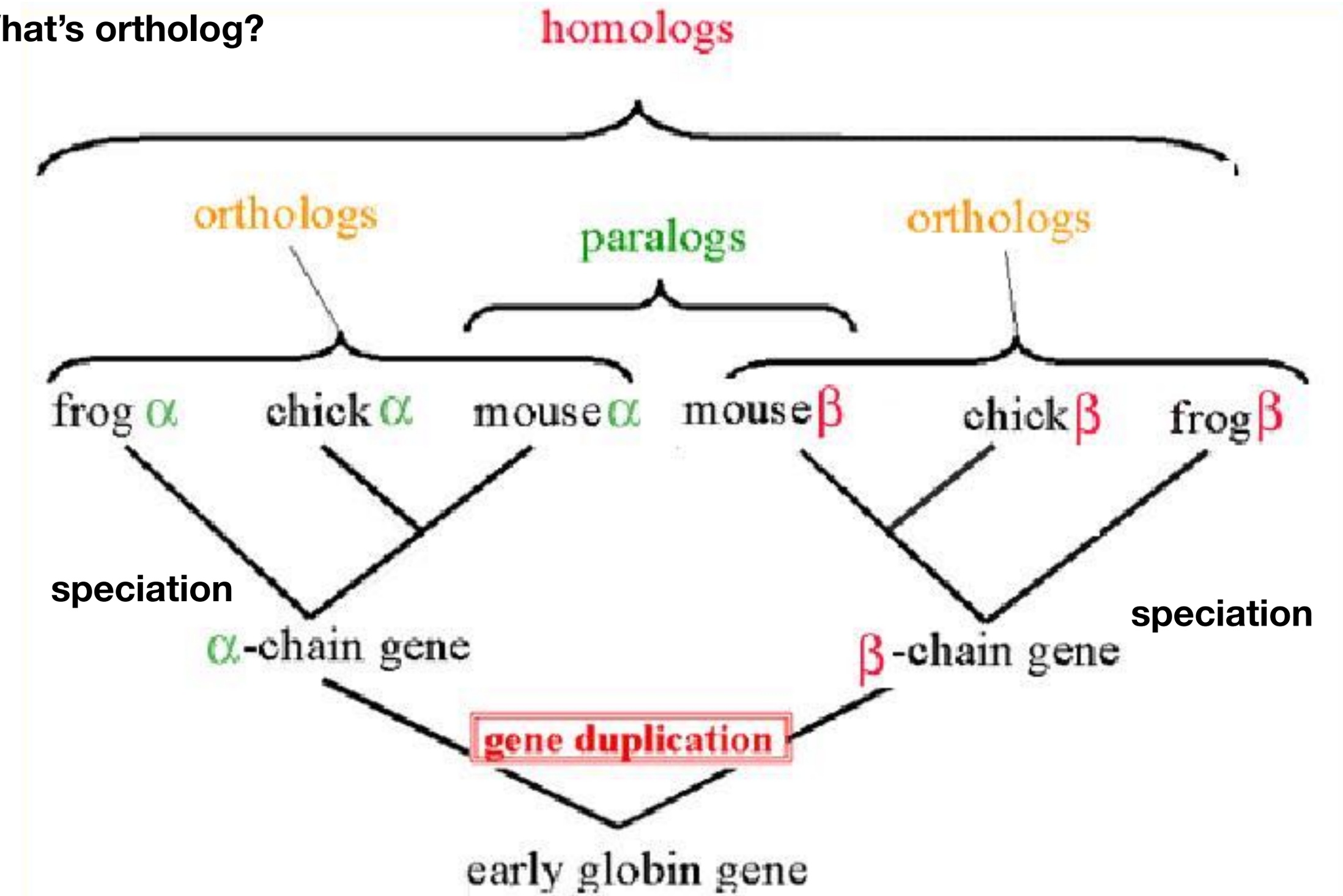
Further assemble



Keep the path with higher score.

Get orthologous assemblies

What's ortholog?



