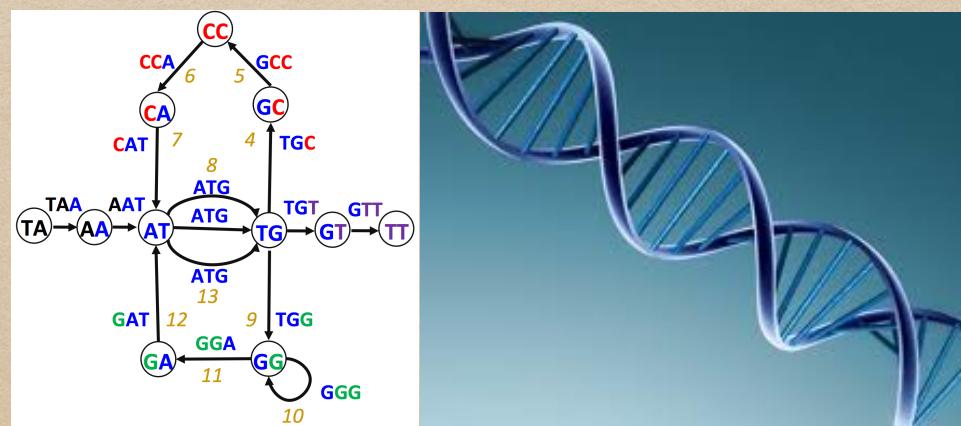


Reconstrução de Genomas

Prof. Dr. Luiz Cláudio Demes da Mata Sousa
Tópicos em Bioinformática/Bioinformática



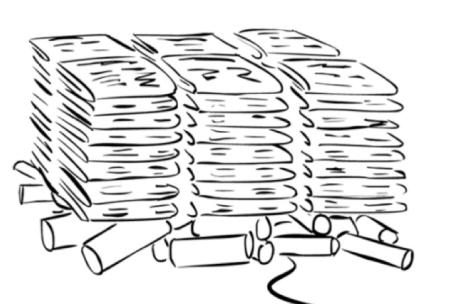
Introdução

- ◆ "Explodindo jornais" - shotgun
- ◆ O problema da reconstrução de strings - parte 1

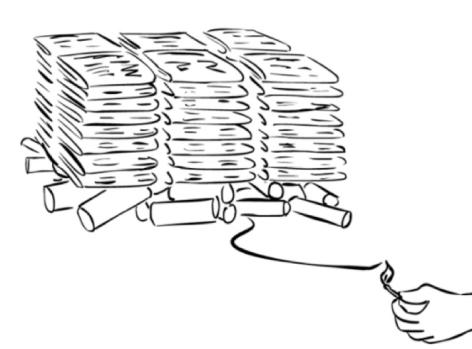
Explodindo Jornais - Shotgun



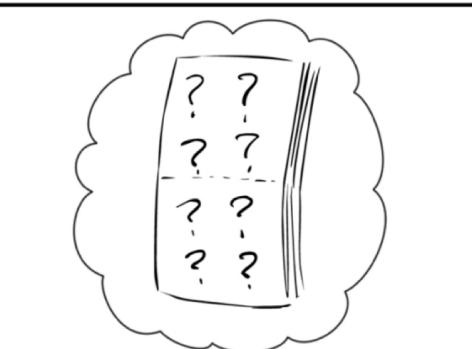
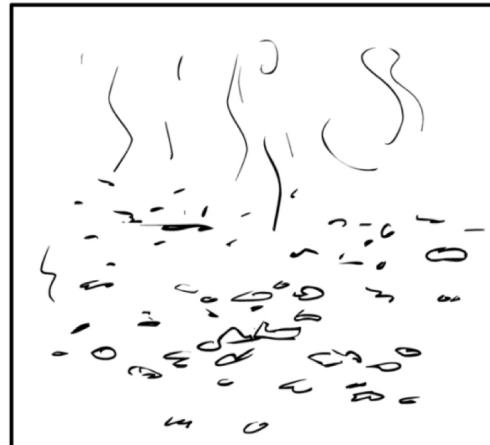
stack of NY Times, June 27, 2000



stack of NY Times, June 27, 2000
on a pile of dynamite



this is just hypothetical



so, what did the June 27, 2000 NY
Times say?

