AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

PAIRED DE BRUIJN GRAPHS

22BIO211: Intelligence of Biological Systems - 2

Dr. Manjusha Nair M
Amrita School of Computing, Amritapuri

Email : manjushanair@am.amrita.edu
Contact No: 9447745519

Constructing Paired de Bruijn graphs from Genome

1. Representing Genome as a Path
2. Labeling Nodes by Paired Prefixes and Suffixes
3. Glue nodes with identical labels

Constructing Paired de Bruijn graphs from paired k-mers

Paired de Bruijn graph for a collection of paired *k*-mers:

- Represent every paired *k*-mer as an edge between its paired prefix and paired suffix.
- Glue **ALL** nodes with identical labels.

Prefix – suffix of (k,d) -mers

- Given a (k, d)-mer (a1 . . . ak | b1, . . . bk), we define its **prefix** and **suffix** as the following (k-1,d+1): mers

$$\text{PREFIX}((a_1 \dots a_k \mid b_1, \dots b_k)) = (a_1 \dots a_{k-1} \mid b_1 \dots b_{k-1})$$
$$\text{SUFFIX}((a_1 \dots a_k \mid b_1, \dots b_k)) = (a_2 \dots a_k \mid b_2 \dots b_k)$$

For example,

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

Prefix – suffix of (k,d) -mers

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

  For example, for the consecutive (k, d)-mers
  (**TAA** | **GCC**) and (**AAT** | **CCA**) in
  TAATGCCATGGGATGTT,

  SUFFIX((**TAA** | **GCC**)) = PREFIX((**AAT** | **CCA**)) = (**AA** | **CC**) .

1. Representing Genome as a Path Graph

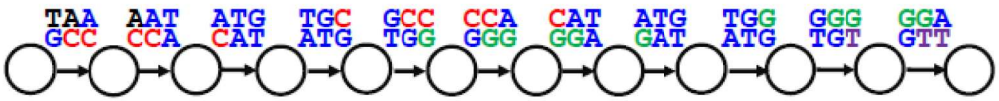
Genome

TAATGCCATGGGATGTT

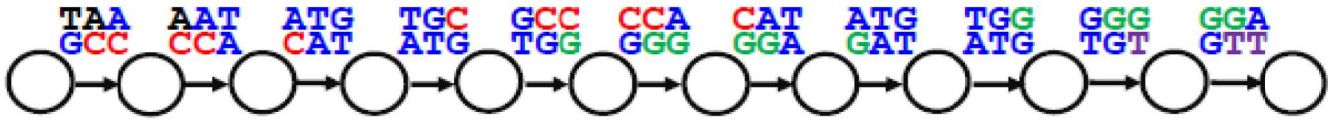
Paired K-mers

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

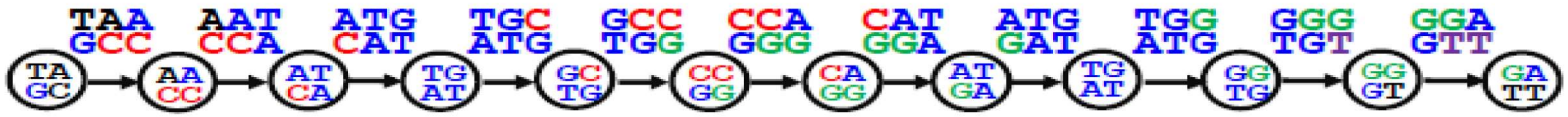
PathGraph : edges corresponding to all (*k*, *d*)-mers.

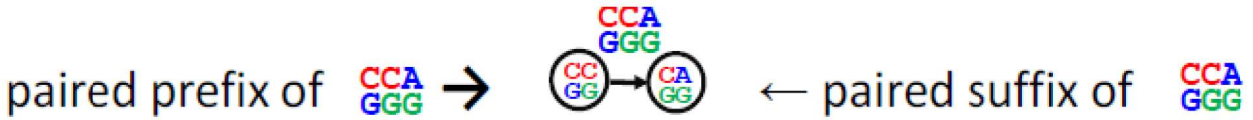


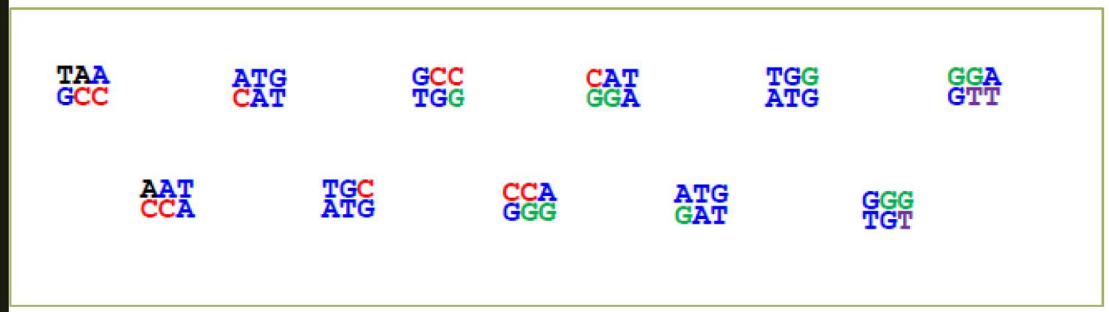
2. Labeling Nodes by Paired Prefixes and Suffixes

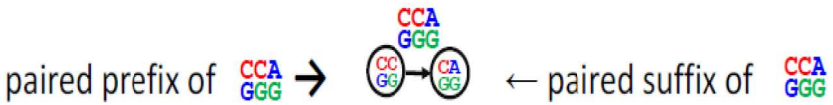


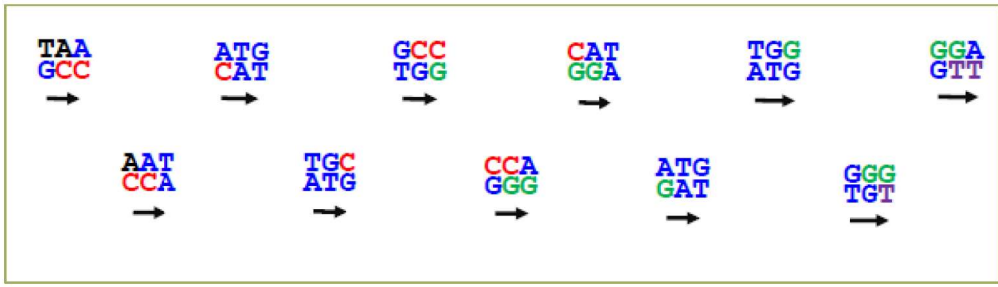
Labeling Nodes by Paired Prefixes and Suffixes

