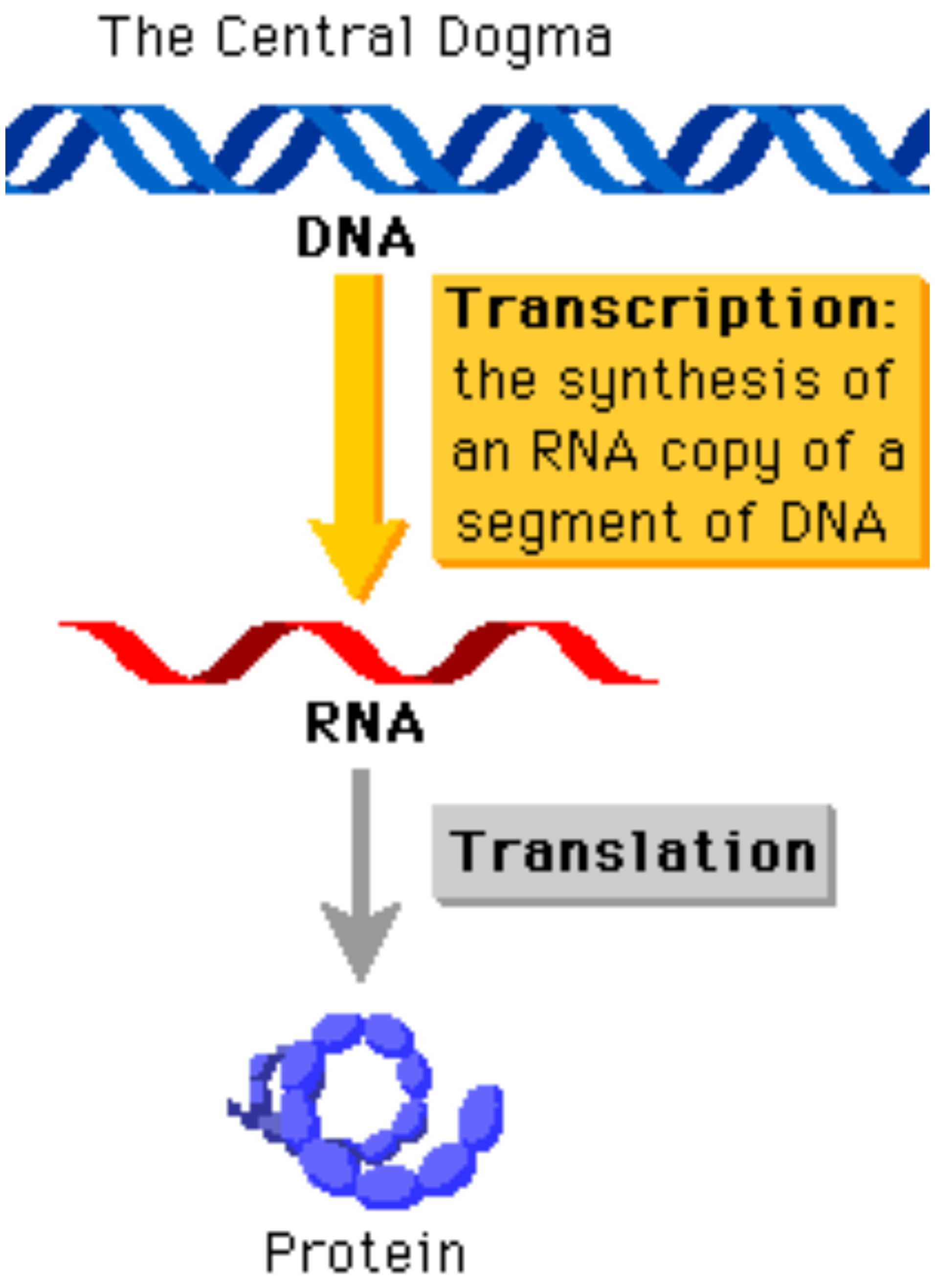
introduction to (de novo) assembly

blerina sinaimeri

The central dogma of molecular biology



The Central Dogma

DNA

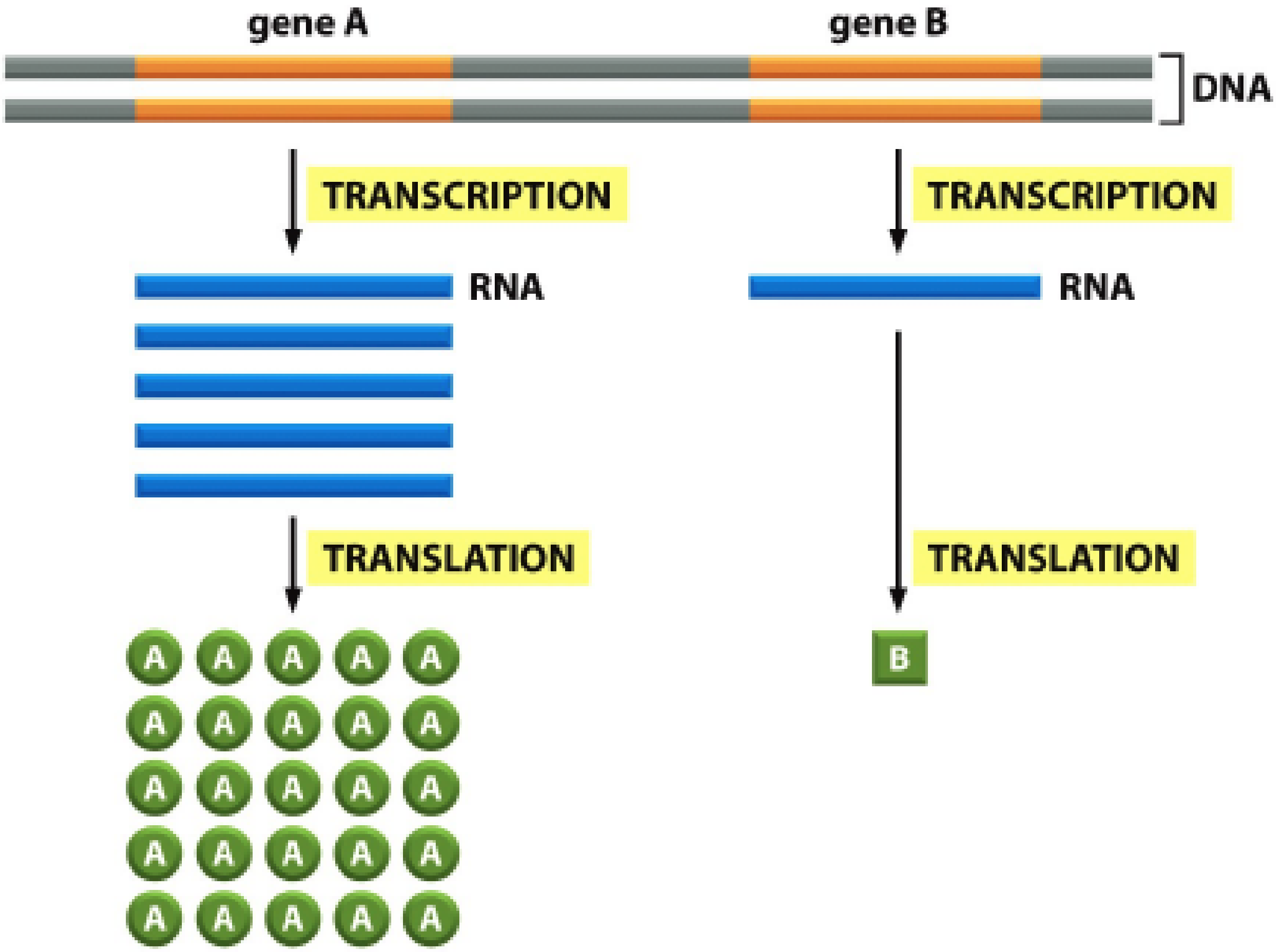
Transcription: the synthesis of an RNA copy of a segment of DNA