Explodindo Jornais - Shotgun

hoodie, appre-
se have not yet named
information is welc

lie, appre-
yet named any suspects, alt
is welc

O problema de remontar o jornal é um pouco mais difícil do que parece à primeira vista: Existiam várias cópias do jornal e, sendo assim, não podemos remontá-lo da mesma forma que faríamos com um simples quebra-cabeça.

O que devemos fazer é tentar sobrepor fragmentos de diferentes cópias

Explodindo Jornais - shotgun

A pergunta principal é: o que remontar jornais tem a ver com Biología?

Tecnologias de leitura de genoma são limitadas. Em geral, uma única leitura do início ao fim do genoma não é possível. Sendo assim temos que realizar diversas cópias realizadas em fragmentos que precisam ser remontados, assim como os jornais.

Uma única leitura de um pedaço de DNA com 500 nucleotídeos em 1988 custava mais de um dólar, tornando o sequenciamento de mamíferos na casa dos bilhões de dólares.

Explodindo Jornaís - Shotgun

Uma resposta a esse custo excessivo foi a criação do **DNA Array**: Leituras do tamanho de 10 nucleotídeos. Dessa forma, geravam uma composição do genoma bem barata.

— DNA Array —

Gera-se todos os 4^k possíveis DNA k-mers e liga-os ao DNA Array.

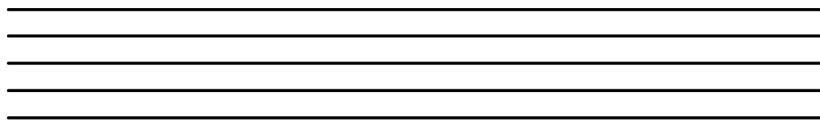
DNA Array é uma matriz onde cada k-mer é associado a uma única localização

AAA	AGA	CAA	CGA	GAA	GGA	TAA	TGA
AAC	AGC	CAC	CCG	GAC	GGC	TAC	TGC
AAG	AGG	CAG	CGG	GAG	GGG	TAG	TGG
AAT	AGT	CAT	CGT	GAT	GGT	TAT	TGT
ACA	ATA	CCA	CTA	GCA	GTA	TCA	TTA
ACC	ATC	CCC	CTC	GCC	GTC	TCC	TTC
ACG	ATG	CCG	CTG	GCG	GTG	TCG	TTG
ACT	ATT	CCT	CTT	GCT	GTT	TCT	TTT

