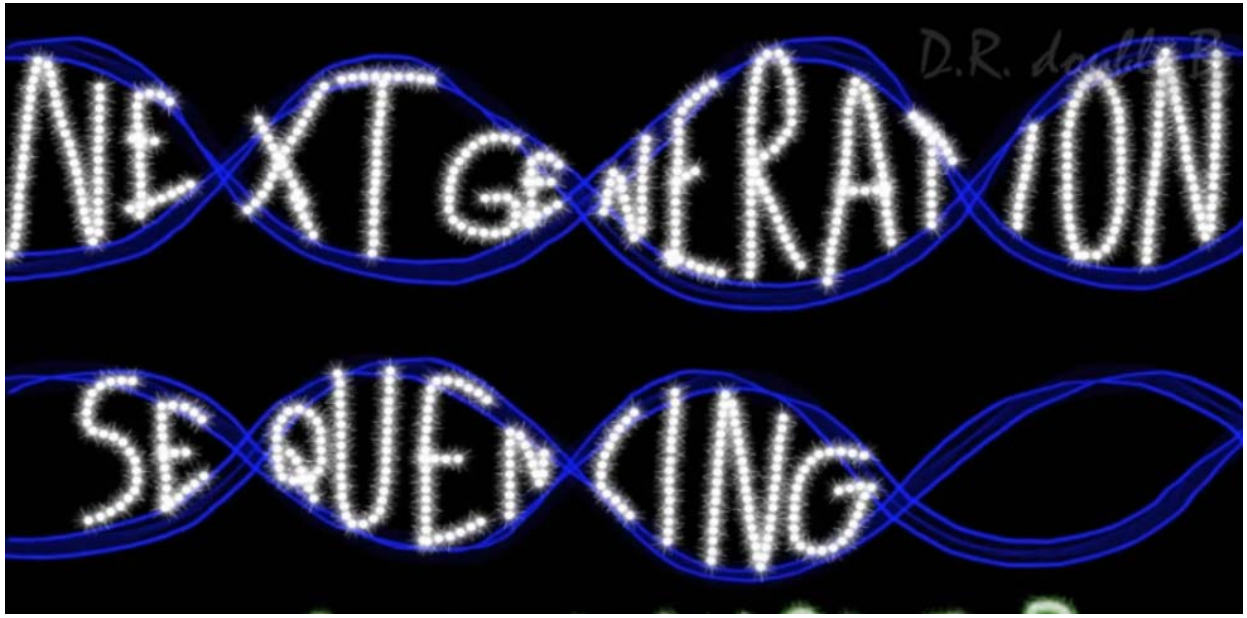
CS123A Bioinformatics Module 5 – Week 15 – Presentation 2

Leonard Wesley

Computer Science Dept

San Jose State Univ



Agenda

- Next Generation Sequencing (cont.)

NGS Technologies

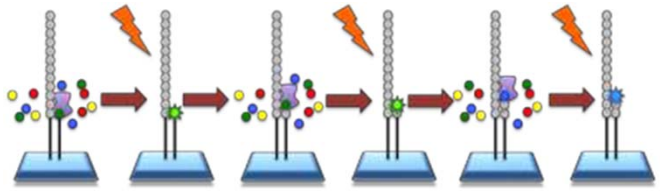
Illumina Sequencing

https://www.youtube.com/watch?v=fCd6B5HRaZ8

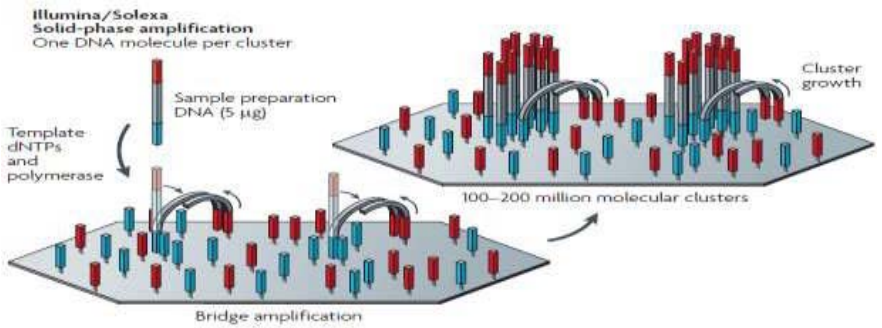
Illumina Sequencing Technology

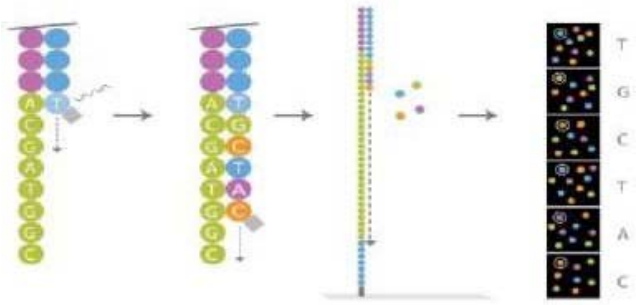
ILLUMINA/SOLEXA SEQUENCING

Solid-phase amplification can produce 100-200 million spatially separated clusters, providing free ends to which a universal sequencing primer can be hybridized to initiate the NGS reaction

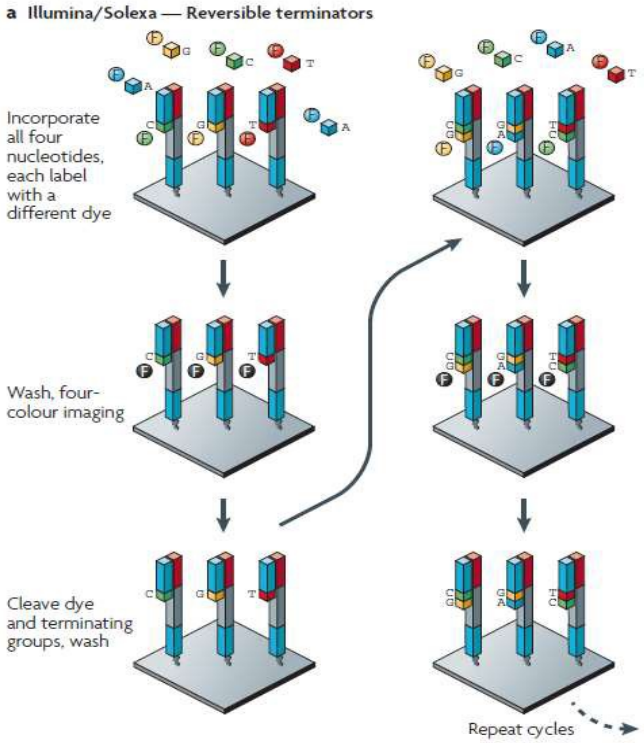


Bridge PCR Clustal Amplification



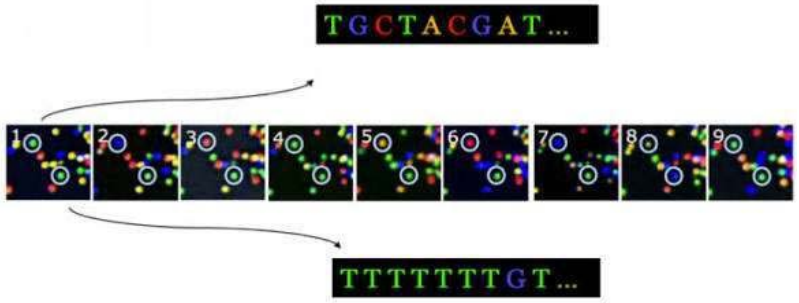


- Run time: 1–10 days
- Produces: 2–1000 Gb of sequence
- Read length: 2 x 50 bp – 2 x 250 bp (paired-end)
- Cost: $0.05–$0.40/Mb



Over 1800 publications.

- **Applications**
  - DNA sequencing
  - Gene Regulation Analysis
  - Sequencing-based Transcriptome Analysis
  - SNPs and SVs discovery
  - Cytogenetic Analysis
  - ChIP-sequencing
  - Small RNA discovery analysis



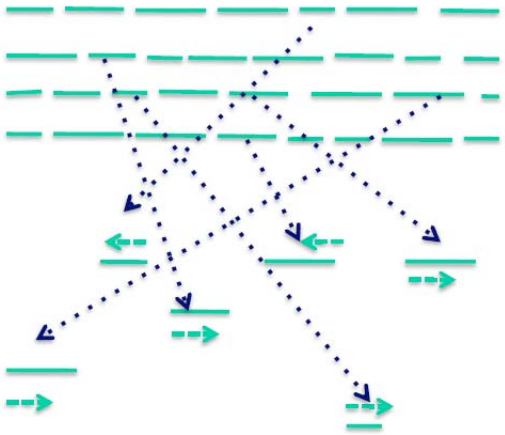
- A whole human genome sequence was determined in **8 weeks** to an average depth of ~ 40X, discovering ~ 4 new million SNPs and ~400000 SVs (with an accuracy <1% for both over-calls and under-calls)

What Is Next?

Assembly Of Sequenced Data

Multiple copies of sample DNA

Randomly fragment DNA



Sequence sample of fragments

Assemble reads

The Fragment Assembly Problem

- Given: A set of reads (strings) {*s1*, *s2*, … , *sn* }
- Do: Determine a large string *s* that "best explains" the reads

- What do we mean by "*best explains*"?
- What *assumptions* might we require?

