Genome Assembly problem in the language for computer scientists

Genome Sequencing Problem

Reconstruction a genome from reads

input: A collection fo string called *Reads*

output: A string *Genome* reconstructed from *Reads*

This is not a computational problem!!!

Reconstruction strings from a k-mers.

String Composition Problem

Generate the k-mer compositions of a string

input: A string Text and a integer k

output: COMPOSITION$_{\mathbf{k}}$(Text), where the k-mers are arranged in lexicographic order (since the Position is not available)

Given a a string *Text*, its k-mer composition COMPOSITION$_{\mathbf{k}}$(*Text*) is the collection of all k-mer substring of Text

COMPOSITION**3** (**TAATGTT**) ={AAT,ATG,GTT,TAA,TGT}

Reconstruction strings from a k-mers.

String Reconstruction Problem, for genome assembly purpose

Reconstruction a string from its k-mer composition

input: A collection of k-mers.

output: A *Genome* such that COMPOSITION$_{\mathbf{k}}$(Genome) is equal to the collection of k-mers.

Reconstruction strings from a k-mers.

String Reconstruction Problem

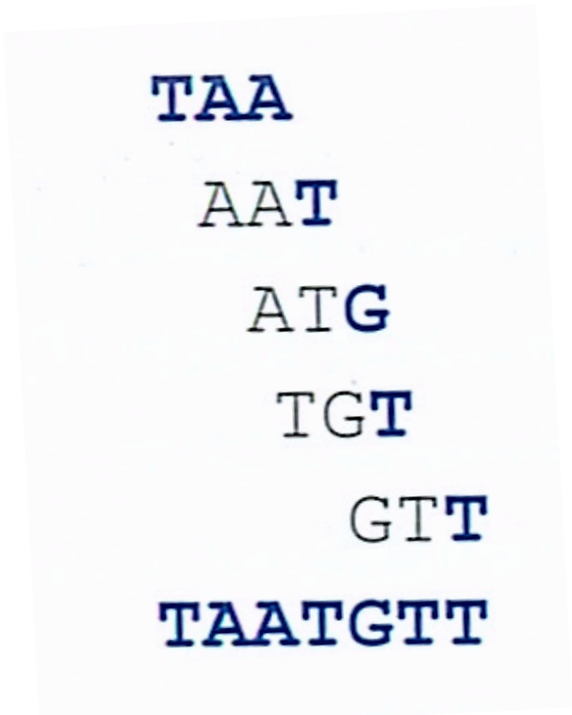
Reconstruct a string from its k-mer composition

input: An integer k and a collection Patterns of k-mers

output: A string Text with k-mer composition equal to Patterns

String reconstruction problem is perform connecting a pair of k-mers if they overlap on k-1 symbols.

TAATGTT



Reconstruction strings from a k-mers.

TAATGCCATGGATGTT

```
TAA
 AAT
  ATG
TAATG
```

```
TAA
 AAT
  ATG
   TGT
TAATGT
```

```
TAA
 AAT
  ATG
   TGT
    GTT
TAATGTT
```

Unfortunately, we are stuck at GTT since no 3-mers in the set of k-mers composition start with TT.

```
TAA
 AAT
  ATG
   TGC
TAATGC
```

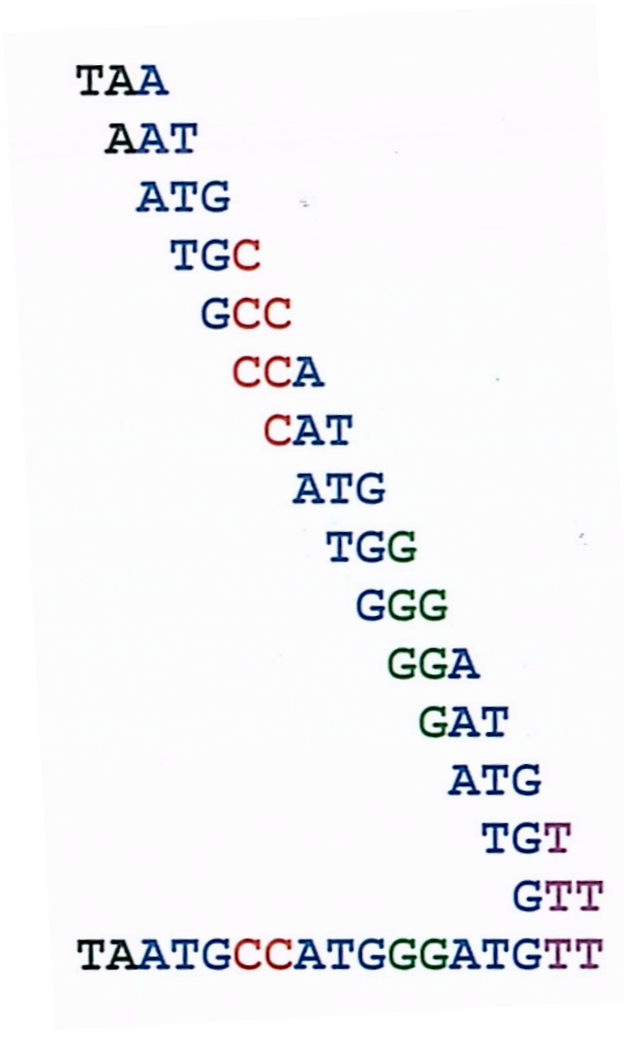
continuing the precess,
we obtain the following assembly

```
TAA
 AAT
  ATG
   TGC
    GCC
     CCA
      CAT
       ATG
        TGG
         GGA
          GAT
           ATG
            TGT
             GTT
TAATGCCATGGATGTT
```

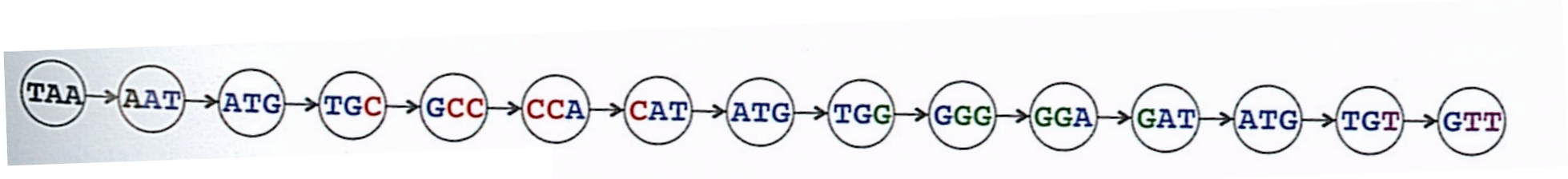
Repeats complicate genome assembly

From a string to a graph

In blue repeated regions.



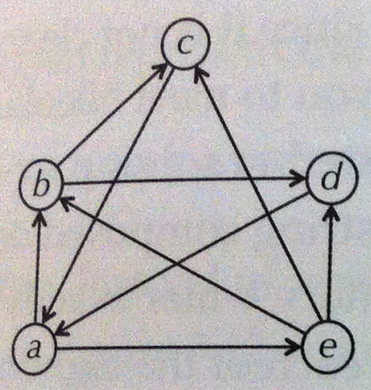
The 15 color-code 3-mers making the sequence as a graph to form a genome path according to their order on the genome.



Two graph representations

There are two standard ways of representing a graph. Fro directed graph with n nodes, the nXn adjacency matrix ($A_{i,j}$) is defined by the following rule: $A_{i,j}=1$ if a directed edge connects node i to node j, and $A_{i,j}=0$ otherwise. Another way of representing a graph is use an adjacency list, for which we simply list all nodes connected to each node.

Graph



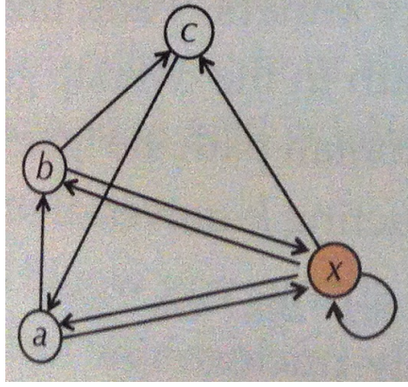
Adjacency Matrix

	a	b	c	d	e
a	0	1	0	0	1
b	0	0	1	1	0
c	1	0	0	0	0
d	1	0	0	0	0
e	0	1	1	1	0

Adjacency List

a is adjacent to *b* and e
b is adjacent to c and *d*
c is adjacent to *a*
d is adjacent to *a*
e is adjacent to *b*, c, and *d*

Graph



Adjacency Matrix

	a	b	c	x
a	0	1	0	1
b	0	0	1	1
c	1	0	0	0
x	1	1	1	1

Adjacency List

a is adjacent to *b* and *x*
b is adjacent to c and *x*
c is adjacent to *a*
x is adjacent to *a*, *b*, *c*, and *x*

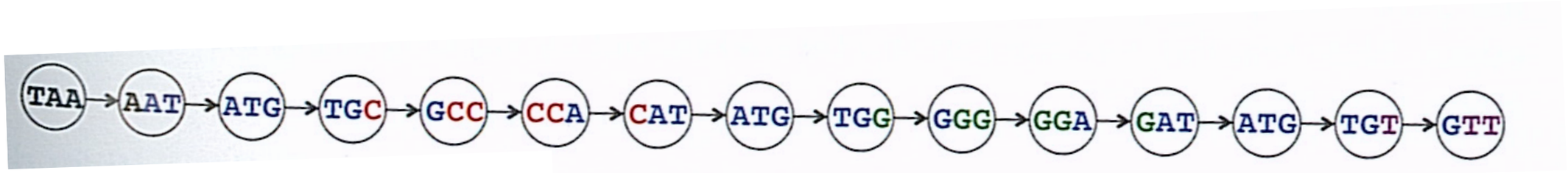
From a string to a graph

String Spelled by a genome path Problem

Reconstruct a string from its genome path

input: A sequence of k-mers $Pattern_1,\ldots Pattern_n$ such that the last k-1 symbols of $Pattern_i$ are equal to the first k-1 symbols of $Pattern_{i+1}$ for 1 <= n-1