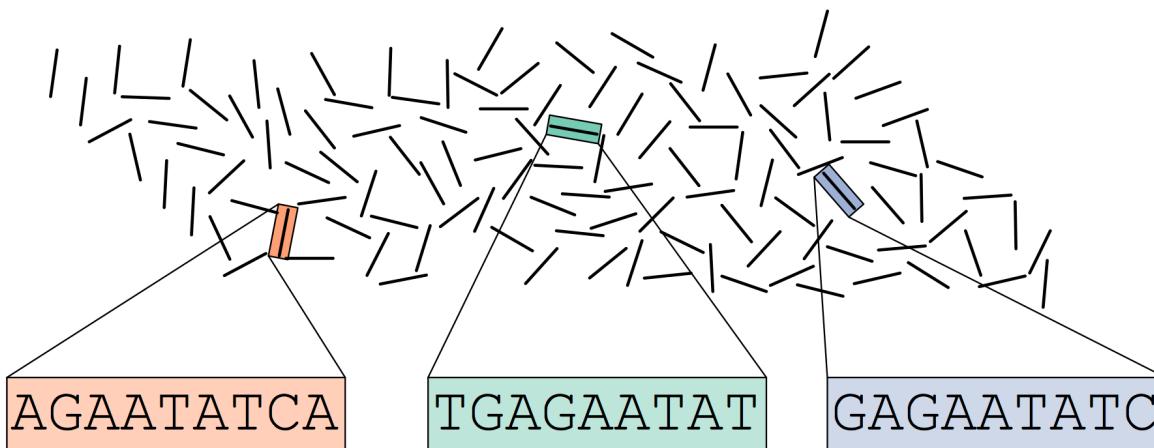
Todos os 64 possíveis 3-mers

Explodindo Jornais - Sequenciamento Tradicional

Multiple copies of a genome



Shatter the genome into reads



Sequence the reads

AGAATATCA

GAGAATATC

TGAGAATAT

... **TGAGAATATCA** ...

Assemble the genome with overlapping reads

Devemos sobrepor as diversas leituras para remontar o genoma.

Mesma idéia aplicada ao tentar remontar os jornais

O problema da reconstrução de strings

— Mundo k-mer —

Primeiro: Considerando que as leituras em uma sequenciadora possuem basicamente o mesmo tamanho, pode-se assumir que as leituras são k-mers para algum valor de k.

Segundo: Os algoritmos a serem estudados levarão em consideração um ambiente ideal e hipotético em que todo o gnoma é lido.

Terceiro: Cada substring k-mer é gerada a partir de uma única leitura.

O problema da reconstrução de strings

Problema computacional: modelando uma montagem de genoma.

Dada um determinada string (texto), sua Composição k -mer ou $\text{Composition}_k(\text{TEXTO})$ é a coleção de todas as substrings k -mer de TEXTO . Por exemplo:

$$\text{Composition}_3(\text{TATGGGGTGC}) = \{\text{TAT}, \text{ATG}, \text{TGG}, \text{GGG}, \text{GGG}, \text{GGT}, \text{GTG}, \text{TGC}\}$$

Os k -mers devem ser listados na ordem lexicográfica (dicionário), aos invés da ordem em que aparecem na seqüência. Sendo assim:

$$\text{Composition}_3(\text{TATGGGGTGC}) = \{\text{ATG}, \text{GGG}, \text{GGG}, \text{GGT}, \text{GTG}, \text{TAT}, \text{TGC}, \text{TGG}\}$$

String Composition Problem: *Generate the k -mer composition of a string*

Input: *An integer k and a string text*

Output: $\text{Composition}_k(\text{TEXT})$, com k -mers em ordem lexicográfica

O problema da reconstrução de strings

Para remontar um genoma, precisa-se resolver o problema inverso.

String Reconstruction Problem: Reconstruct a string from 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

Antes de trabalhar o problema da reconstrução, consideremos o exemplo a seguir: **AAT ATG GTT TAA TGT**

A maneira natural de resolver o problema da reconstrução é imitar a solução dada para remontar o jornal : Conecte um par de k -mer se eles se sobrescrevem em $k-1$ símbolos (letras). Para o exemplo acima, é fácil ver que devemos começar pelo k -mer TAA.

O problema da reconstrução de strings

AAT ATG GTT TAA TGT

TAA
AAT
ATG
TGT
GTT

TAATGTT

(SIMPLES? - Vamos com calma!)

Exemplo 2:

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

TAA
AAT
ATG
TGT
GTT
TAATGTT

Problema: Não há um k-mer que inicie com as letras TT

A dificuldade em montar essa seqüência reside no fato de haver três k-mers ATG. Dessa forma teremos três caminhos a seguir: TGG, TGC e TGT.

Resumo - Aula 1

- ◆ Reconstrução de strings - parte 2
 - ◆ Genoma Path
 - ◆ Grafo Sobrepostos (Overlap graph)
 - ◆ Caminhos Hamiltonianos

O problema da reconstrução de strings - parte 2

Repetições no genoma tornam a sua remontagem mais complexa.

O problema de remontar exige que se “veja à frente” para que se consiga montar o genoma corretamente - existem vários caminhos **inviáveis** que devem se previstos.