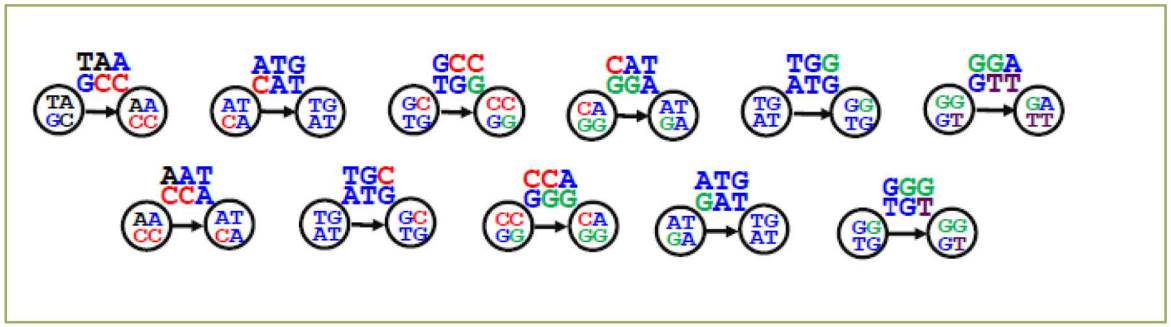




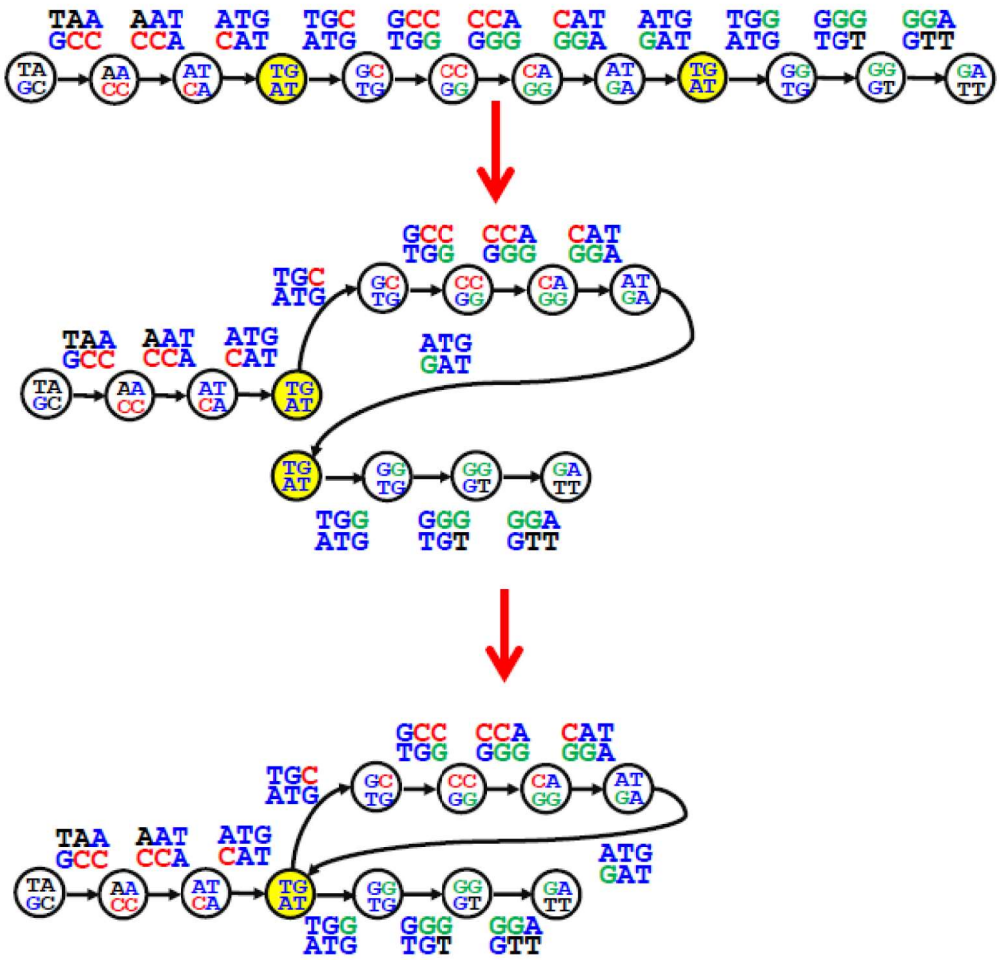

TAA GCC
ATG CAT
GCC TGG
CAT GGA
TGG ATG
GGA GTT
AAT CCA
TGC ATG
CCA GGG
ATG GAT
GGG TGT

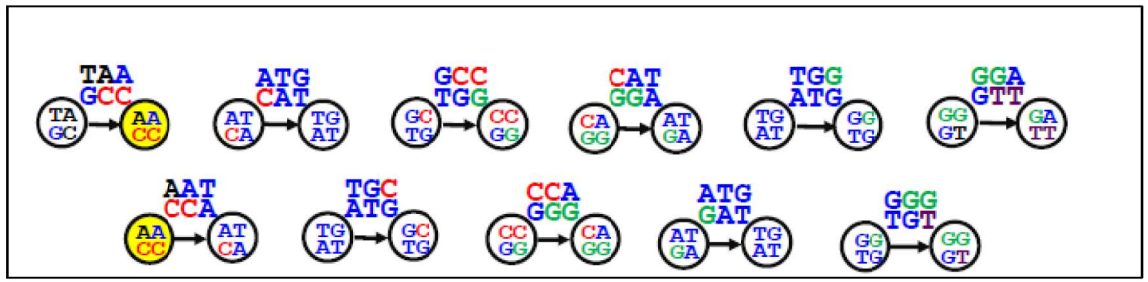

paired prefix of CCA GGG
CCA GGG
CC GG
CA GG
← paired suffix of CCA GGG

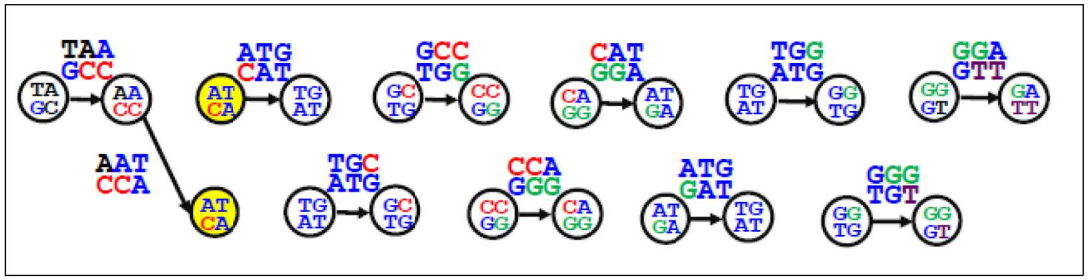

TAA GCC
ATG CAT
GCC TGG
CAT GGA
TGG ATG
GGA GTT
AAT CCA
TGC ATG
CCA GGG
ATG GAT
GGG TGT

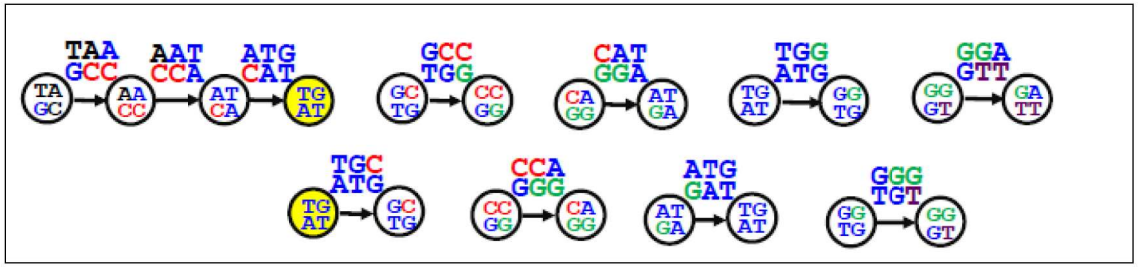

TAA GCC
TA GC
AA CC
ATG CAT
AT CA
TG AT
GCC TGG
GC TG
CC GG
CAT GGA
CA GG
AT GA
TGG ATG
TG AT
GG TG
GGA GTT
GG GT
GA TT
AAT CCA
AA CC
AT CA
TGC ATG
TG AT
GC TG
CCA GGG
CC GG
CA GG
ATG GAT
AT GA
TG AT
GGG TGT
GG TG
GG GT