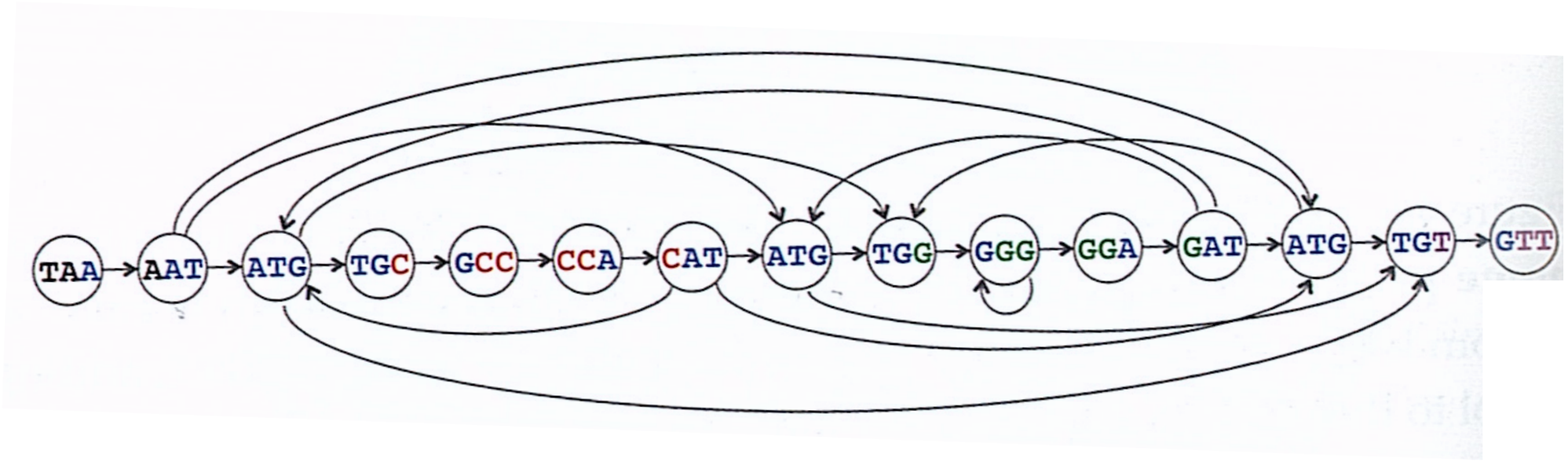
output: A string Text of length k+n-1 such that the i-th k-mer in Text is equal to $Pattern_i$ (for i <= i <=n)



The string's genome path from its k-mer composition we will use an arrow to connect any k-mer Pattern to a k-mer Pattern' if the suffix of Pattern' is equal to the prefix of Pattern'.

Suffix(TAA) = Prefix(AAT) = AA

From a string to a graph



This direct graph represents all overlap connection between nodes representing the 3-mer composition of the string **TAATGCCATGGGATGTT**

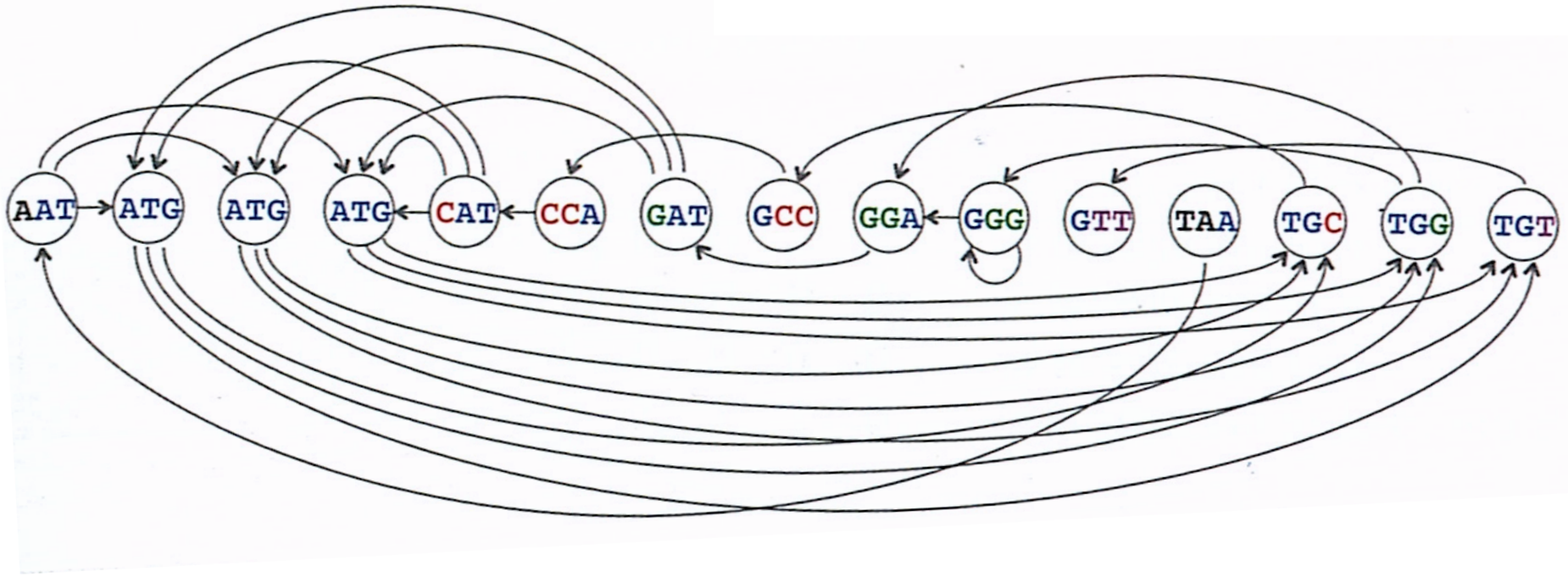
Note that: **ATG** can be connected with: **TGG**

TGC

TGT

From a string to a graph

In the previous graph it is possible follows the horizontal path to define the genome sequence, but if we want define the overlap graph with 3-mer o**rdered in lexicographical way**, the path vanish.



Overlap graph problem To generalise the construction of the graph, we form a node for each k-mer in Patterns and connect the k-mers by a direct edge if the suffix is equal to the prefix => **overlap graph**

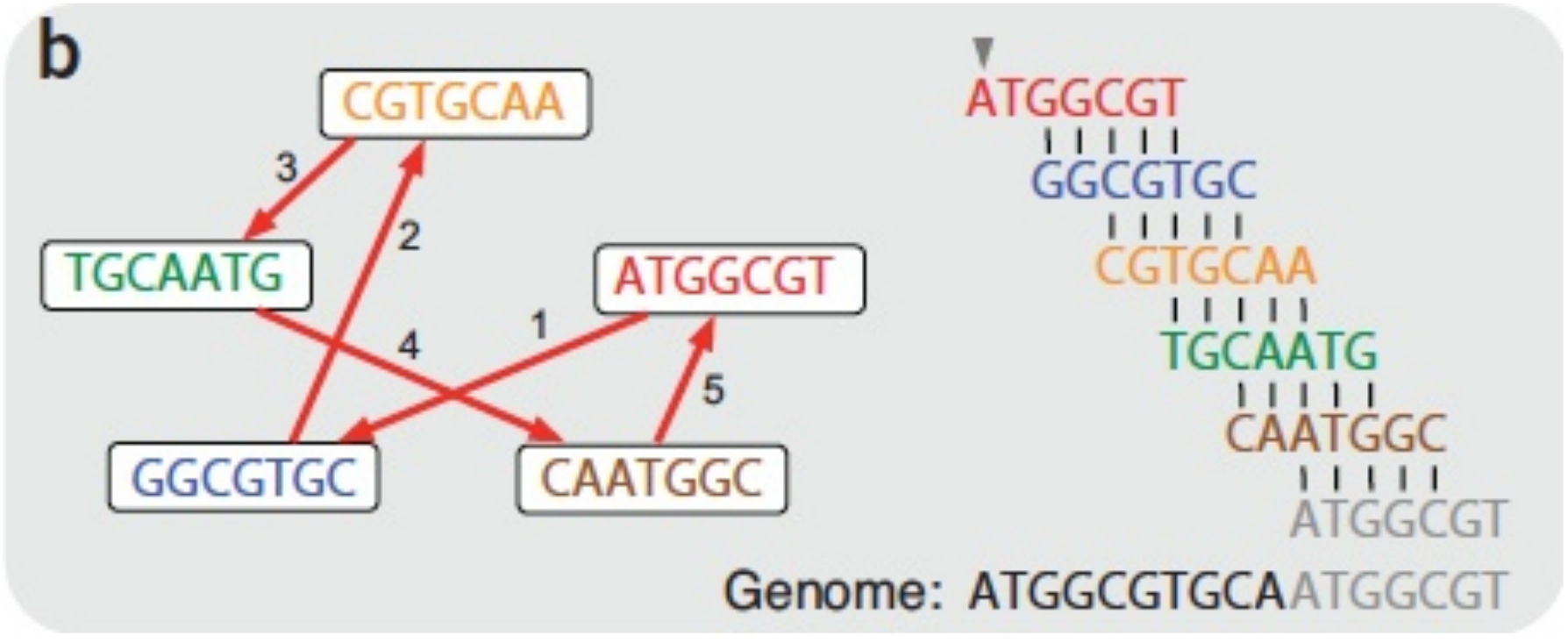
Overlap Graph Problem

Construct the overlap graph of a collection of k-mers

input: A collection Patterns of k-mers

output: The overlap graph OVERLAP (Patterns)

The **Overlap graph** (**Hamiltonian graph)** is a graph in which each read is represented by a node and overlap between reads is represented by an arrow (called a directed edge) joining two reads. For instance, two nodes representing reads may be connected with a directed edge if the reads overlap by at least five nucleotides.
The **Hamiltonian cycle**, is *a path that travels to every node exactly once and ends at the starting node, meaning that each read will be included once in the assembly*.