Get orthologous assemblies

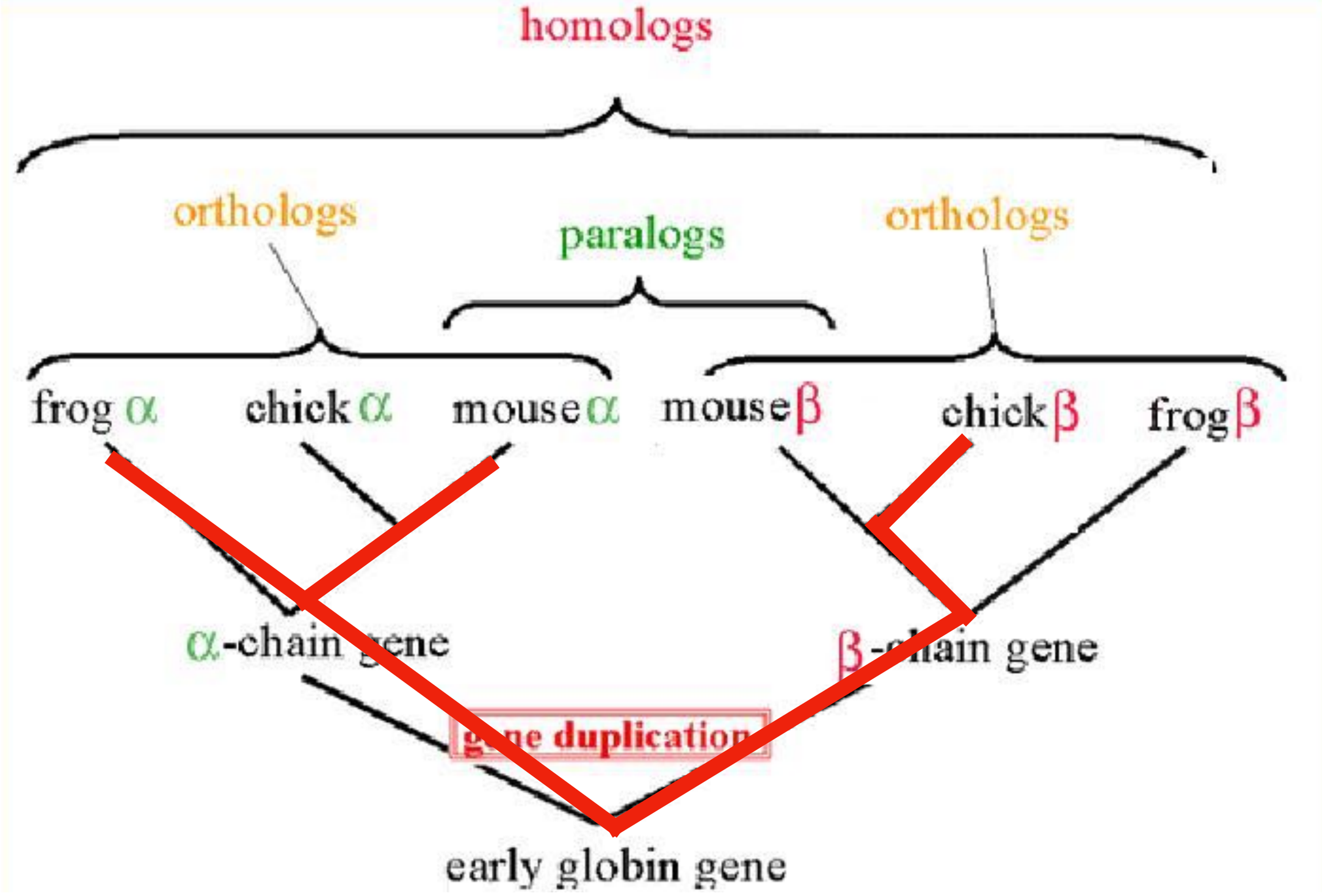
Why we need it?

Our final aim is to reveal evolutionary history among our enriched species

Orthologues genes are derived from “speciation event”. So, the evolutionary history of these genes are identical with the evolutionary history of species

