The choice must be computed considering the most **Efficient algorithm**

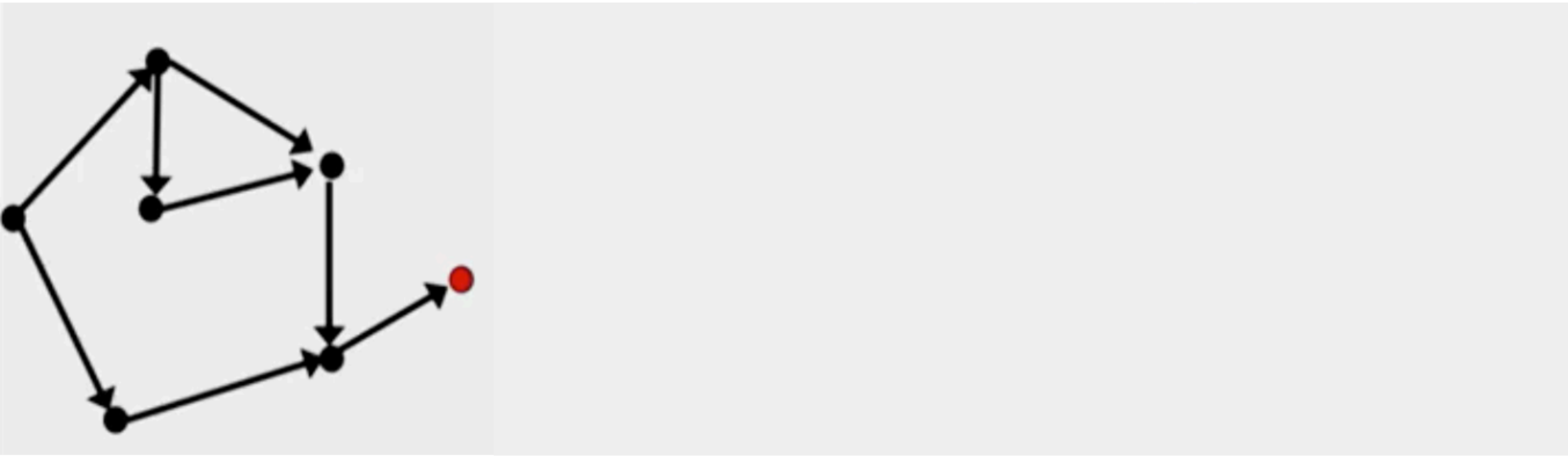
While Euler solver the **Eulerian Path Problem**, nobody has developed a fast algorithm for the **Hamiltonian Path Problem** yet.

The **Hamiltonian Path Problem** belongs to a collection containing thousands of computational problems for which no fast algorithms are known, namely **NP(NonDeterministic Polynomial Time)-Complete Problems**

Several computational problems fall in the **NP-Complete Category,** and the **Hamitonian Path problem** is one of the seven **Millennium Problem** in mathematics.

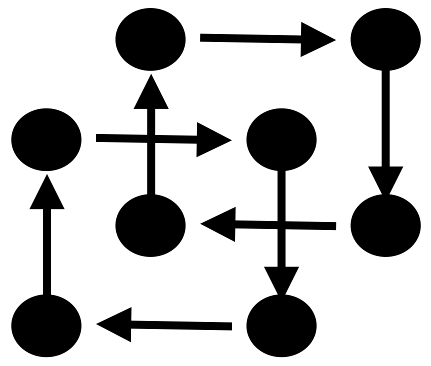
A graph is **Eulerian** if it contains an *Eulerian Cycle*

Has this graph an Eulerian cycle or is this graph Eulerian?

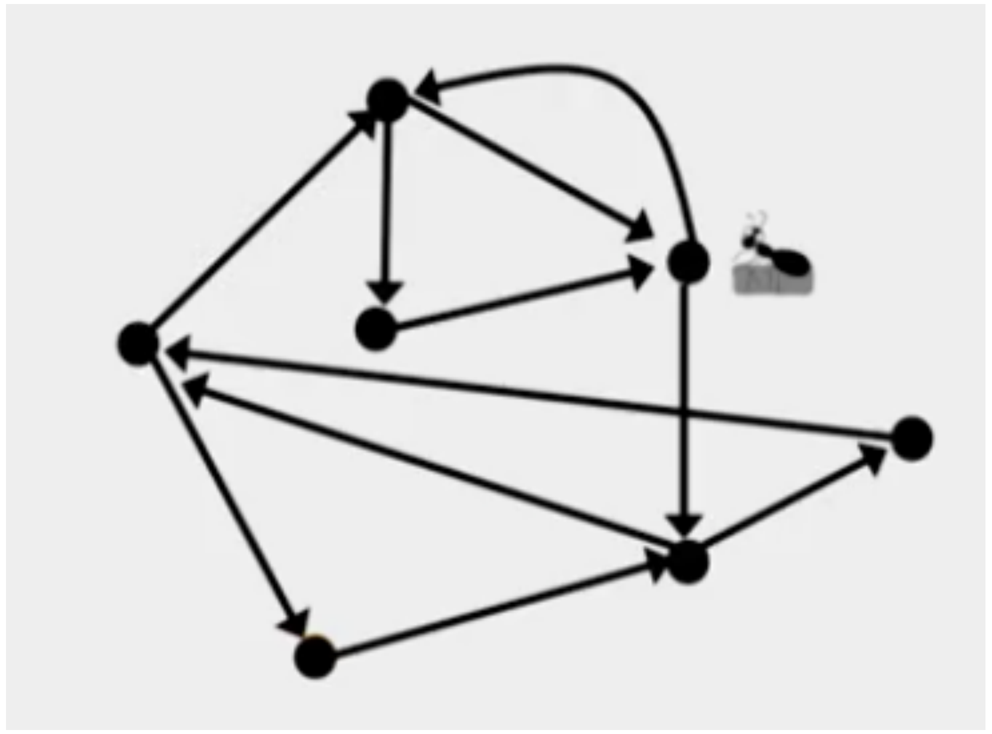


A graph is balance if **incoming** arcs are equals to **outcoming** arcs for each node

Balance but
disconnect graph

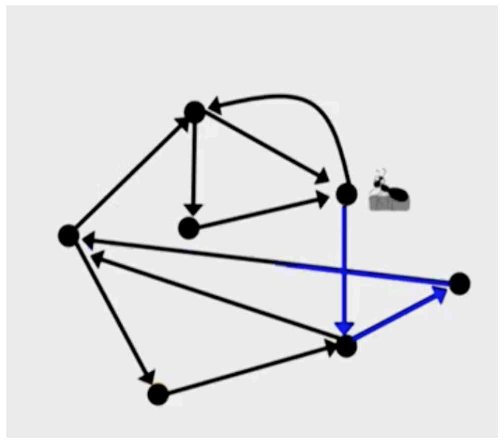
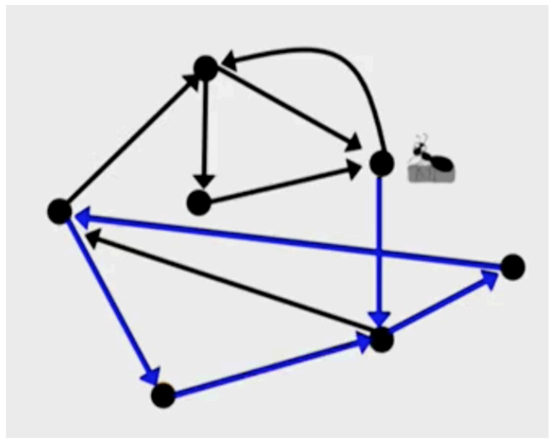
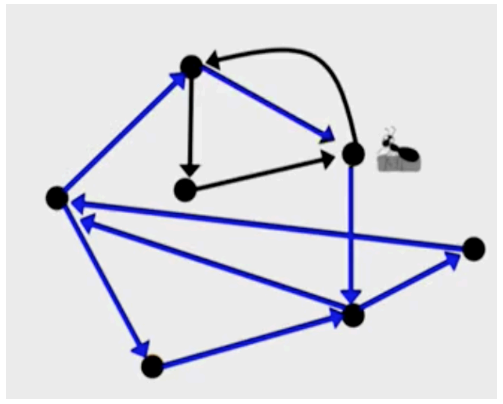
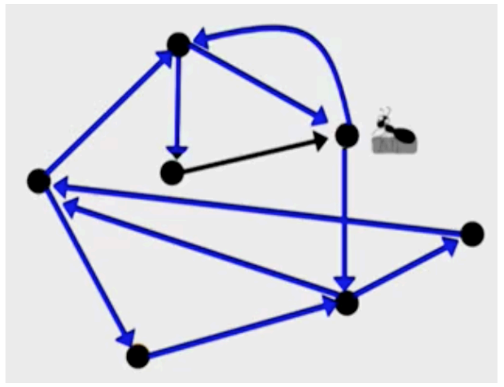


Eulerian graph must be *balance* and *strongly connected*, i.e. it is possible to reach any node from every other node

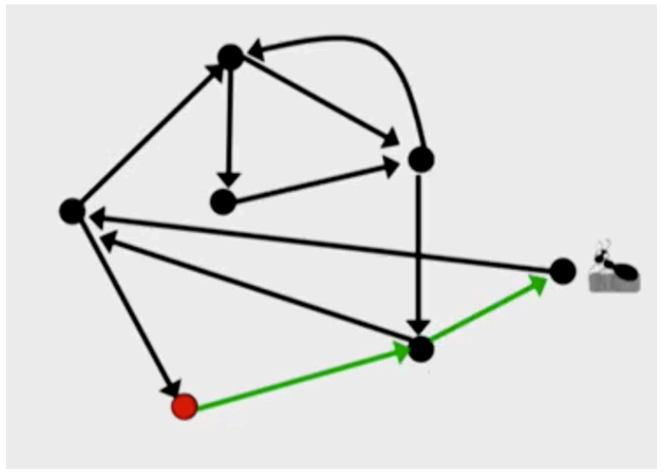
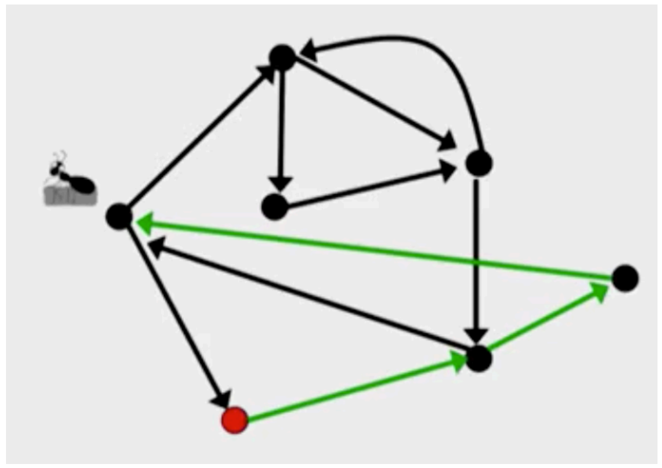
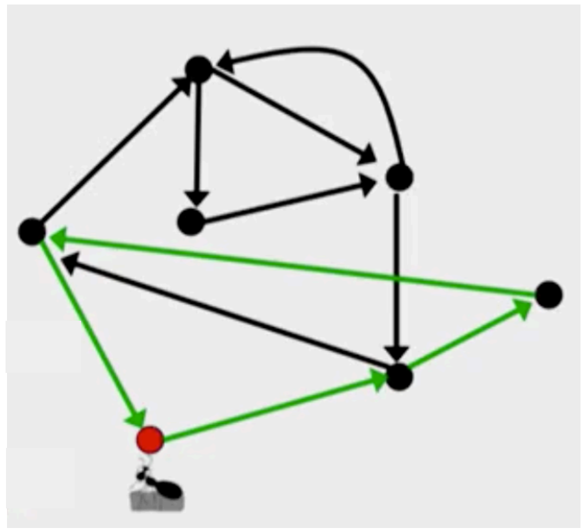