Genoma path - 3-mers da sequência TAATGCCATGGGATGTT na ordem genômica

O problema da reconstrução de strings - parte 2



Reconstruir o genoma a partir do **Genoma path** é simples, basta ler da esquerda para a direita.

Formalismo necessário:

Prefixo = primeiros $k-1$ nucleotídeos. Exemplo: Prefixo(TAA) = TA

Sufixo = últimos $k-1$ nucleotídeos. Exemplo: Sufixo(TAA) = AA

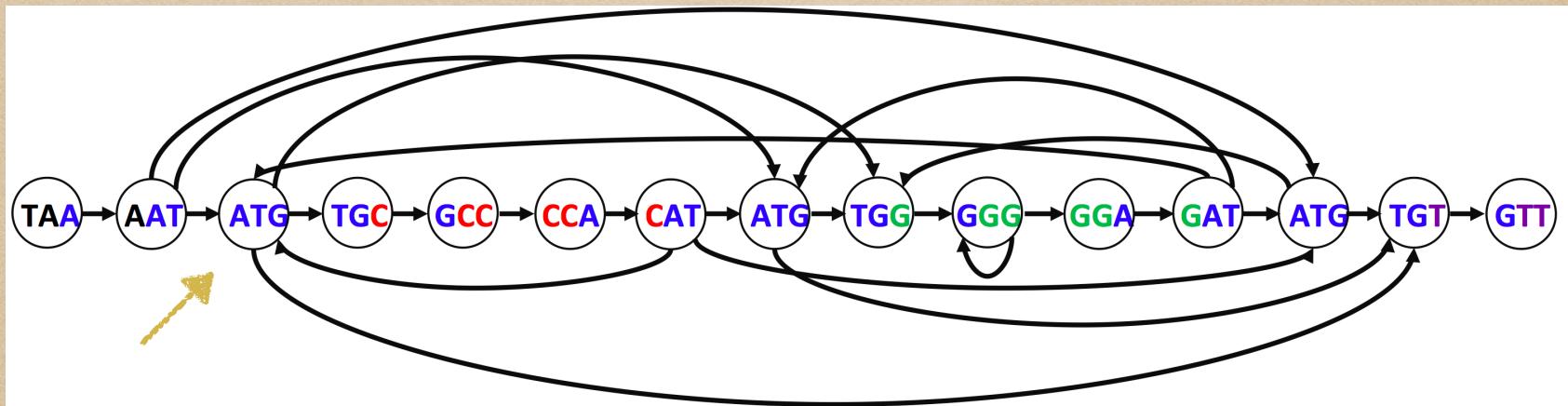
Observe que no **Genoma path** que o sufixo do anterior = prefixo do próximo.

Sendo assim, pode-se afirmar que:

Regra 1 - **Sufixo (Padrão K-mer₁) = Prefixo (Padrão K-mer₂)**

Pergunta: É possível reconstruir o genoma conhecendo-se apenas a composição k-mer e a Regra 1?

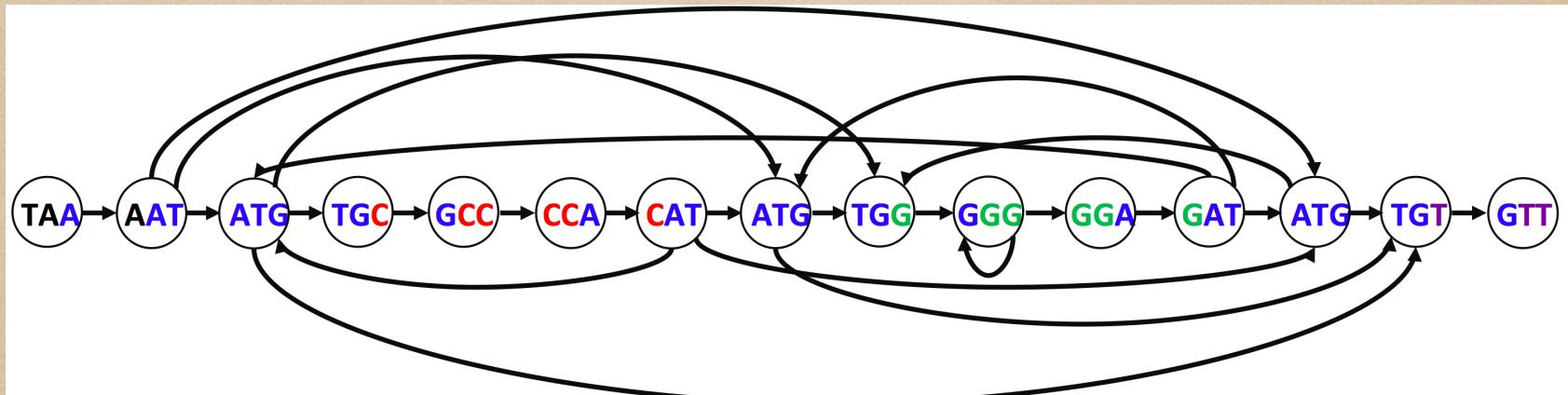
O problema da reconstrução de strings - parte 2