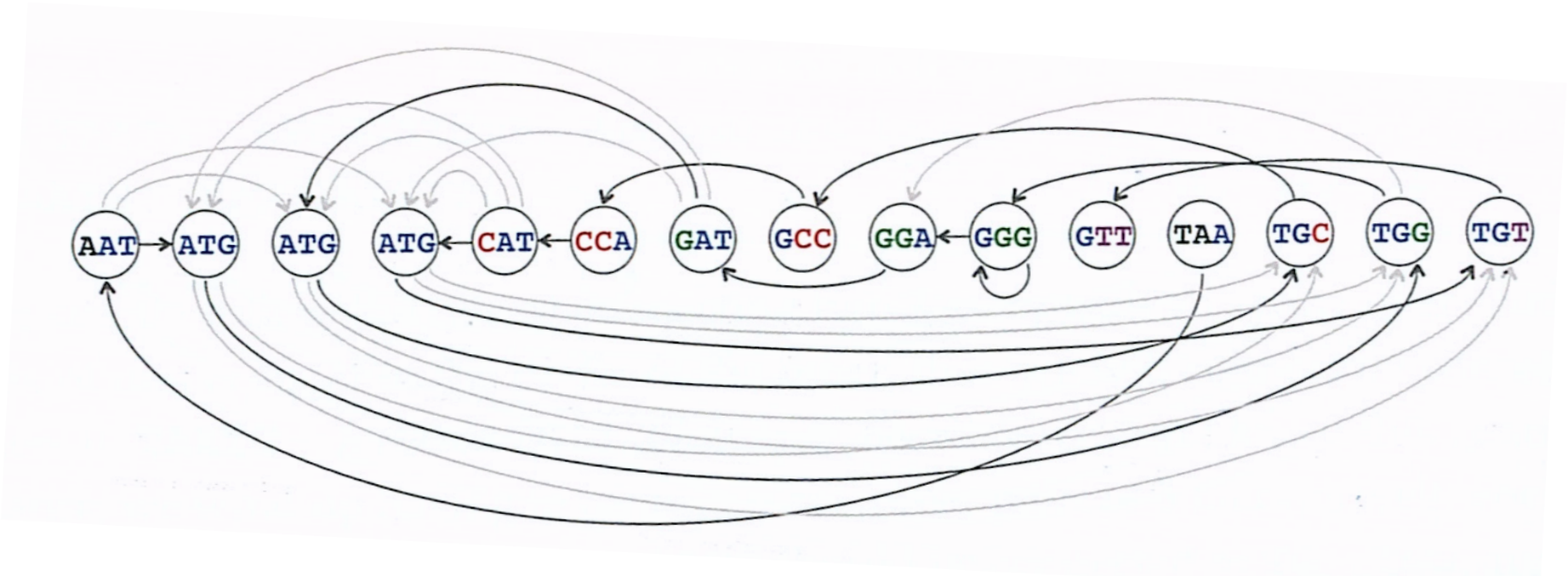


Hamiltonian Path Problem

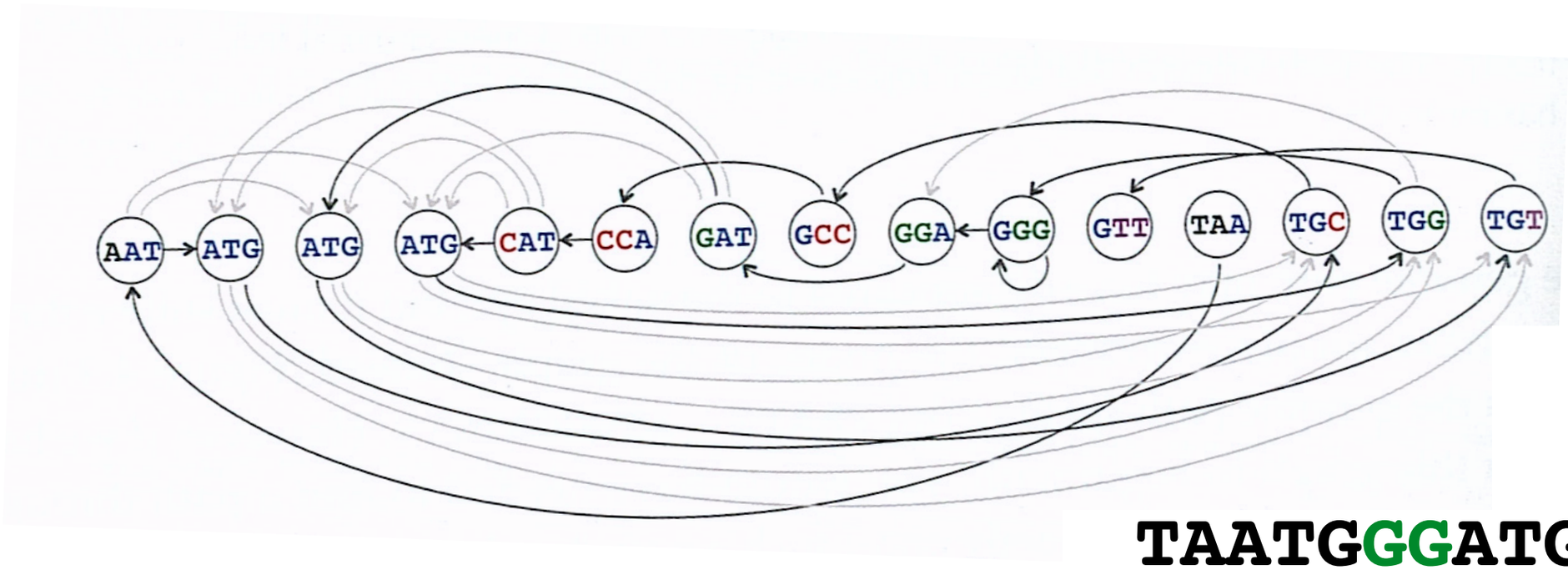
Construct a Hamiltonian path in a graph

input: A direct graph
output: A path visiting every node in the graph exactly once (if such a path exists).

In the **String Reconstruction Problem** the path are generated by visiting every node exactly once.



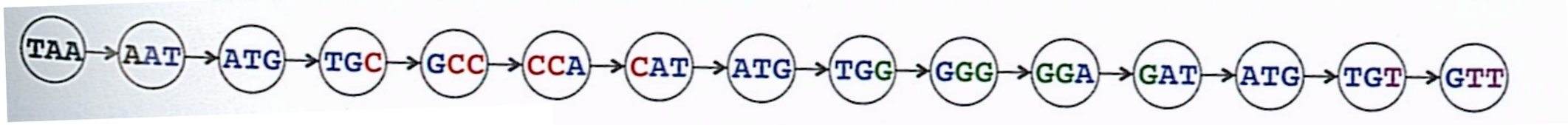
TAATGCCATGGGATGTT



TAATGGGATGCCATGTT

Representing a Genome as a Path

Composition 3 (TAATGCCATGGGATGTT) =



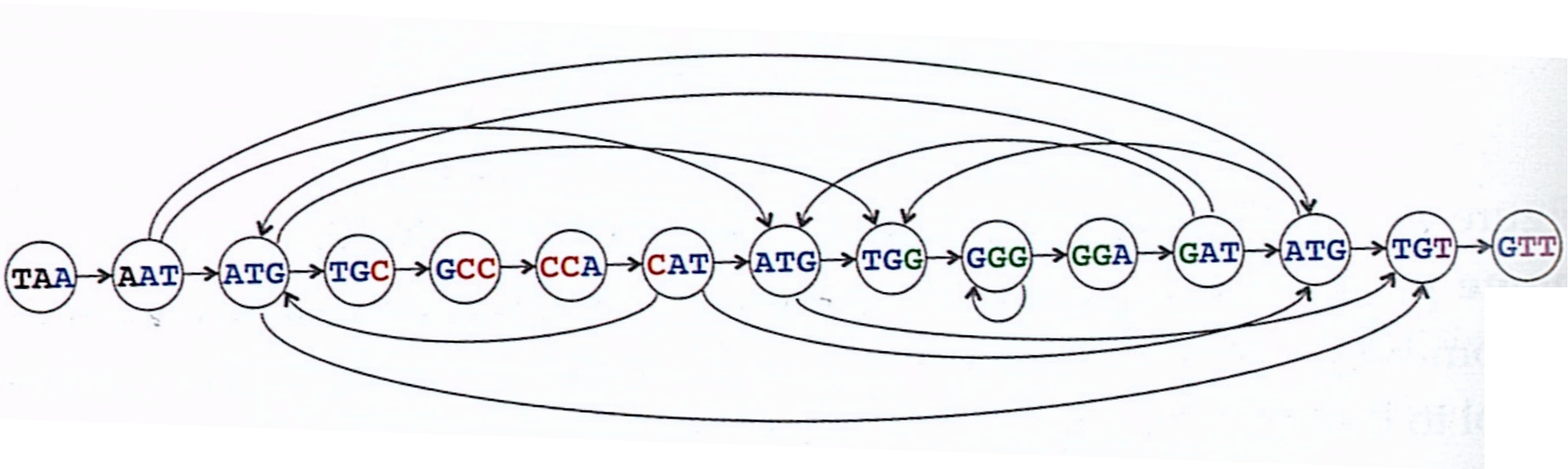
Can we construct this genome path without knowing the genome **TAATGCCATGGGATGTT**, only from its composition?