Euler's Theorem: Every balanced, strongly connected directed graph is Eulerian

To prove that the Graph has an Eulerian cycle, place the ant at any node *v_0* of the graph and let him randomly walk through the graph under the condition that he cannot traverse the same edge twice

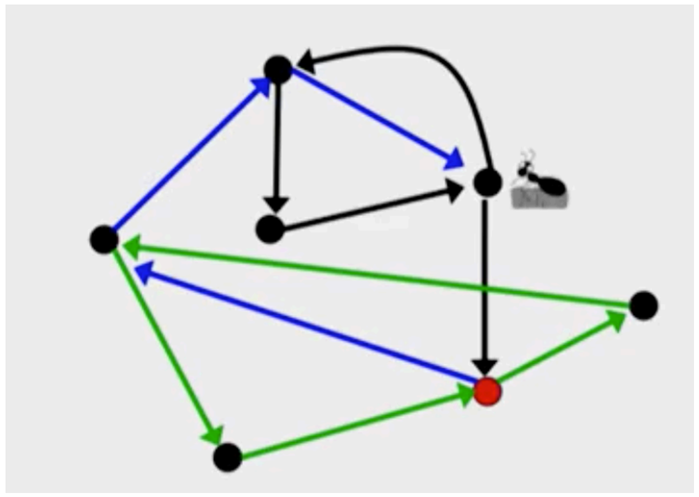
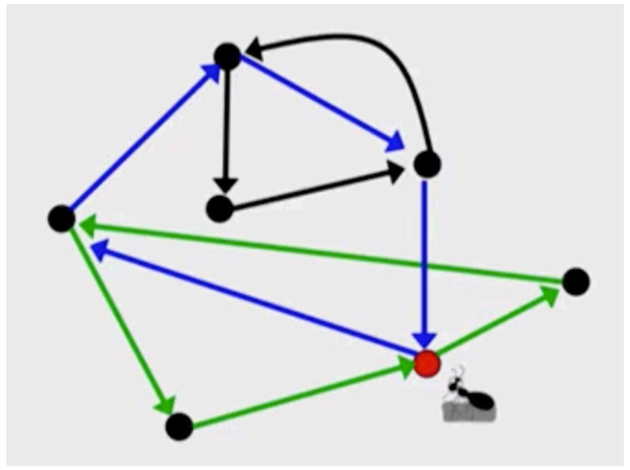





The ant is lucky!!

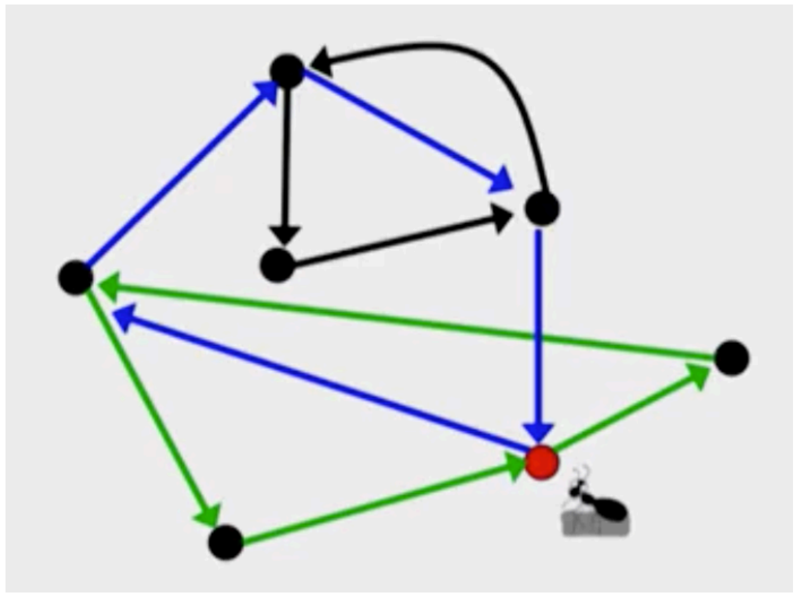




The ant get stuck when he is in the starting node v_0. Why?
The constructed cycle is not Eulerian, can we enlarge it?

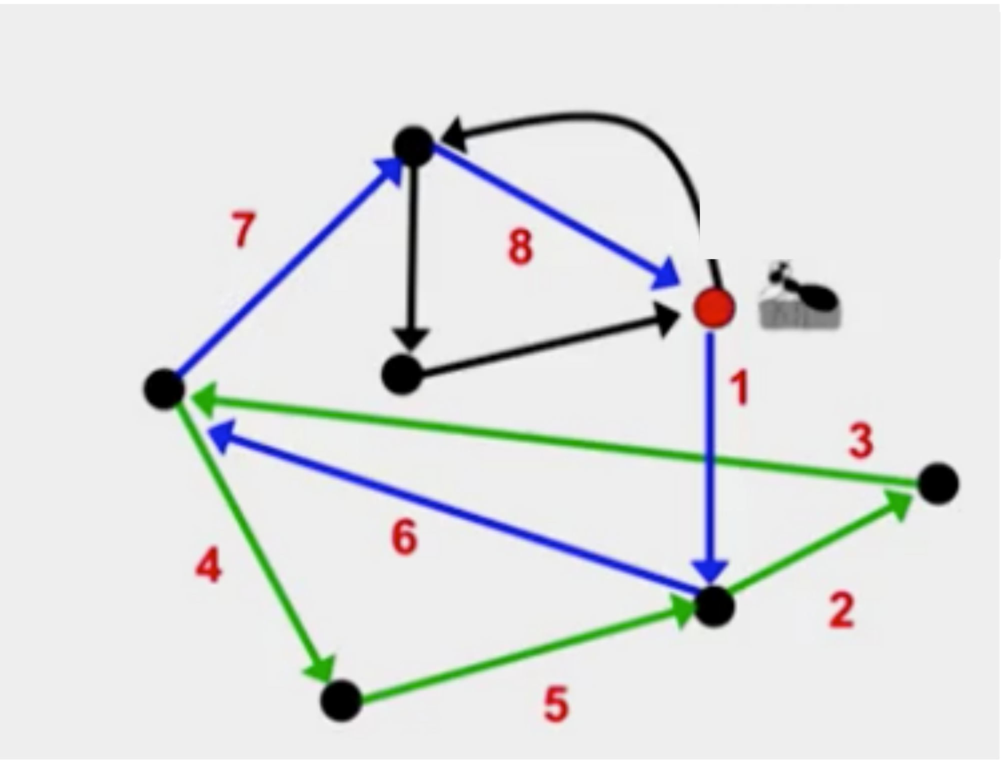
Let's start at a different node in the Green Cycle, in a node with still unexplored edges. Starting at a node that has used edge, traverse the already constructed (green cycle) and return back to the starting node

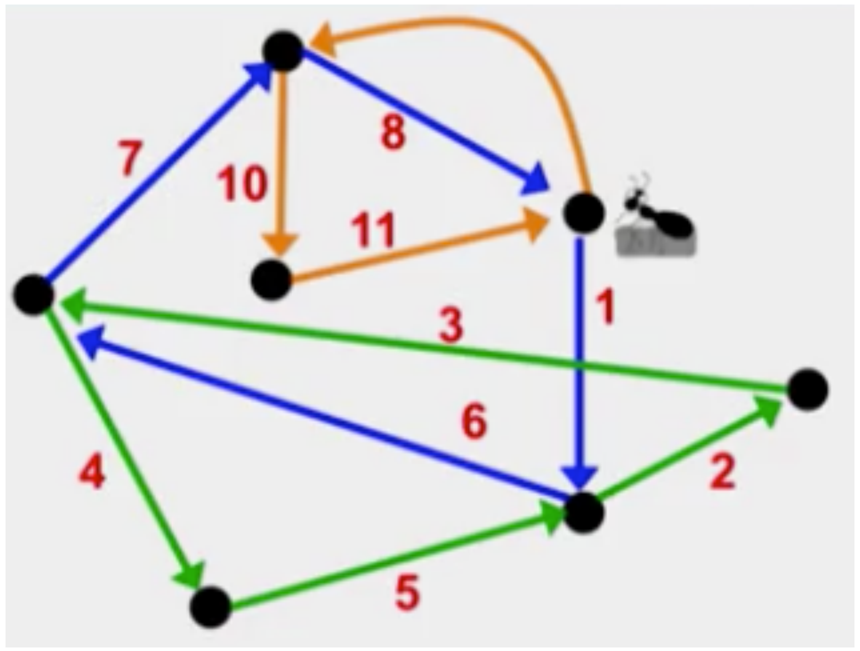
He is get stuck in again the starting node

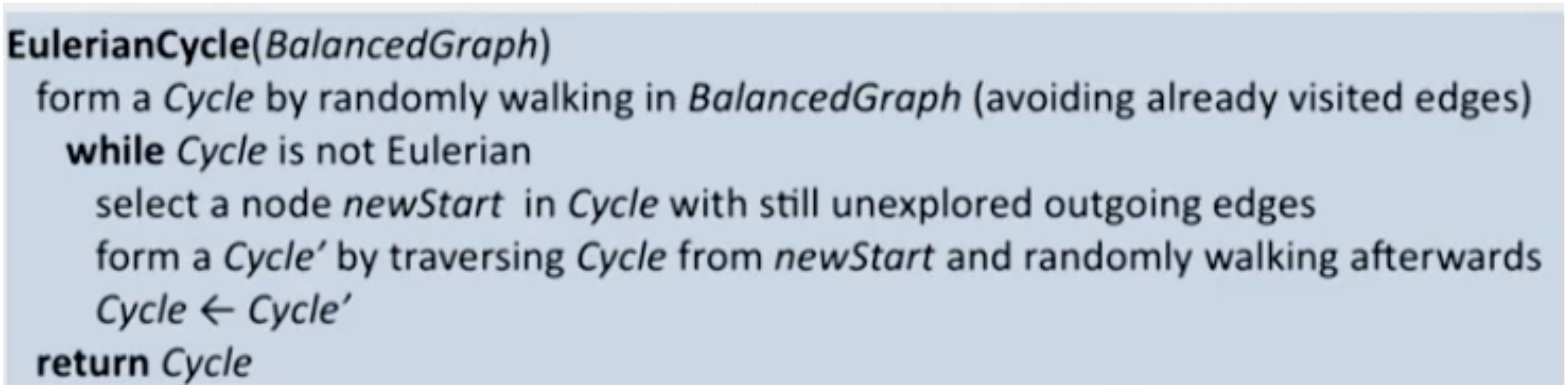


Staring a new node and traversing the previously constructed green and blue cycles



When the ant arrive at the starting node there is the possibility to explore other edges and to enlarge the Eulerian cycle