Shortest superstring problem

- Objective: Find a string *s* such that
  - all reads *s1*, *s2*, … , *sn* are substrings of *s*
  - *s* is as short as possible

- Assumptions:
  - Reads are 100% accurate
  - Identical reads must come from the same location on the genome
  - “best” = “simplest”

Shortest superstring example

- Given the reads:
  - {ACG, CGA, CGC, CGT, GAC, GCG, GTA, TCG}

- What is the shortest superstring you can come up with? (HINT: Length=10)

ANSWER: Shortest superstring example

- Given the reads:
  - {ACG, CGA, CGC, CGT, GAC, GCG, GTA, TCG}

- What is the shortest superstring you can come up with? (HINT: Length=10)
  - **TCGACGCGTA**

Algorithms for shortest superstring problem

- This problem turns out to be *NP*-complete
  - Simple *greedy* strategy:

    while # strings > 1 do

        merge two strings with maximum overlap

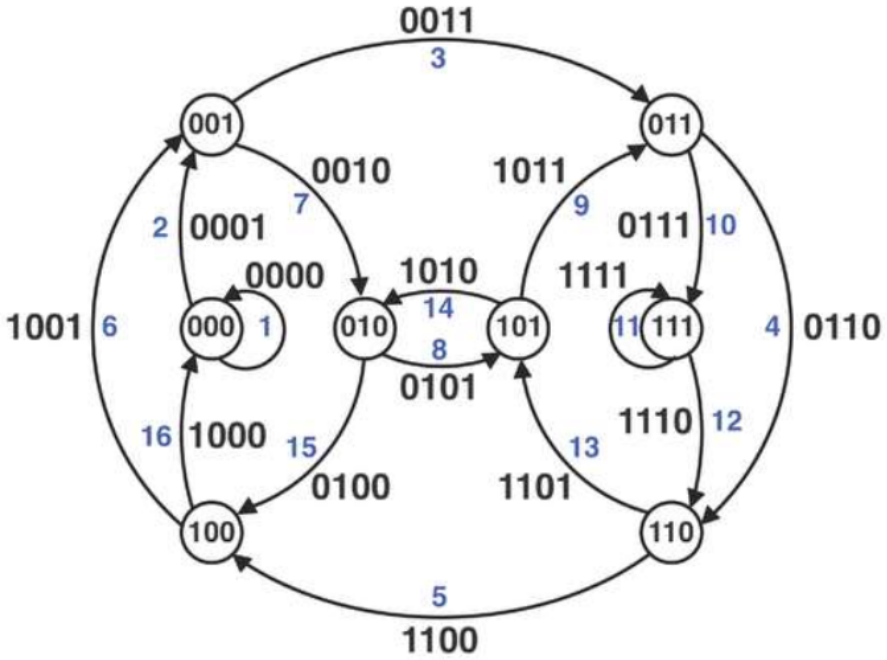
    loop
- Conjectured to give strings with

    length $\leq$ 2 x minimum length

- Other approaches are based on *graph theory… e.g., de Bruijn graphs*

What are de Bruijn graphs ?

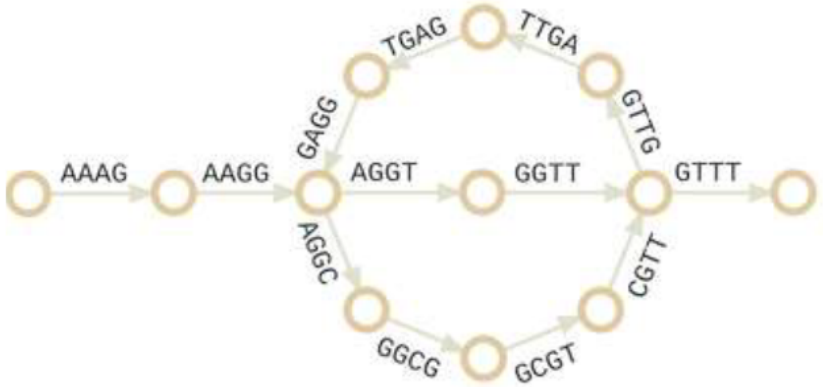
De Bruijn graph is a **directed graph** representing overlaps between sequences of symbols



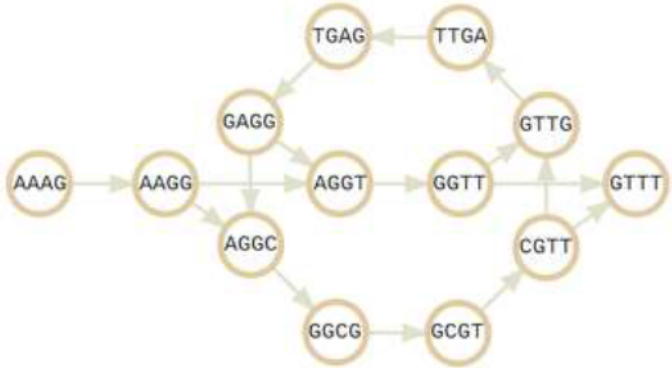
Types of De Bruijn Graphs

- Eulerian Path
- Hamiltonian Trail

B. Eulerian de Bruijn graph



C. Hamiltonian de Bruijn graph



INPUT/OUTPUT

Input :

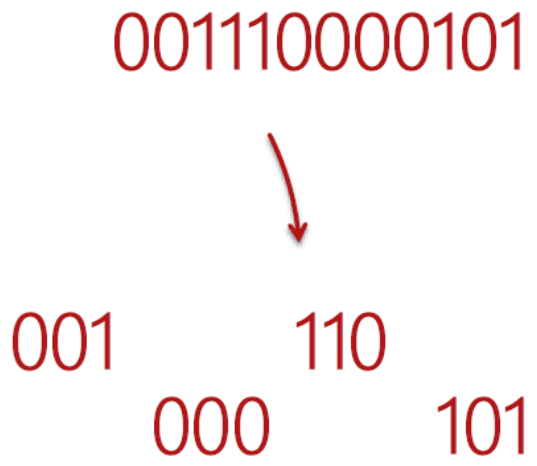
A set of k-mers patterns.

Output :

De Bruijn graphs

K-mers

- All the possible subsequences (of length k) from a read obtained through **DNA Sequencing**
- Small chunks



INPUT

- **Whole Sequence :**

  If whole sequence is given we have to divide it into k-mers

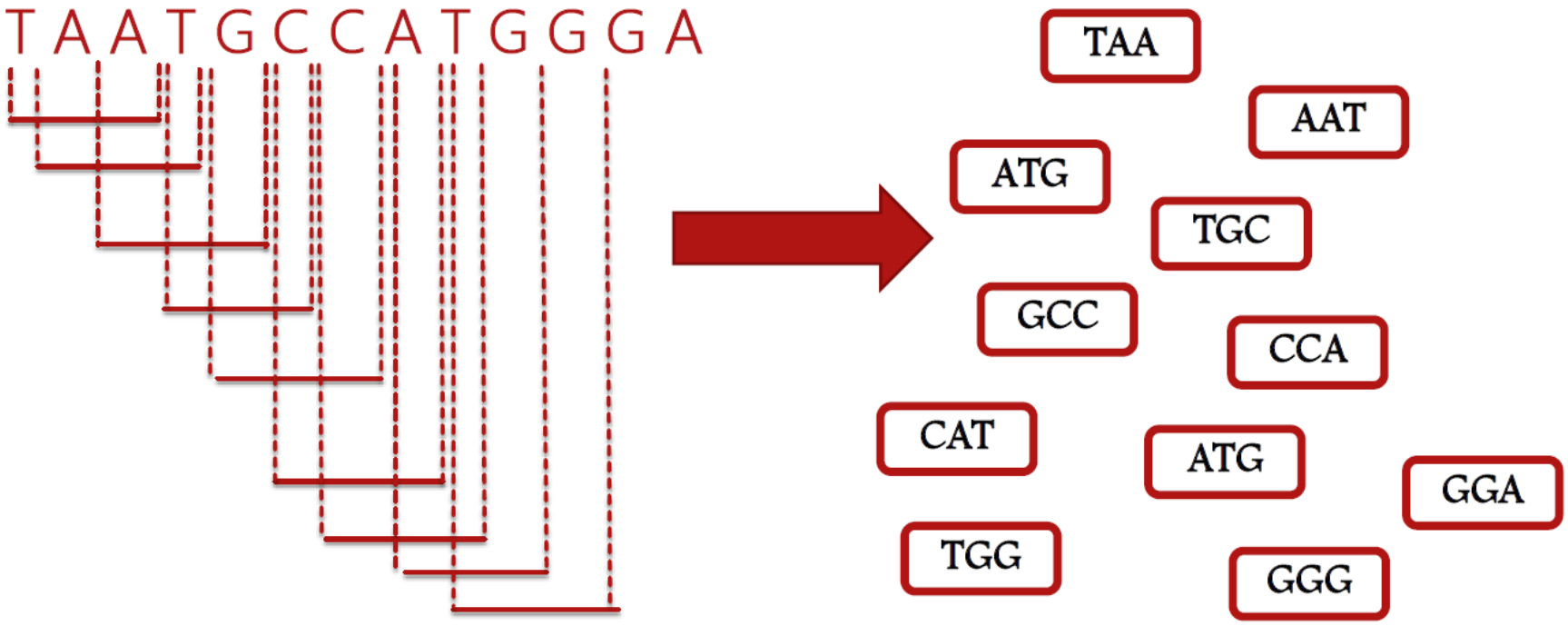
- **K-mers**

1st Step

The first step is to choose a k-mer size, and split the original sequence into its k-mer components(if sequence is given)