YES. We simply need to connect k-mer1 with k-mer2 if suffix (k-mer1)=prefix(k-mer2).

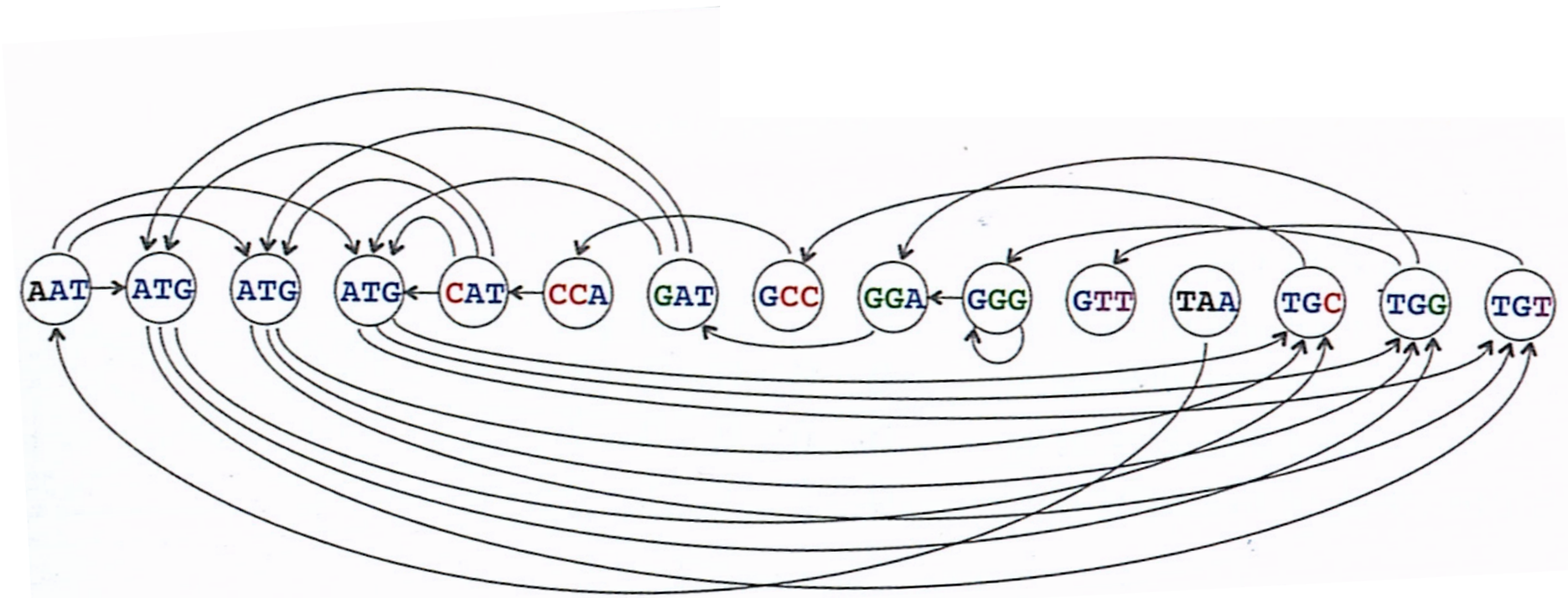
i.e. **TAA -> AAT**

suffix (k-mer1)=prefix(k-mer2).

i.e. **TA**A **->** **A**AT



Can we still find the genome path in this graph?

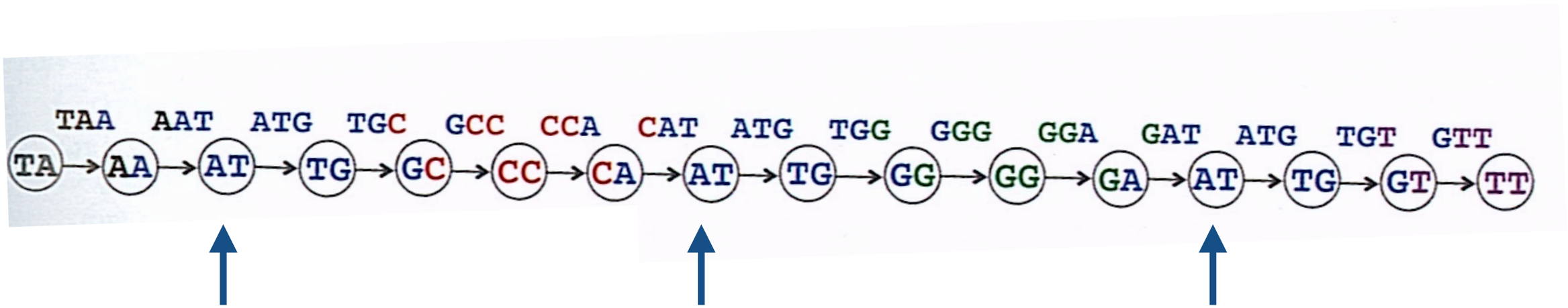


de Bruijn graphs

TAATGCCATGGGATGTT

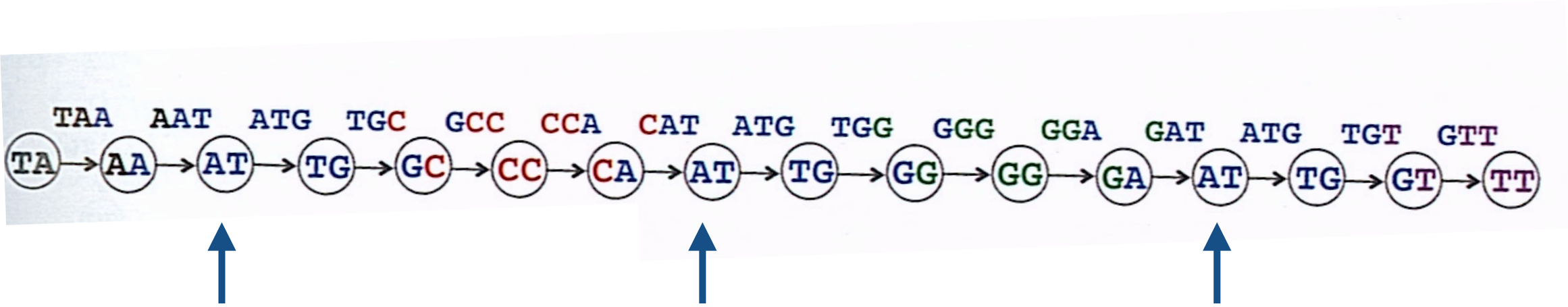
TAA AAT ATG TGC GCC CCA CAT ATG TGG GGG GGA GAT ATG TGT GTT

New graph representation: assign the 3-mers to the **edges**. Since each pair of consecutive edges represent two consecutive 3-mers, they overlap in two nucleotides.