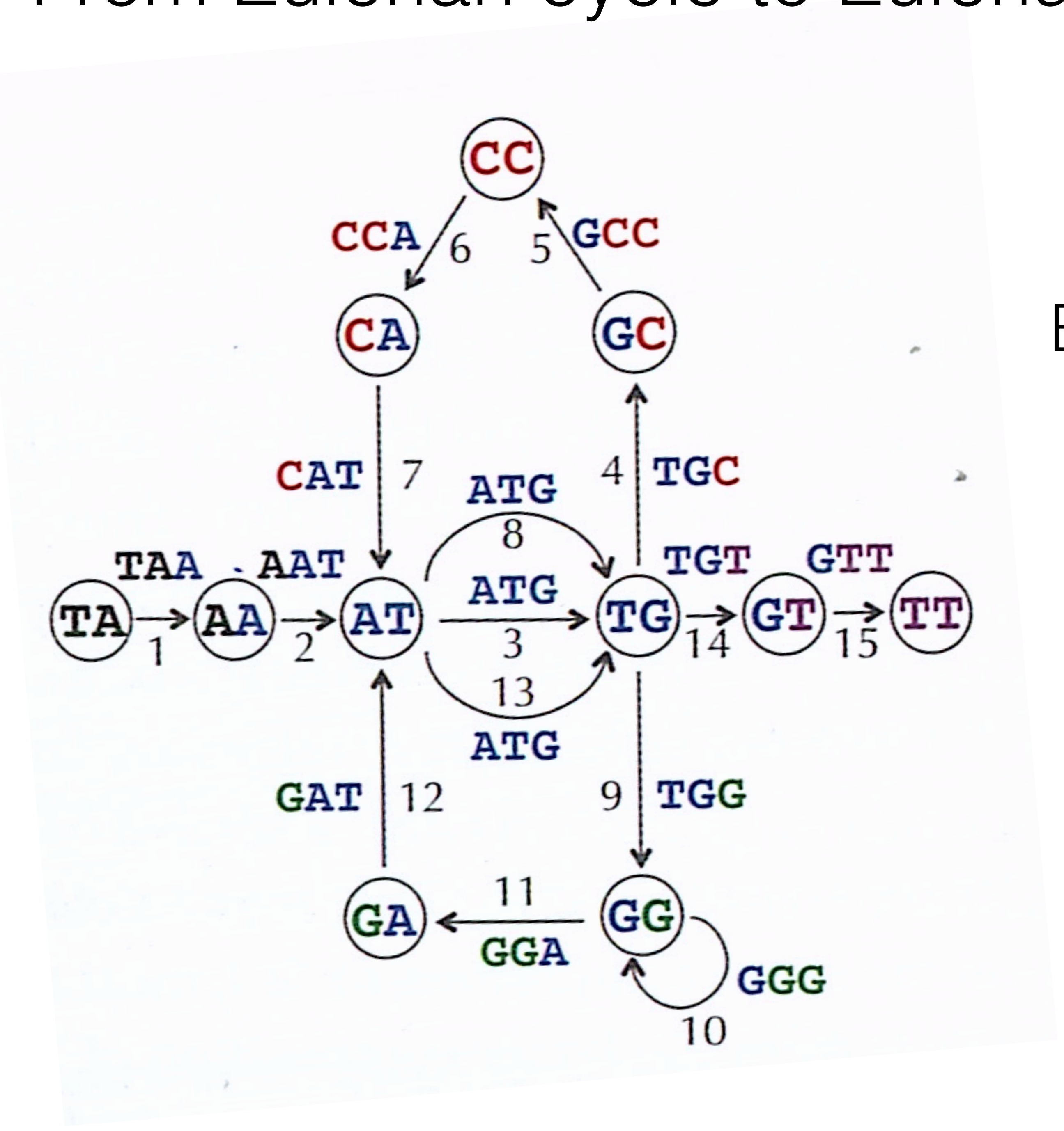




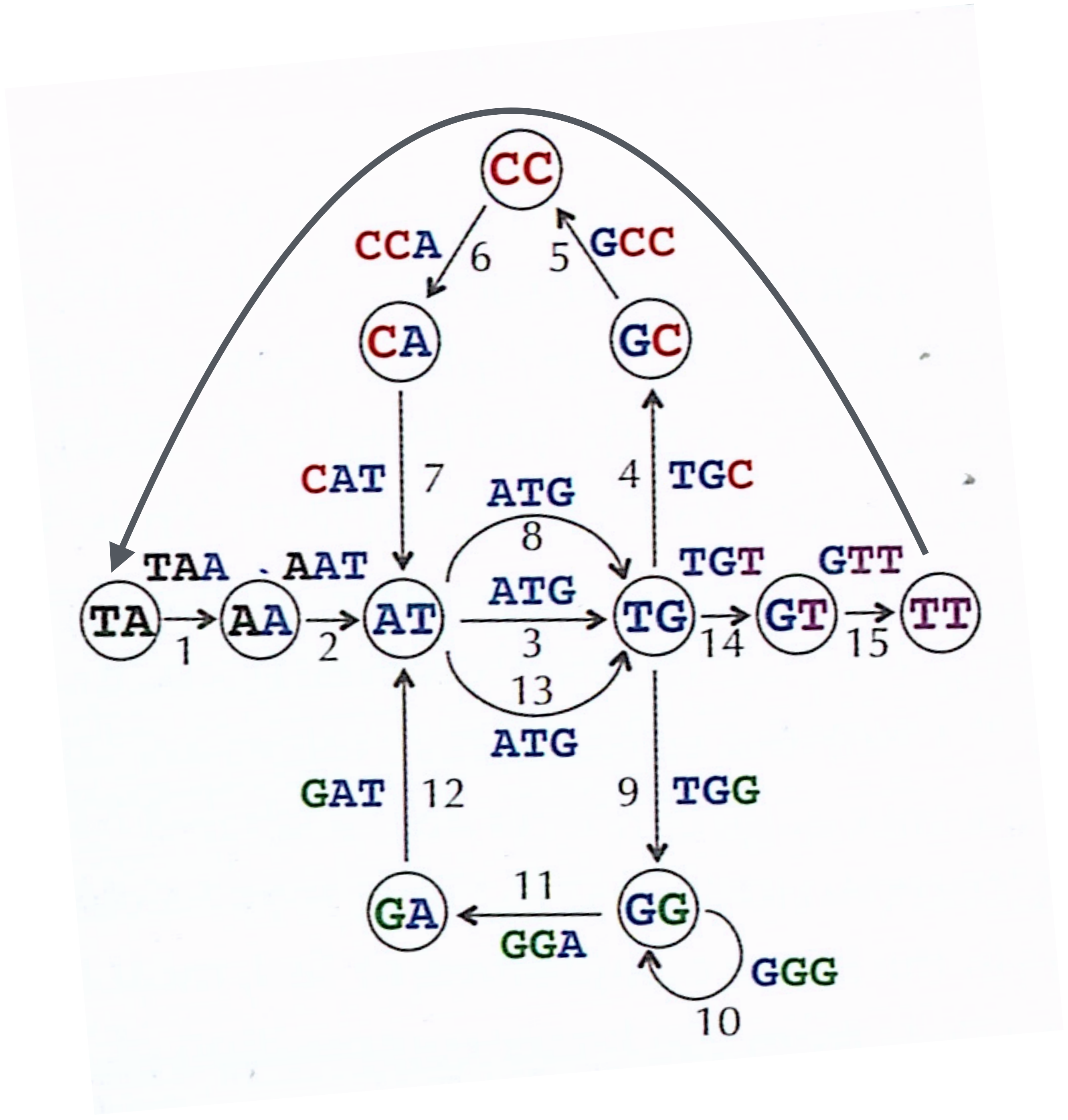
```
EulerianCycle(BalancedGraph)
 form a Cycle by randomly walking in BalancedGraph (avoiding already visited edges)
   while Cycle is not Eulerian
     select a node newStart in Cycle with still unexplored outgoing edges
     form a Cycle' by traversing Cycle from newStart and randomly walking afterwards
     Cycle ← Cycle'
 return Cycle
```

The proof of the Eulerian theorem offers and example of a **constructive proof,** which proves the desired results and provide also a method for constructing the object we need.

From Eulerian cycle to Eulerian paths



Has an eulerian path and we can transform this Eulerian path into Eulerian cycle by adding a connection

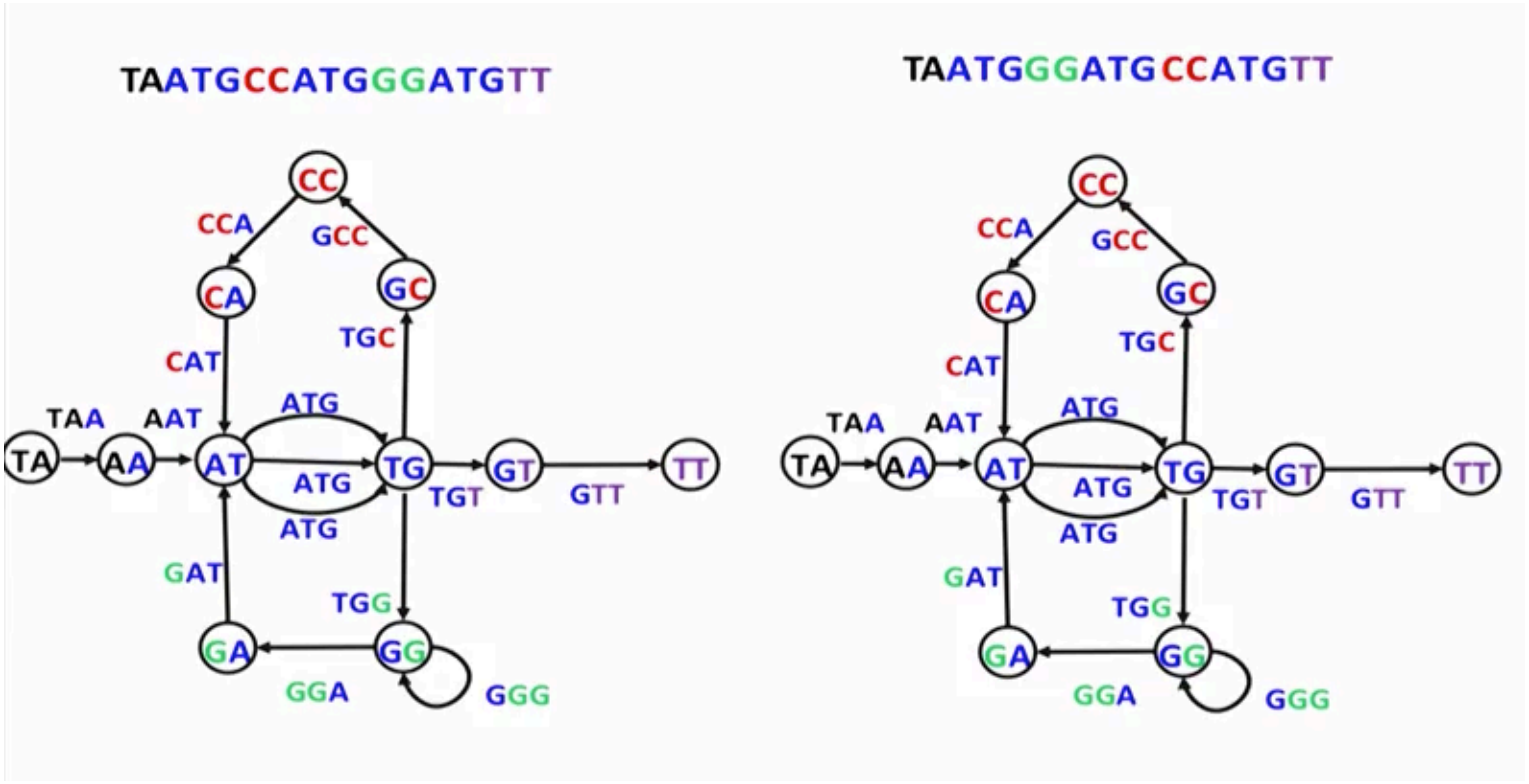


Multiple Eulerian Paths

A given string cannot be uniquely reconstructed for its 3-mres composition since another string has the same 3-mers composition.