Size = 3

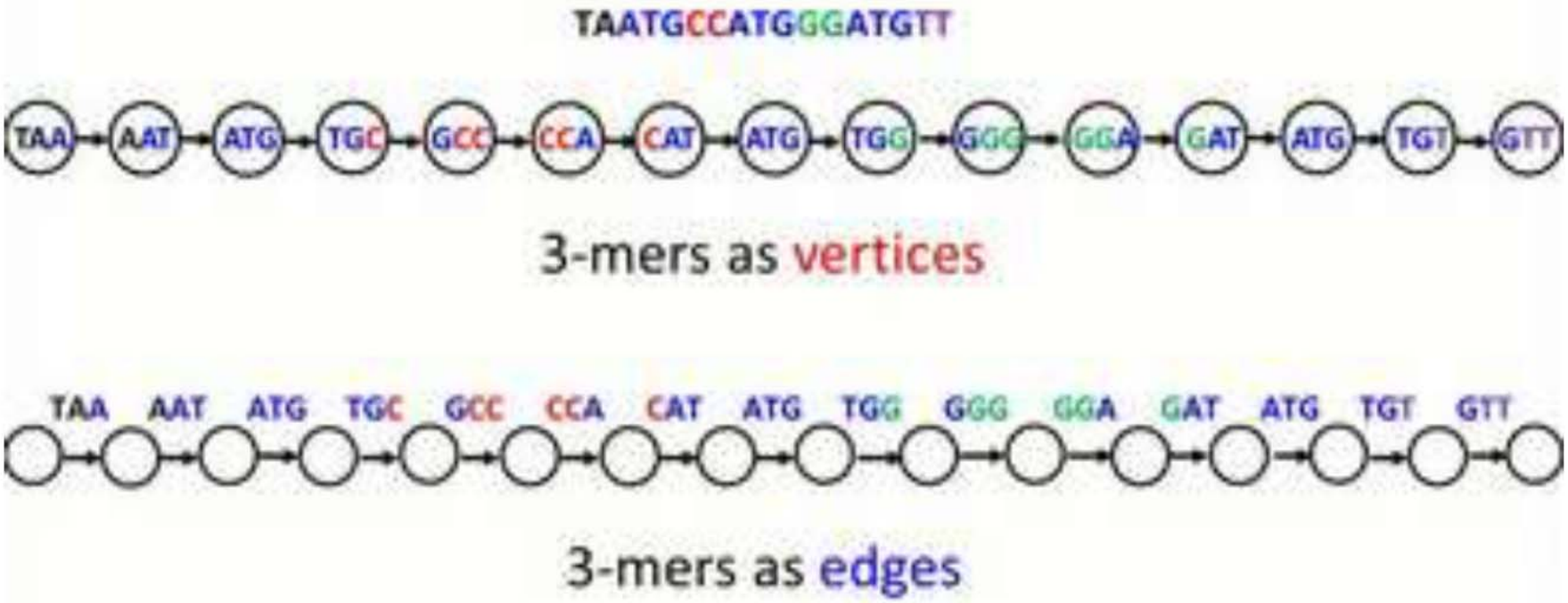
Whole sequence



Construction of graph

- Create Nodes.
- Build edges.
- Create directions.