RNA

Translation

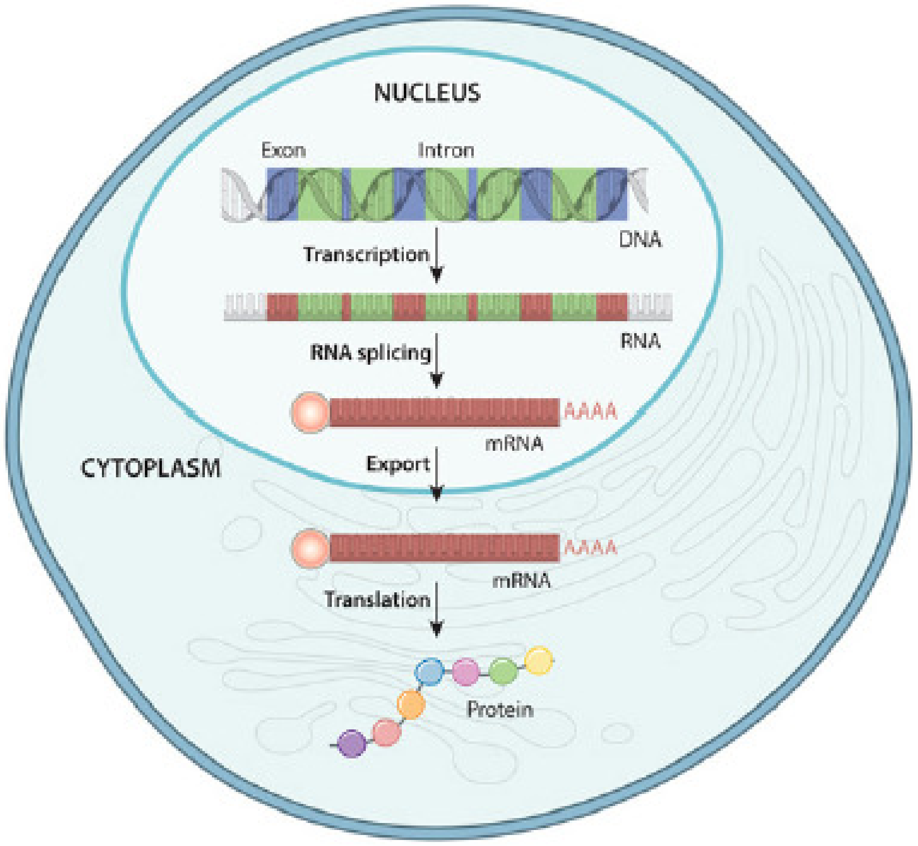
Protein

Gene Expression

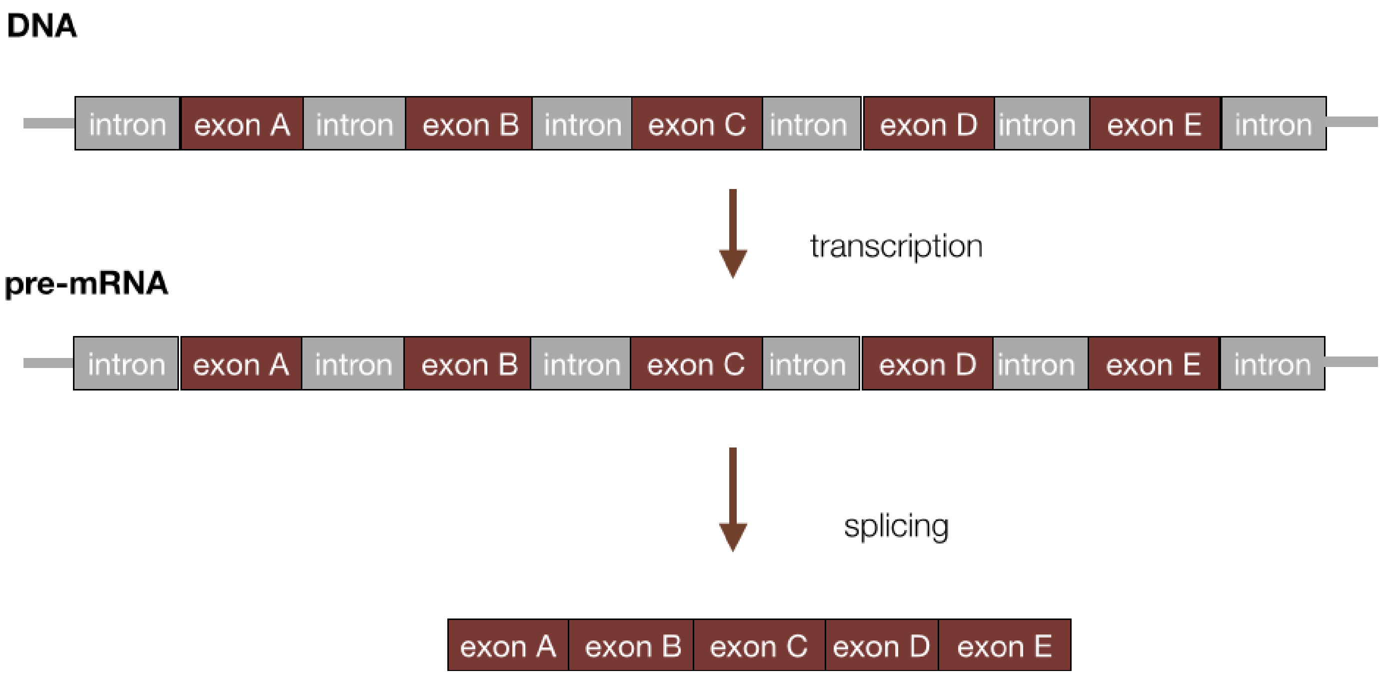


From DNA to RNA to proteins in eucaryotes

From DNA to proteins in eucaryotes



RNA-splicing in eucaryotes



DNA

intron | exon A | intron | exon B | intron | exon C | intron | exon D | intron | exon E | intron

transcription

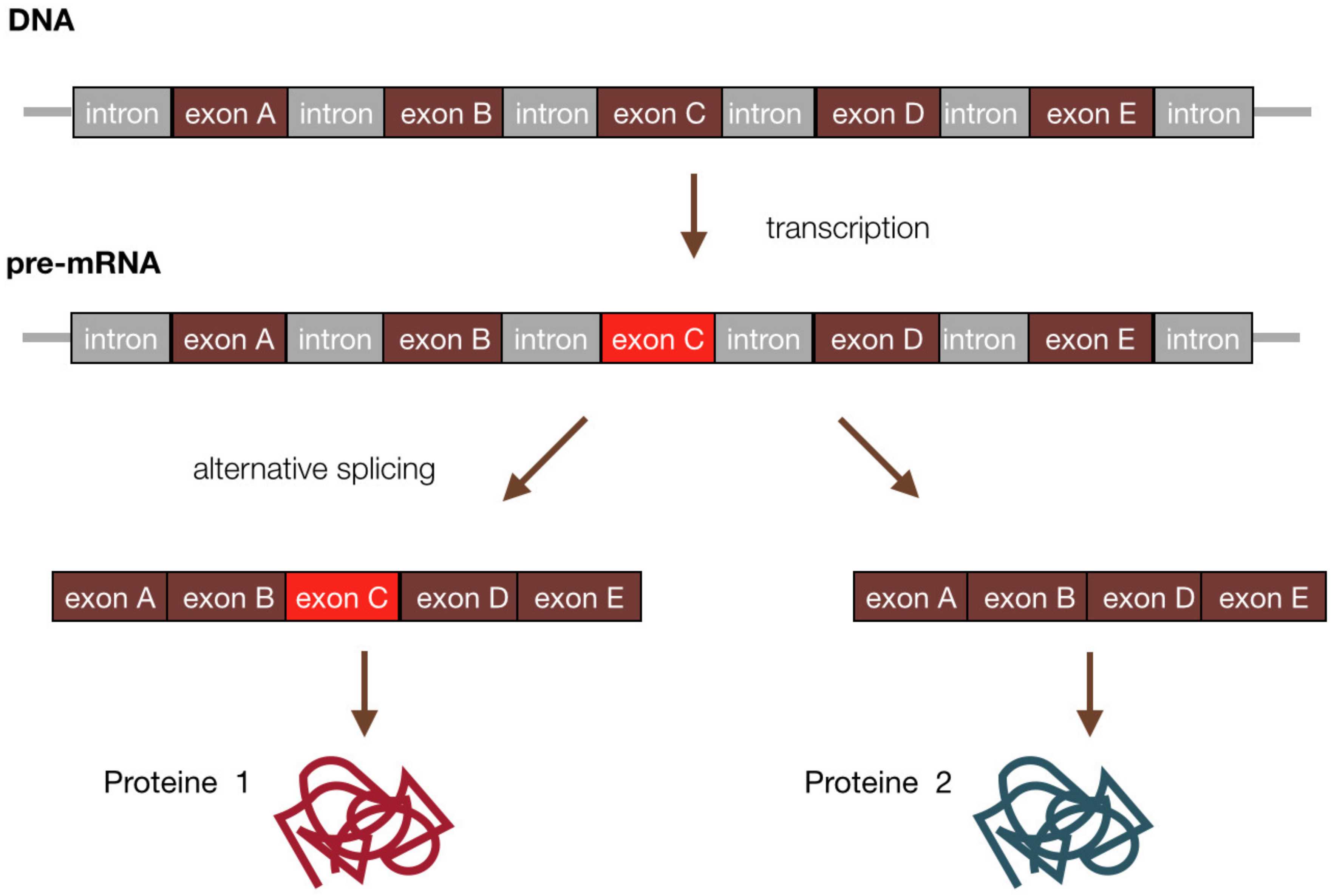
pre-mRNA

intron | exon A | intron | exon B | intron | exon C | intron | exon D | intron | exon E | intron

splicing

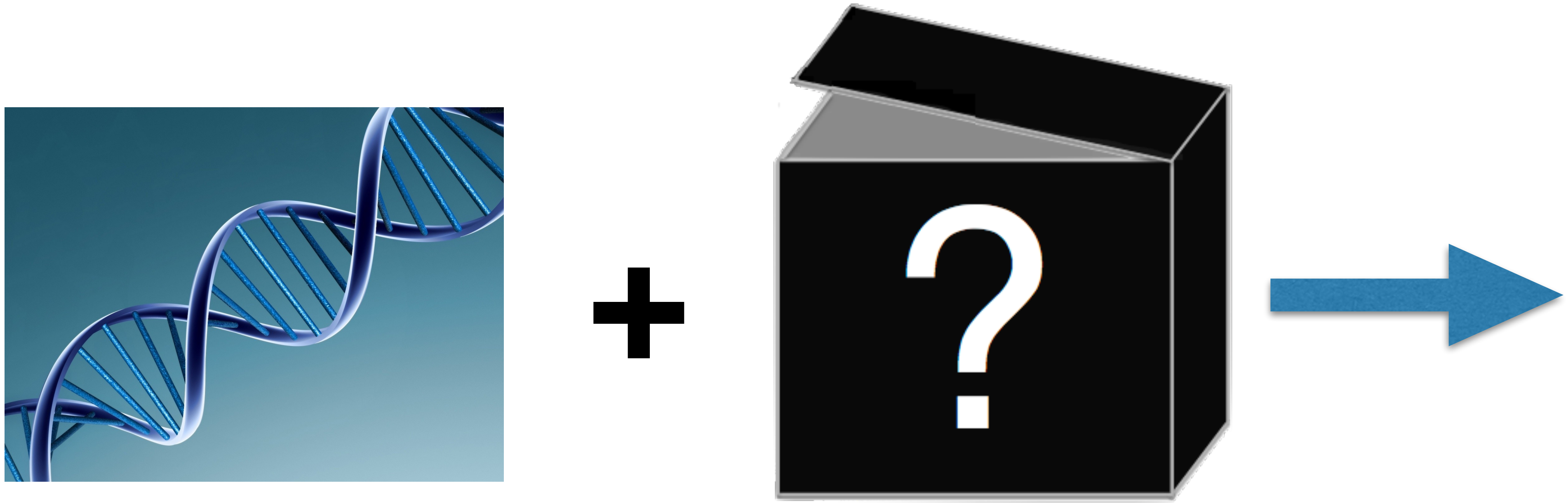
exon A | exon B | exon C | exon D | exon E

Alternative splicing (AS) in RNA



Why sequencing?