TAATGCCATGGGATGTT

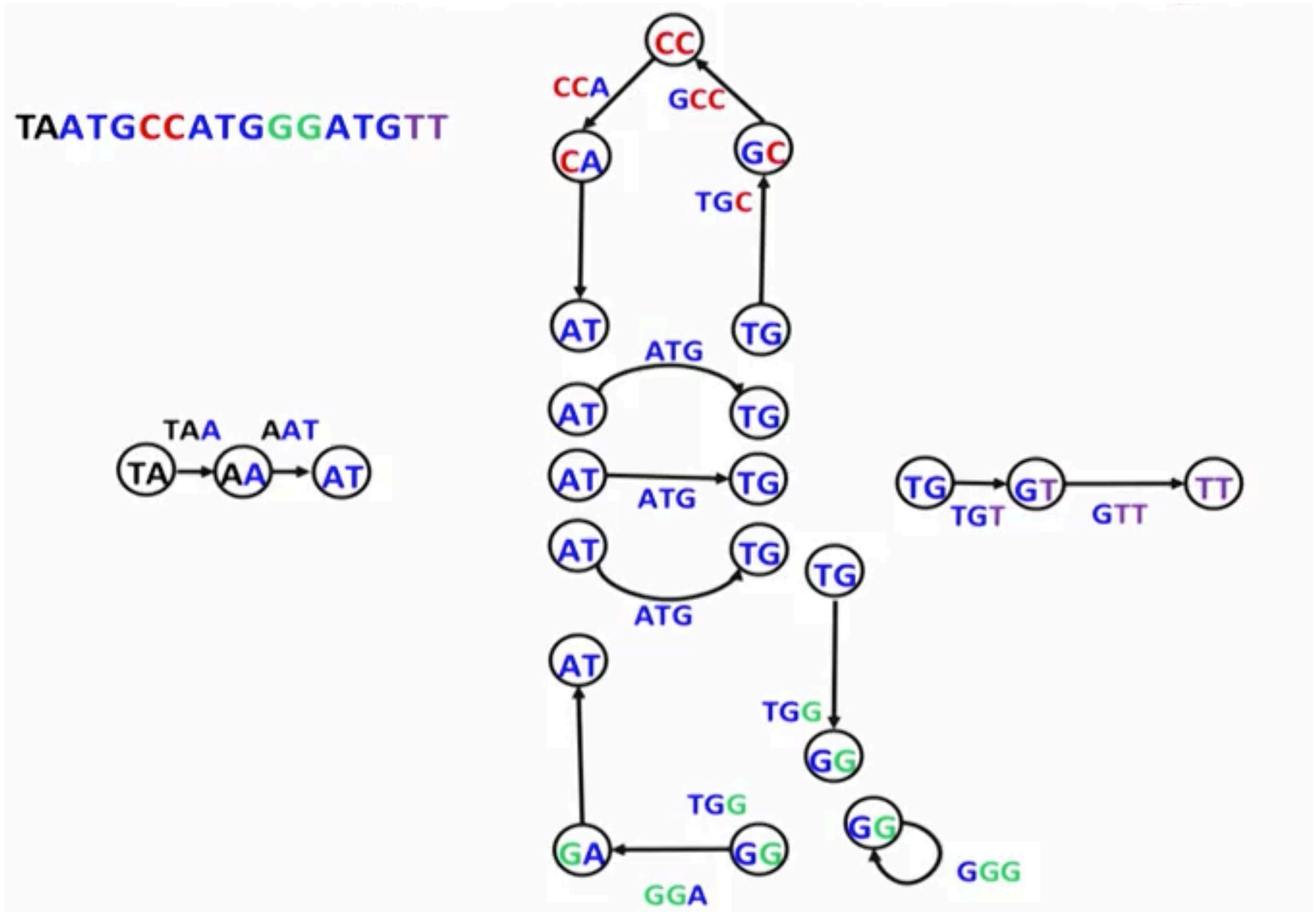
TAATGGGATCCGATGTT



Which is the correct?

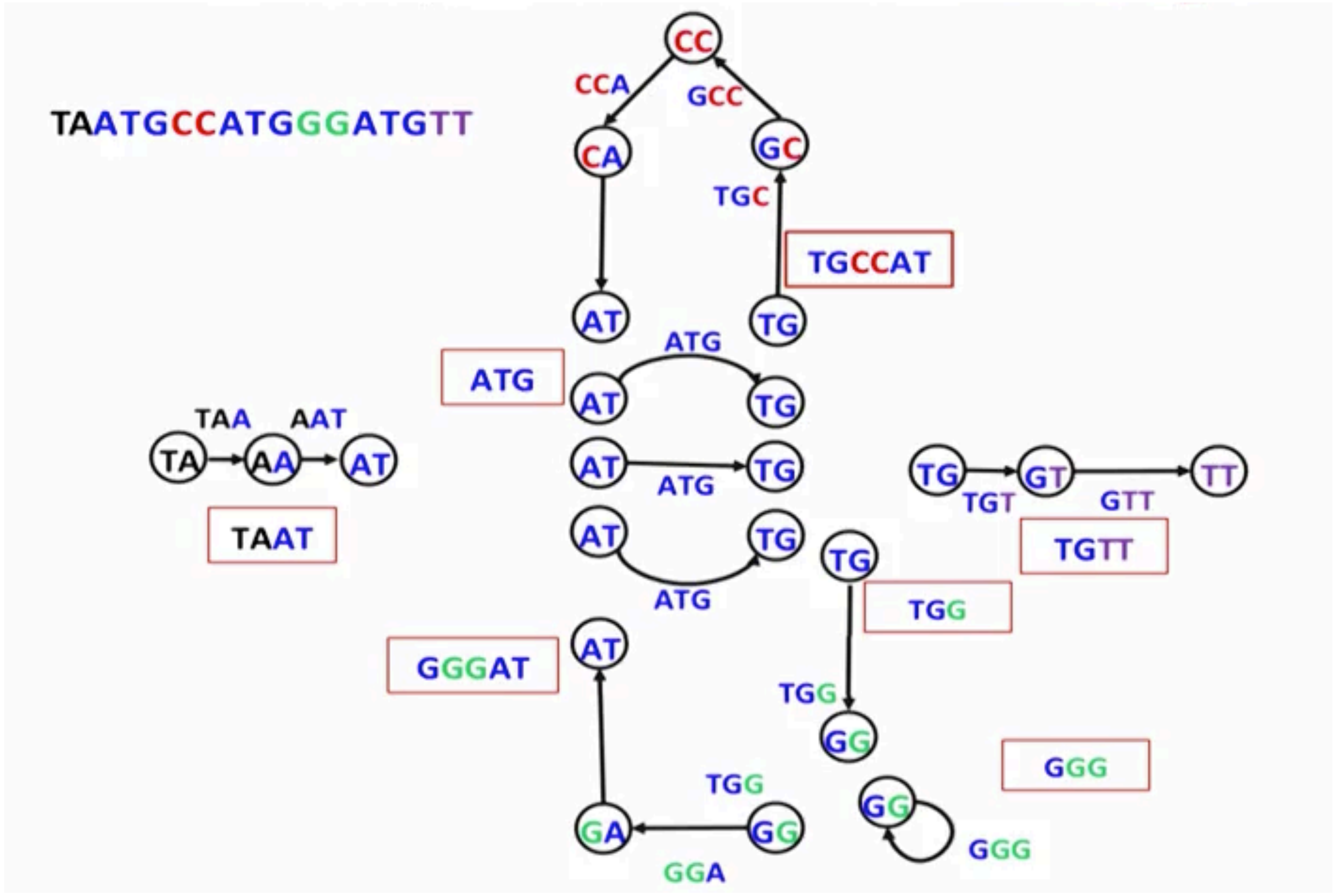
Breaking genome into Contigs

Force the graph to decompose it in multiple components



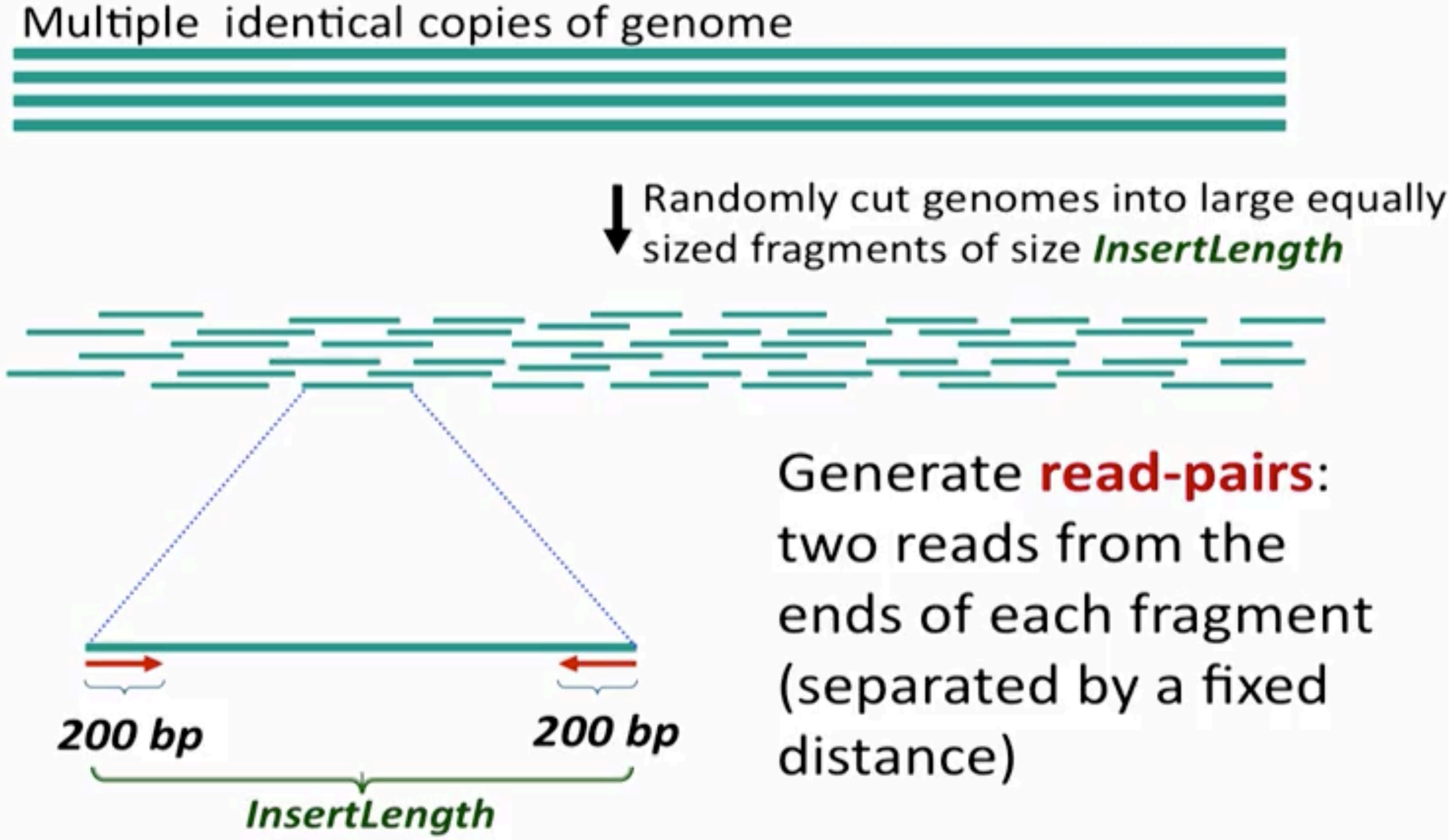
Breaking genome into Contigs

Multiple Eulerian paths



How can we connect these contigs?