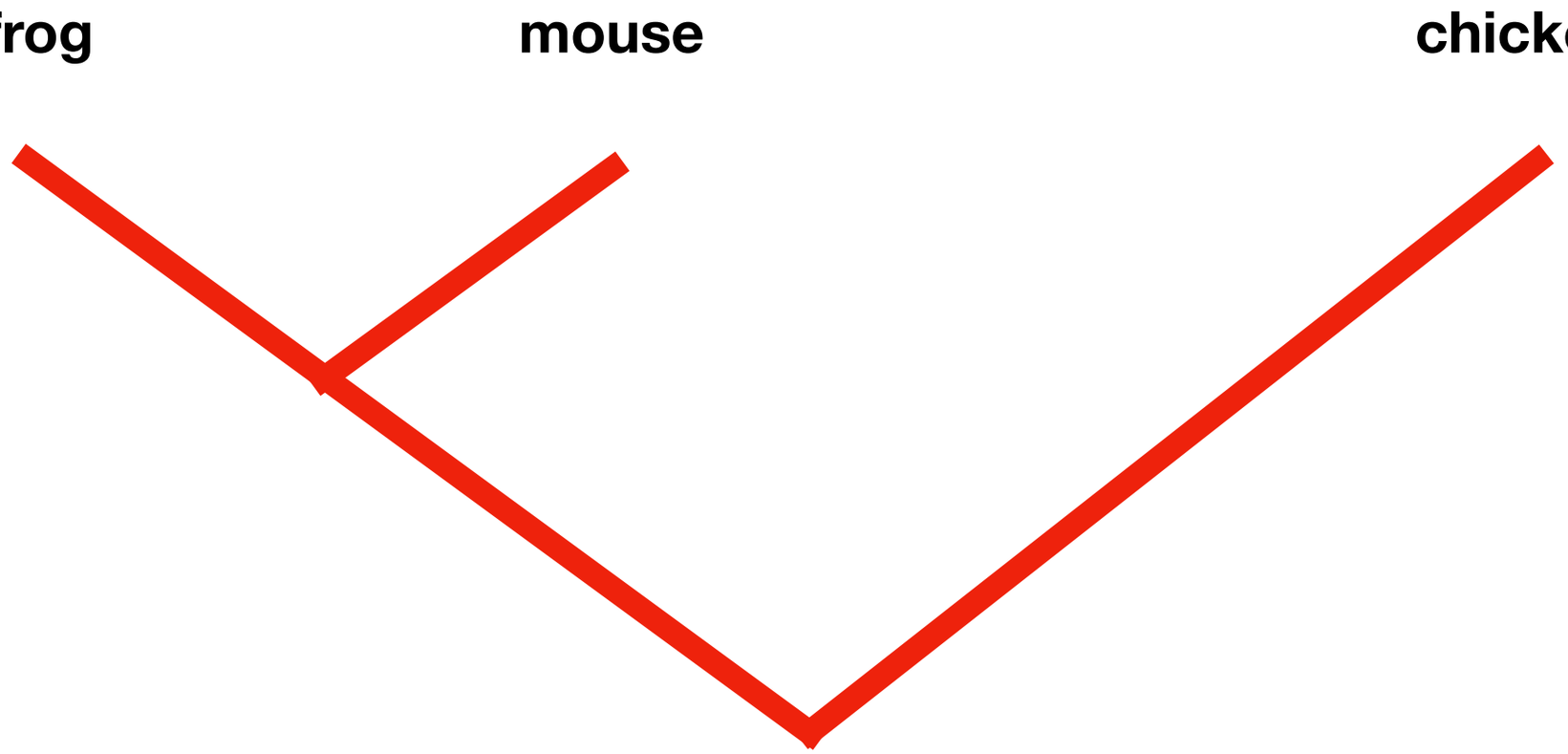
Get orthologous assemblies

What will happen if we use paralog genes to reveal evolutionary history



Get orthologous assemblies

What will happen if we use paralog genes to reveal evolutionary history



Get orthologous assemblies

There's various way to find orthologues. The method we used here called reciprocal blast