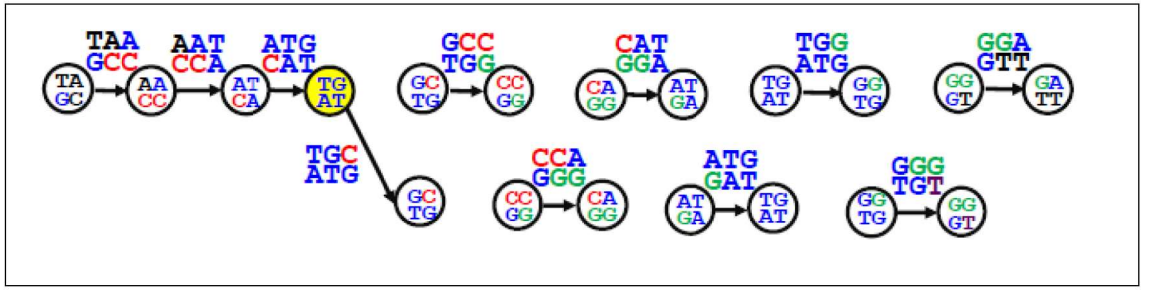
3. Glue nodes with identical labels



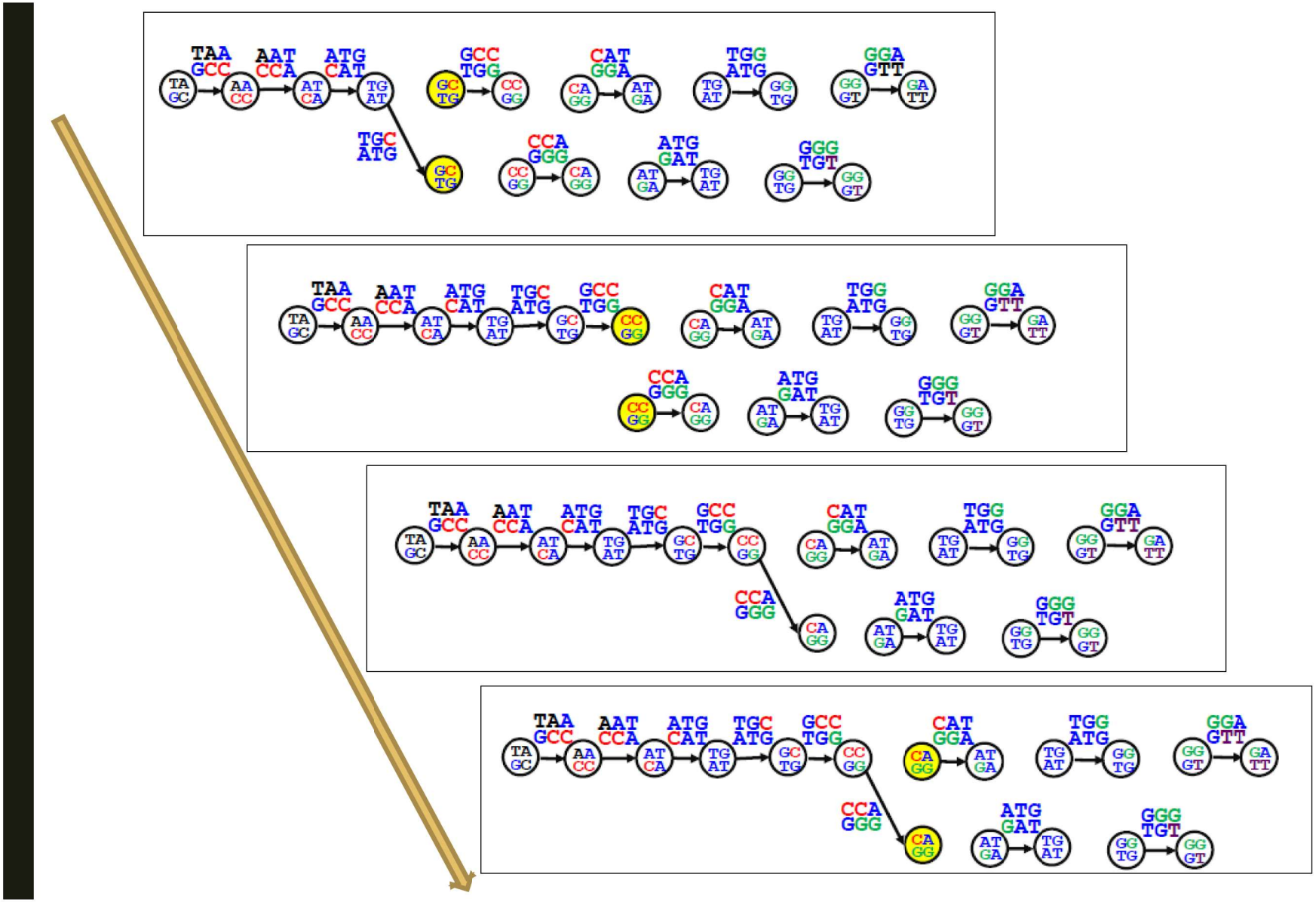


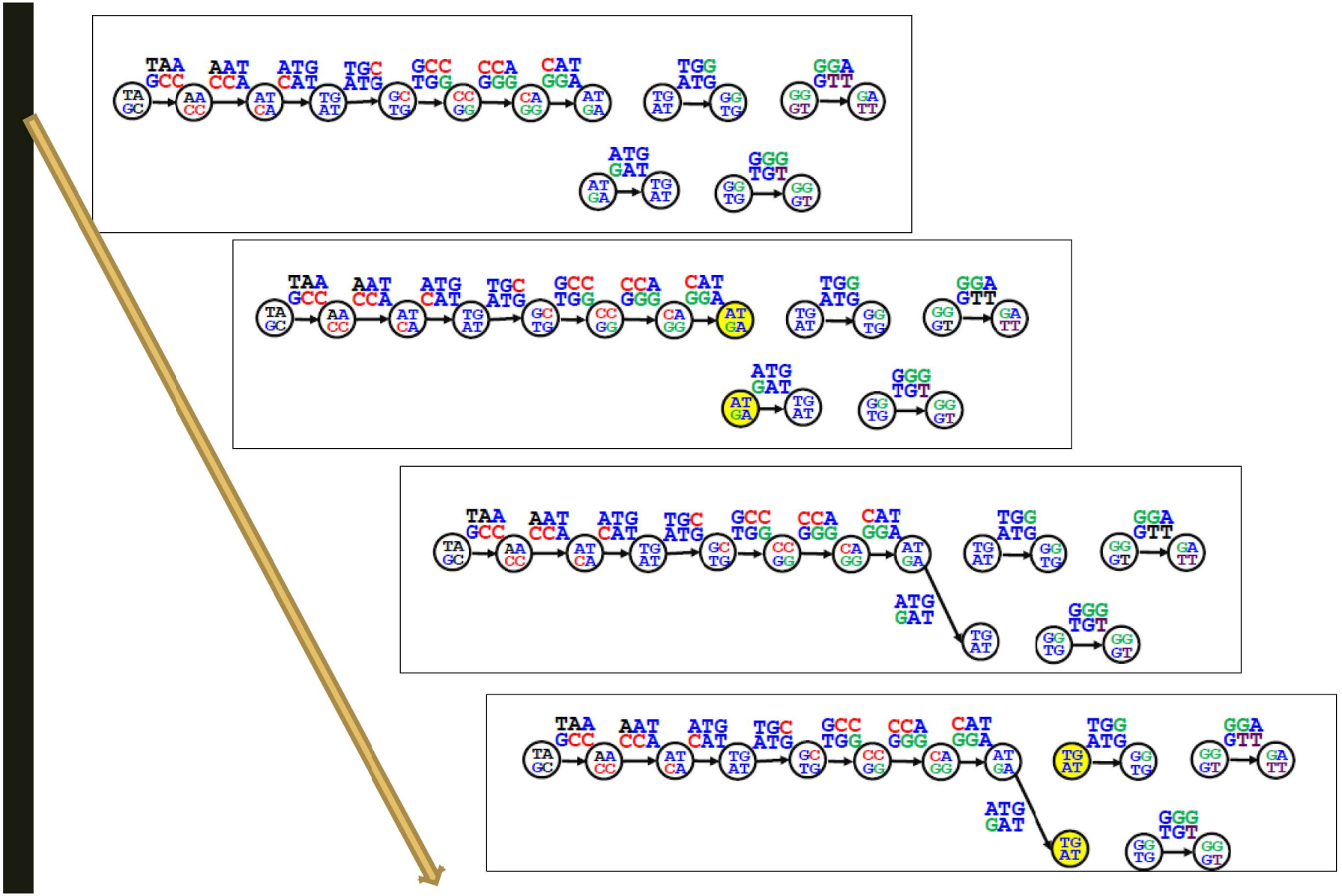


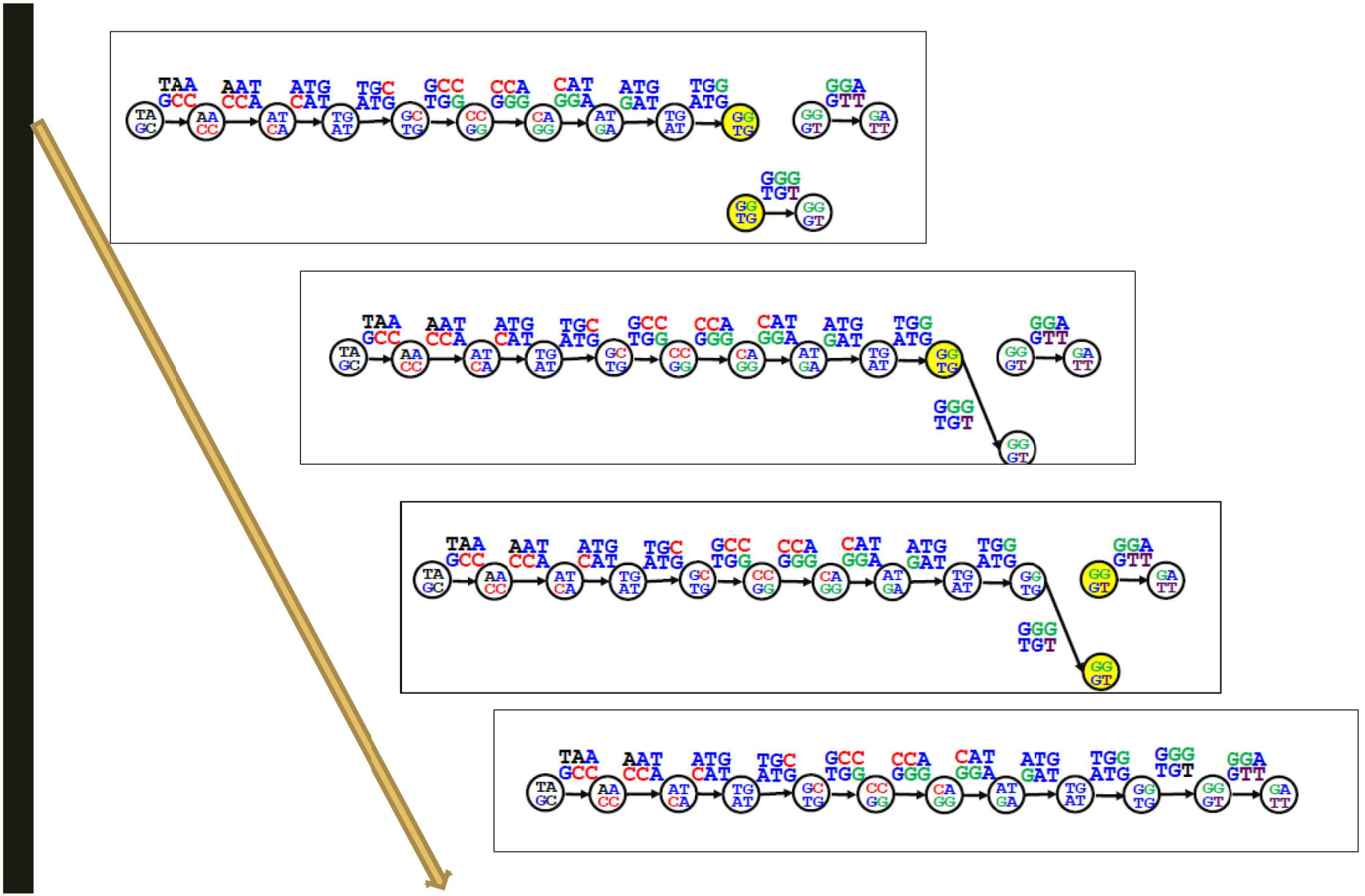




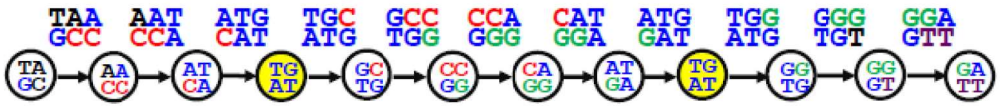