Now: labeling edges by *k*-mers



Nodes

- We consider each k-mer as edge.
- 2 nodes for each edge

Nodes

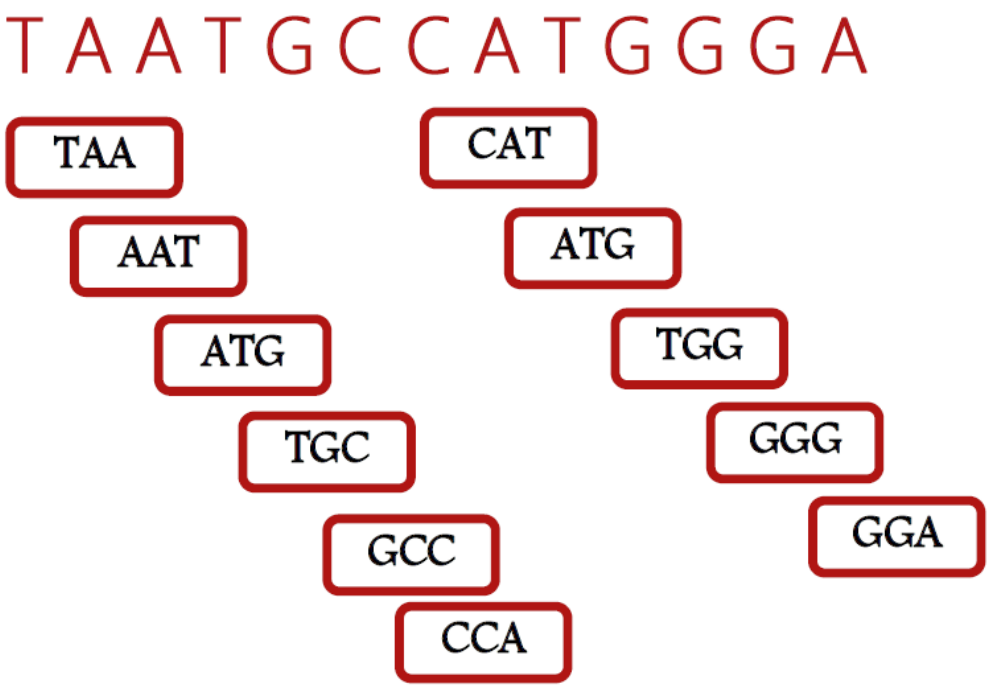
- **First :**

  Prefix of edge

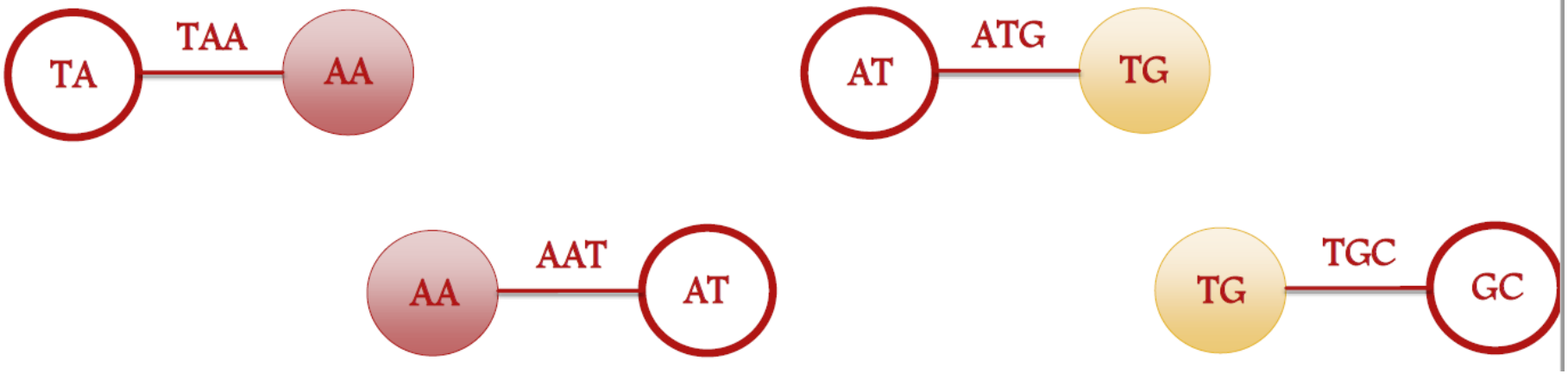
- **Second :**

  Suffix of edge

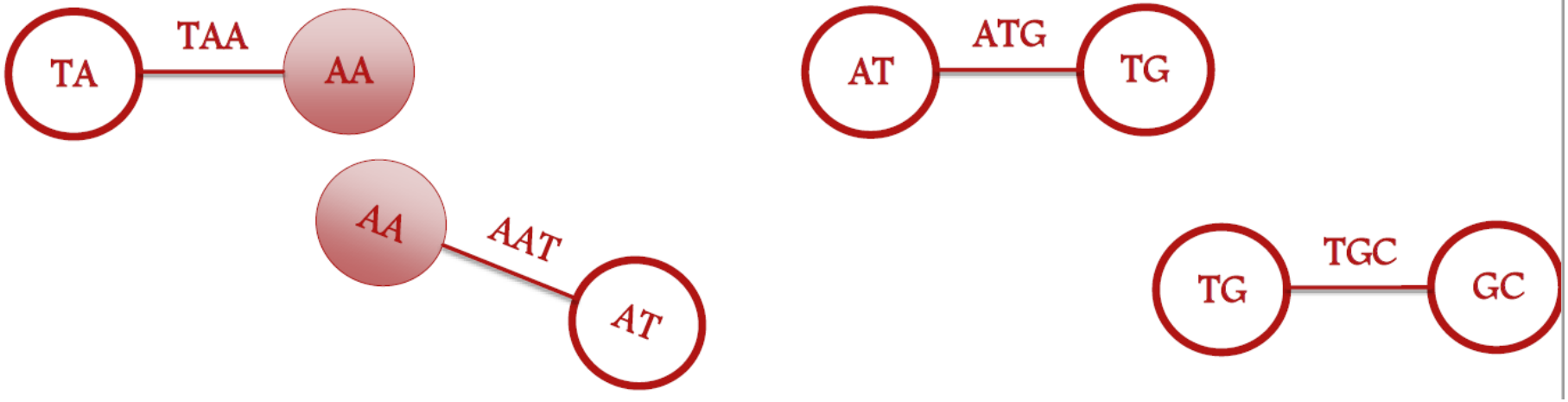
K-mers



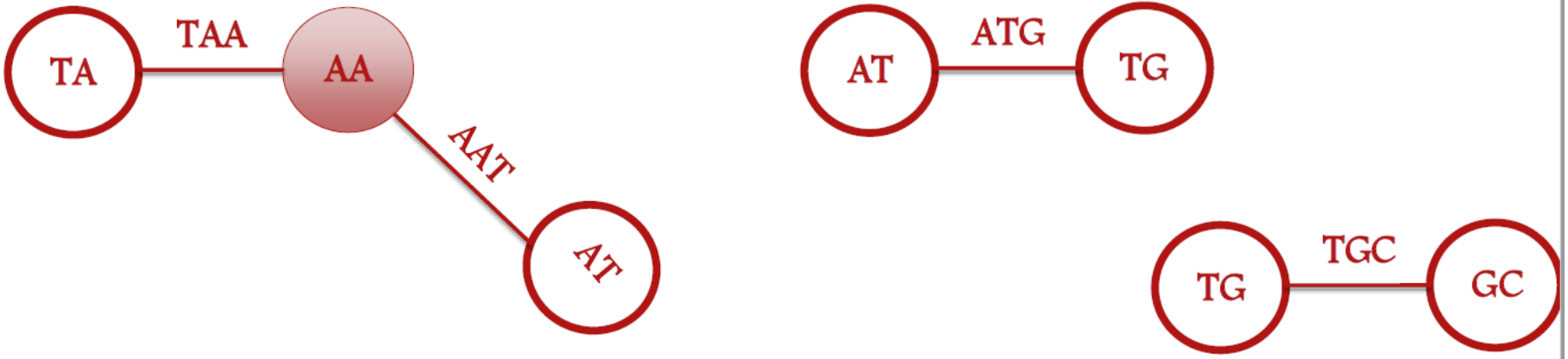
Highlight the similar nodes closer



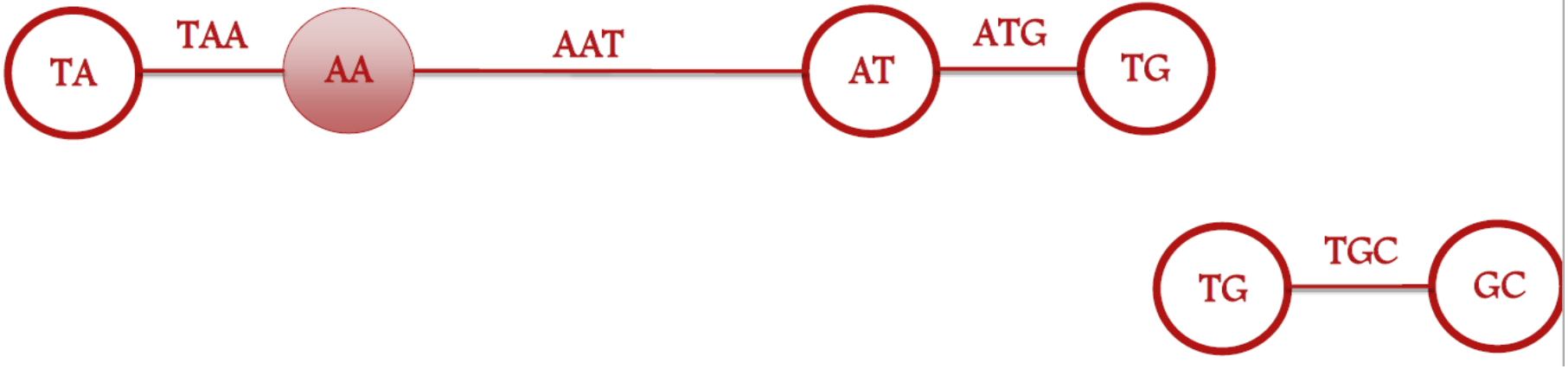
Make the similar nodes closer



Glue the similar nodes



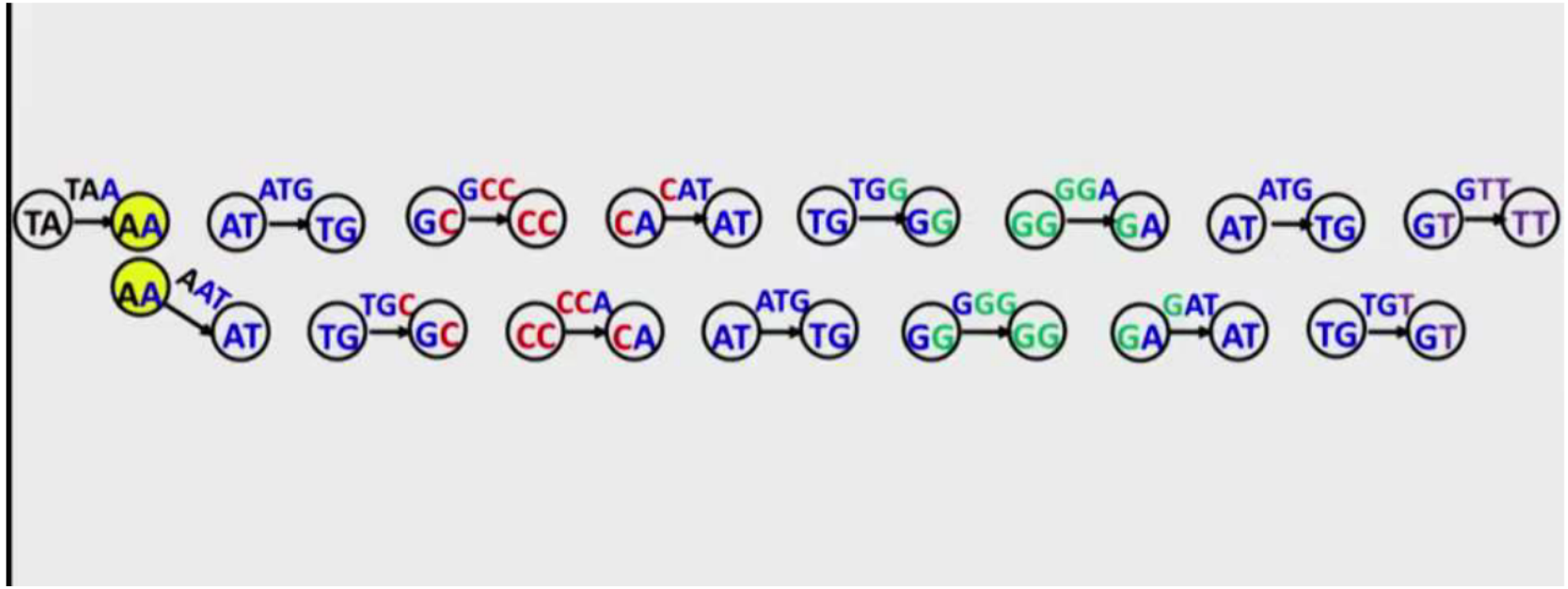
Glue the similar nodes



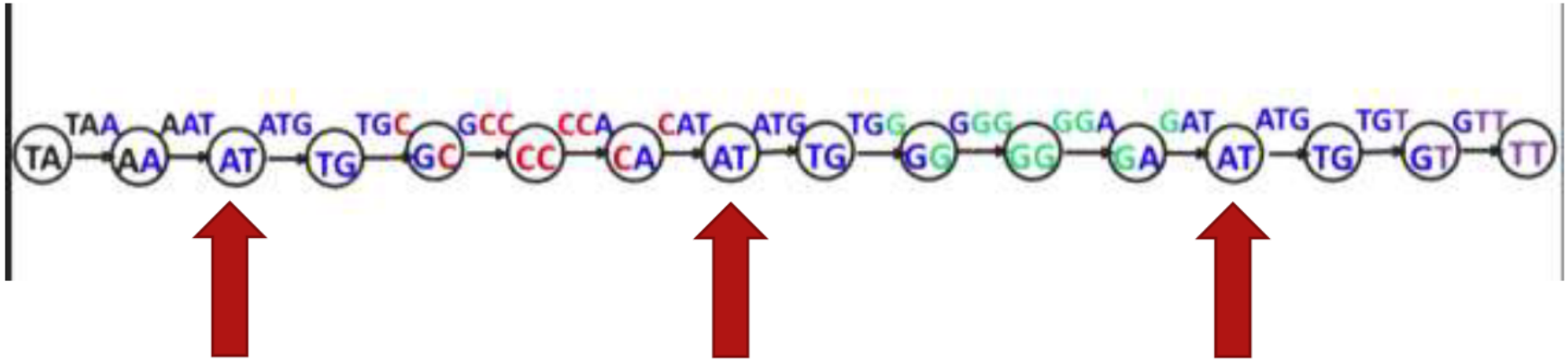
2nd Step

Then a directed graph is constructed by connecting pairs of k-mers with overlaps between the first k-1 nucleotides and the last k-1 nucleotides