DNA Sequencing with Read-pairs



Multiple identical copies of genome
Randomly cut genomes into large equally sized fragments of size *InsertLength*
200 bp
200 bp
InsertLength

Generate **read-pairs**: two reads from the ends of each fragment (separated by a fixed distance)

Assembling genomes from read-pairs

Which type of information can help?

->read length

->paired-end read

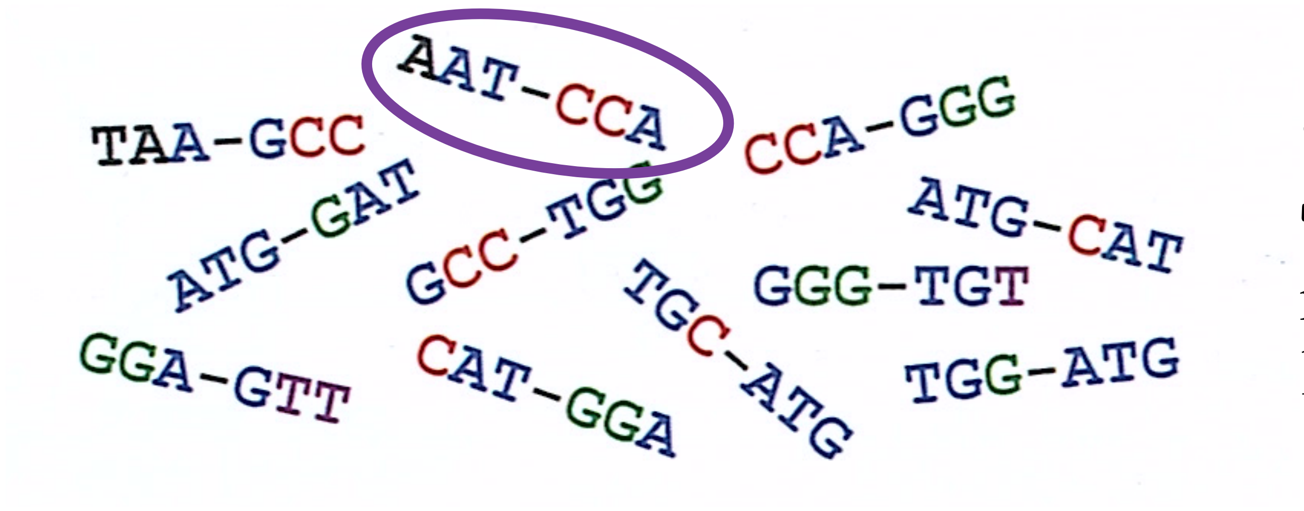
k= length sequence

d=fixed distance in the genome

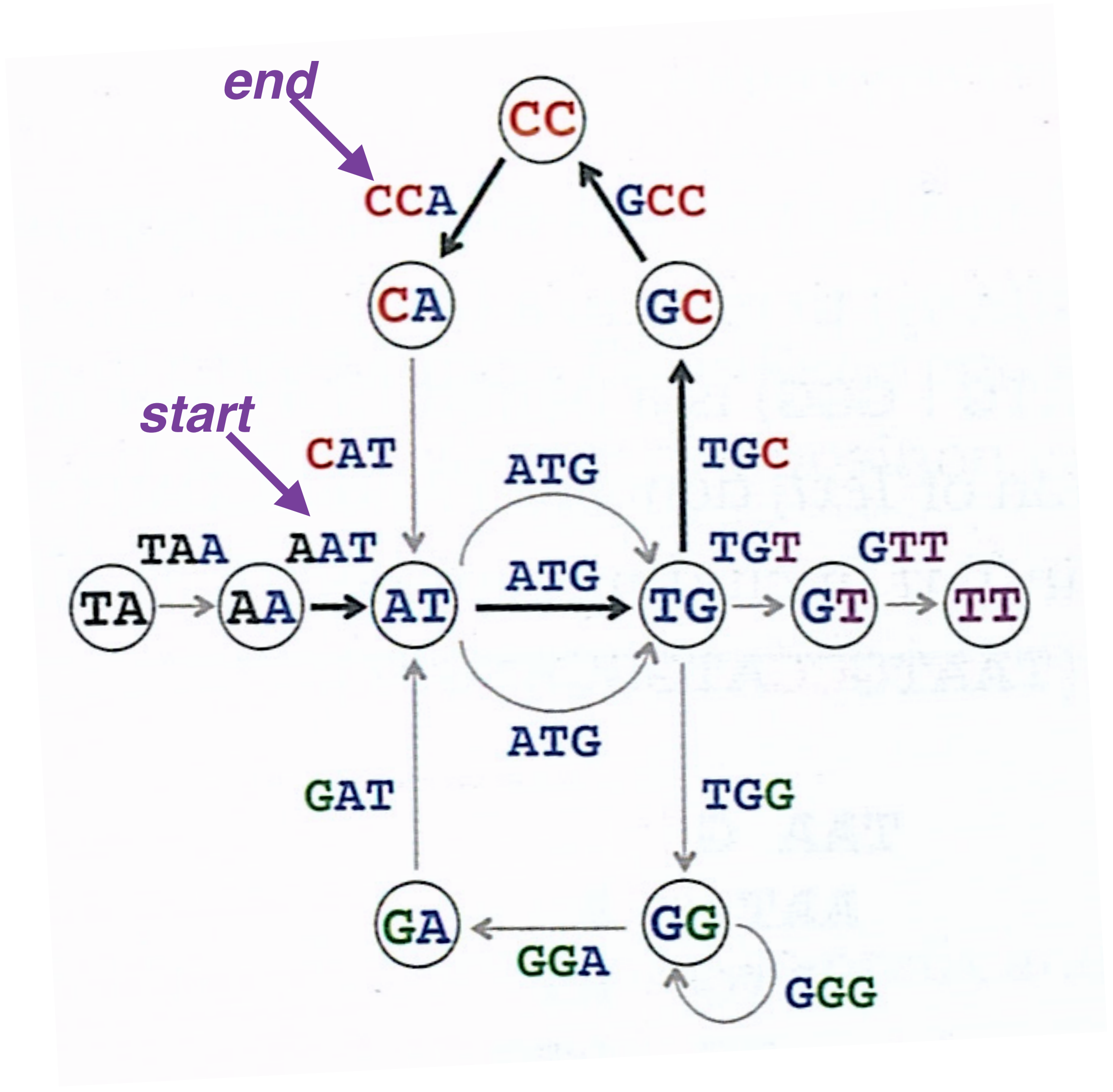
$k+d+k \longrightarrow$ If we infer the nucleotide sequence between the two reads the read length increase to 2k+d

Let Reads be the collection of all 2N k-mers reads taken from N read-pair, a read-pair formed by k-mer reads Read1 and Read2 corresponds to two edges in the deBruijn graph.

Assembling genomes from read-pairs



Set of Reads generated from **TAATGCCATGGGATGTT** and formed by reads of length 3 separated by a gap of length .



How transforming read-pairs into long virtual reads

The highlighted path of length k +d +k =3+1+3=5 between the edges labeled ATT and CCA spells out **AATGCCA**, then a long read has been created.