Perfect Word



AACGGTAACTGAAC
GTACGTACTACGTA
CGTACCATGAACGG
TAACTGAACGTACG
TACTACGTACGTAC
CATGAACGGTAACT
GAACGTACGTACTA
CGTACGTACCATG

Reality

- Cannot sequence full length DNA
- But we can sequence short fragments of it

DNA sequencing

1

2

3

4

5

The New York Times

MEN WALK ON MOON

ASTRONAUTS LAND ON PLAIN; COLLECT ROCKS, PLANT FLAG

DNA sequencing

The New York Times

MEN WALK ON MOON

ASTRONAUTS LAND ON PLAIN;
COLLECT ROCKS, PLANT FLAG

Voice From Moon:
Eagle Has Landed

A Powdery Surface
Is Closely Explored

the univer

r the new tech

makes a n

ver

ew technology

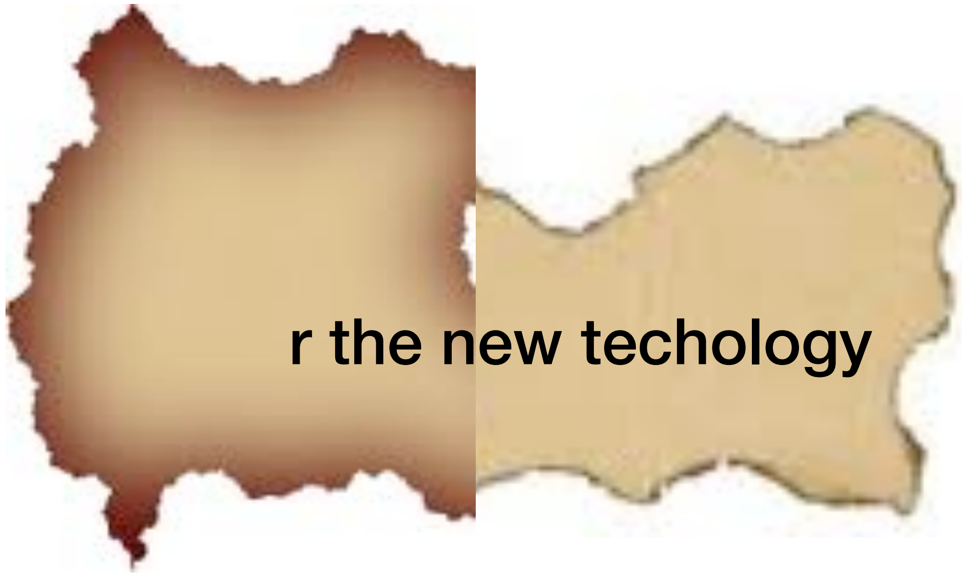
thre

DNA sequencing

The New York Times

MEN WALK ON MOON

ASTRONAUTS LAND ON PLAIN;
COLLECT ROCKS, PLANT FLAG



r the new techology

HORSE PROVES EXPENSIVE

How to Get Ri
of Ca

LOOK FOR THIEVES HERE

Billions of short reads

AACGGTAACTGAACGTACGTACTACGTACGTACCATG

AACGGTAACTGAACGTACGTACTACGTACGTACCATG

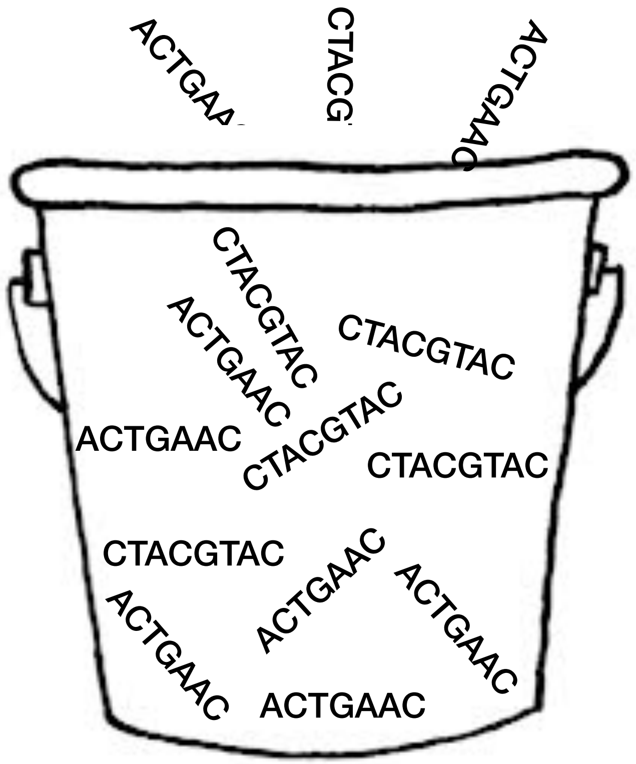
AACGGTAACTGAACGTACGTACTACGTACGTACCATG

Billions of short reads

AACGGTAACTGAACGTACGTACTACGTACGTACCATG.....

AACGGTAACTGAACGTACGTACTACGTACGTACCATG.....

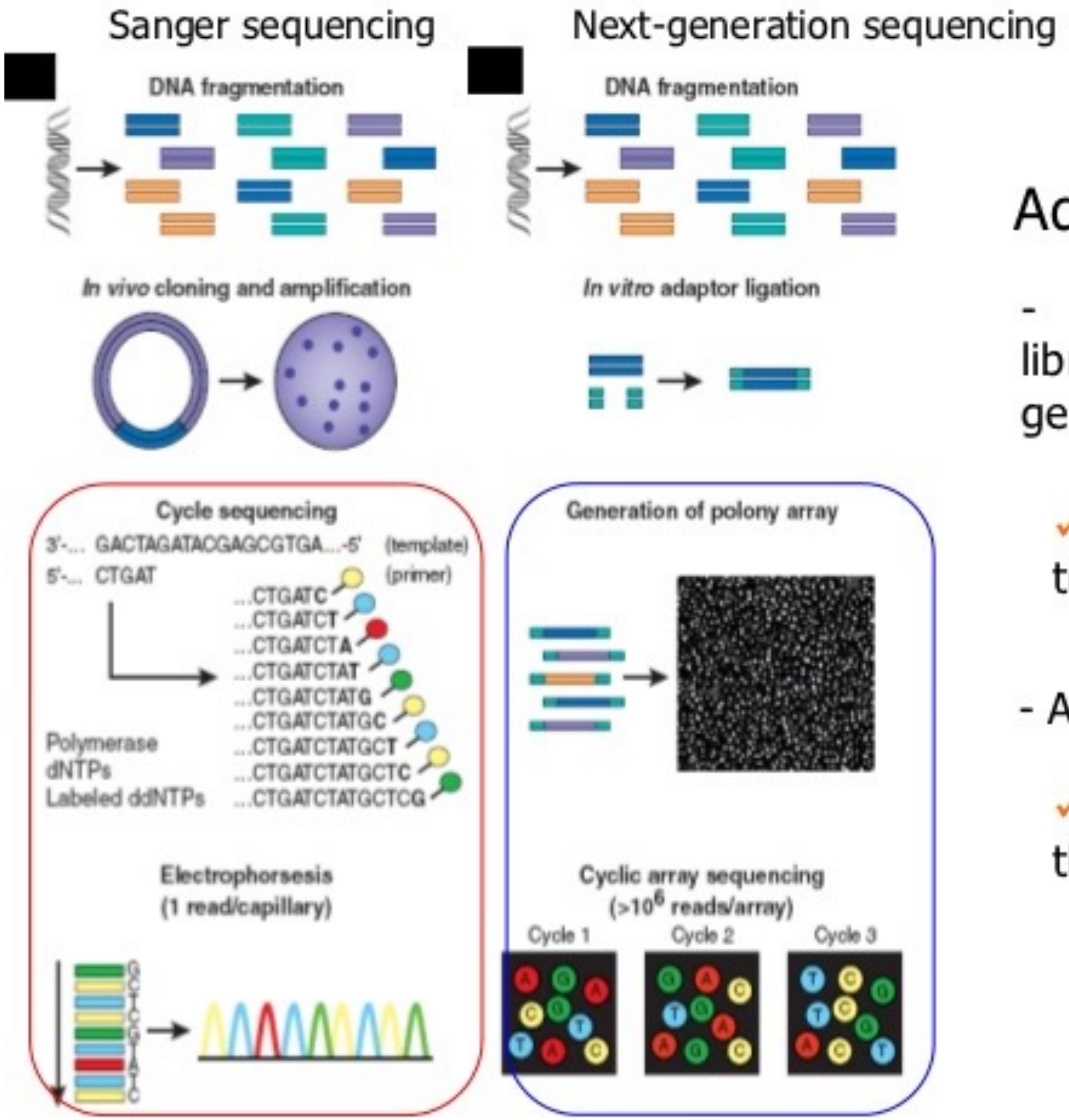
AACGGTAACTGAACGTACGTACTACGTACGTACCATG.....



AACGGTAACTGAACGTACGTACTACGTACGTACCATG

?