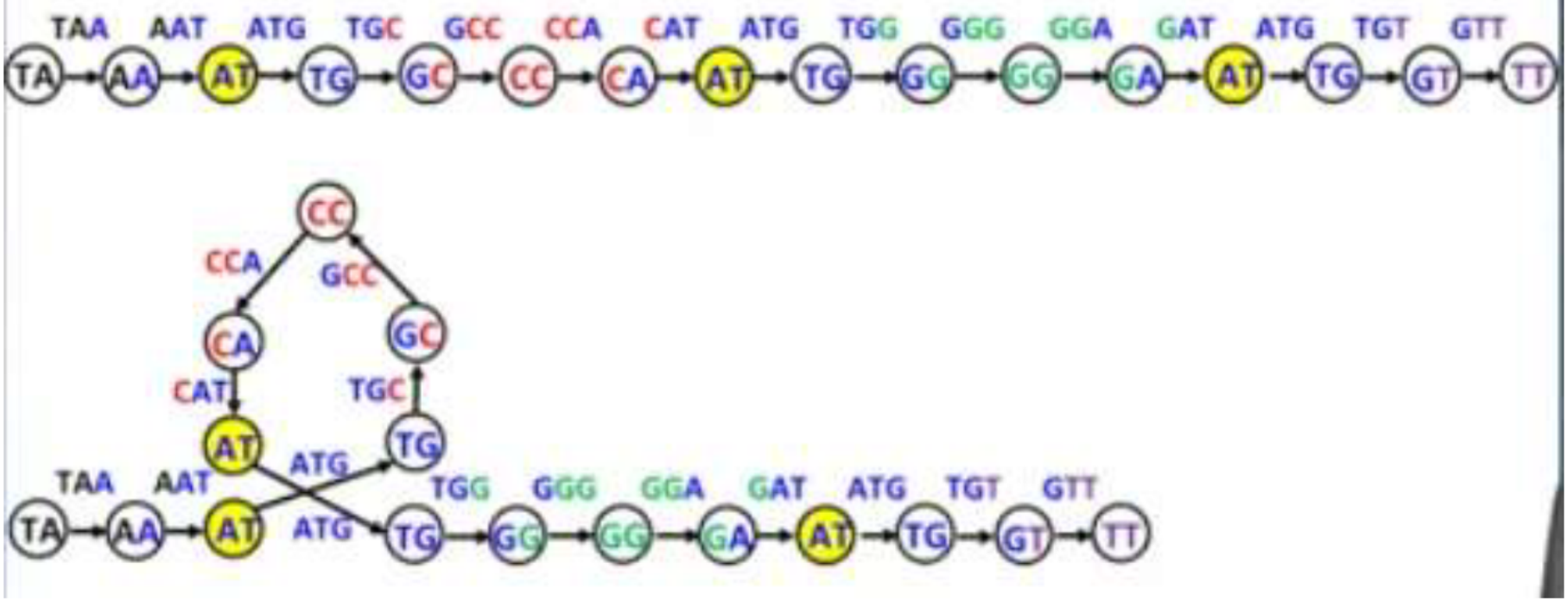
For All K-mers

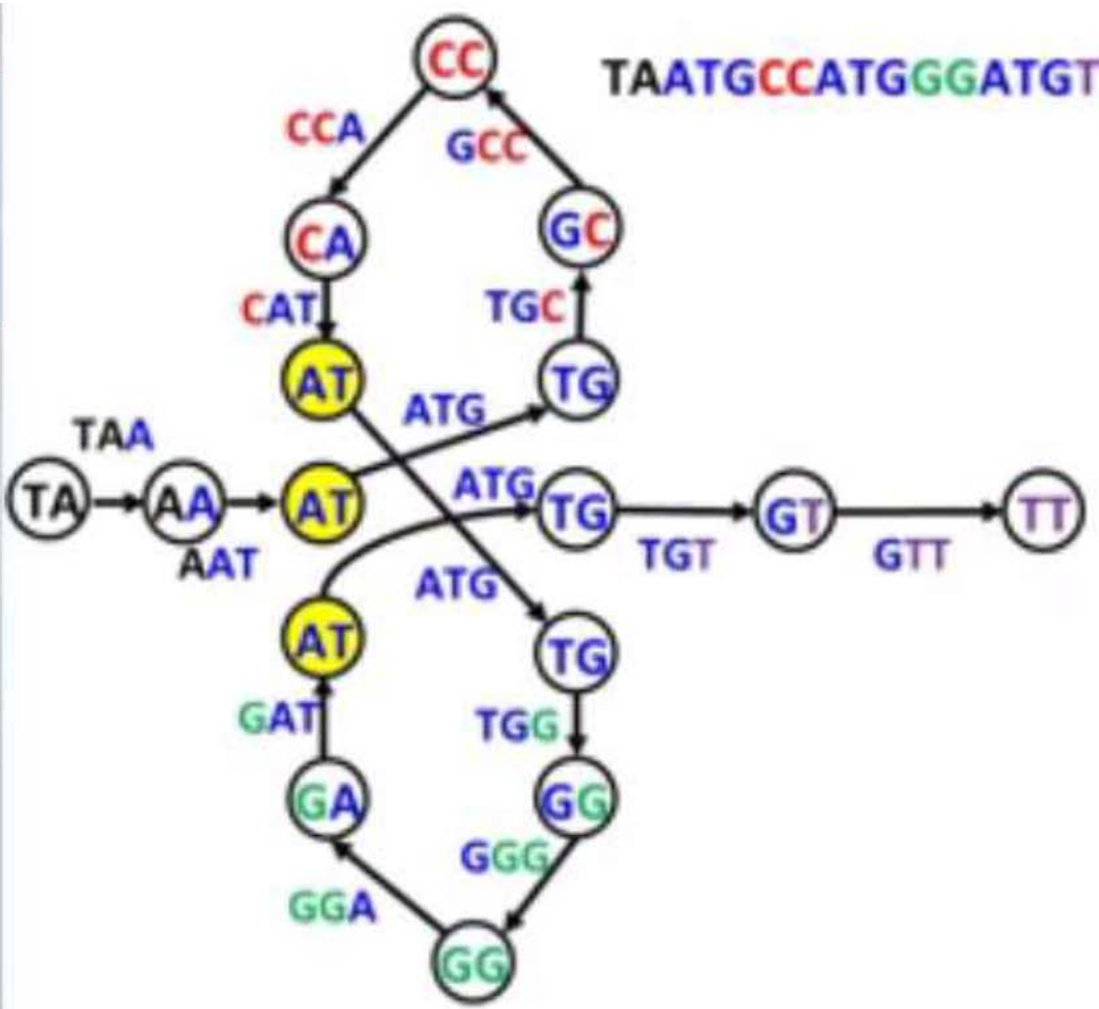


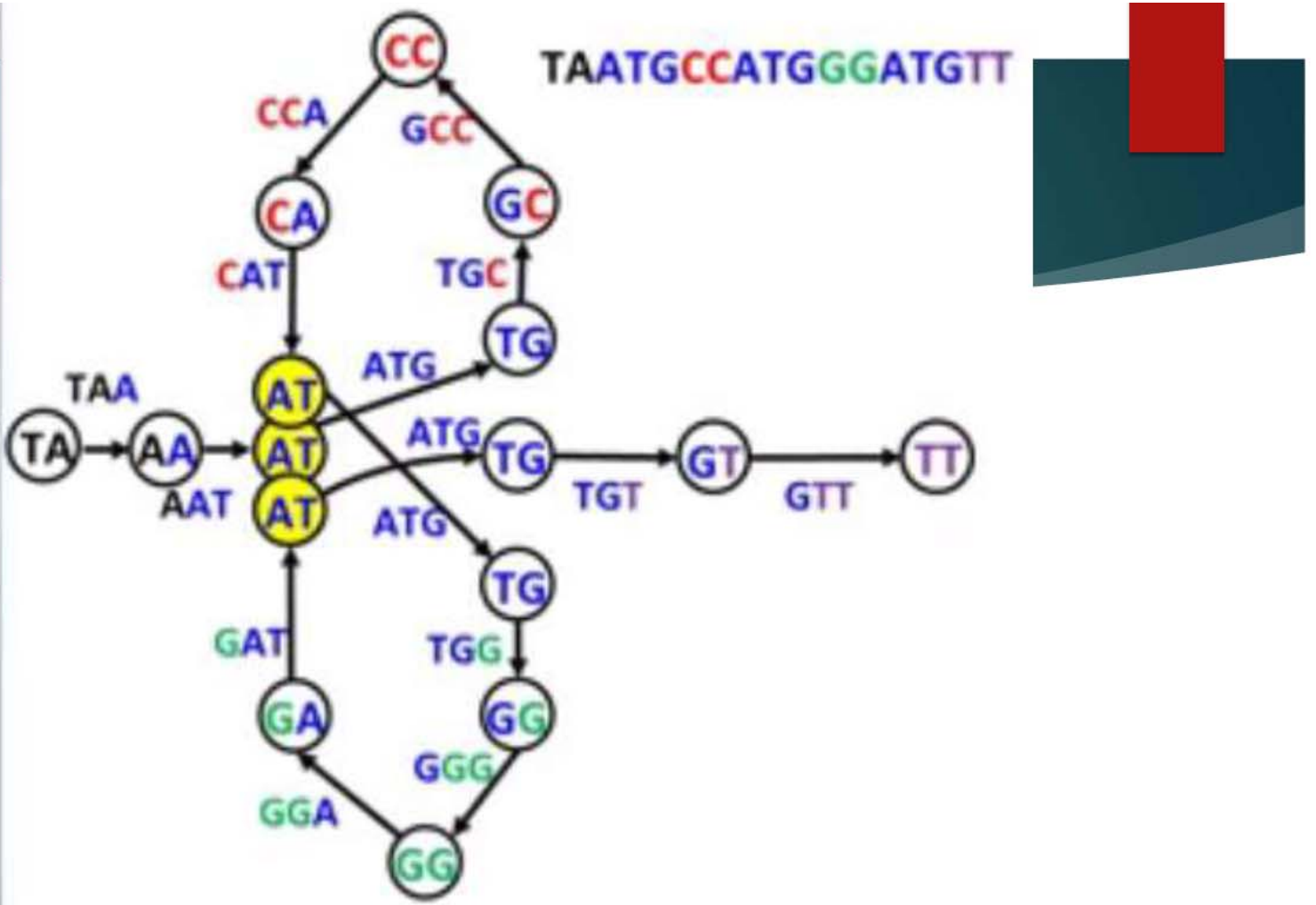
Glue the similar nodes again.