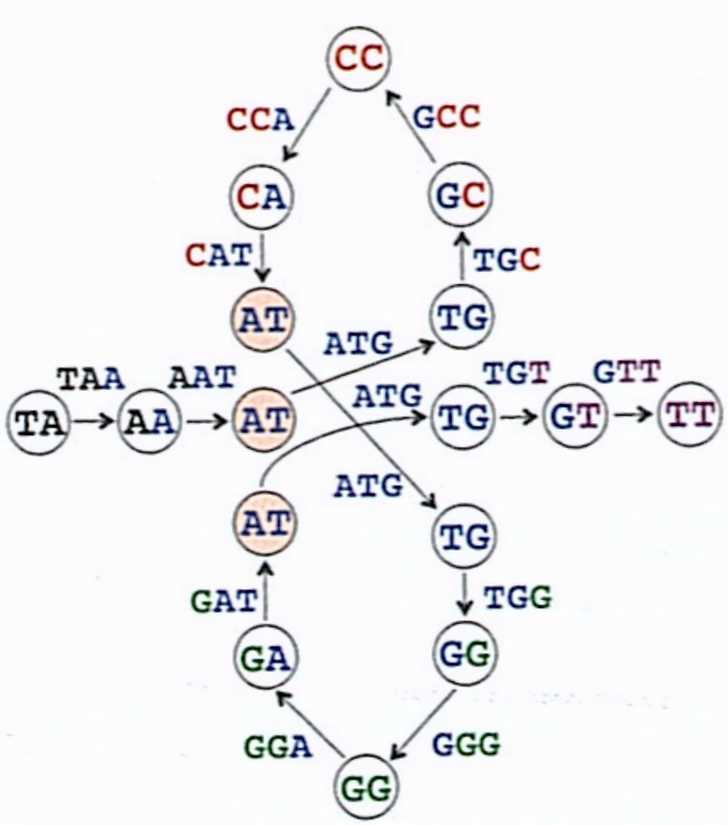


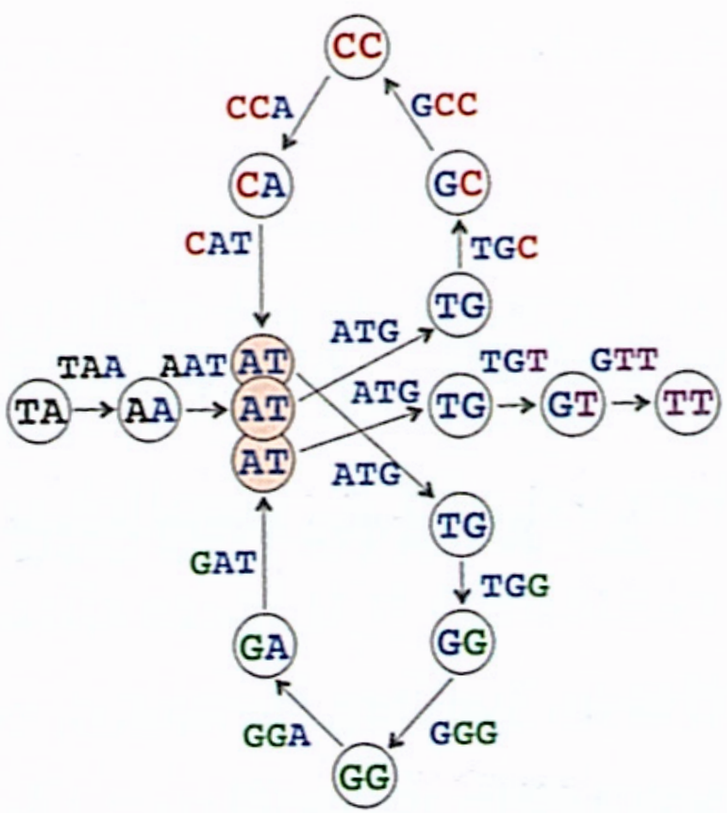
From this graph it is possible build a complex version by **gluing** the identical nodes.

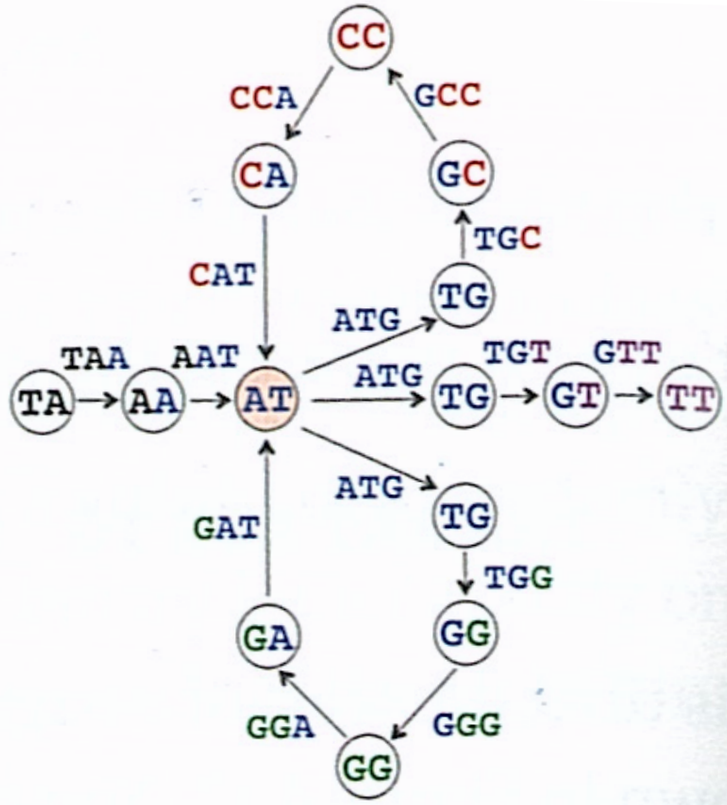
de Bruijn graphs



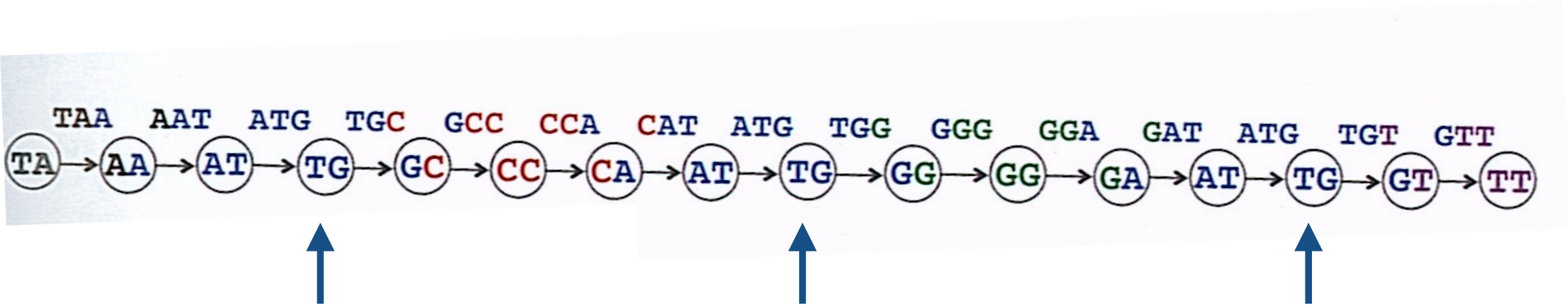
In the this graph there are 3 identical nodes **AT**, that can be collapsed in a single node.