Next Generation Sequencing



Advantages:

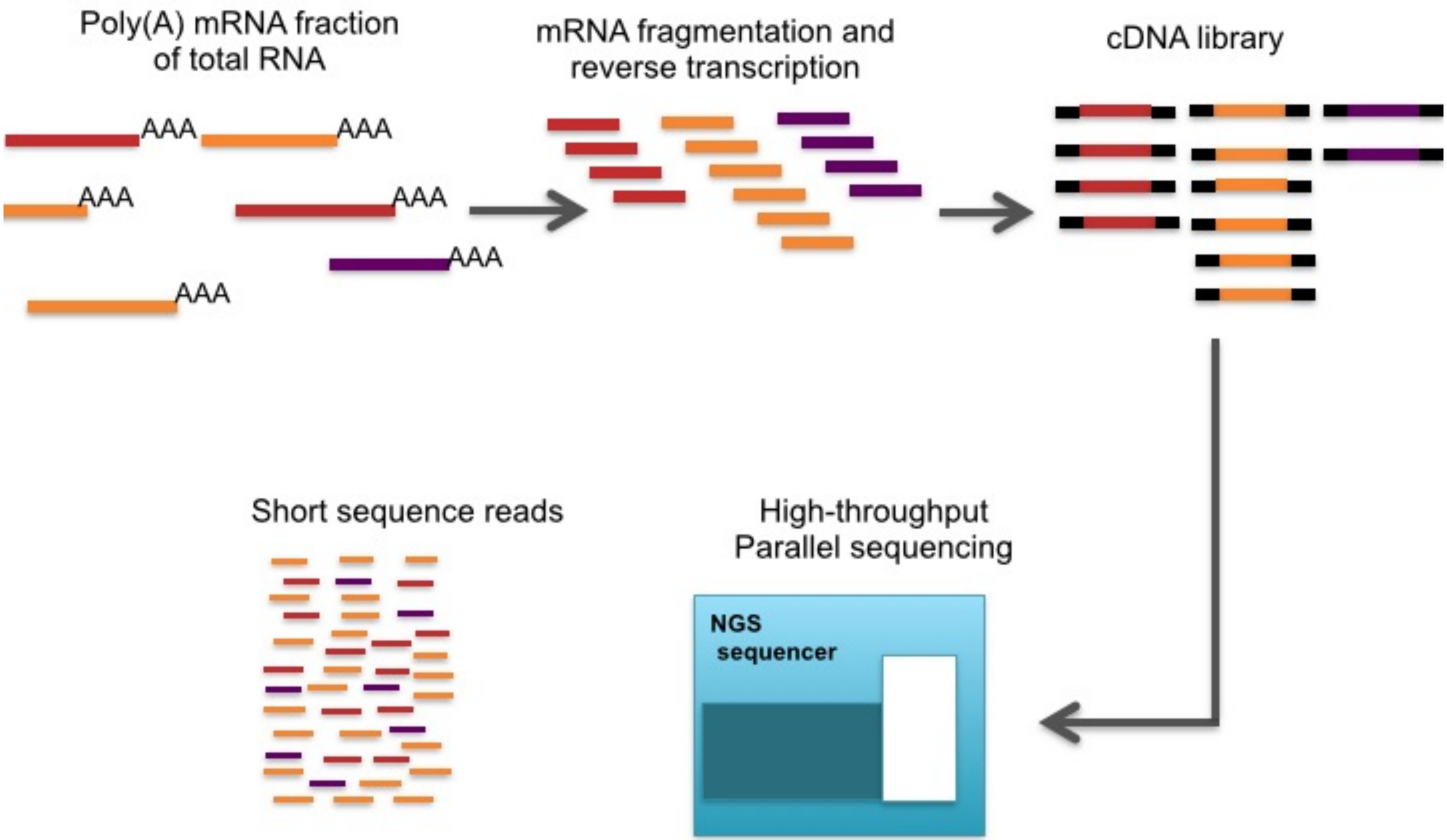
- Construction of a sequencing library → clonal amplification to generate sequencing features

  ✓ No in vivo cloning, transformation, colony picking...

- Array-based sequencing

  ✓ Higher degree of parallelism than capillary-based sequencing

RNA-seq



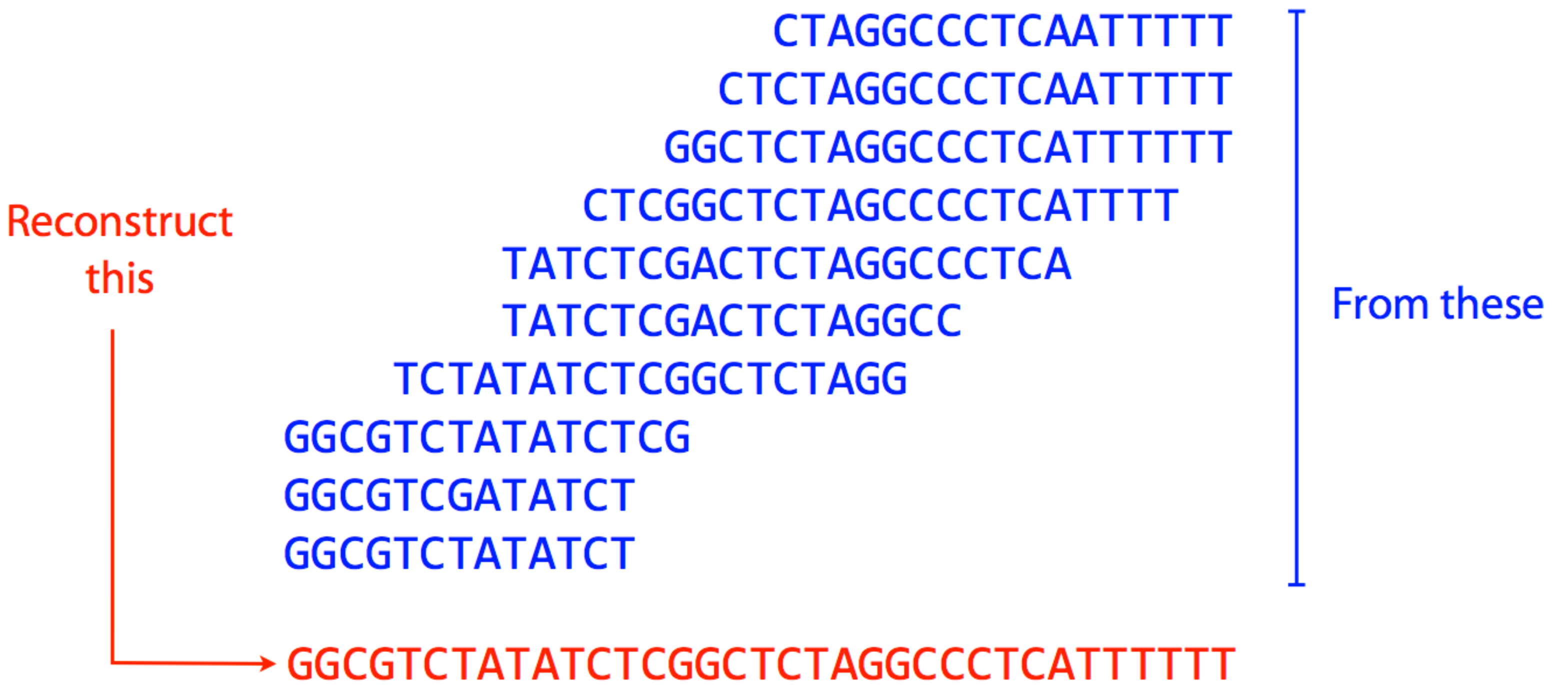
Poly(A) mRNA fraction of total RNA
AAA
AAA
AAA
AAA
AAA
AAA
mRNA fragmentation and reverse transcription
cDNA library
High-throughput Parallel sequencing
NGS sequencer
Short sequence reads

Why RNA-sequencing?

- Which region of the genome is transcribed
- Variability (AS events) of the mRNAs of the same gene
- Expression level of the mRNAs

coverage

- All genome positions are **covered** many times.
- The **coverage** is the number of reads covering a fixed position.



CTAGGCCCTCAATTTTT
CTCTAGGCCCTCAATTTTT
GGCTCTAGGCCCTCATTTTTT
CTCGGCTCTAGCCCCTCATTTT
TATCTCGACTCTAGGCCCTCA
TATCTCGACTCTAGGCC
TCTATATCTCGGCTCTAGG
GGCGTCTATATCTCG
GGCGTCGATATCT
GGCGTCTATATCT