Glue **ALL** nodes with identical labels.

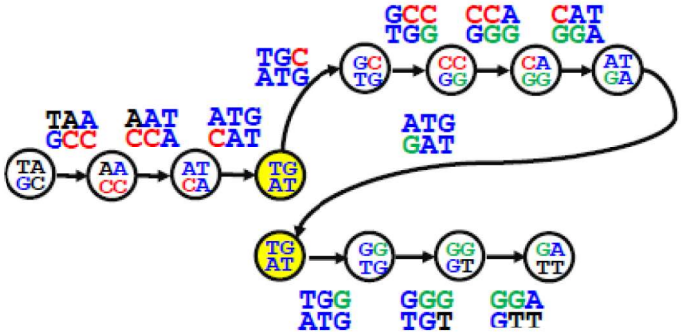

TAA
GCC
AAT
CCA
ATG
CAT
GCC
TGG
CAT
GGA
TGG
ATG
GGA
GTT
TGC
ATG
CCA
GGG
ATG
GAT
GGG
TGT

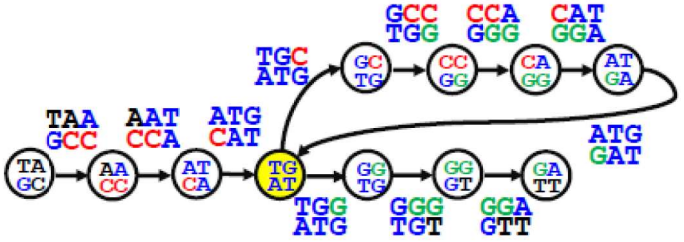

TAA GCC
AAT CCA
ATG CAT
TGC ATG
GCC TGG
CCA GGG
CAT GGA
TGG ATG
GGA GTT
ATG GAT
GGG TGT


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

We Are Not Done with Gluing Yet

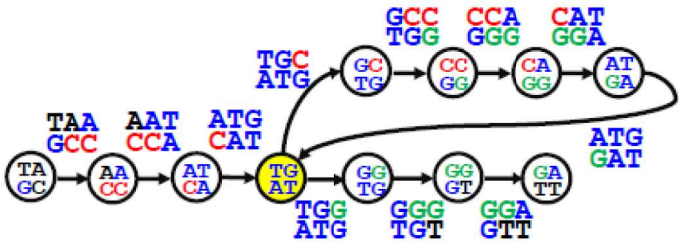






Which Graph Represents a Better Assembly?