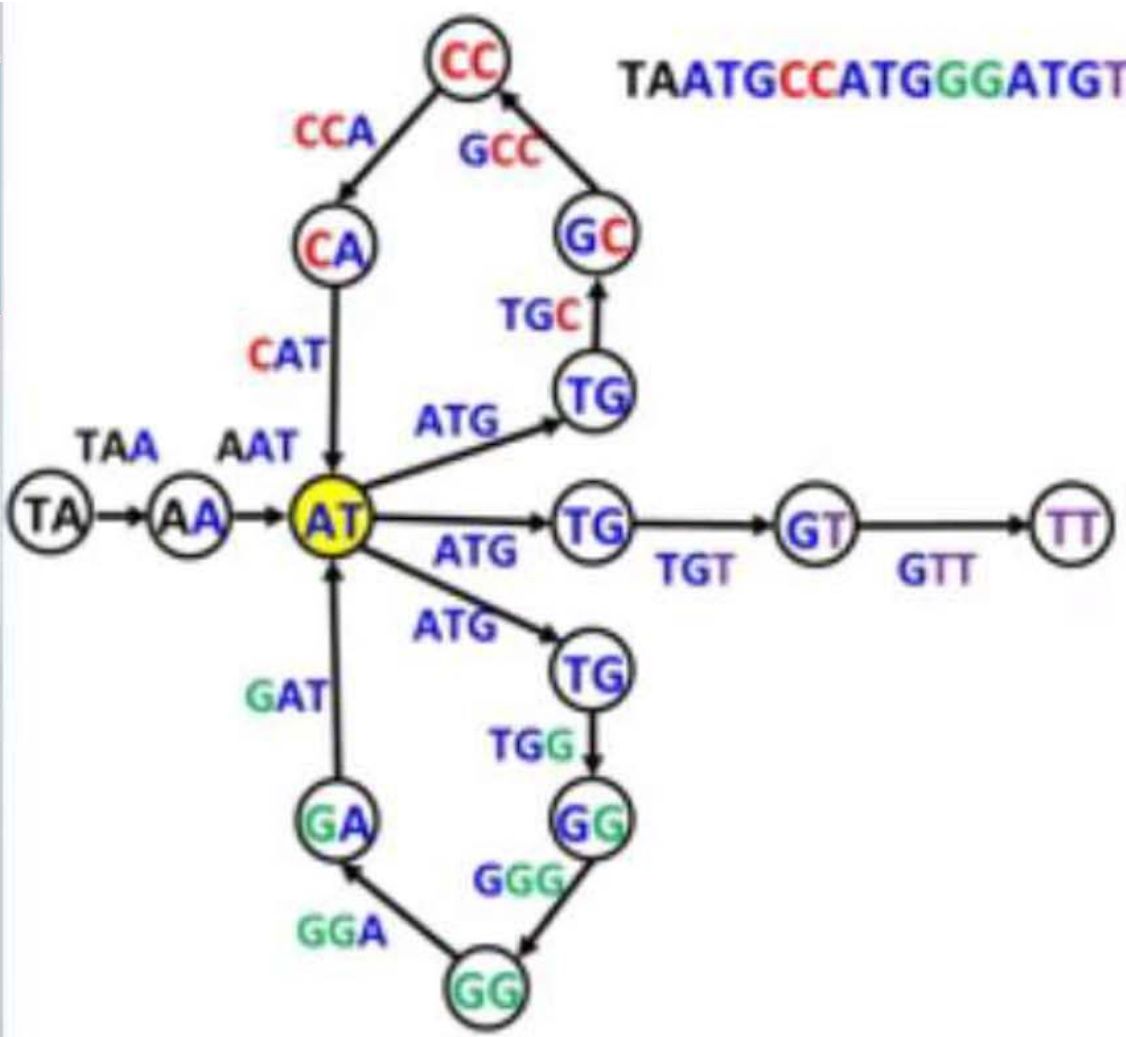


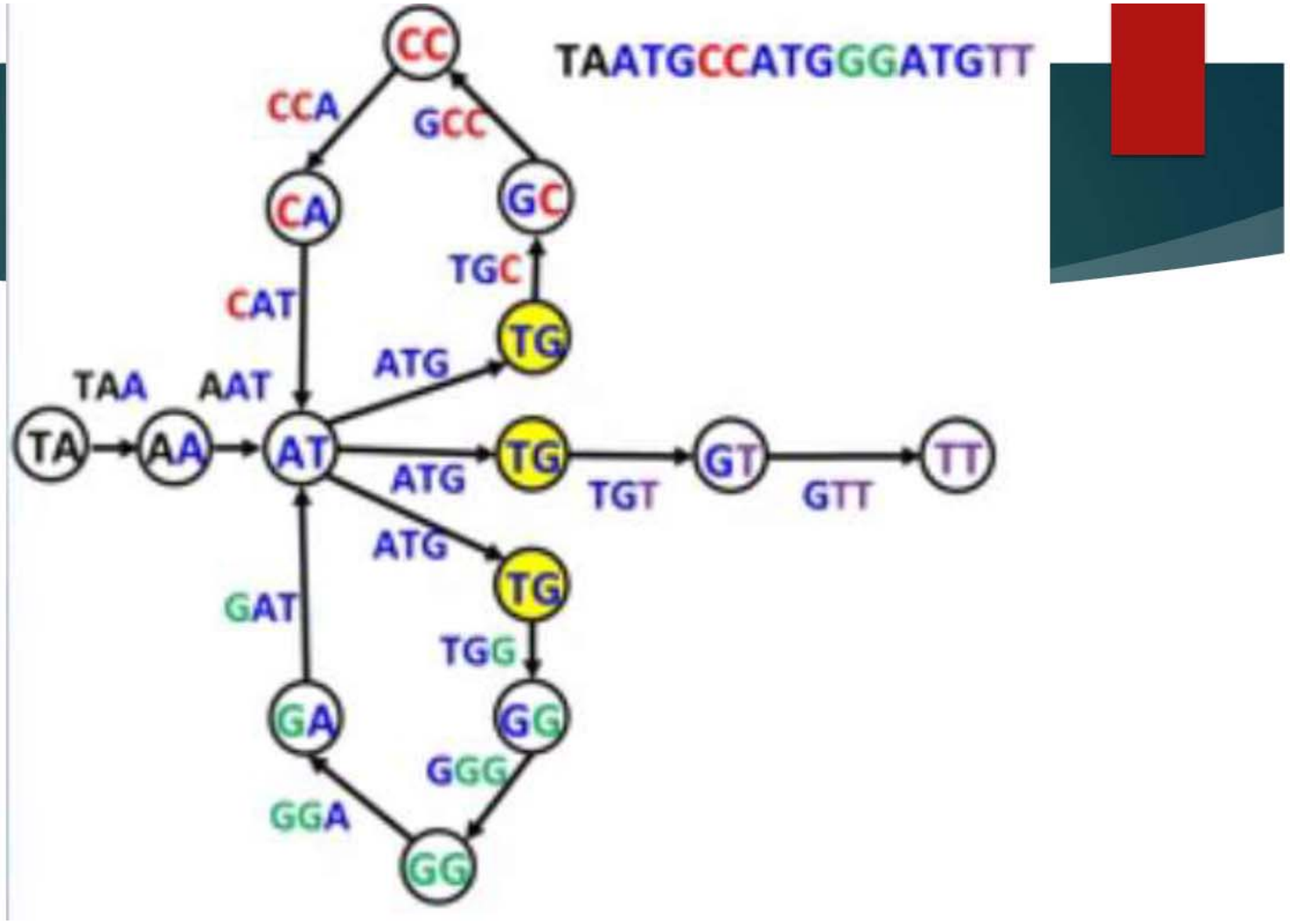
Gluing identically labeled vertices

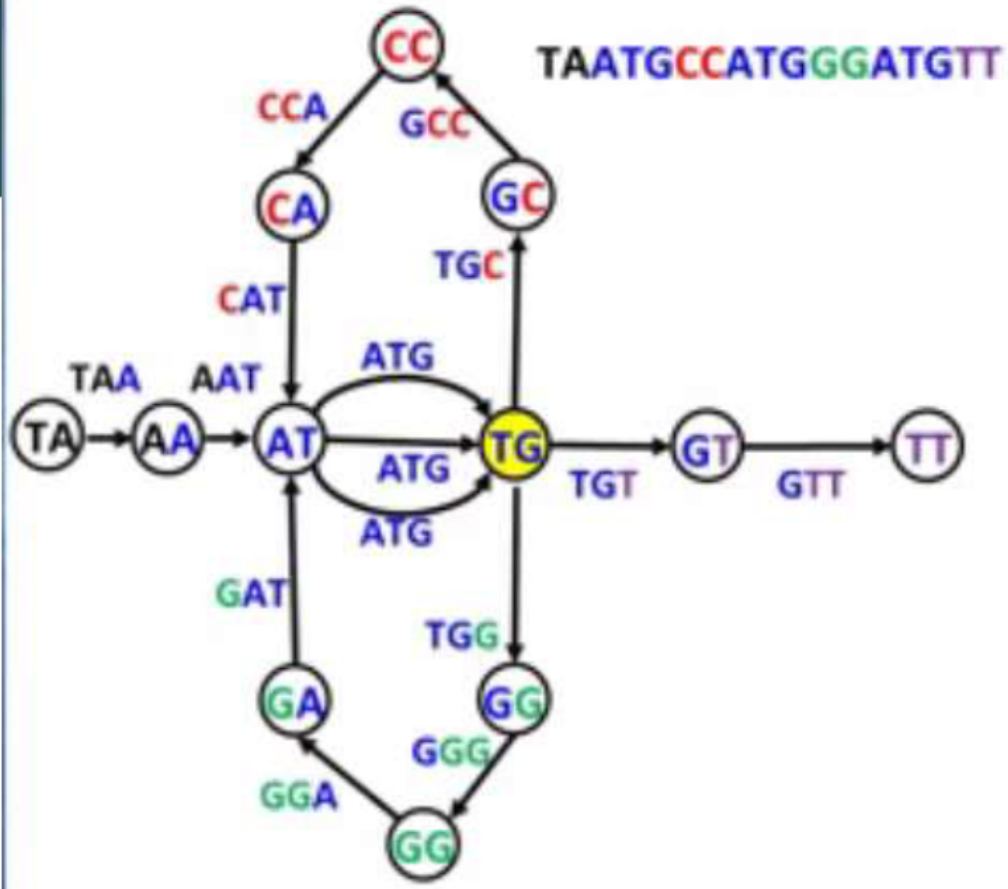


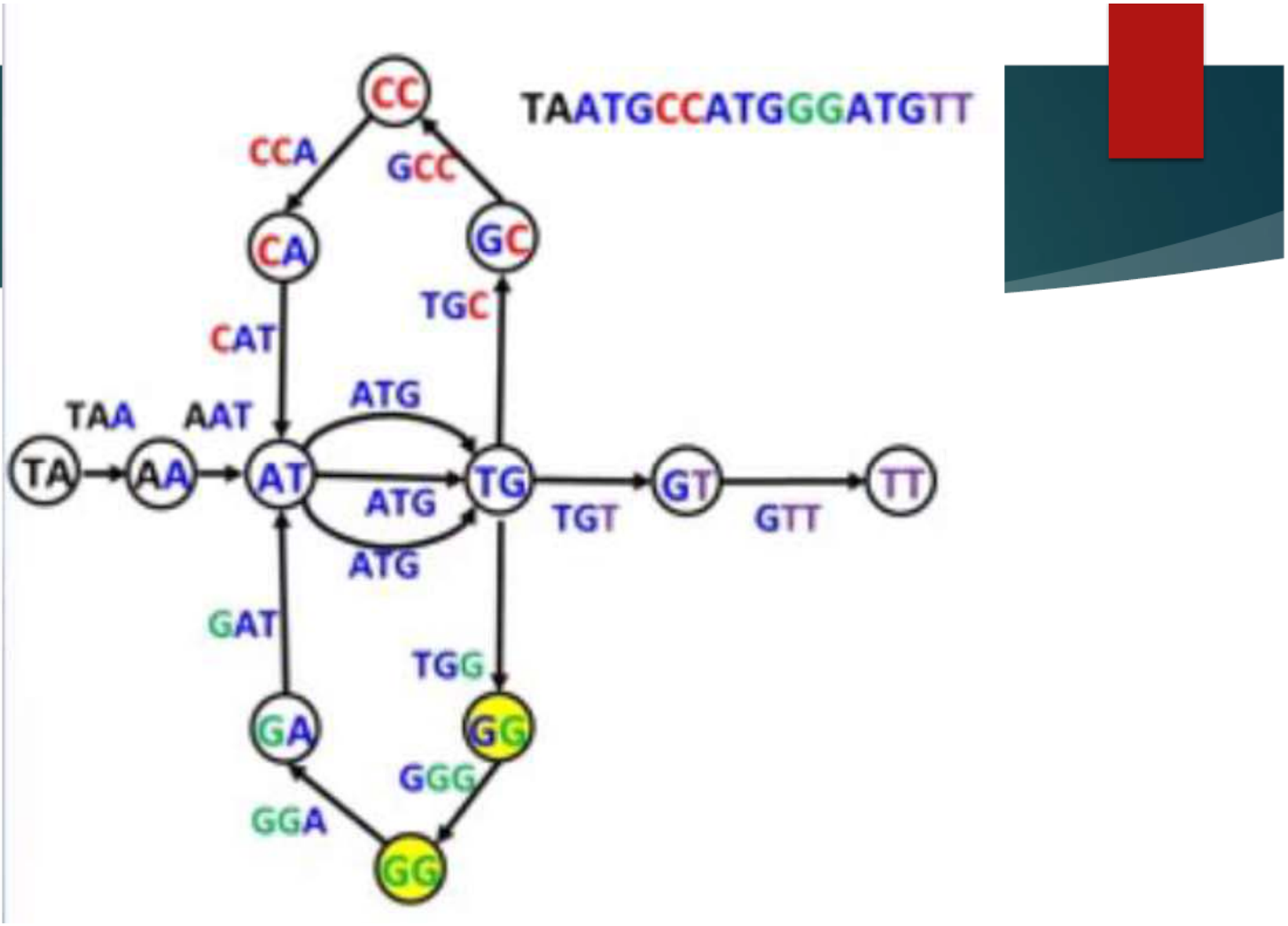

TAATGCCATGGGATGTT
CC
CCA
GCC
CA
GC
CAT
TGC
AT