From these

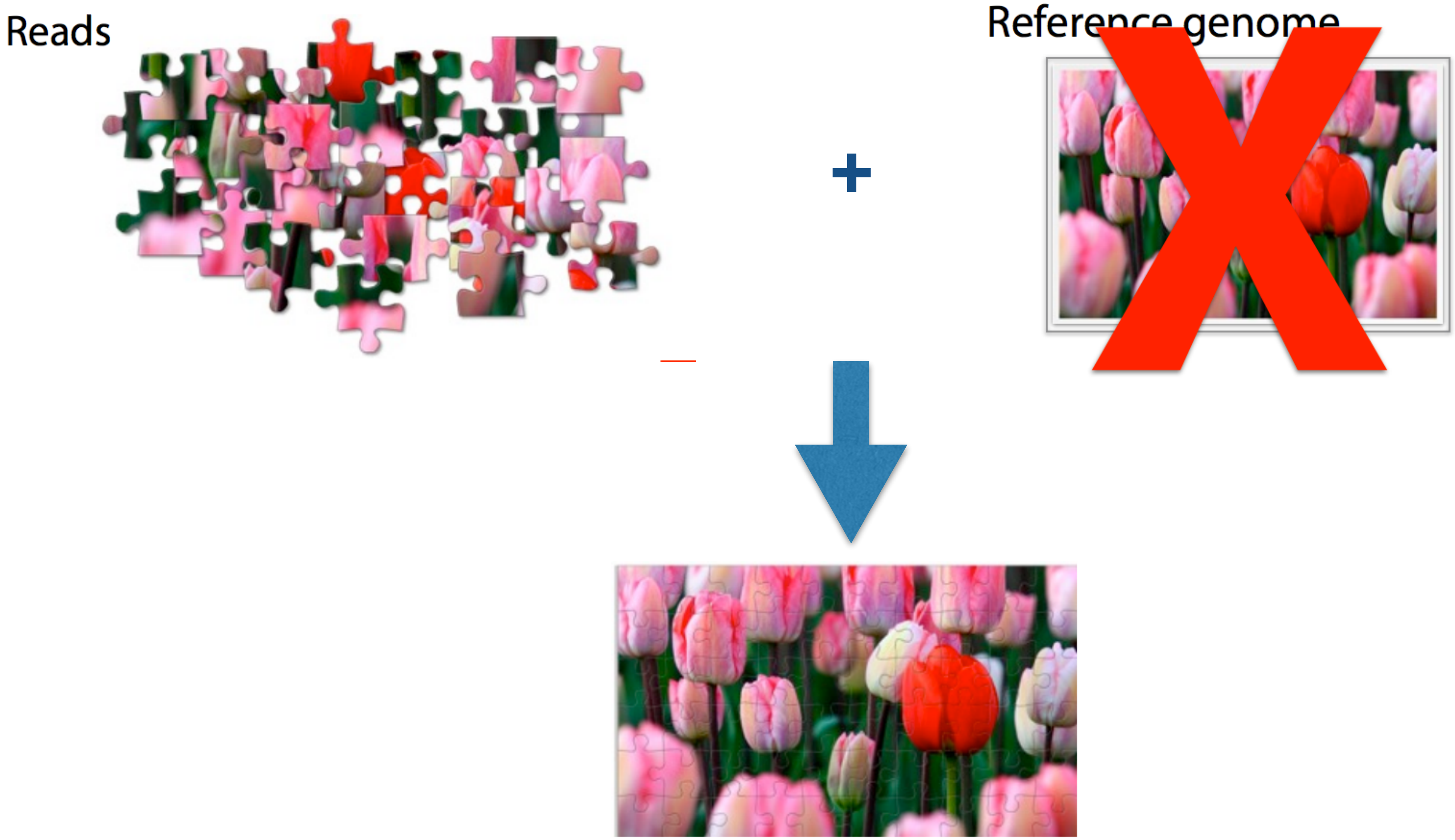
Reconstruct this

GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT

Some terminology

- **read** - a 100-250 long word that comes out of a NGS machine
- **coverage** - the average number of reads (or inserts) that cover a position in the target DNA piece
- **shotgun sequencing** - the process of obtaining many reads from random locations in DNA, to detect overlaps and assemble
- **mate pair** - a pair of reads from two ends of the same insert fragment (we know approx. distance)
- **contig** - a contiguous sequence formed by several overlapping reads with no gaps
- **consensus sequence** - sequence derived from the multiple alignment of reads in a contig

de novo assembly vs referenced-based mapping



de novo assembly

Methods

Different approaches

- greedy assembly
- Overlap-layout-consensus
- de bruijn graphs
- strings graphs

Basic idea

- Find all overlaps between reads
- Build a graph
- Simplify the graph (sequencing errors)
- Traverse a graph to produce a consensus.

Modelling and assembling NGS data (I)

- Given a set of strings $\mathcal{R} \subset \Sigma^* = \{A, C, T, G\}^*$ each $r \in \mathcal{R}$ is part of an unknown string $S \in \Sigma^*$ reconstruct the original string $S \in \Sigma^*$

Modelling and assembling NGS data (I)

- Given a set of strings $\mathcal{R} \subset \Sigma^* = \{A, C, T, G\}^*$ each $r \in \mathcal{R}$ is part of an unknown string $S \in \Sigma^*$ reconstruct the original string $S \in \Sigma^*$

- **Shortest superstring problem (SSP)**
  - Given a set of strings $r \in \mathcal{R}$ find a **minimum** length string $S \in \Sigma^*$ such that every $r \in \mathcal{R}$ is a substring of S

Shortest Superstring Problem

- **Shortest superstring problem (SSP)**
  - Given a set of strings $r \in \mathcal{R}$ find a **minimum** length string $S \in \Sigma^*$ such that every $r \in \mathcal{R}$ is a substring of S

Exercise 1

- SSP is NP-hard. (Garey and Johnson 1979)

SSP: Greedy Algorithm (I)

Algorithm

- **Overlap** $ov(s; t)$ of two strings s, t is the longest string y such, that $s = xy$ and $t = yz$ for some non-empty x, z.
- **Prefix** $pr(s; t)$ of s w.r.t. t be the string x in the previous definition.
- $s = pr(s; t)ov(s; t)$. Notice that $pr(s; t)t$ is the shortest string containing s and t in that order. This string is usually called a merge of s and t.
- **Greedy approach:** We pick two strings s_i s_j with largest overlap from R (breaking ties arbitrarily) and replace them with their merge. Stop when there is only one string left.

SSP: Greedy Algorithm (II)

Algorithm

- **Greedy approach:** We pick two strings s_i s_j with largest overlap from R (breaking ties arbitrarily) and replace them with their merge. Stop when there is only one string left.
  - How good is this algorithm? Can it have an approximation factor better than two?

Approximation algorithm

- A minimization problem is a problem where we want to find a solution with minimum value.
  - An algorithm for a minimization problem is called a ρ-approximation algorithm, for some $\rho > 1$, if the algorithm produces for any input I a solution whose value is at most $\rho \cdot \mathrm{opt}(I)$.
- A maximization problem is a problem where we want to find a solution with maximum value.
  - An algorithm for a maximization problem is called a ρ-approximation algorithm, for some $\rho < 1$, if the algorithm produces for any input I a solution whose value isat least $\rho \cdot \mathrm{opt}(I)$. The factor ρ is called the approximation factor (or the approximation ratio) of the algorithm.

SSP: Greedy Algorithm (II)

Algorithm

- **Greedy approach:** We pick two strings s_i s_j with largest overlap from R (breaking ties arbitrarily) and replace them with their merge. Stop when there is only one string left.
  - How good is this algorithm? Can it have an approximation factor better than two?

$$\{ab^k, b^k c, b^{k+1}\}$$

SSP: Greedy Algorithm (II)

Algorithm

- **Greedy approach:** We pick two strings s_i s_j with largest overlap from R (breaking ties arbitrarily) and replace them with their merge. Stop when there is only one string left.
  - How good is this algorithm? Can it have an approximation factor better than two?

The Greedy Conjecture [Blum et al. 1991]: The Greedy Algorithm has approximation factor 2.

SSP: Greedy Algorithm (II)

Algorithm

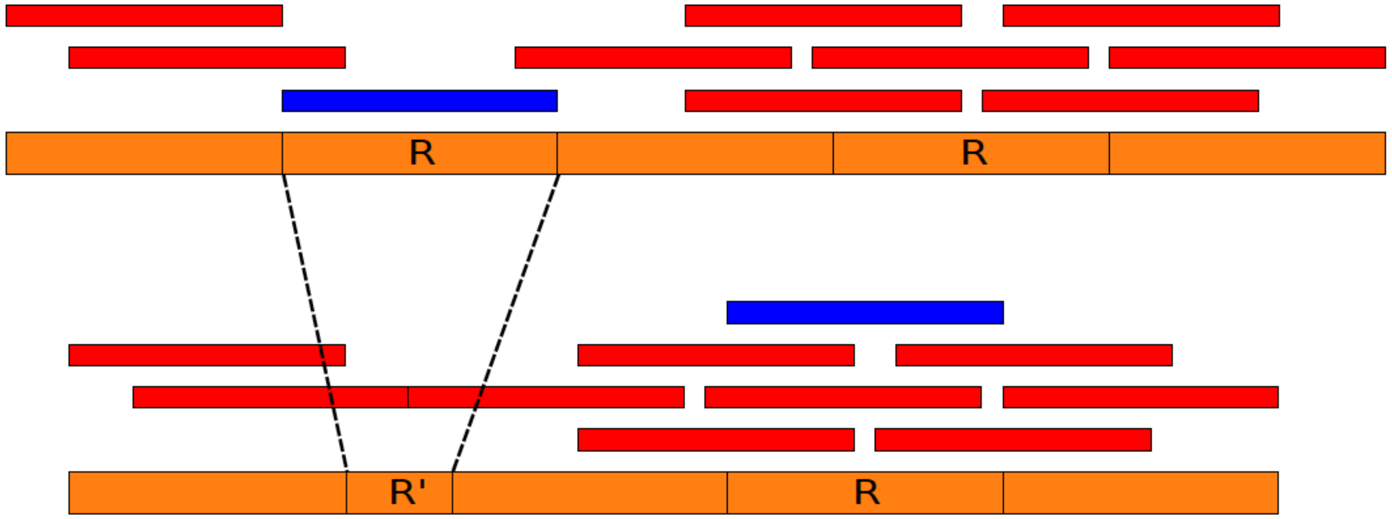
- **Greedy approach:** We pick two strings s_i s_j with largest overlap from R (breaking ties arbitrarily) and replace them with their merge. Stop when there is only one string left.
  - How good is this algorithm? Can it have an approximation factor better than two?