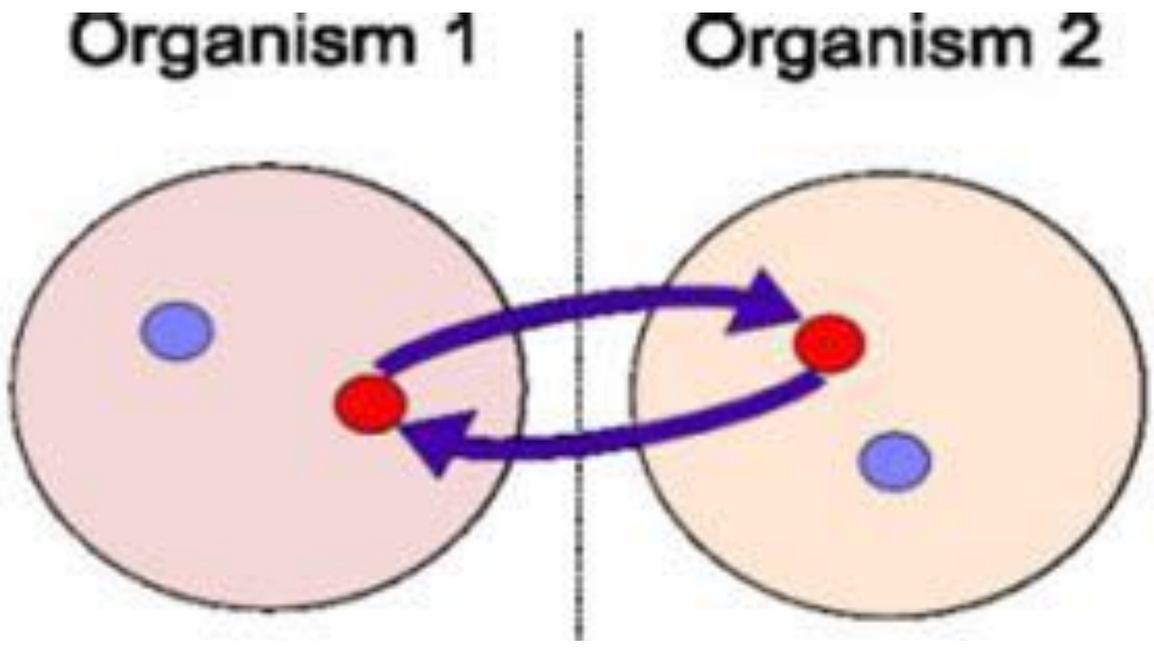
This method is built on the assumption that orthologous genes have identical or highly related functions and this sharing is greater than for paralogs.

Closest gene between 2 species are potential orthologous gene

Get orthologous assemblies

Reciprocal blast in general case

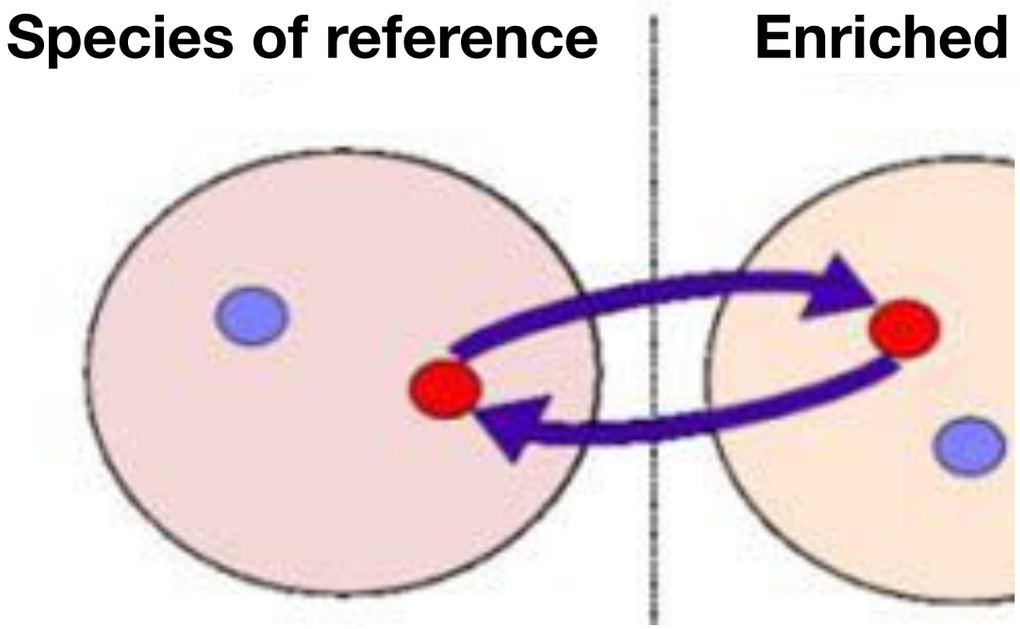
(1) sequence of gene which we want to find its orthologous sequence in other organisms
(2) genome of these 2 organisms