TG
ATG
TAA
TA
AA
AT
ATG
TG
GT
TT
TGT
GTT
AAT
ATG
AT
TG
GAT
TGG
GA
GG
GGG
GGA
GG

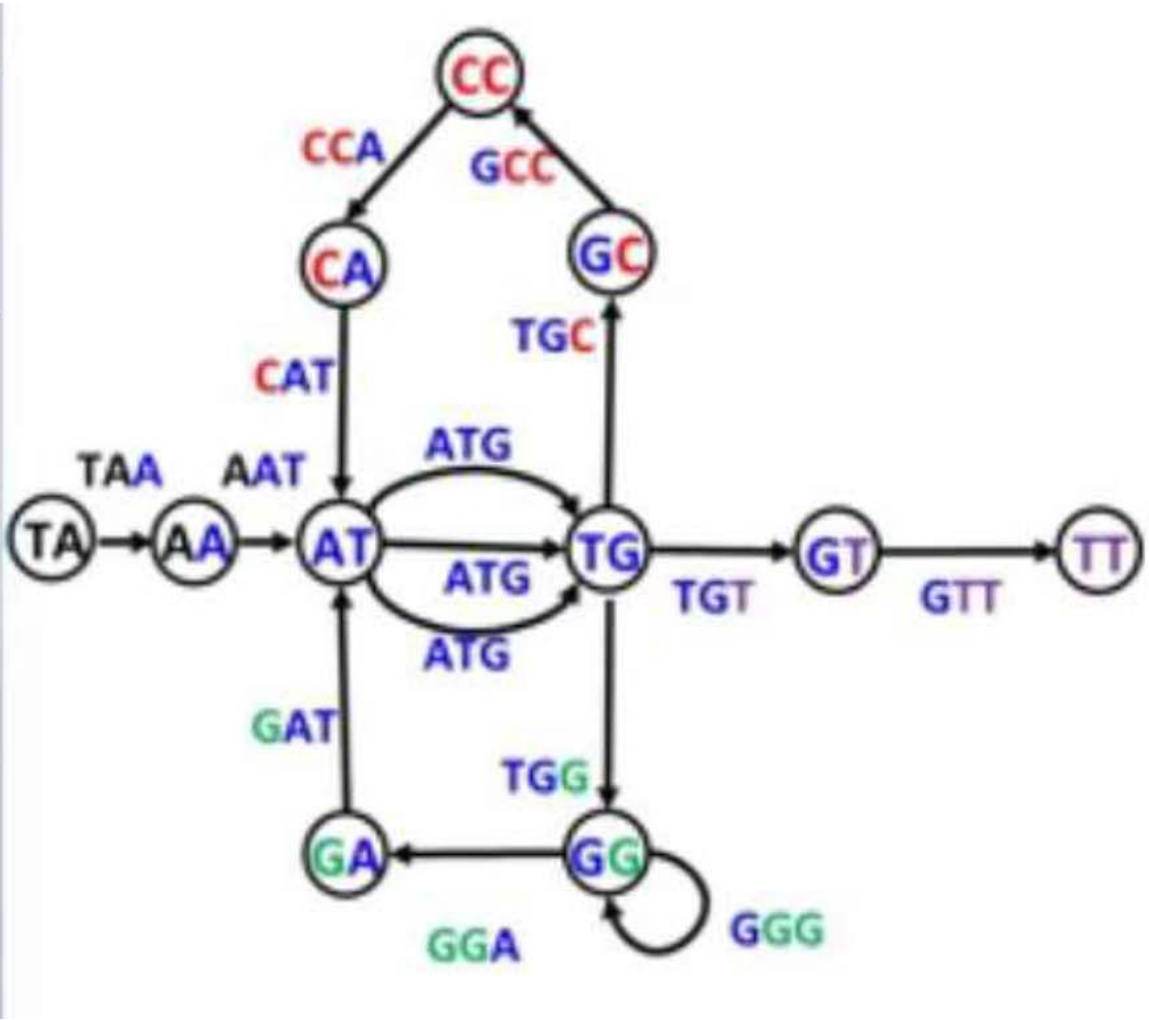

TAATGCCATGGGATGTT
CC
CCA
GCC
CA
GC
CAT
TGC
TG
ATG
TAA
AT
TA
AA
AT
ATG
TG
GT
TT
AAT
AT
TGT
GTT
ATG
TG
GAT
TGG
GA
GG
GGG
GGA
GG