Exercise 2

- **What is the best approximation value that you can prove for the greedy algorithm?**

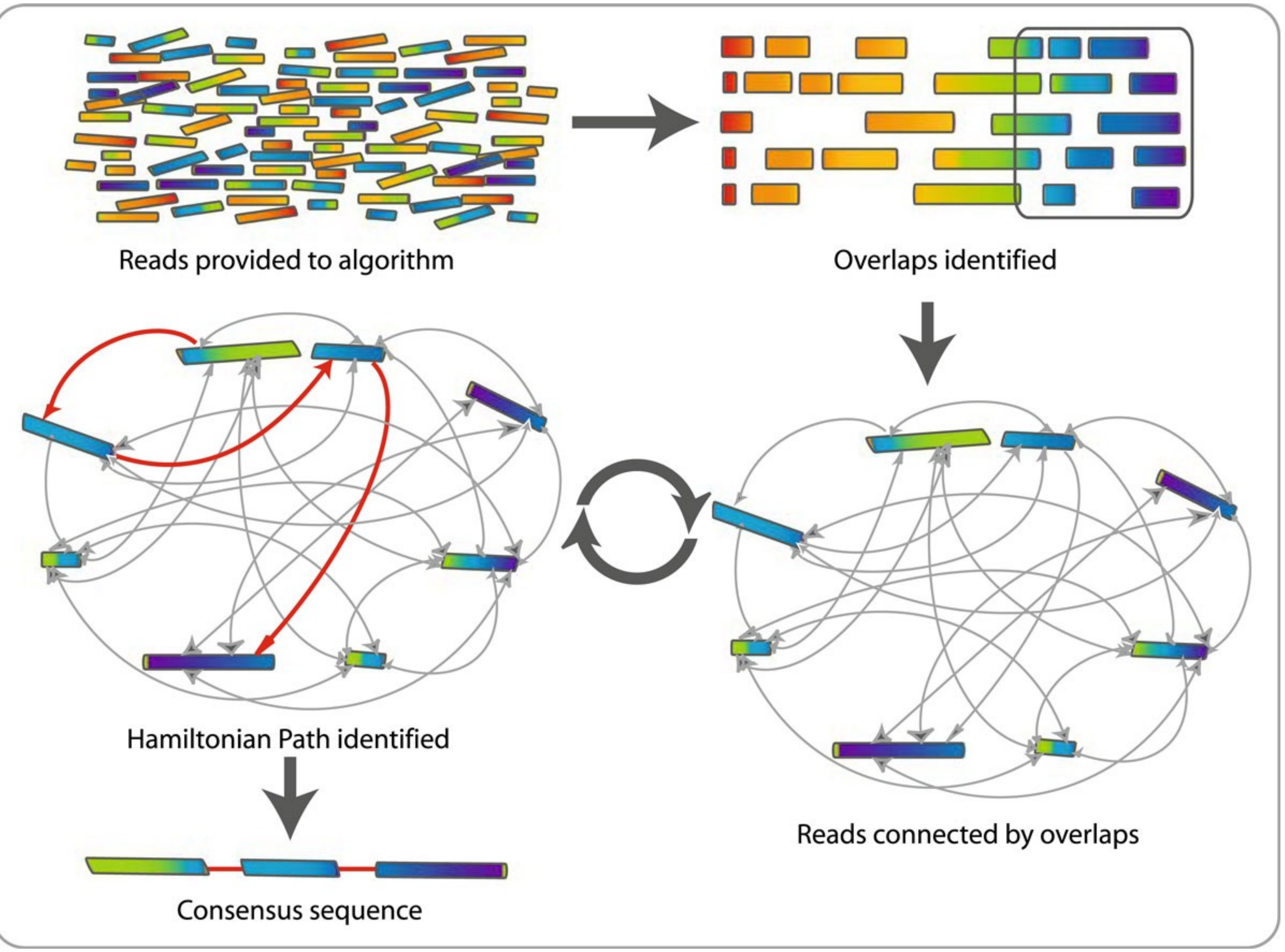
SSP: Problems (III)

- **Minimality :** requiring the superstring to be of minimal length, although motivated by parsimony, is in the best case questionable due to *repeats*.



- **Local choices :** in the true solution, the genome from which the reads were generated, several suffix-prefix read overlaps are not locally optimal.

Modelling and assembling NGS data (II) Overlap-Layout-Consensus



- **Overlap :** Build the overlap graph (find potentially overlapping reads)
- **Layout :** merge reads into contigs and simplify the graph.
- **Consensus :** Derive the DNA sequence and correct read errors

Building the overlap graph

Definition (Overlap Graph)

- Given a set of reads $\mathcal{R} \subset \Sigma^* = \{A, C, T, G\}^*$ the overlap graph is a complete weighted directed graph such that:
- $V = \mathcal{R}$ and $E = \mathcal{R}^2;$
- $w(u, v) =$ length of the maximal suffix of *u* that is equal to a prefix of *v*

Building the overlap graph

Definition (Overlap Graph)

- Given a set of reads $\mathcal{R} \subset \Sigma^* = \{A, C, T, G\}^*$ the overlap graph is a complete weighted directed graph such that:
- $V = \mathcal{R}$ and $E = \mathcal{R}^2$;
- $w(u, v) =$ length of the maximal suffix of *u* that is equal to a prefix of *v*

In a more general definition a small number of mismatches is allowed for the suffix-prefix overlap. (Use DP to find the optimal overlap alignment)

Finding the optimal overlap alignment

- Overlap: suffix of *X* matches a prefix of *Y*
- We want to allow mismatches between *X* and *Y*.

X : **CTCGGCCCTGG - -**
Y : **- - CGACCCTAGTT**

```
D[i,j]= min {
                D[i-1,j]+score(X[i-1],-),
                D[i, j-1] + score(-,Y[j-1]),
                D[i-1, j-1] + score(X[i-1],Y[j-1])
}
```

score(a,b)

	-	A	C	T	G
-	8	8	8	8	8
A	8	0	4	4	2
C	8	4	0	2	4
T	8	4	2	0	4
G	8	2	4	4	0

Finding the optimal overlap alignment

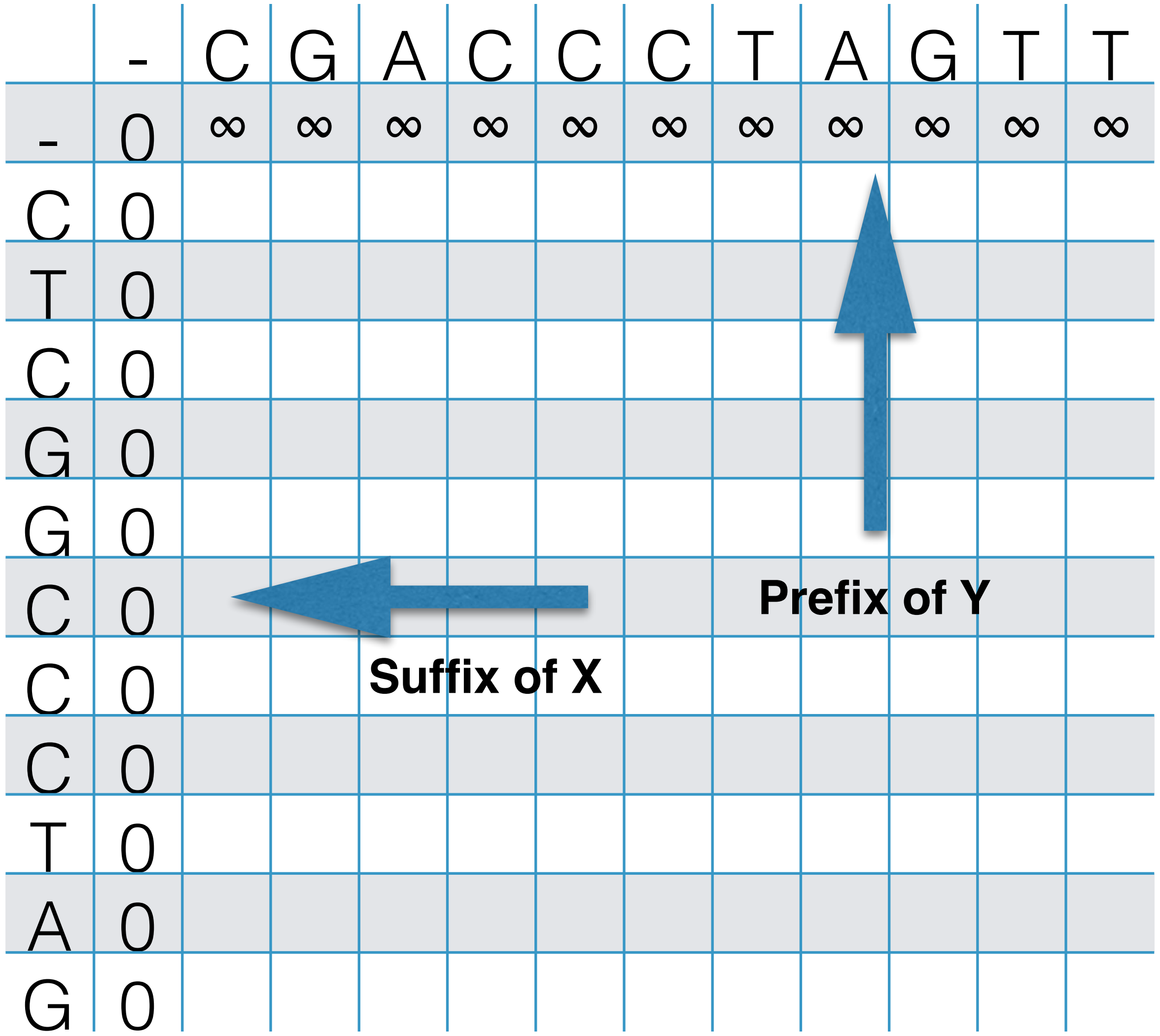
- Overlap: suffix of *X* matches a prefix of *Y*
- We want to allow mismatches between *X* and *Y*.

X : **CTCGGCCCTGG - -**
Y : **- - CGACCCTAGTT**

```
D[i,j]= min {
            D[i-1,j]+score(X[i-1],-),
            D[i, j-1] + score(-,Y[j-1]),
            D[i-1, j-1] + score(X[i-1],Y[j-1])
}
```

score(a,b)

	-	A	C	T	G
-	8	8	8	8	8
A	8	0	4	4	2
C	8	4	0	2	4
T	8	4	2	0	4
G	8	2	4	4	0



	-	C	G	A	C	C	C	T	A	G	T	T
-	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
C	0											
T	0											
C	0											
G	0											
G	0											
C	0											
C	0											
C	0											
T	0											
A	0											
G	0											

Prefix of Y

Suffix of X

Finding the optimal overlap alignment

- Overlap: suffix of *X* matches a prefix of *Y*
- We want to allow mismatches between *X* and *Y*.