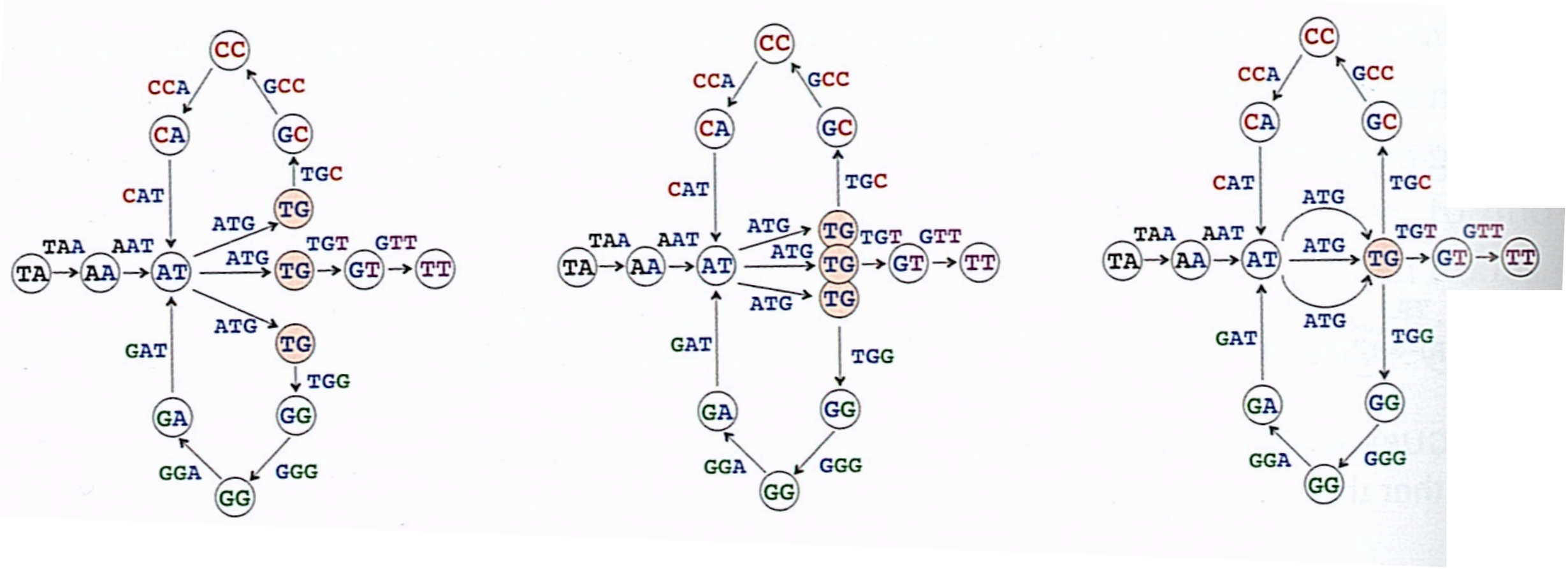




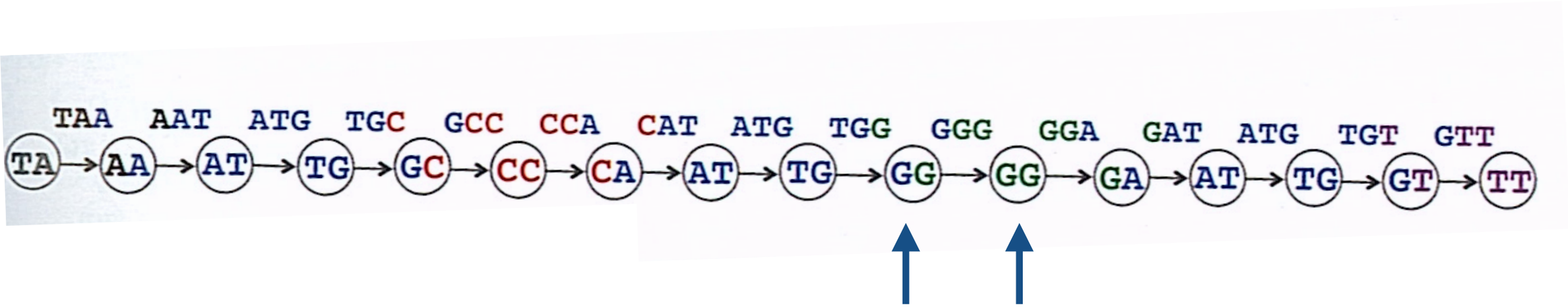
de Bruijn graphs



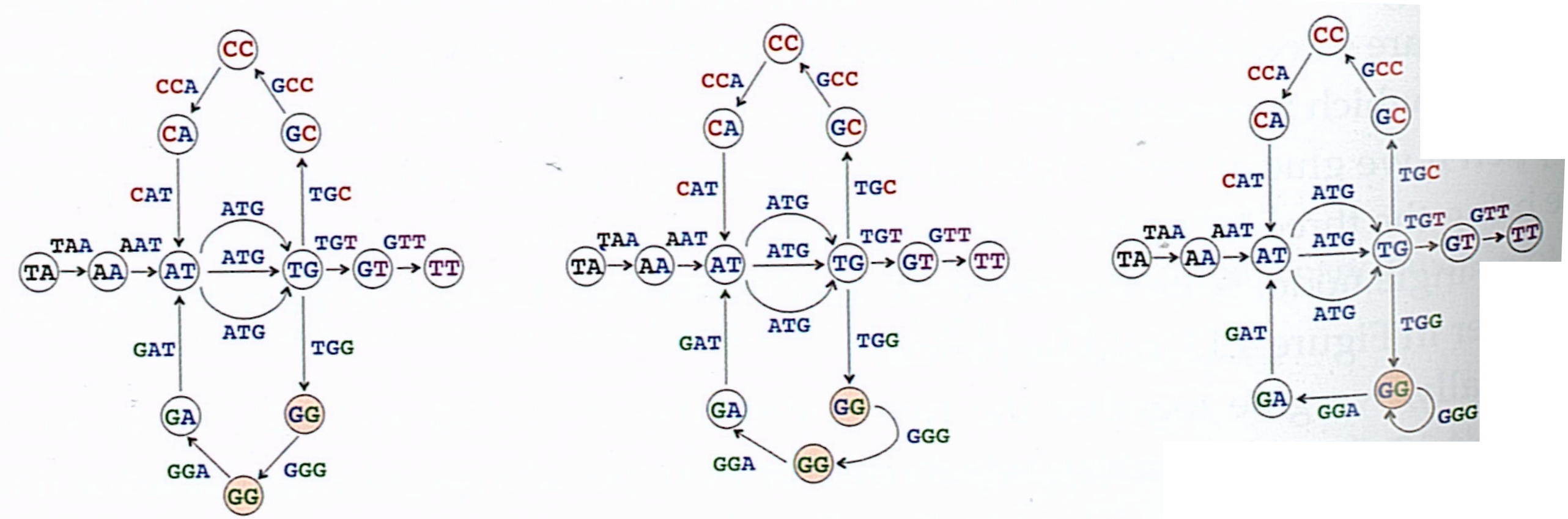
Same glue procedure for the 3 identical nodes **TG**



de Bruijn graphs

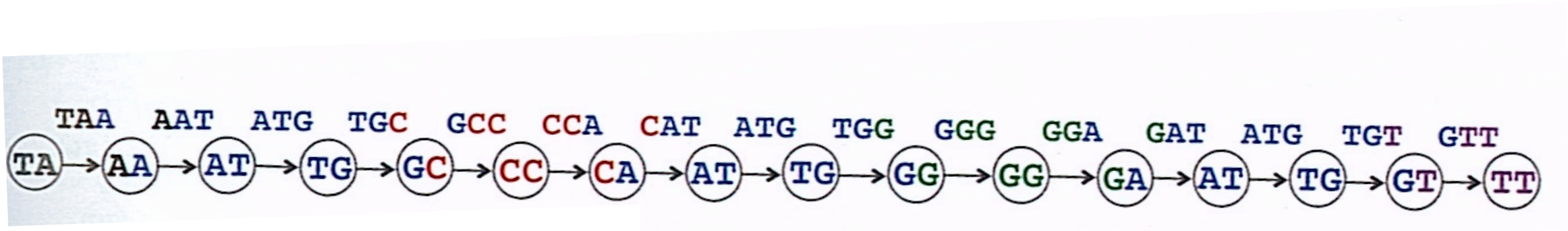


Same glue procedure for the 2 identical nodes **GG**



The final graph is called **de Bruin graph: `DeBRUIJN3(TAATGCCATGGATGTT)`**
This procedure has reduced from 16 to 11 node, while the number of edges stayed the same.

de Bruijn graphs



Given a genome Text, $PATHGRAPH_k(Text)$ is the path consisting of
|Text| - k +1 edges, where the i-th edge of this path is labeled by the
i-th k-mer in Text and the i-th node of the path is labeled by the i-th (k-1)-mer in Text.

De Bruijn graph from String Problem

Construction the de Bruijn graph of a string

input: A string Text and an integer k

output: $DEBruijn_k(Text)$

de Bruijn graphs - Construction from a k-mer composition without gluing

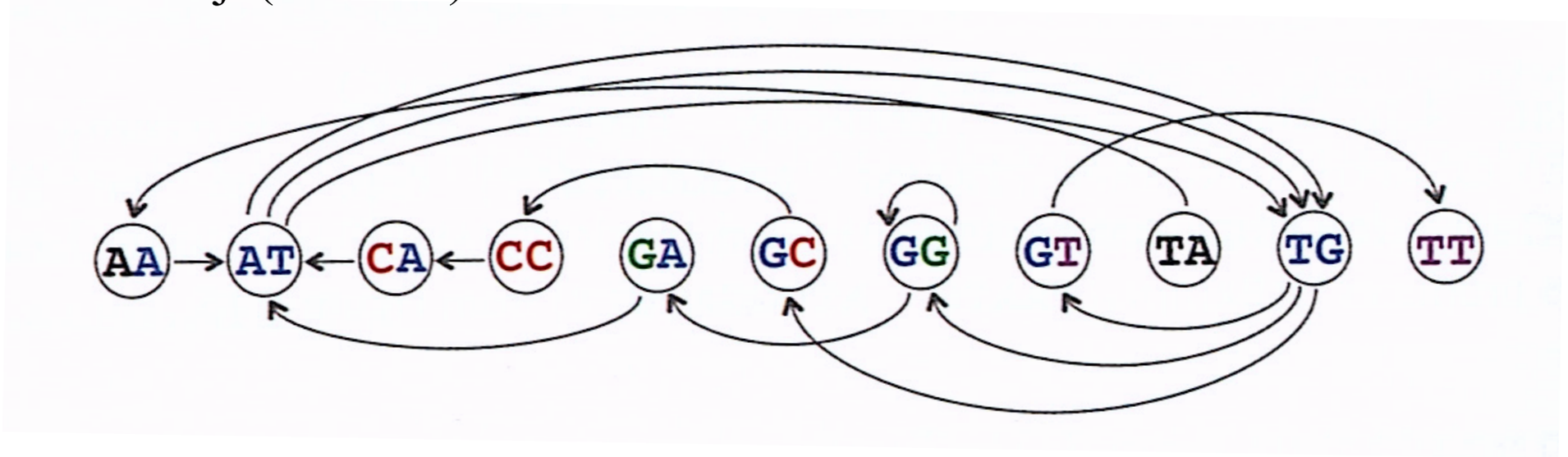
Give a collection of k-mers *Pattern*, the node of DeBruijnk(*Patterns*) are simply all unique (k-1)-mers occurring as a prefix or suffix in *Patterns*.

Patterns AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG GTT TAA TGC TGG TGT

The set of 11 unique 2-mers occurring as a prefix or suffix:

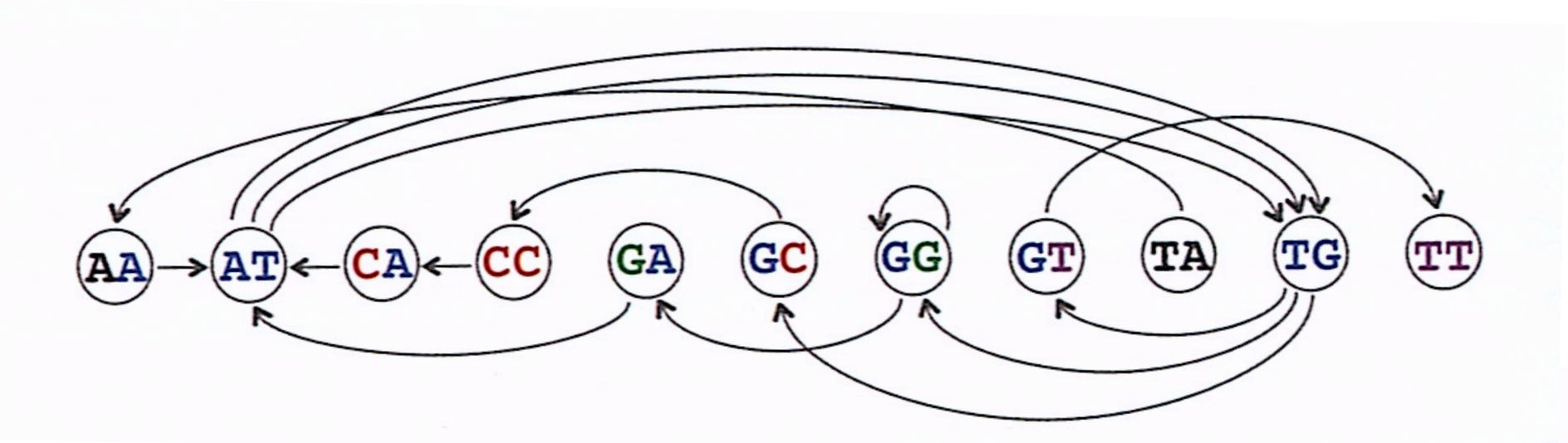
AA AT CA CC GA GC GG GT TA TG TT

For every k-mer in *Patterns*, we connect its prefix node to its suffix node by a directed edge in order to produce DeBruijn(*Patterns*).