Basic assumption: the path that links two reads is *unique*

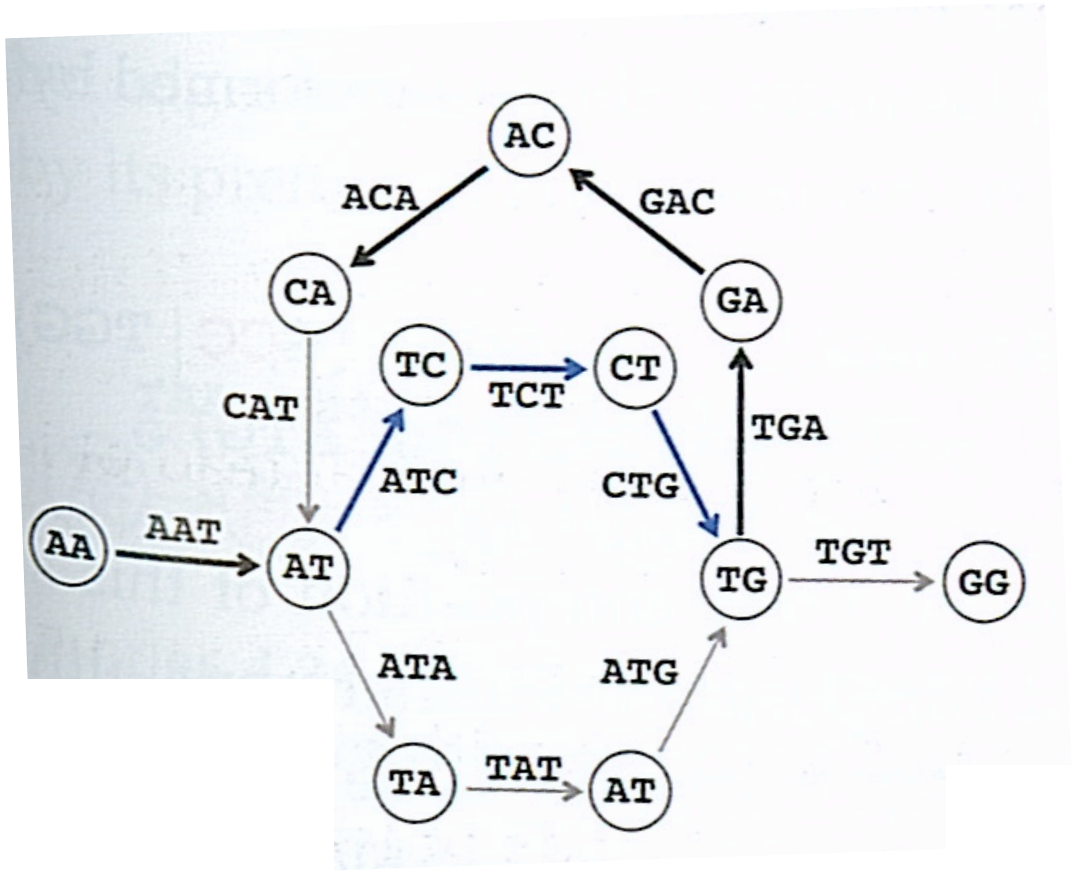
Assembling genomes from read-pairs

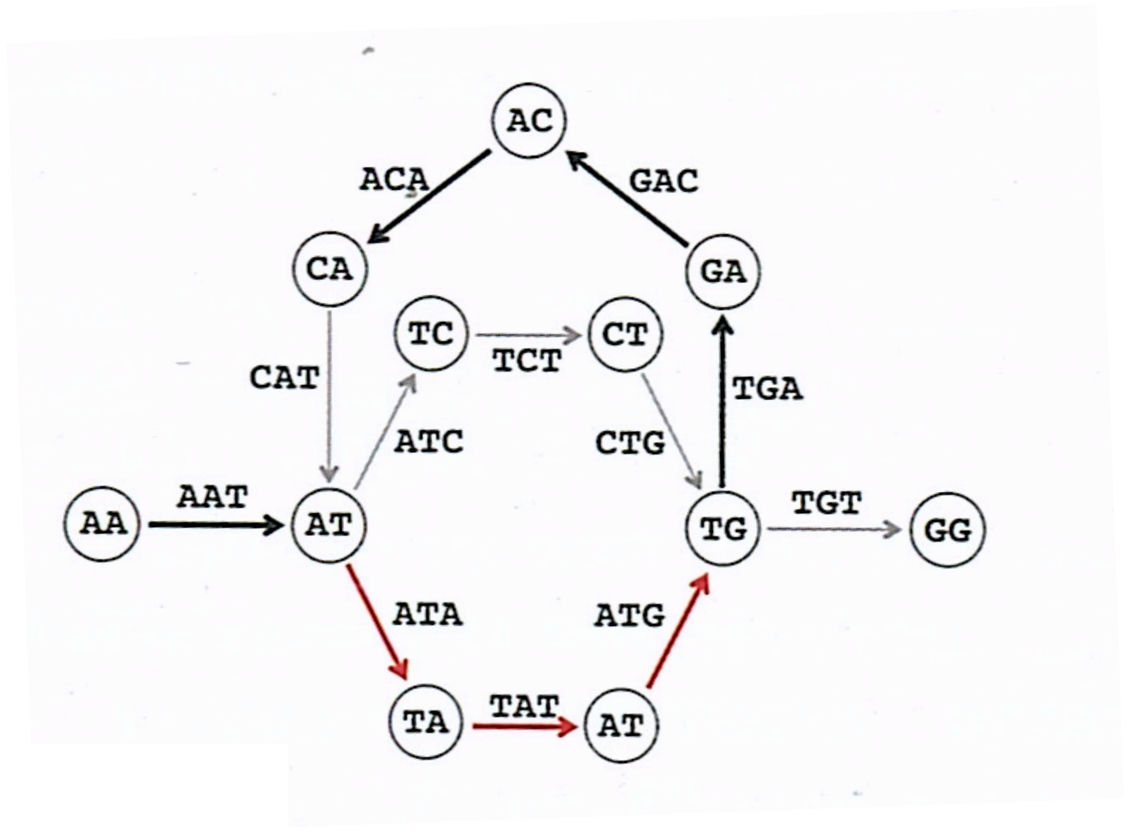
Basic assumption: the path that links two reads is *unique*

This assumption **limits the application** of this techniques since in the genome there are several repetitive genomic regions.

The **basic assumption** limits the application of long virtual reads approach to assembly read-pairs because repetitive genomic regions often contain multiple path of the same length

DeBruijn Graph for the string AATCTGACATATGG

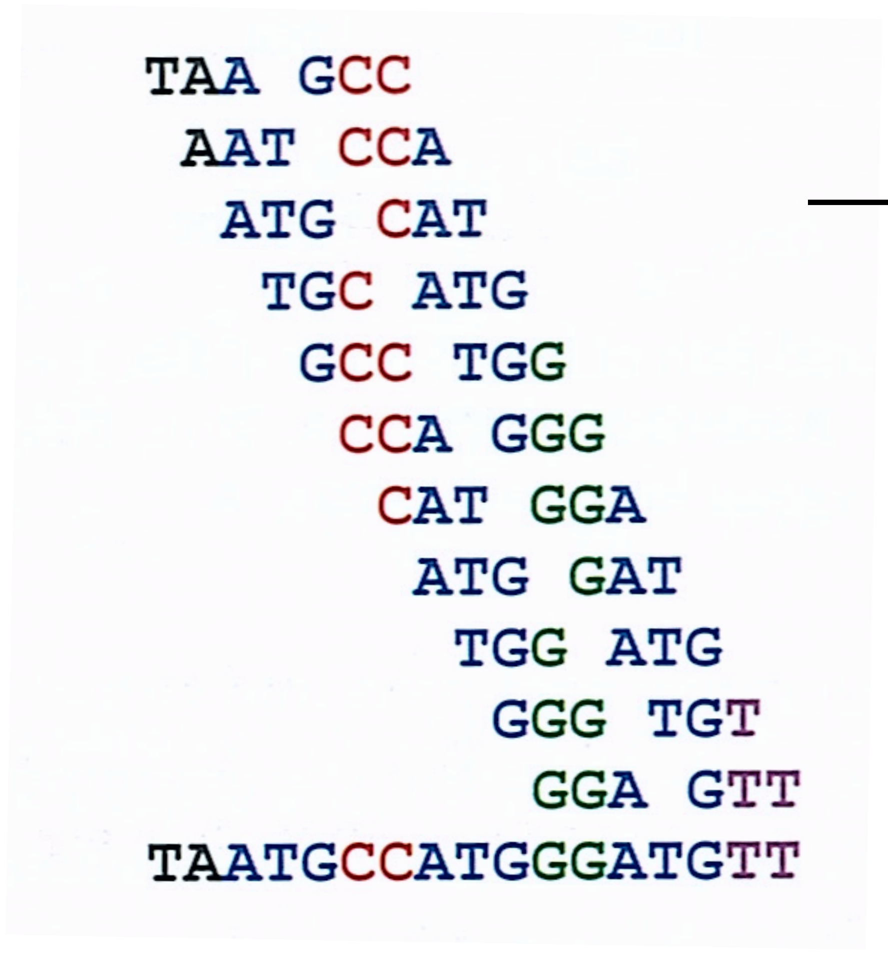




Assembling genomes from read-pairs - *Alternative approach*

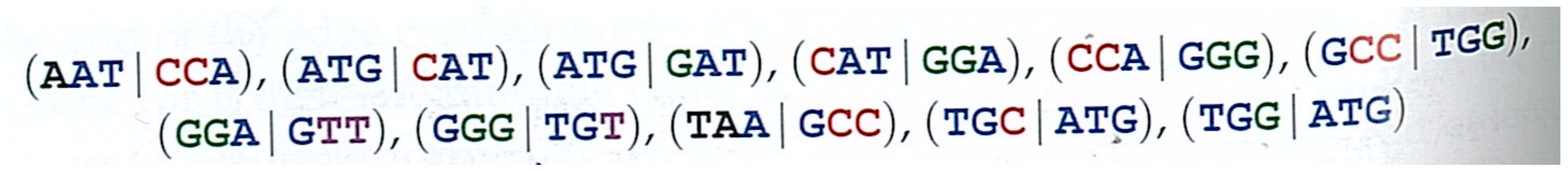
Given a string Text, a (k,d)-mer is a pair of k-mers on Text separated by distance d.
Eg (ATG | GGG) is paired (3,4)-mer in TAATGCCATGGGATGTT

PaierdComposition 3,1(TAATGCCATGGGATGTT)



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

List of the (3,1)-mers according to the lexicographic order



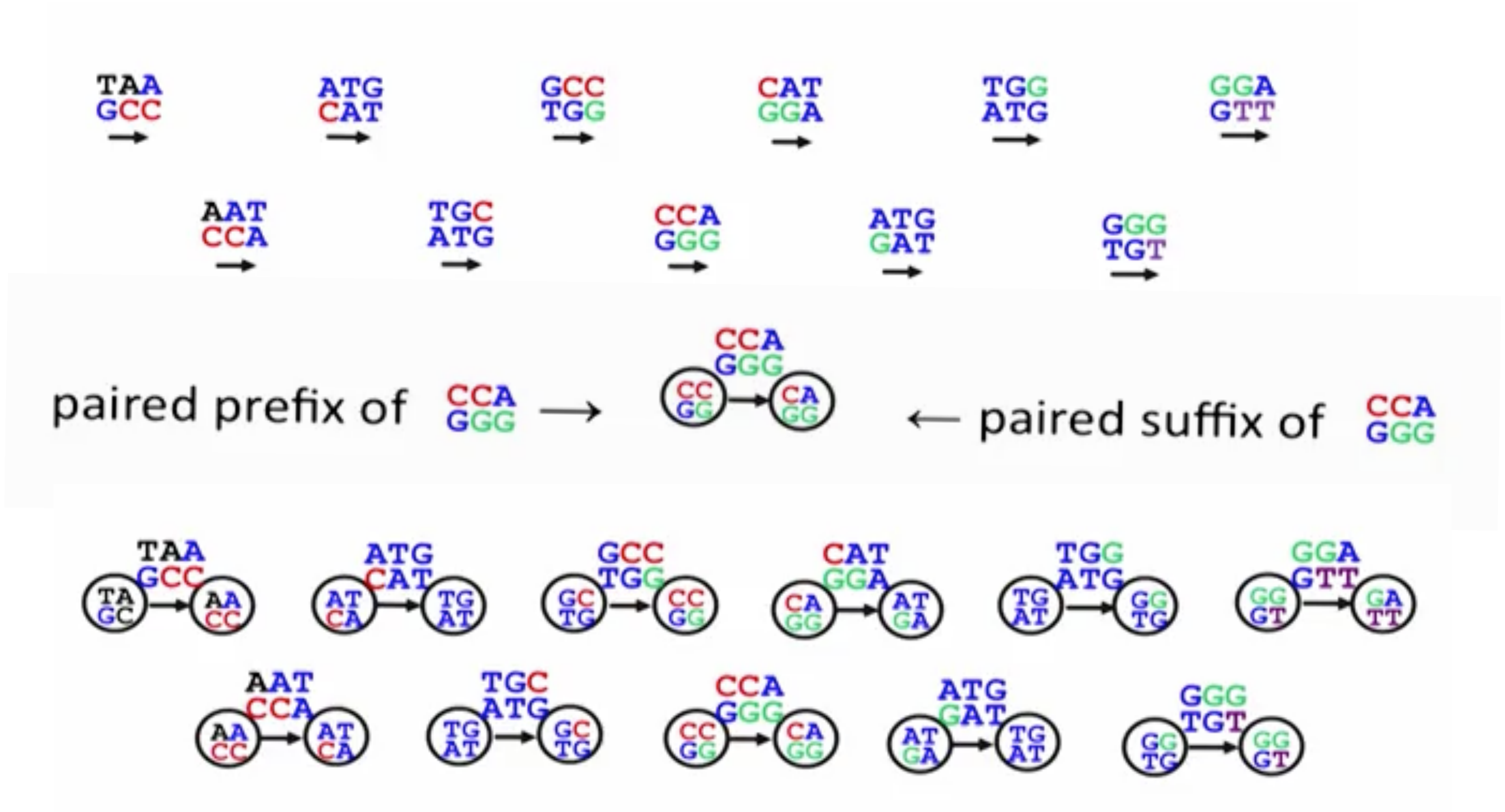
(AAT | CCA), (ATG | CAT), (ATG | GAT), (CAT | GGA), (CCA | GGG), (GCC | TGG), (GGA | GTT), (GGG | TGT), (TAA | GCC), (TGC | ATG), (TGG | ATG)

TAATGCCATGGGATGTT

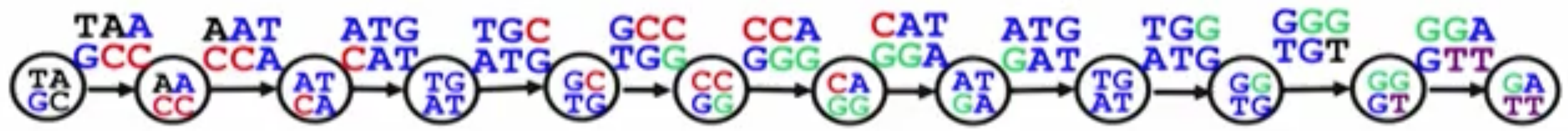
TAATGGGATCCGATGTT

The list of the (3,1)-mers of these two string is different, while it is possible that 3-mers can be repeated

Assembling genomes from read-pairs - *Alternative approach*



glue identical nodes



Paired de Bruin graph

Given a (k,d)-mer $(a_1,\dots a_k|b_1,\dots b_k)$

$\text{PREFIX}((a_1\dots a_k|b_1,\dots b_k))=((a_1\dots a_{k-1}|b_1,\dots b_{k-1}))$

$\text{SUFFIX}((a_1\dots a_k|b_1,\dots b_k))=((a_2\dots a_k|b_2,\dots b_k))$

Define the set of ***prefix*** and ***suffix*** of (k,d)-mers.