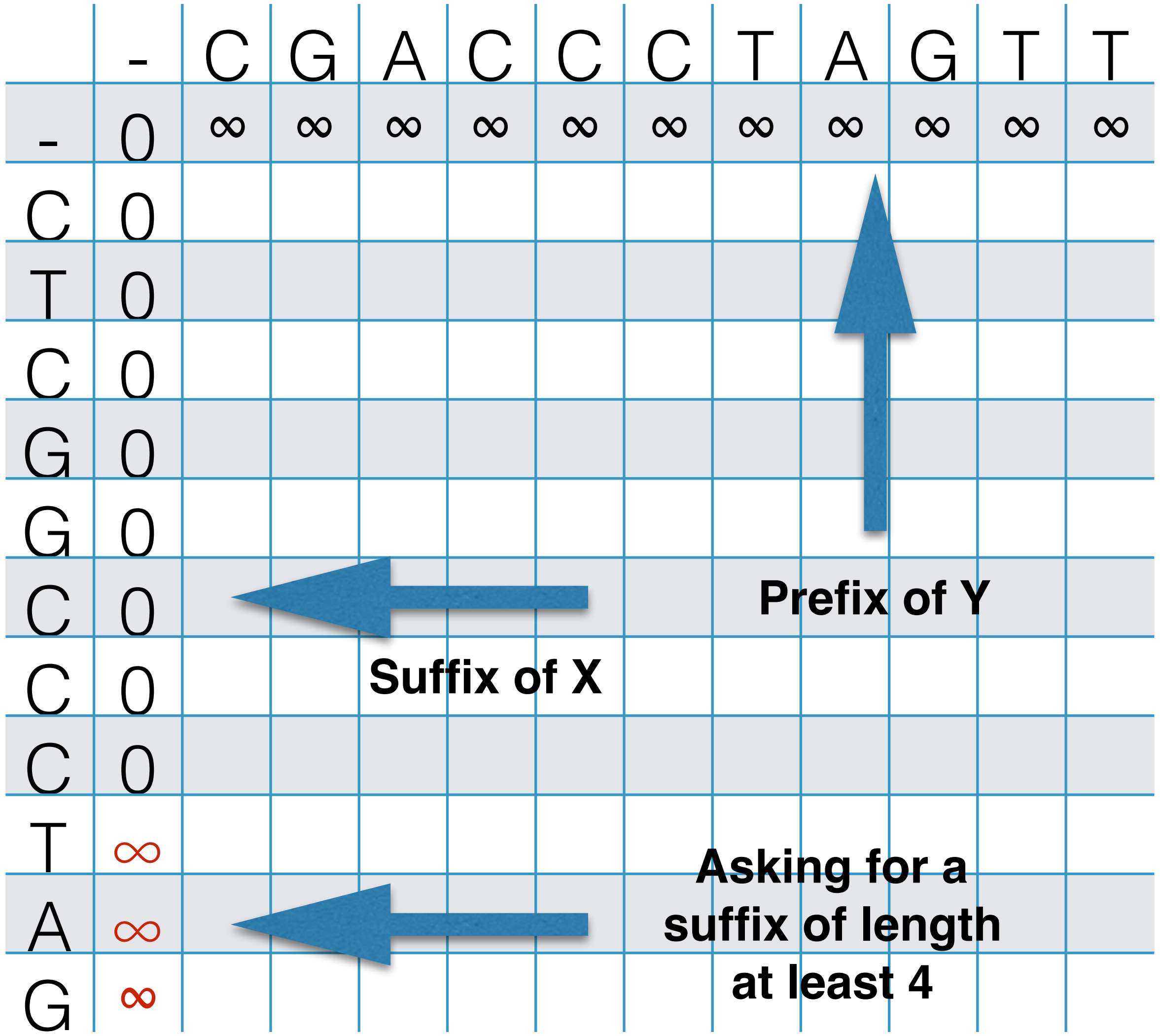
X : **CTCGGCCCTGG - -**

Y : **- - CGACCCTAGTT**

D[i,j]= min {
D[i-1,j]+score(X[i-1],-),
D[i, j-1] + score(-,Y[j-1]),
D[i-1, j-1] + score(X[i-1],Y[j-1])
}

score(a,b)

	-	A	C	T	G
-	8	8	8	8	8
A	8	0	4	4	2
C	8	4	0	2	4
T	8	4	2	0	4
G	8	2	4	4	0



	-	C	G	A	C	C	C	T	A	G	T	T
-	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
C	0											
T	0											
C	0											
G	0											
G	0											
C	0											
C	0											
C	0											
T	∞											
A	∞											
G	∞											

Prefix of Y

Suffix of X

Asking for a suffix of length at least 4

Layout

- Graph simplification
    - remove all arcs that have weights below a given threshold.
    - remove from the graph all the edges that are transitively inferable. (**String graph**)

Consensus

- Find a *constrained* walk of minimum length in the graph
    - We define a *selection* function *s* that classifies the arcs:
        - *optional* : no constraint
        - *required* : present at leat once
        - *exact* : present exactly once

- Selection function is defined using A-statistics (Meyers et al. 2000)

Overlap-Layout-Consensus
and
Hamiltonian Path

- Given a selection function *s* , a string graph *G* find an *s-walk.*

Finding a minimum s-walk is NP-hard from a reduction from Hamiltonian path (Medvedev et.al. 2007)

- Many assemblers use heuristics however for shorter reads and much deeper coverages, the overlap computation step is a computational bottleneck.

Finding overlaps

- For **N reads of length L** we need:
  - **O(N^2)** comparisons
  - each comparison **O(L^2)** alignment

Given N reads…
Where N ~ 100 million…

We need to use a linear-time algorithm

Modelling and assembling NGS data (III)
de Bruijn graph

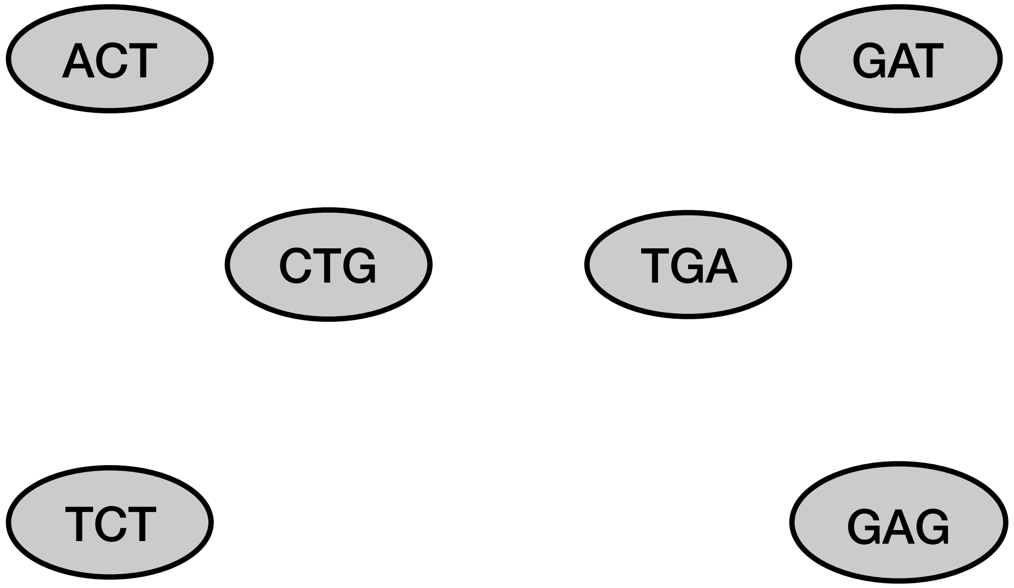
De Bruijn graph

Given a set of reads *R* and an integer *k* we define the de Bruijn graph *B(R,k)*

- Vertices are substrings of length **k** (**k**-mers)
- Arcs are *k-1* suffix-prefix overlaps that appear as a substring in *R*.

Example

R={ACTGAT,TCTGAG}, k=3



de Bruijn graph

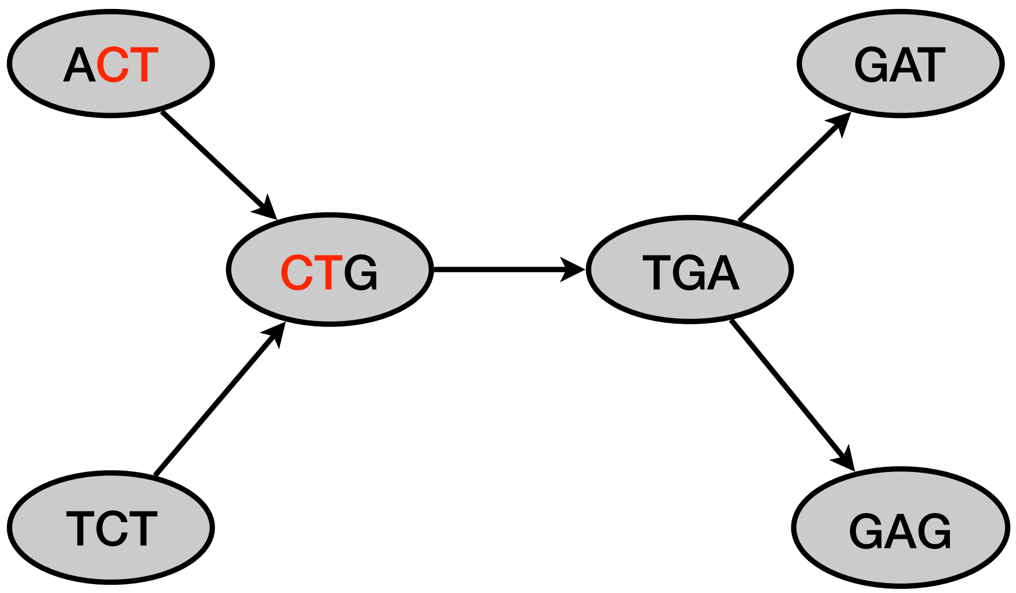
De Bruijn graph

Given a set of reads *R* and an integer *k* we define the de Bruijn graph *B(R,k)*

- Vertices are substrings of length **k** (**k**-mers)
- Arcs are *k-1* suffix-prefix overlaps that appear as a substring in *R*.