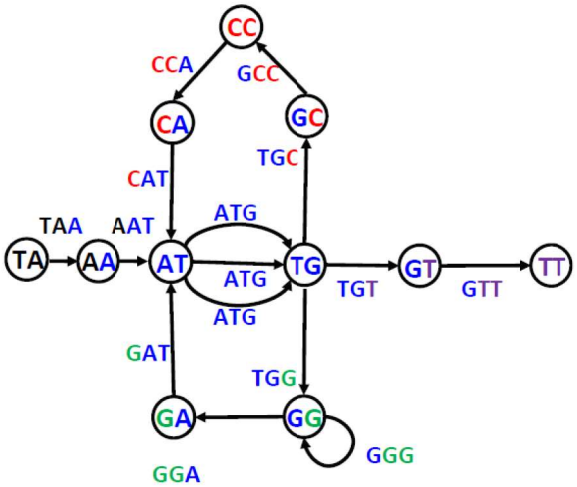
Paired de Bruijn Graph



Unique genome reconstruction

TAATGCCATGGGATGTT

de Bruijn Graph



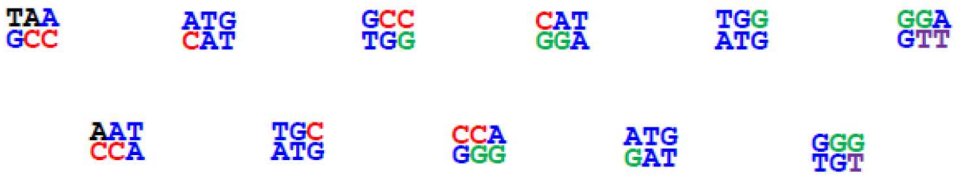
Multiple genome reconstructions

TAATGCCATGGGATGTT

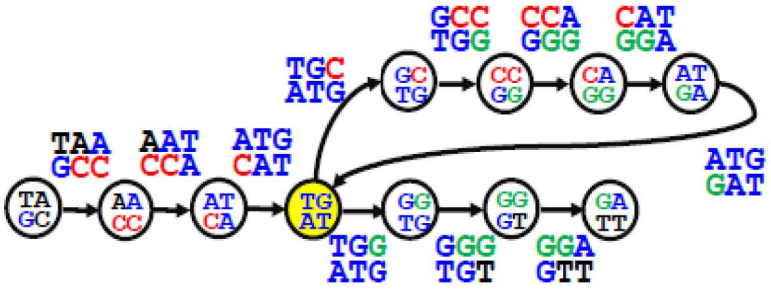
TAATGGGATGCCATGTT

Paired deBruijn Graphs

Paired K-mers



Paired de Bruijn Graphs



String Reconstruction

Solution of the String Reconstruction Problem:

- Eulerian path in the de Bruijn graph constructed from a *k*-mer composition.

Solution of the String Reconstruction Problem from Read Pairs:

- Eulerian path in the de Bruijn graph constructed from a *(k,d)* - mer composition.

Eulerian Path through Paired deBruijn Graphs

We saw that every Eulerian path in the de Bruijn graph constructed from a k-mer composition spells out a solution of the String Reconstruction Problem.

But is this the case for the paired de Bruijn graph?

String Reconstruction using Eulerian path in paired de Bruijn graphs

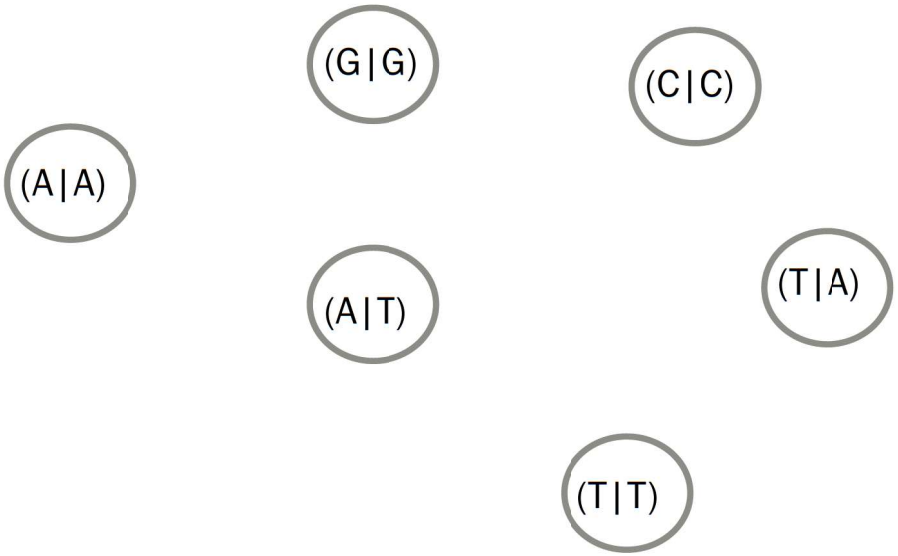
Create de Bruijn Graph: constructed from the collection of nine (2,1) mers :
(AG|AG), (AG|TG), (CA|CT), (CT|CA), (CT|CT), (GC|GC), (GC|GC), (GC|GC), (TG|TG).