PREFIX(GAC|TCA) =(GA|TC)
SUFFIX(GAC|TCA) =(AC|CA)

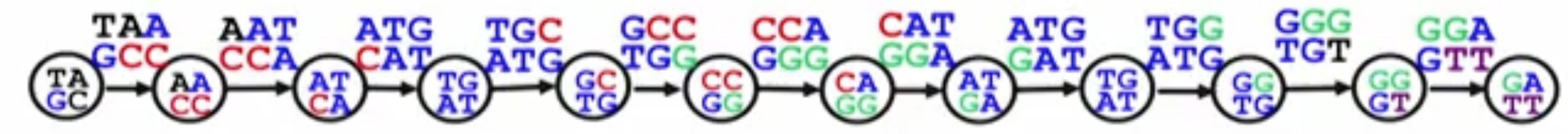
Note that for consecutive (k,d)-mers appearing in the Text, the suffix of the first (k,d)mer is equal to the prefix of the second (k,d)-mer.

$\text{SUFFIX}((\mathbf{TAA}\,|\,\mathbf{GCC})) = \text{PREFIX}((\mathbf{AAT}\,|\,\mathbf{CCA})) = (\mathbf{AA}\,|\,\mathbf{CC})$.

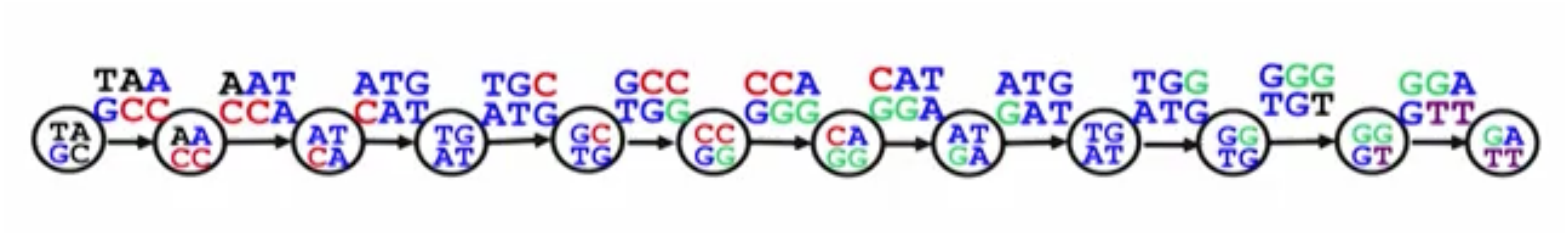
TAATGCCATGGGATGTT

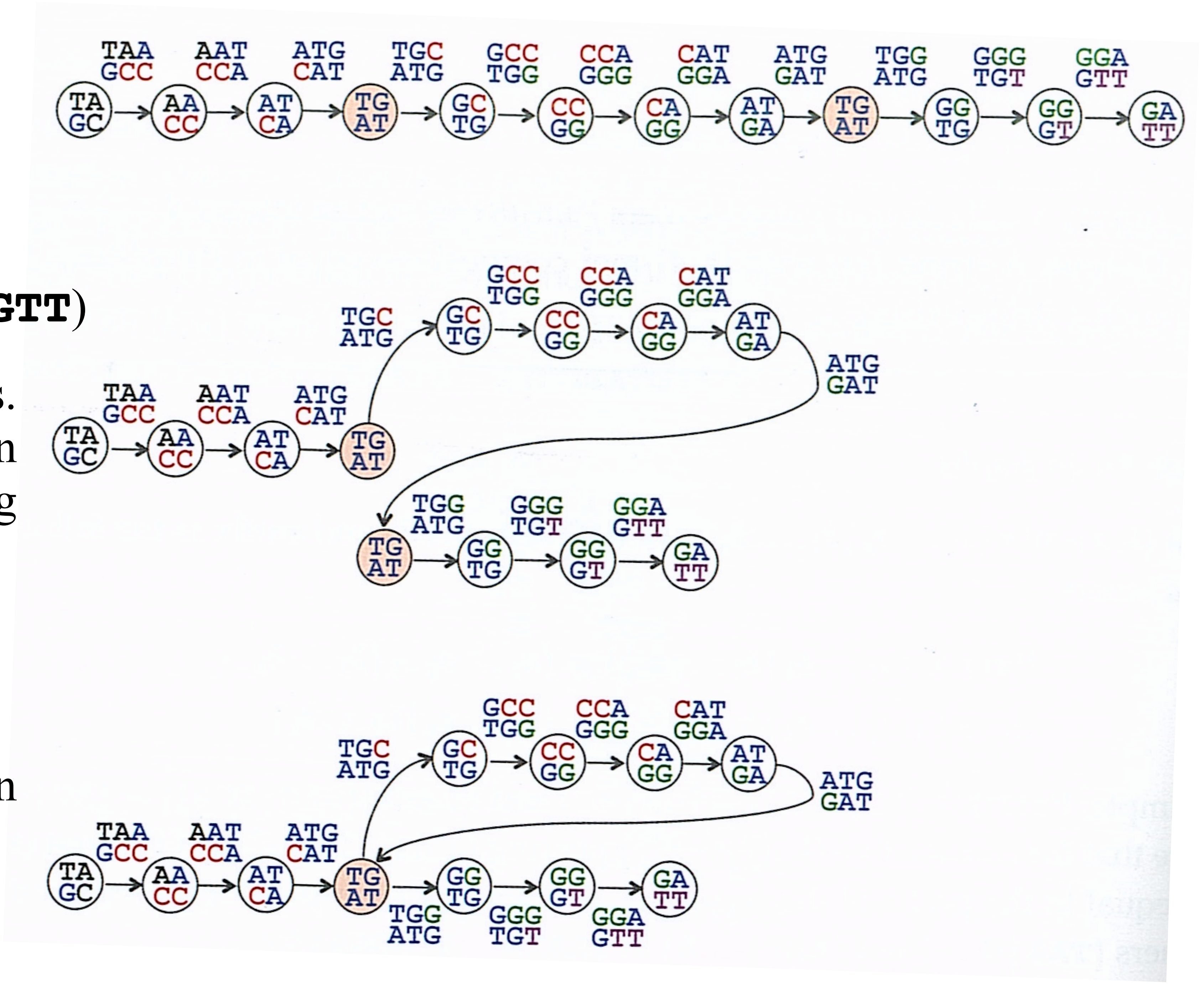
A **PathGraph** k,d(*Text*) that represents a path formed by *Text* is creating by nodes in which are stored ***suffix*** and ***prefix*** and edges that are labeled by the (k,d)mers

$\text{PATHGRAPH}_{3,1}$(TAATGCCATGGGATGTT)



Paired de Bruijn graph



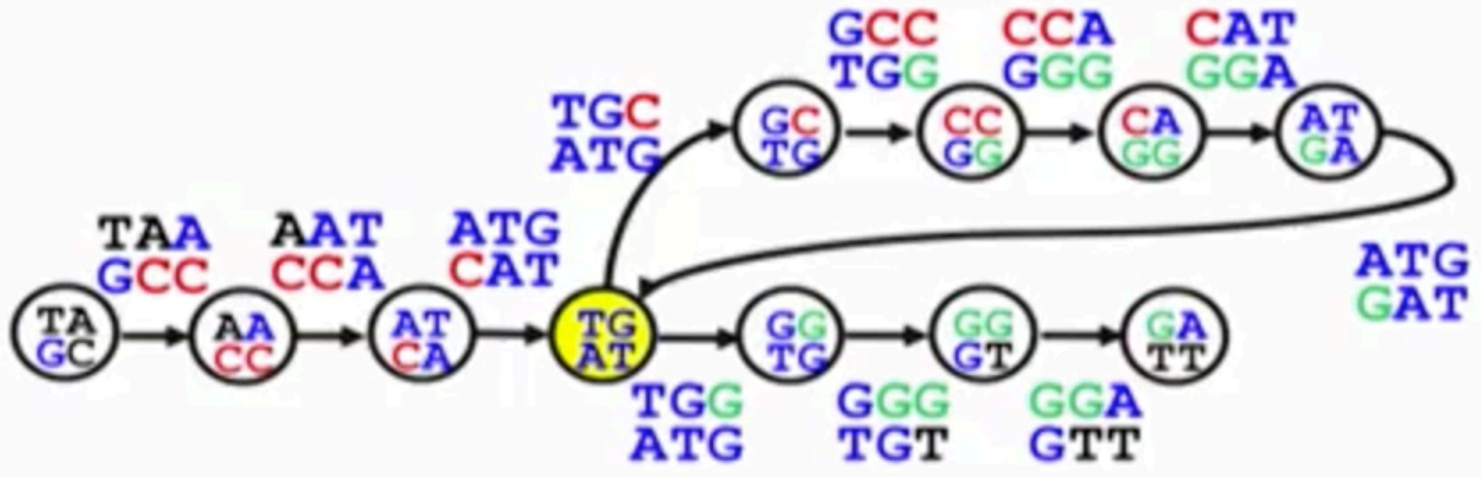


From the
PathGraph3,1(TAATGCCATGGGATGTT)

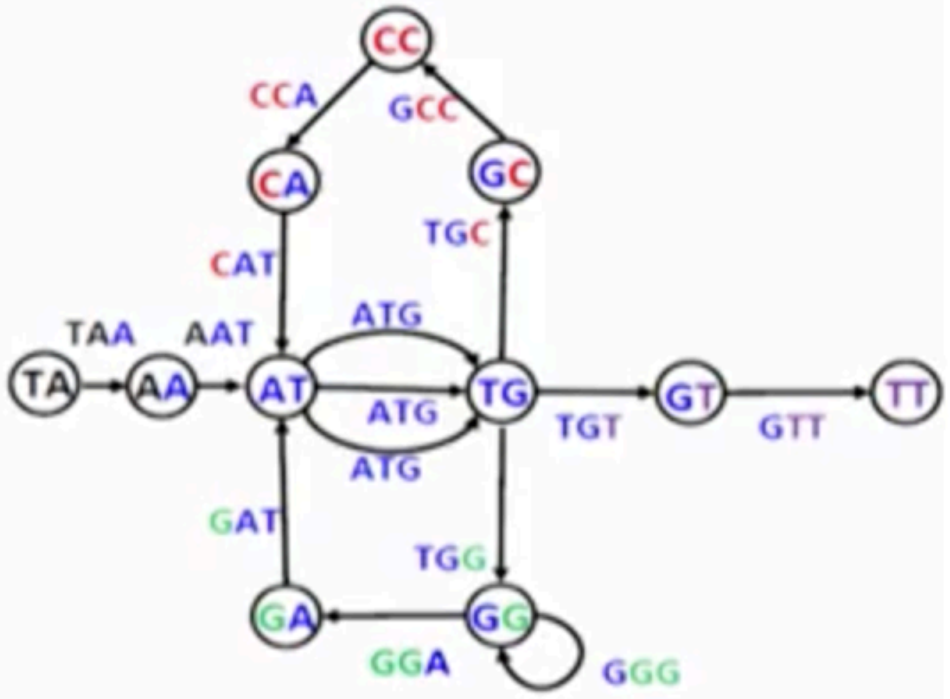
is formed by 11 edges and 12 nodes. The Paired deBruijn graphics again formed by gluing the nodes sharing the same label.

and follow the Eulerian path we can found the complete *Text*.