If a pair of genes in different species are the closest to each other, these 2 genes have "reciprocal best hit"

Get orthologous assemblies

Reciprocal blast in our pipeline

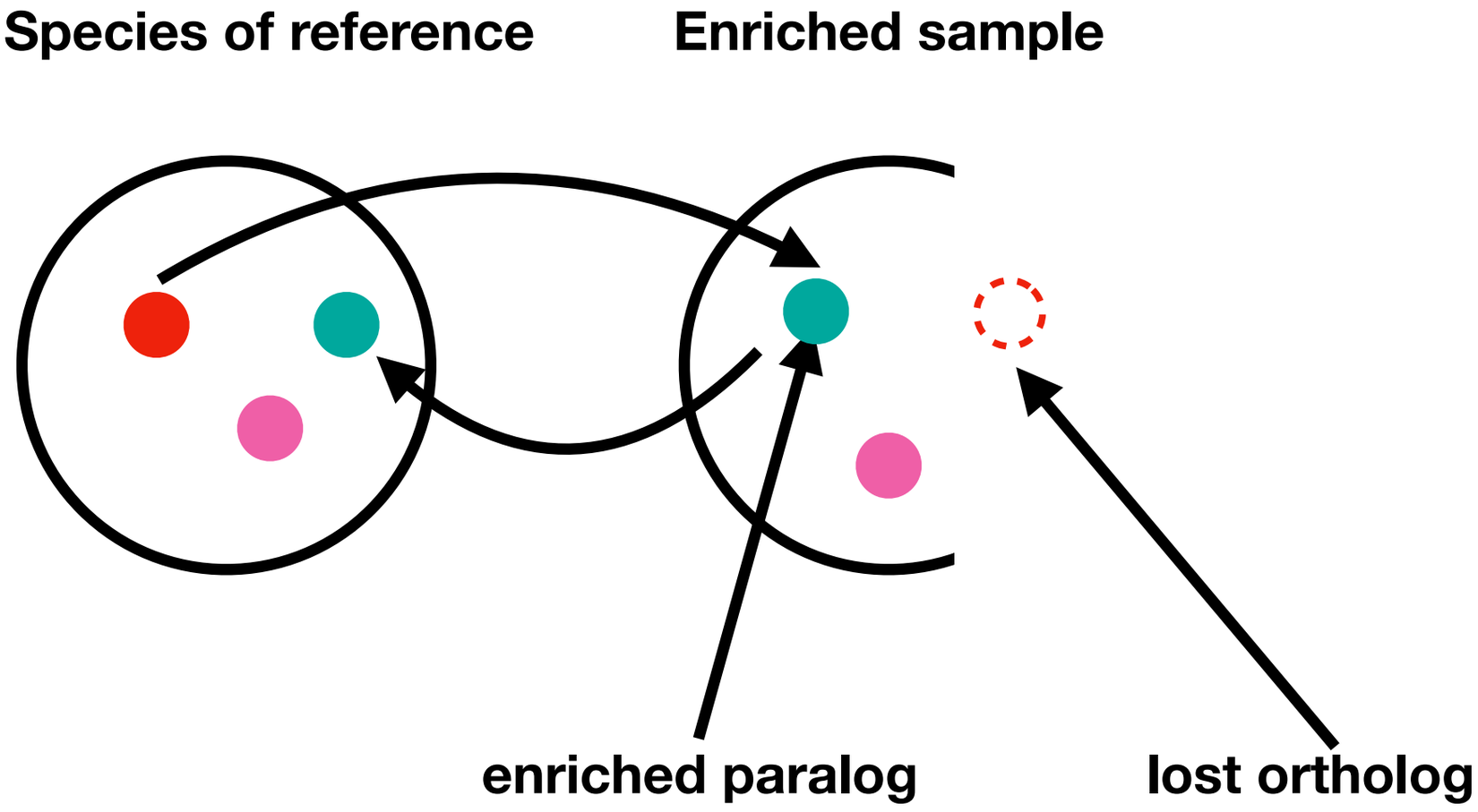


In our situation, we do not have the genome, only got several contigs only

But, loci of reference and contig have reciprocal best hit, then they are also putative orthologues

Get orthologous assemblies

Reciprocal blast in our pipeline



Exclude the contig if it does not have reciprocal blast hit with reference loci

Until here we've been reached our first goal