Step 1 : Find Prefix suffix pair of (K,d)-mers

(K,d) -mer	Prefix pair	Suffix pair
(AG|AG)	(A|A)	(G|G)
(AG|TG)	(A|T)	(G|G)
(CA|CT)	(C|C)	(A|T)
(CT|CA)	(C|C)	(T|A)
(CT|CT)	(C|C)	(T|T)
(GC|GC)	(G|G)	(C|C)
(GC|GC)	(G|G)	(C|C)
(GC|GC)	(G|G)	(C|C)
(TG|TG)	(T|T)	(G|G)

String Reconstruction using Eulerian path in paired de Bruijn graphs

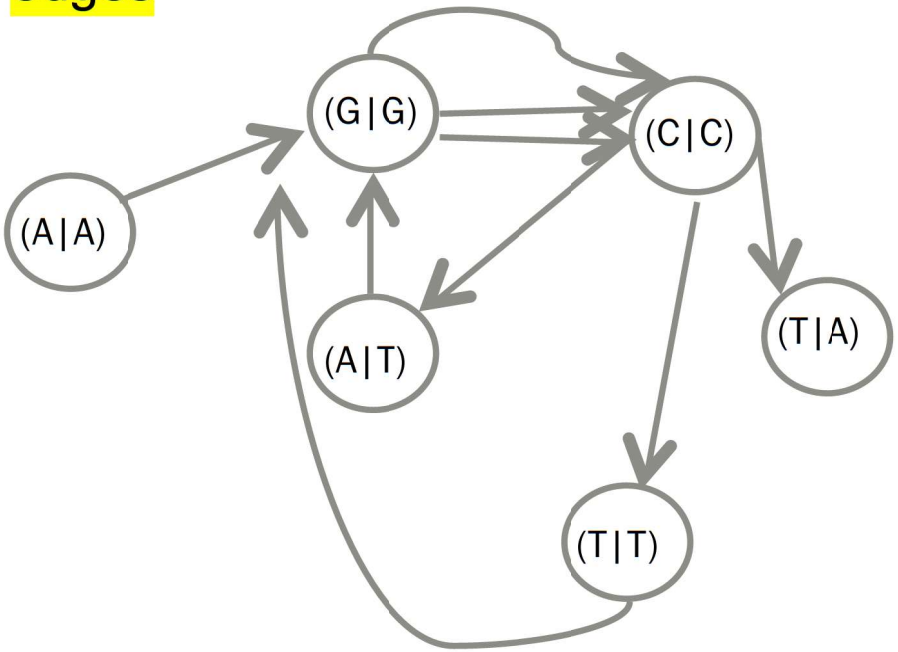
Step 2 : Select Unique Vertices



(K,d) -mer	Prefix pair	Suffix pair
(AG|AG)	(A|A)	(G|G)
(AG|TG)	(A|T)	(G|G)
(CA|CT)	(C|C)	(A|T)
(CT|CA)	(C|C)	(T|A)
(CT|CT)	(C|C)	(T|T)
(GC|GC)	(G|G)	(C|C)
(GC|GC)	(G|G)	(C|C)
(GC|GC)	(G|G)	(C|C)
(TG|TG)	(T|T)	(G|G)

String Reconstruction using Eulerian path in paired de Bruijn graphs

Step 3 : Preserve the edges between each Prefix-Suffix pair and label the edges

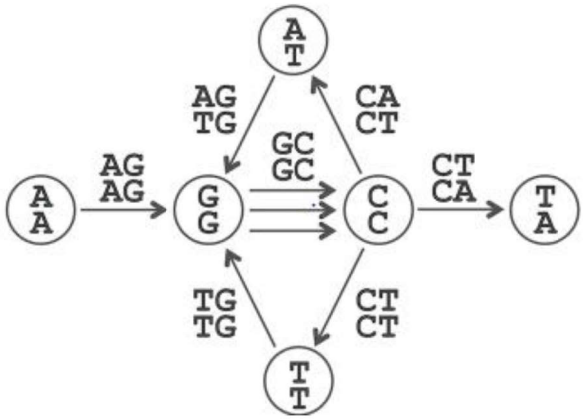


(K,d) -mer	Prefix pair	Suffix pair
(AG|AG)	(A|A)	(G|G)
(AG|TG)	(A|T)	(G|G)
(CA|CT)	(C|C)	(A|T)
(CT|CA)	(C|C)	(T|A)
(CT|CT)	(C|C)	(T|T)
(GC|GC)	(G|G)	(C|C)
(GC|GC)	(G|G)	(C|C)
(GC|GC)	(G|G)	(C|C)
(TG|TG)	(T|T)	(G|G)

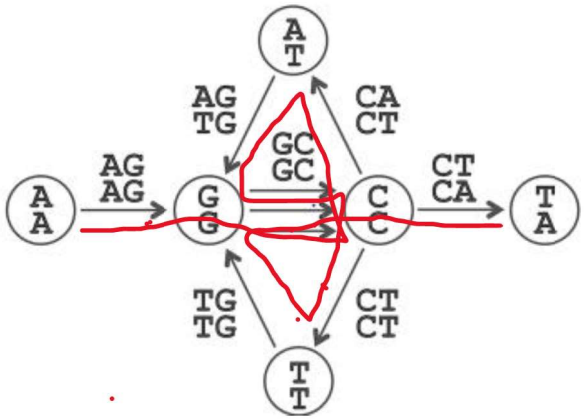
String Reconstruction using Eulerian path in paired de Bruijn graphs

Create de Bruijn Graph: constructed from the collection of nine (2,1)-mers

(AG|AG), (AG|TG), (CA|CT), (CT|CA), (CT|CT), (GC|GC), (GC|GC), (GC|GC), (TG|TG).