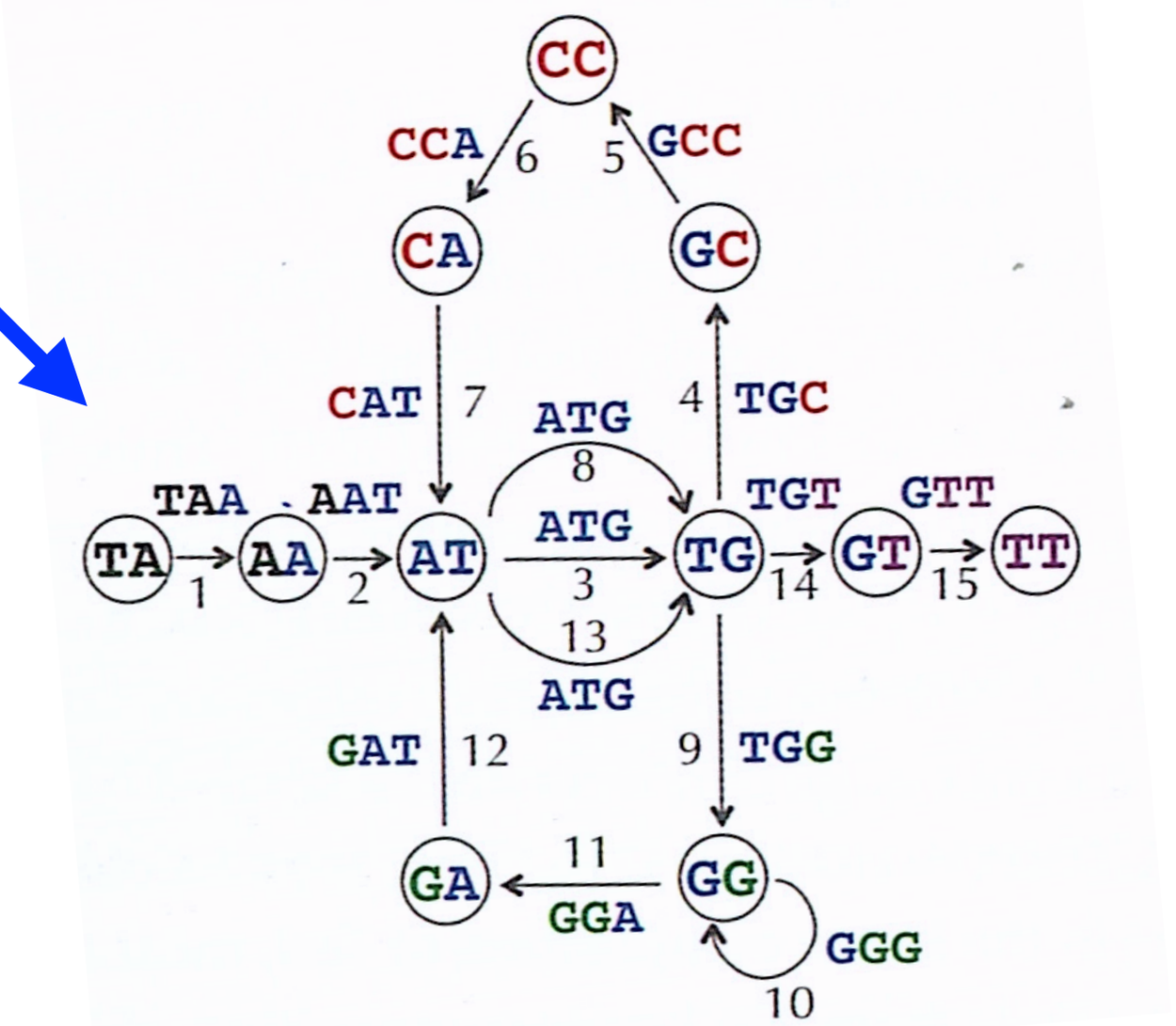


de Bruijn graphs - Construction from a k-mer composition

The previous graph is the same that we obtained with the previous approach.



same graph although it has been drawn differently



de Bruijn graphs - Eulerian Path Problem

We must solve the **String Reconstruction Problem** to find a path that visit every edge exactly once. (**Eularian Path**)

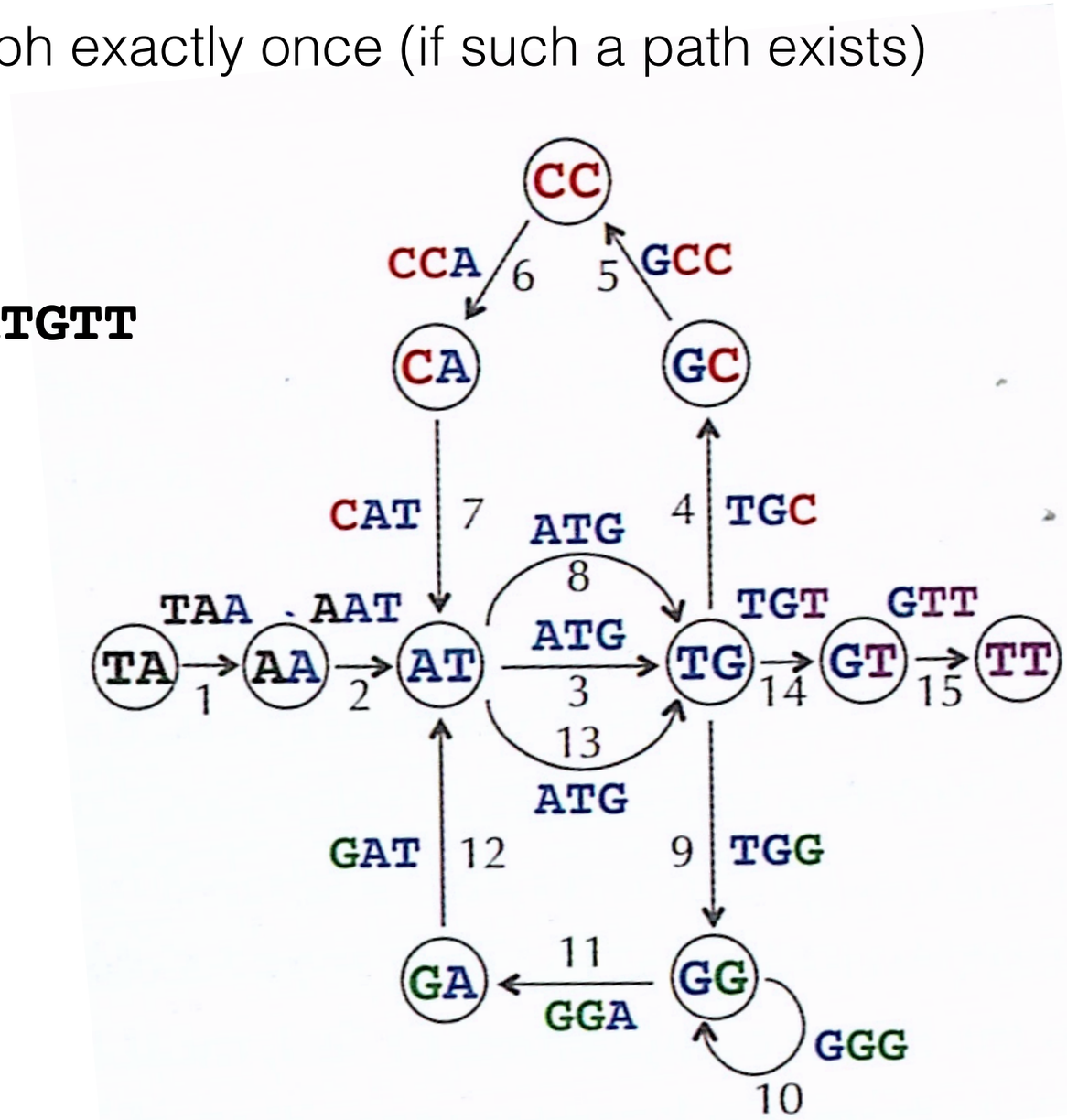
Eulerian Path Problem

Construction an Eulerian path in a graph

input: A direct graph

output: a path visiting every edge in graph exactly once (if such a path exists)