TAATGCCATGGGATGTT
CC
CCA
GCC
CA
GC
TGC
CAT
TG
ATG
TAA
AAT
TA
AA
AT
TG
GT
TT
ATG
TGT
GTT
ATG
TG
GAT
TGG
GA
GG
GGG
GGA
GG


TAATGCCATGGGATGTT
CC
CCA
GCC
CA
GC
TGC
CAT
TG
ATG
TAA
AAT
TA
AA
AT
TG
GT
TT
ATG
TGT
GTT
ATG
TG
GAT
TGG
GA
GG
GGG
GGA
GG


CC
TAATGCCATGGGATGTT
CCA
GCC
CA
GC
TGC
CAT
ATG
TAA
AAT
TA
AA
AT
TG
GT
TT
ATG
TGT
GTT
ATG
GAT
TGG
GA
GG
GGG
GGA
GG


TAATGCCATGGGATGTT
CC
CCA
GCC
CA
GC
TGC
CAT
TAA
AAT
ATG
TA
AA
AT
TG
GT
TT
ATG
TGT
GTT
ATG
GAT
TGG
GA
GG
GGG
GGA
GG

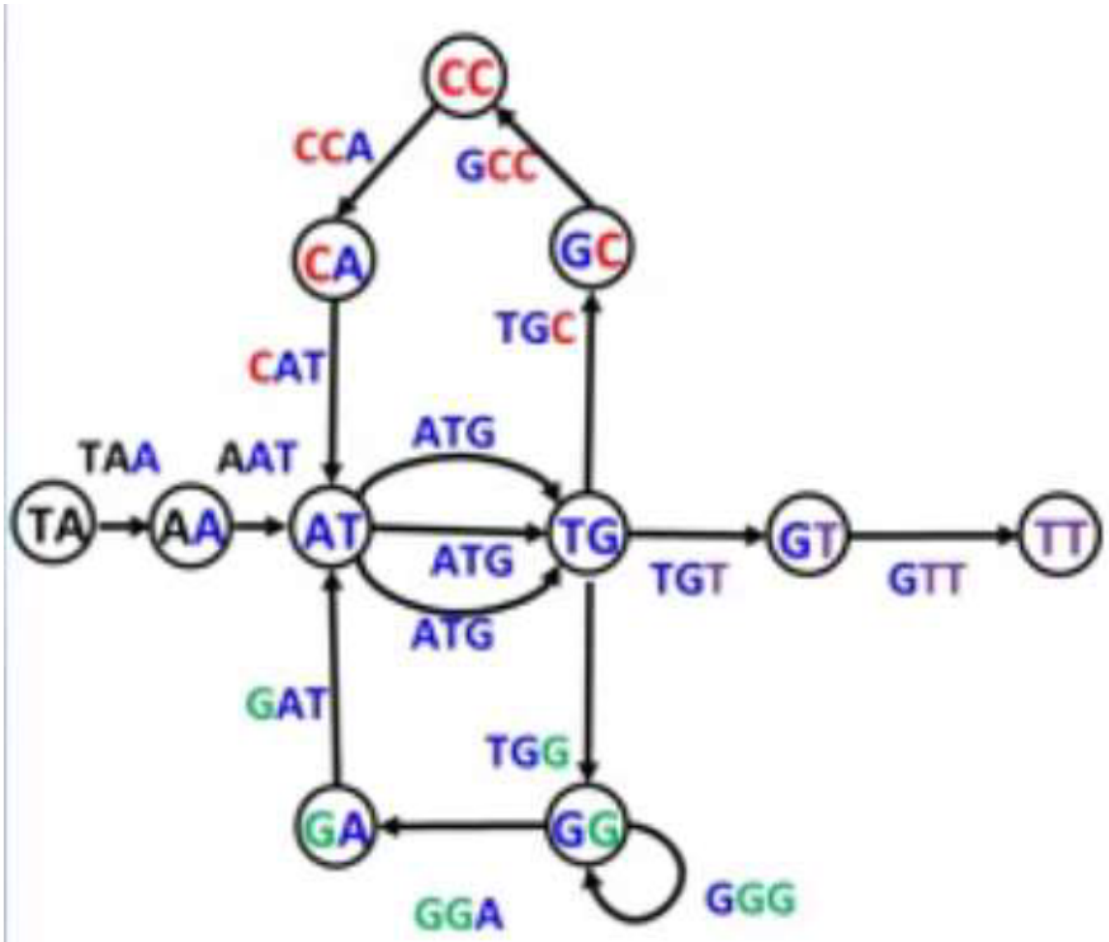

CC
CCA
GCC
CA
GC
TGC
CAT
ATG
TAA
AAT
TA
AA
AT
TG
GT
TT
ATG
TGT
GTT
ATG
GAT
TGG
GA
GG
GGA
GGG

Construction of Genome

- Use whole sequence of first edge

Then

- The Only Use the last alphabet from the intermediate edges.
- Build the genome.



T A A T G C C A T G G G A

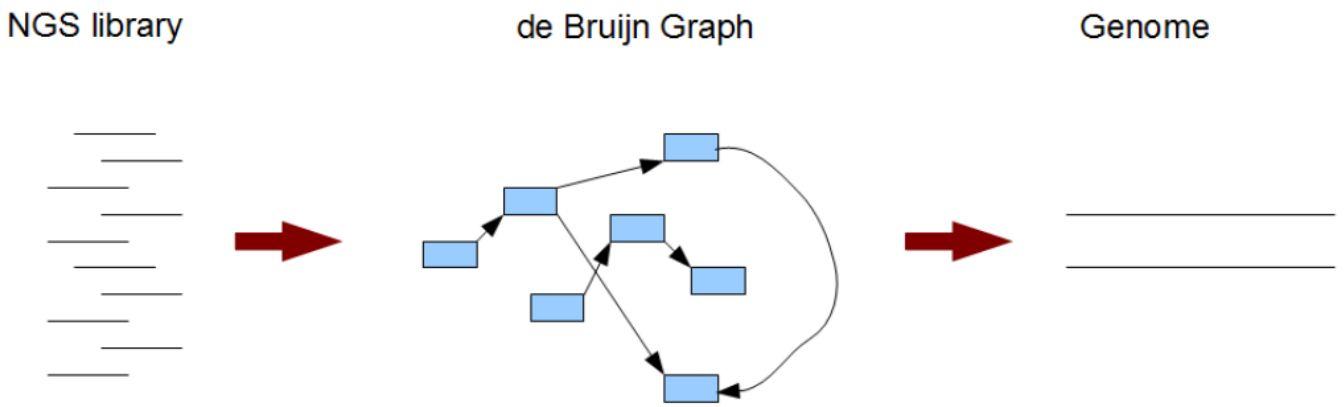
Lets Try An Example

- Consider the read: ATGGCGTGCA
- Build the de Bruijn graph for k=3

ANSWER: Lets Try An Example

- Consider the read: ATGGCGTGCA
- Build the de Bruijn graph for k=3

De bruijn graph based genome assembly algorithm



De Bruijn graph

No: graph can have multiple Eulerian walks, only one of which corresponds to original superstring

Right: graph for ZABCDABEFABY, $k = 3$

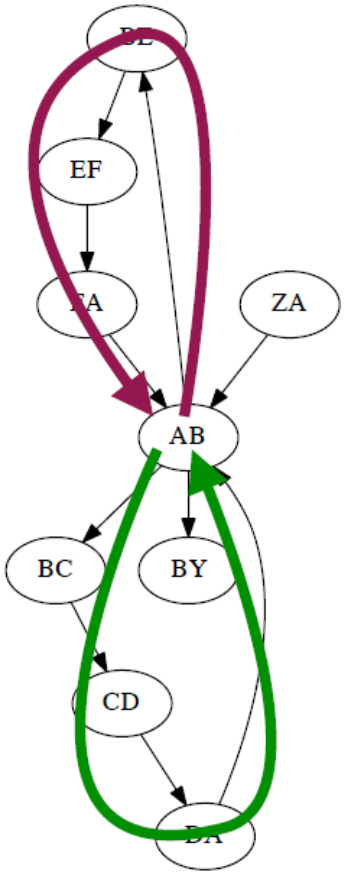
Alternative Eulerian walks:

ZA → AB → BE → EF → FA → AB → BC → CD → DA → AB → BY

ZA → AB → BC → CD → DA → AB → BE → EF → FA → AB → BY

These correspond to two edge-disjoint directed cycles joined by node AB

AB is a repeat: ZABCDABEFABY



De Bruijn graph

Casting assembly as Eulerian walk is appealing, but not practical

- Uneven coverage, sequencing errors, etc make graph non-Eulerian
- Even if graph were Eulerian, repeats yield many possible walks

> Kingsford, Carl, Michael C. Schatz, and Mihai Pop. "Assembly complexity of prokaryotic genomes using short reads." *BMC bioinformatics* 11.1 (2010): 21.

De Bruijn Superwalk Problem (DBSP) is an improved formulation where we seek a walk over the De Bruijn graph, where walk contains each read as a *subwalk*

- Proven NP-hard!

> Medvedev, Paul, et al. "Computability of models for sequence assembly." *Algorithms in Bioinformatics*. Springer Berlin Heidelberg, 2007. 289-301.

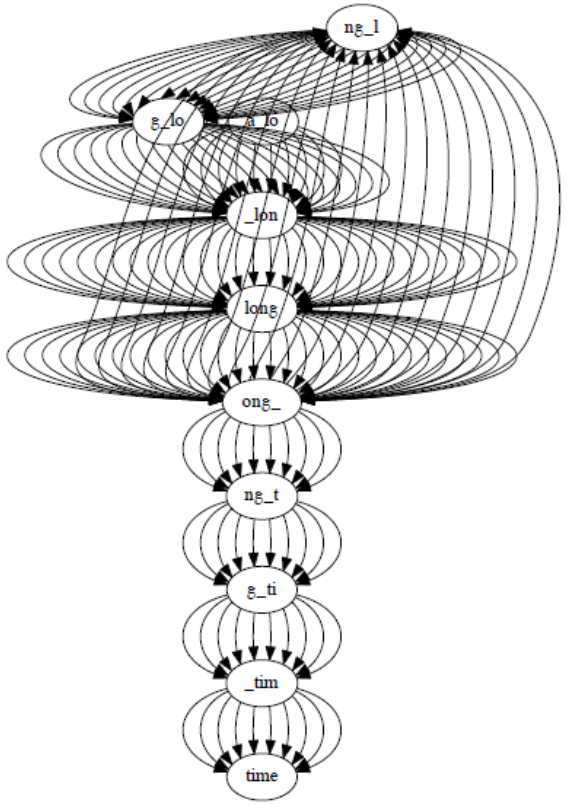
De Bruijn graph

In practice, De Bruijn graph-based tools give up on unresolvable repeats and yield fragmented assemblies, just like OLC tools.

But first we note that using the De Bruijn graph representation has **other advantages**...

De Bruijn graph

In typical assembly projects,
average coverage is ~ 30 - 50



Remaining Genome Assembly Steps

- Mapping: Identify genes and where they are on the genome.
- Annotation: Describe mapped genes, chromosomes, …
- SNV calling: Identify variations

de Bruijn Graph Building Steps

- Break read into k-mers
- Make k-mers edges in graph and prefix node the k-1 prefix of the k-mer, and the suffix node the k-1 suffix. E.g., ABC –ABCD-> BCD
- Merge similar nodes
- Traversed through each edge only once to retrieve the smallest encompassing sequence that yielded the reads.

NEXT: CRISPR