String Reconstruction using Eulerian path in paired de Bruijn graphs

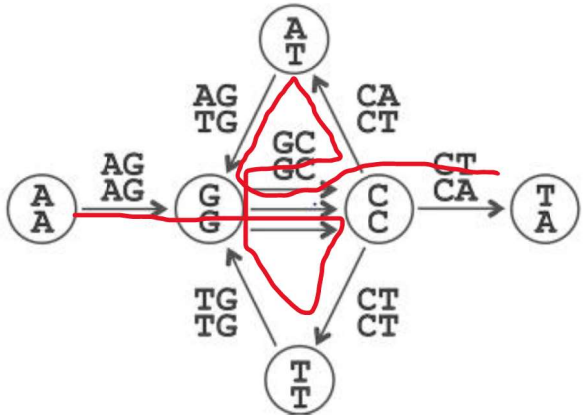


Reconstruct the genome spelled by the following Eulerian path

(AG|AG) →(GC|GC) → (CA|CT) → (AG|TG) → (GC|GC) →
(CT|CT) → (TG|TG) → (GC|GC) → (CT|CA)

```
AG-AG
 GC-GC
  CA-CT
   AG-TG
    GC-GC
     CT-CT
      TG-TG
       GC-GC
        CT-CA
AGCAGCTGCTGCA
```

A pitfall of paired de Bruijn graphs



- Look again at the same graph, and consider the following Eulerian Path:
- (AG|AG) →(GC|GC) → (CT|CT) → (TG|TG) → (GC|GC) → (CA|CT) → (AG|TG) → (GC|GC) → (CT|CA).

Can you reconstruct a genome spelled by this path?

```
AG-AG
 GC-GC
  CT-CT
   TG-TG
    GC-GC
     CA-CT
      AG-TG
       GC-GC
        CT-CA
AGC?GC?GCTGCA
```

Here, not every column has the same nucleotide

A pitfall of paired de Bruijn graphs

Not every Eulerian path in the paired de Bruijn graph constructed from (k,d) -mer composition spells out a solution of the String Reconstruction from Read-Pairs Problem.

String Spelled by a Gapped Genome Path Problem

String Spelled by a Gapped Genome Path Problem:
Reconstruct a sequence of (k, d)-mers corresponding to a path in a paired de Bruijn graph.

Input: A sequence of (k, d)-mers $(a_1|b_1), \ldots, (a_n|b_n)$ such that $\text{SUFFIX}((a_i|b_i)) = \text{PREFIX}((a_{i+1}|b_{i+1}))$ for $1 \leq i \leq n-1$.
Output: A string *Text* of length $k + d + k + n - 1$ such that the i-th (k, d)-mer of *Text* is equal to $(a_i|b_i)$ for $1 \leq i \leq n$ (if such a string exists).

String Spelled by a Gapped Genome Path Problem

Our approach

1. Split the given (k, d)-mers (a1|b1), . . . , (an|bn) into their initial k-mers, ***FirstPatterns = (a1, . . . , an),*** and their terminal k-mers, ***SecondPatterns = (b1, . . . , bn).***
2. Assemble ***FirstPatterns*** and ***SecondPatterns*** into strings ***PrefixString*** and ***SuffixString***
3. Concatenate the two strings

String Spelled by a Gapped Genome Path Problem : Examples

AG-AG
GC-GC
CA-CT
AG-TG
GC-GC
CT-CT
TG-TG
GC-GC
CT-CA

PrefixString = AGCAGCTGCT

SuffixString = AGCTGCTGCA

These strings perfectly overlap until starting at the fourth nucleotide of PrefixString:

PrefixString = AGCAGCTGCT
SuffixString = AGCTGCTGCA
Genome = AGCAGCTGCTGCA

AG-AG
GC-GC
CT-CT
TG-TG
GC-GC
CA-CT
AG-TG
GC-GC
CT-CA

PrefixString = AGCTGCAGCT

SuffixString = AGCTGCTGCA

There is no perfect overlap

PrefixString = AGCTGCAGCT
SuffixString = AGCTGCTGCA
Genome = AGC?GC?GCTGCA

String Spelled by a Gapped Genome Path Problem

```
StringSpelledByGappedPatterns(GappedPatterns, k, d)
    FirstPatterns ← the sequence of initial k-mers from GappedPatterns
    SecondPatterns ← the sequence of terminal k-mers from GappedPatterns
    PrefixString ← StringSpelledByPatterns(FirstPatterns, k)
    SuffixString ← StringSpelledByPatterns(SecondPatterns, k)
    for i = k + d + 1 to |PrefixString|
        if the i-th symbol in PrefixString ≠ the (i − k − d)-th symbol in SuffixString
            return "there is no string spelled by the gapped patterns"
    return PrefixString concatenated with the last k + d symbols of SuffixString
```

String Spelled by a Gapped Genome Path Problem

AG-AG
GC-GC
CA-CT
AG-TG
GC-GC
CT-CT
TG-TG
GC-GC
CT-CA

Perfect overlap until K+d index

- PrefixString = AGCAGCTGCT
- SuffixString = AGCTGCTGCA

From K_d+1 index to end of PrefixString, check matching in i-k-d th position of SuffixString

- PrefixString = AGCAGCTGCT
- SuffixString = AGCTGCTGCA

PrefixString Concatenated with last K+d Symbols of SuffixString

- PrefixString = AGCAGCTGCT
- SuffixString = AGCTGCTGCA

Reconstructed Genome = AGCAGCTGCTGCA

String Spelled by a Gapped Genome Path Problem

AG-AG
GC-GC
CT-CT
TG-TG
GC-GC
CA-CT
AG-TG
GC-GC
CT-CA

Perfect overlap until K+d index

- PrefixString =AGCTGCAGCT
- SuffixString =AGCTGCTGCA

From K_d+1 index to end of PrefixString, check matching in i-k-d th position of SuffixString

- PrefixString =AGCTGCAGCT
- SuffixString =AGCTGCTGCA

No Possible String Reconstruction

Summary

- String Reconstruction from Read-Pairs
- Paired de Bruijn graphs
  - *Constructing Paired de Bruijn graphs from Genome*
  - *Constructing Paired de Bruijn graphs from paired k-mers*
- Pitfall of Paired de Bruijn graphs
- Eulerian path through Paired de Bruijn graphs
- String Spelled by a Gapped Genome Path Problem