Example

R={ACTGAT,TCTGAG}, k=3



de Bruijn graph *vs* overlap graph

- If all the reads of R have length exactly $k + 1$, the line graph of $G_k(R)$ is exactly the overlap graph of R with the arcs of weight zero removed.

- In a de Bruijn graph there is a loss of information with regard to the overlap graph: in de Bruijn graphs we do not have the information that two k-mers came from the same read.
    - As a consequence there are walks in the de Bruijn graph that are not read coherent (not entirely covered by reads).

Problem: Find an Eulerian path (visit each arc of the graph once).

- Finding an Eulerian path is polynomial but it may be not “read coherent”.

- A graph may have an exponential number of Eulerian paths.

Computing a de Bruijn graph

- Given a read set *R*, we can build a de Bruijn graph $G_k(R)$ using a hash table to store all (k +1)-mers present in *R*.
- As each insertion and membership query in the hash table takes *O(1)* (expected) time, the de Bruijn graph can be built in time linear in the size of *R*, i.e. $O(\sum_{r \in \mathcal{R}} |r|)$.

We can compute a de Bruijn graph for a set of reads in linear time in the size of the reads.

de Bruijn graph

de Bruijn graph : Theory

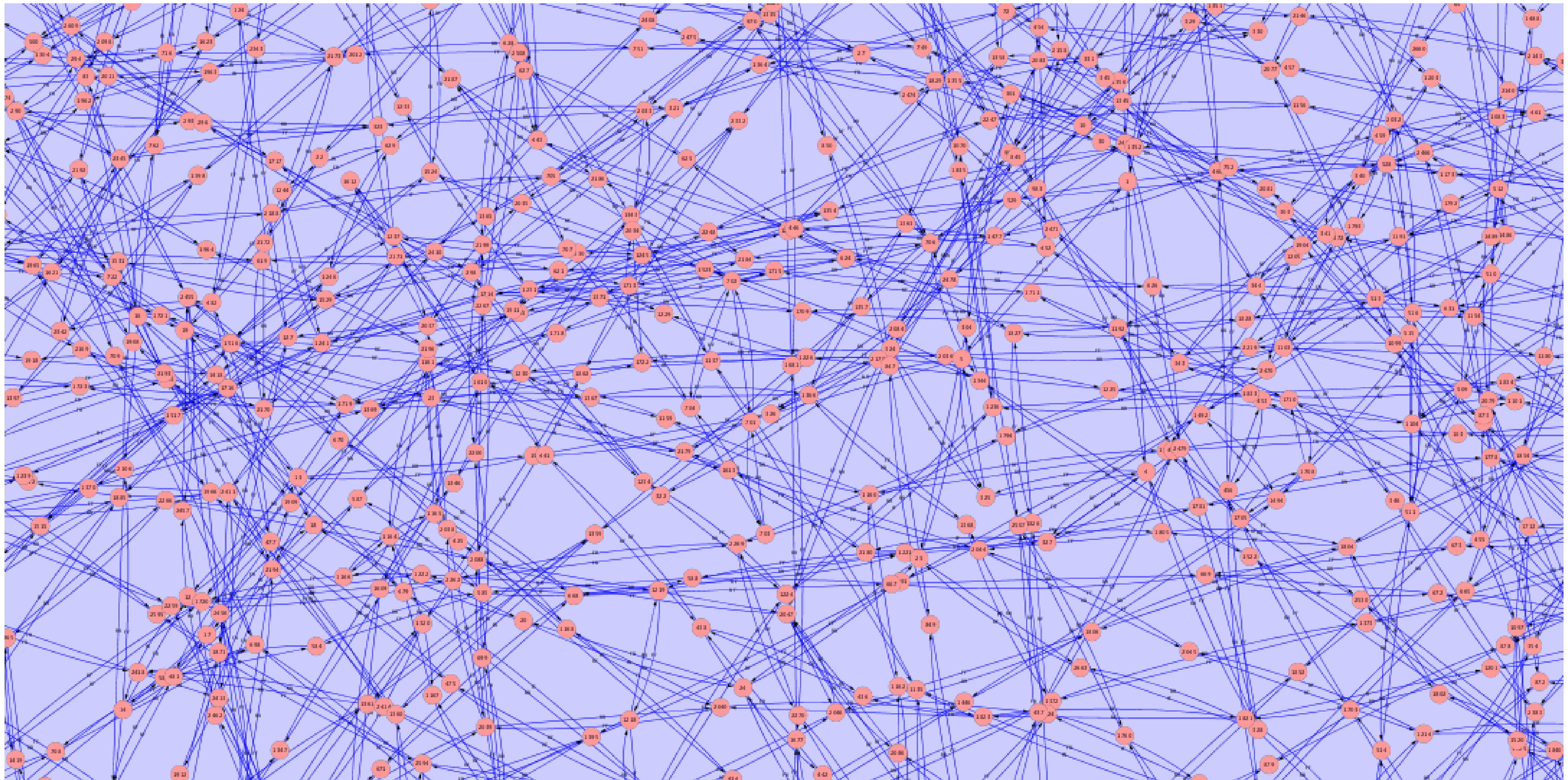
- *Sparse* Graph: out and in degree ≤ 4

de Bruijn graph

de Bruijn graph : Theory

- *Sparse* Graph: out and in degree ≤ 4

de Bruijn graph : Reality



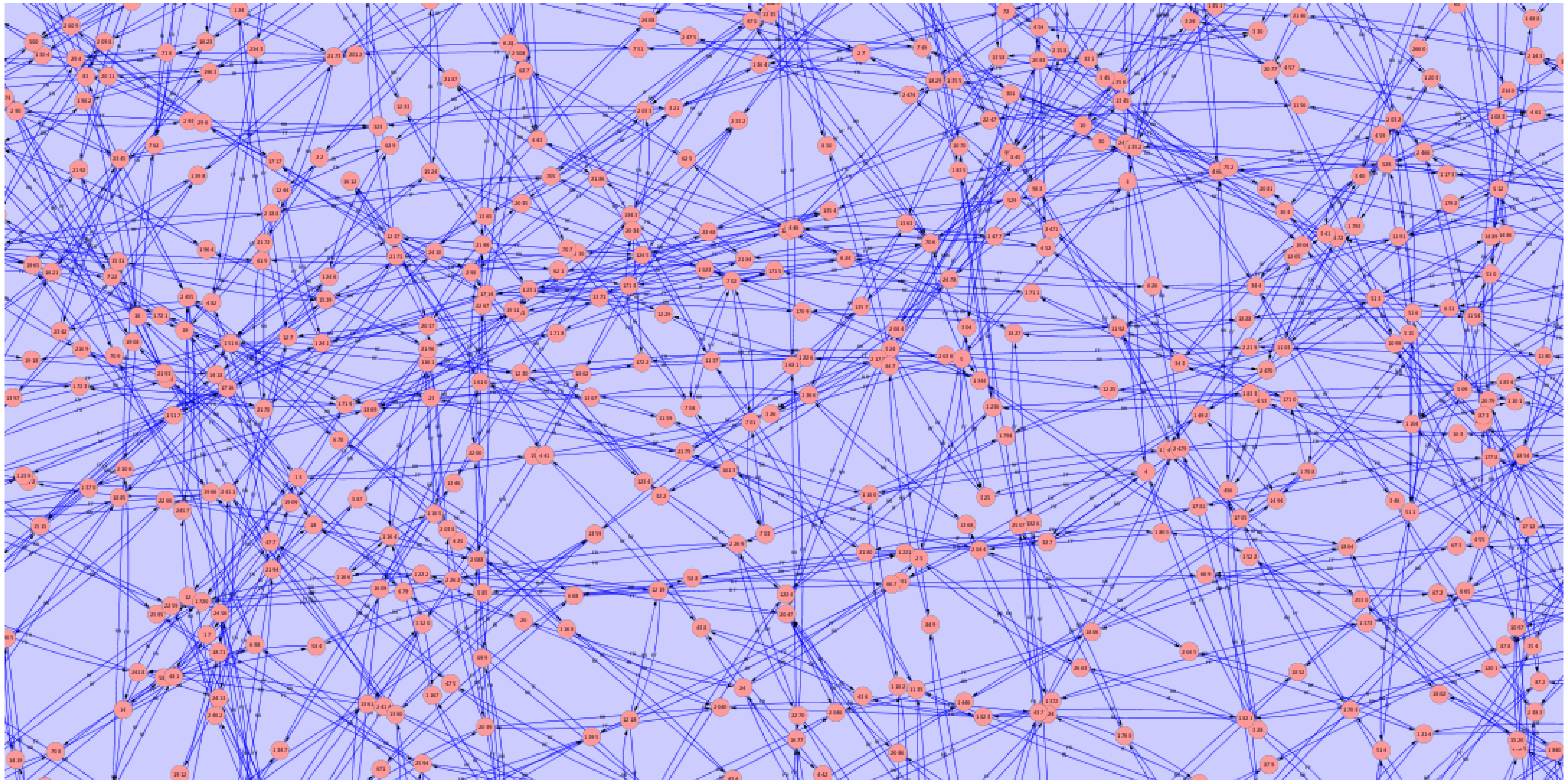
de Bruijn graph

de Bruijn graph : Theory

- *Sparse* Graph: out and in degree ≤ 4

What creates the complexity?

de Bruijn graph : Reality



repeats challenge in assembly