TAATGCCATGGATGTT

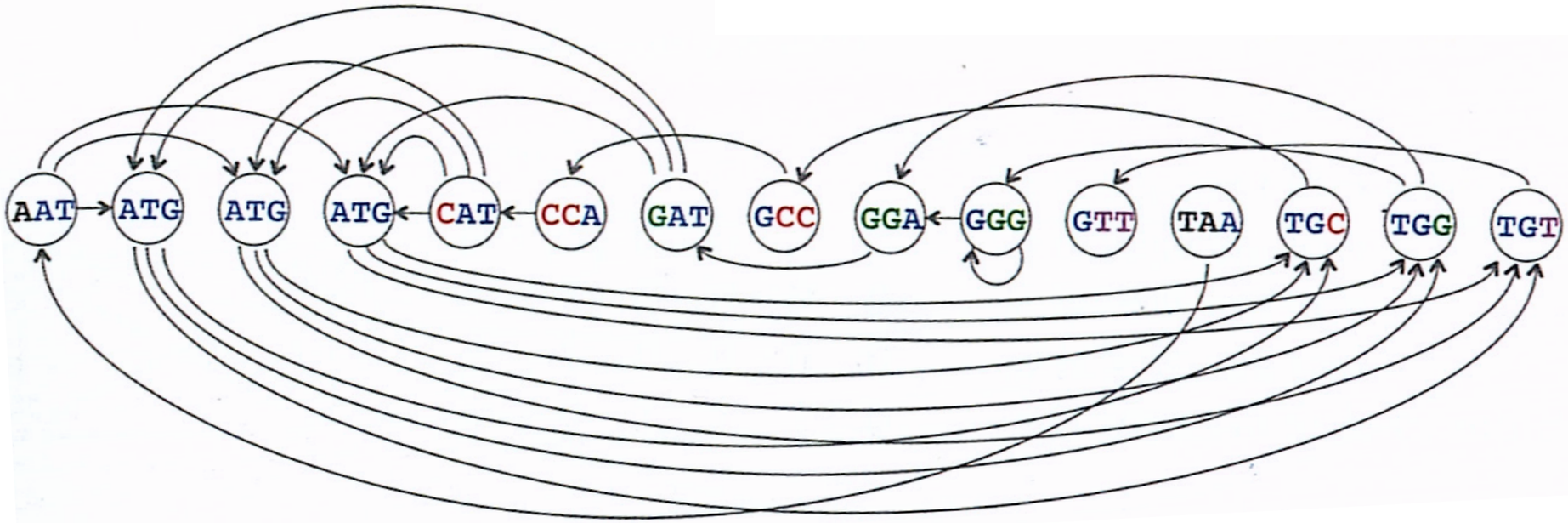


DEBRUIJN(Patterns)

Represent every k-mer in Patterns as an isolated edge between its prefix and suffix glue all nodes with identical labels, yielding the graph DEBRUIJN(Patterns)

return DEBRUIJN (Patterns)

Overlap graph versus De Bruijn graph

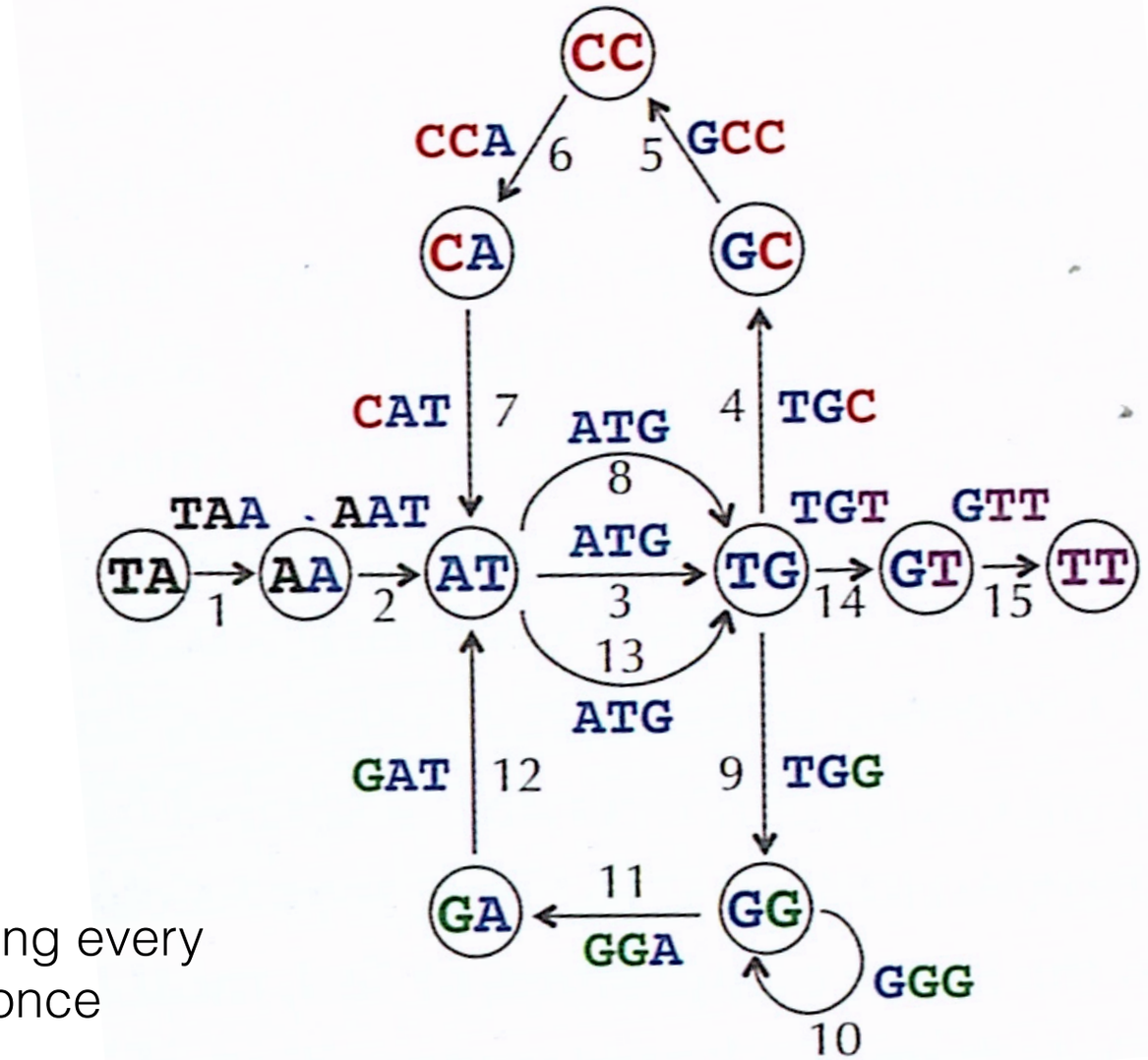


Overlap graph -> Hamiltonian path

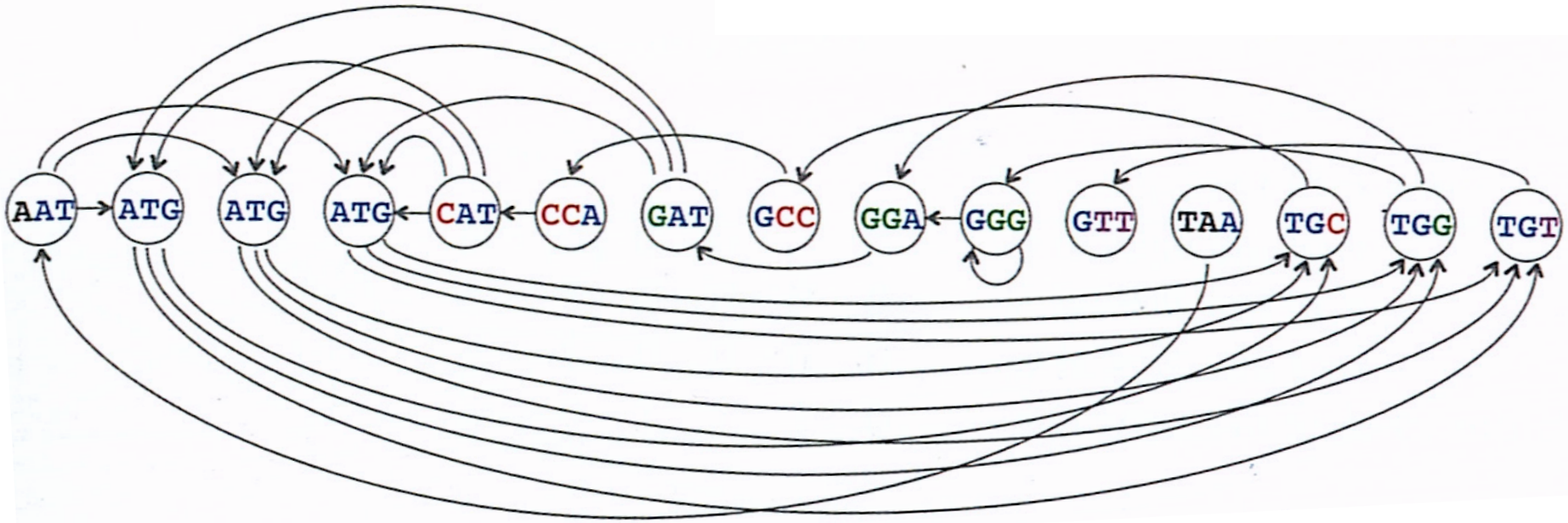
finding a path visiting every
node exactly once

De Bruijn graph -> Eulerian path

finding a path visiting every
edge exactly once



Overlap graph versus De Bruijn graph



Overlap graph -> Hamiltonian path

What problem would you prefer to Solve?

De Bruijn graph -> Eulerian path