Which Graph Represents a Better Assembly?



Paired de Bruijn Graph



De Bruijn Graph

Which Graph Represents a Better Assembly?

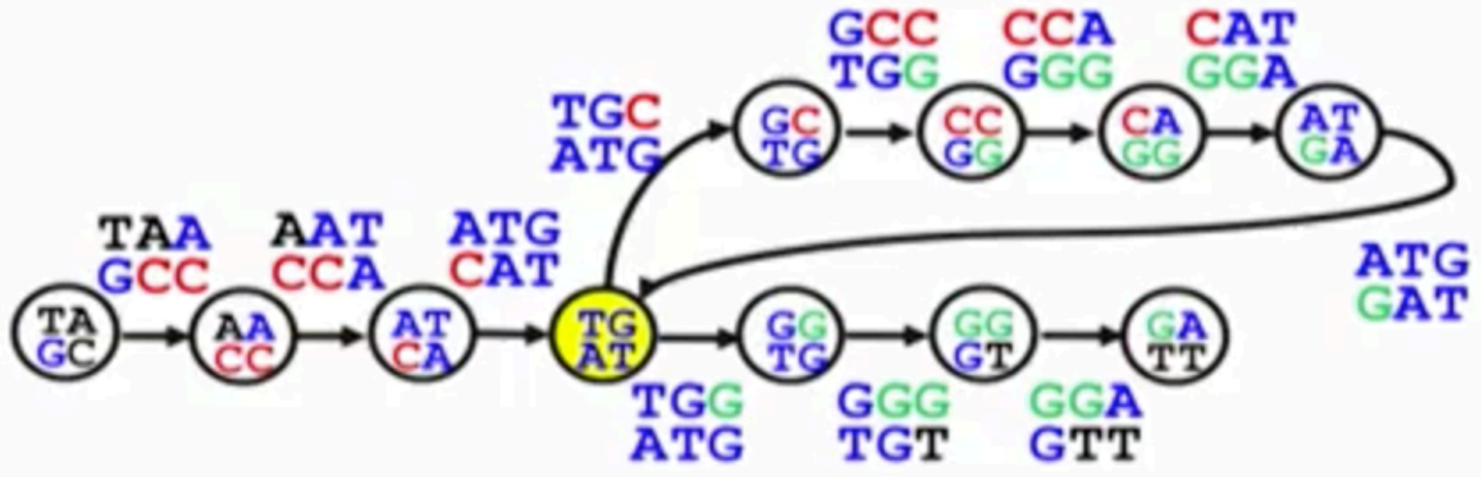
Unique genome reconstruction

TAATGCCATGGATGTT

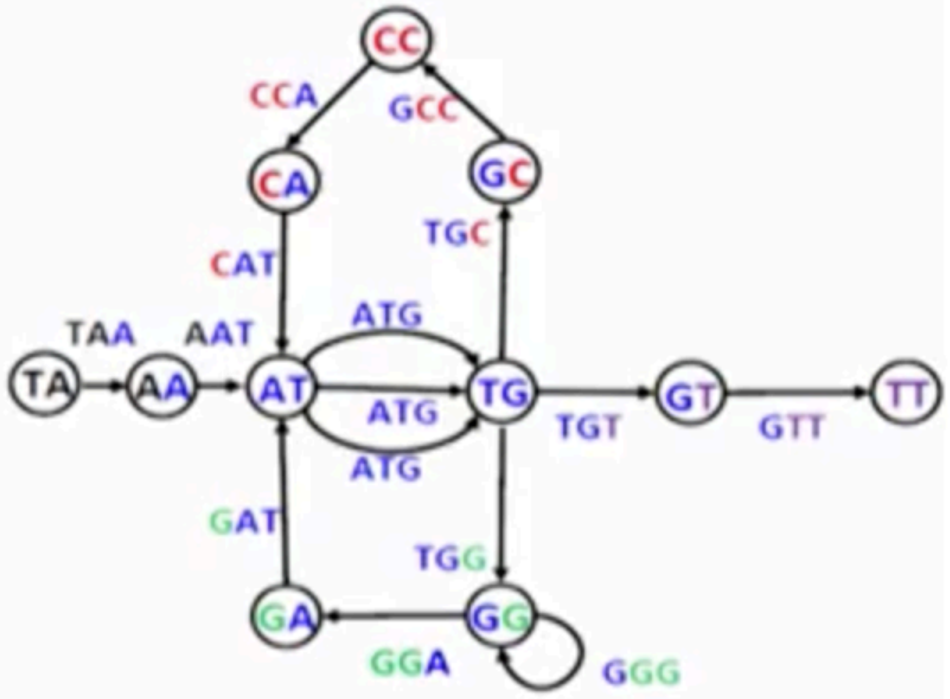
Multiple genome reconstructions

TAATGCCATGGATGTT

TAATGGATGCCATGTT



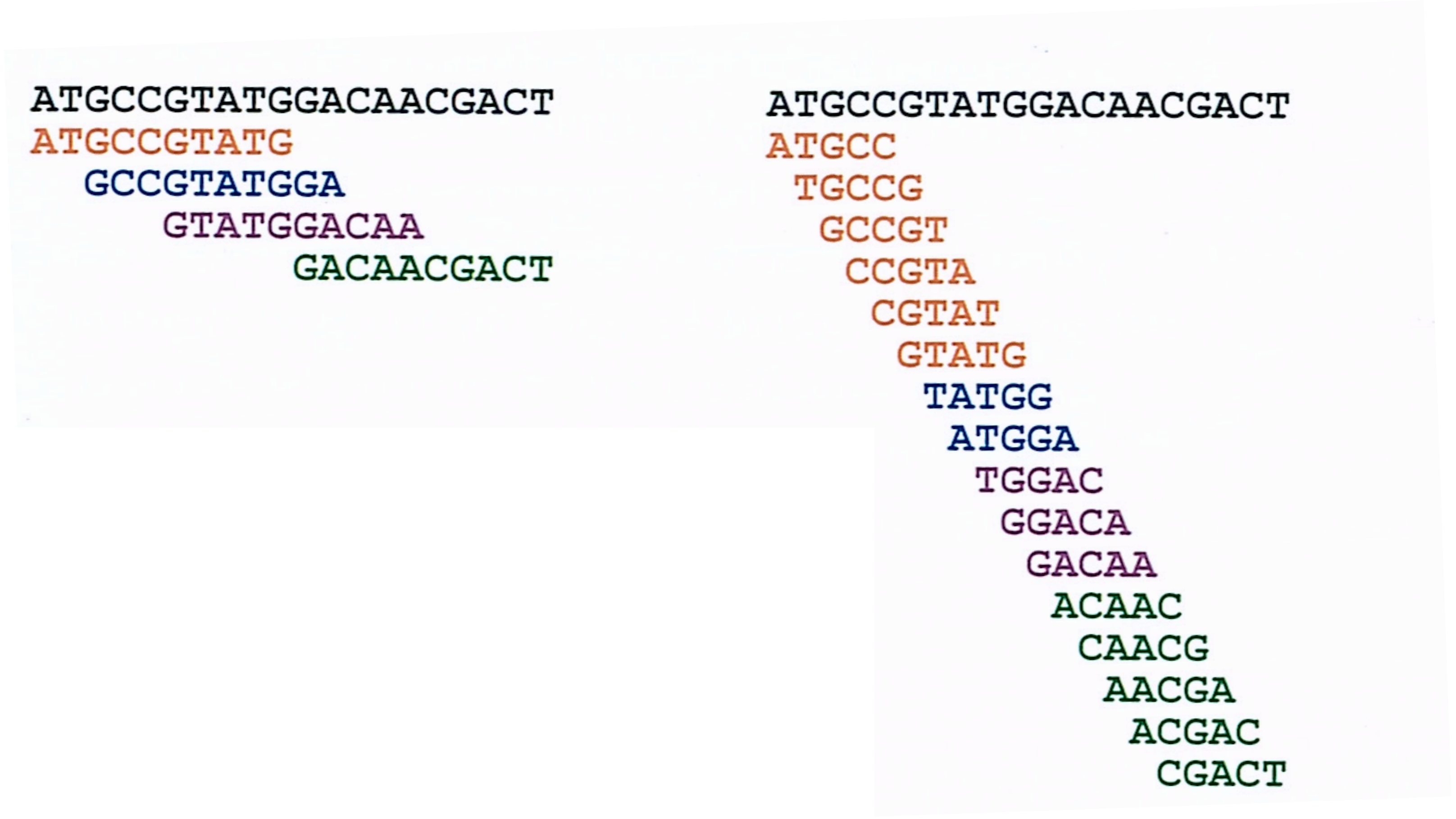
Paired de Bruijn Graph



De Bruijn Graph

Genome assembly

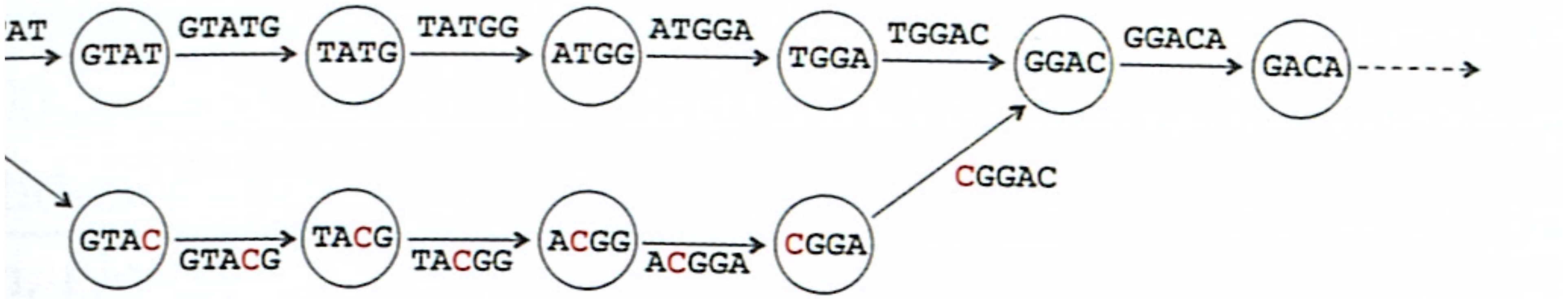
Assumption of perfect coverage $\xrightarrow{\textbf{solved by}}$ read breaking



```
ATGCCGTATGGACAACGACT          ATGCCGTATGGACAACGACT
ATGCCGTATG                    ATGCC
  GCCGTATGGA                   TGCCG
     GTATGGACAA                 GCCGT
          GACAACGACT             CCGTA
                                  CGTAT
                                   GTATG
                                    TATGG
                                     ATGGA
                                      TGGAC
                                       GGACA
                                        GACAA
                                         ACAAC
                                          CAACG
                                           AACGA
                                            ACGAC
                                             CGACT
```

Connect long contigs $\xrightarrow{\textbf{solved by}}$ de Bruijn graph

Read Errors $\xrightarrow{\text{solved by}}$ bubble identification


AT
GTAT
GTATG
TATG
TATGG
ATGG
ATGGA
TGGA
TGGAC
GGAC
GGACA
GACA
CGGAC
GTAC
GTACG
TACG
TACGG
ACGG
ACGGA
CGGA