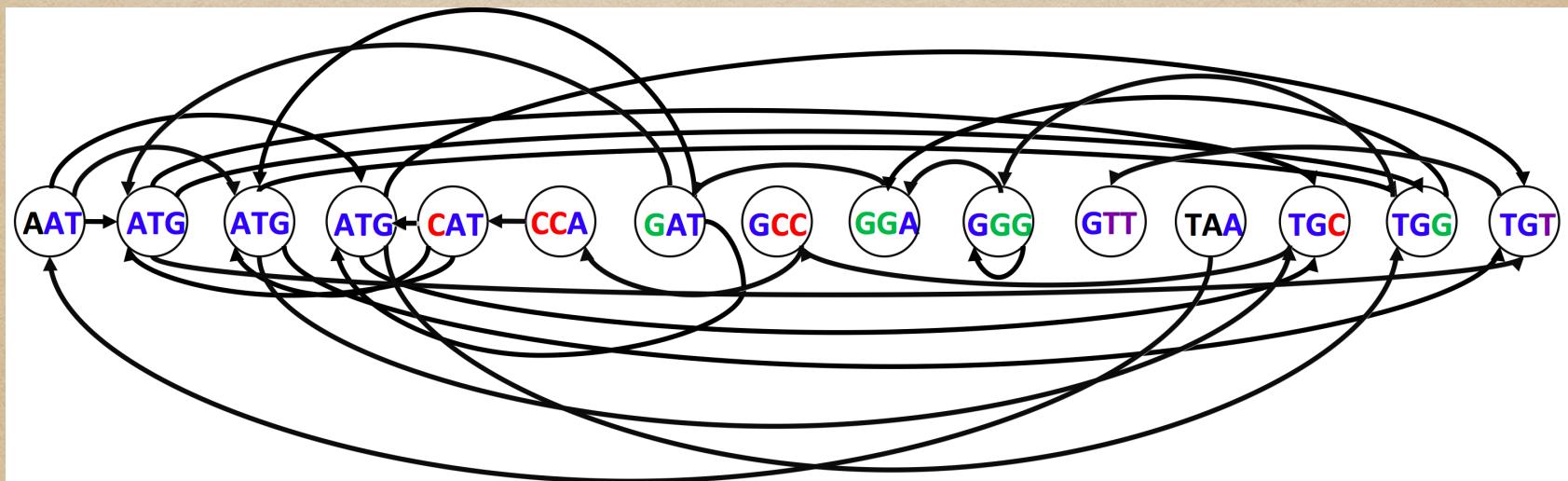
1. Se aplicarmos a **regra 1**, o grafo acima é criado.
2. Muitos outros pares são conectados
3. Novas conexões: ATG conecta-se a TGC, TGG e TGT
4. O novo grafo tem 25 arestas e os 15 nós originais.

A figura acima representa um grafo ou rede de nós conectados por arestas, arcos ou setas

O problema da reconstrução de strings - parte 2

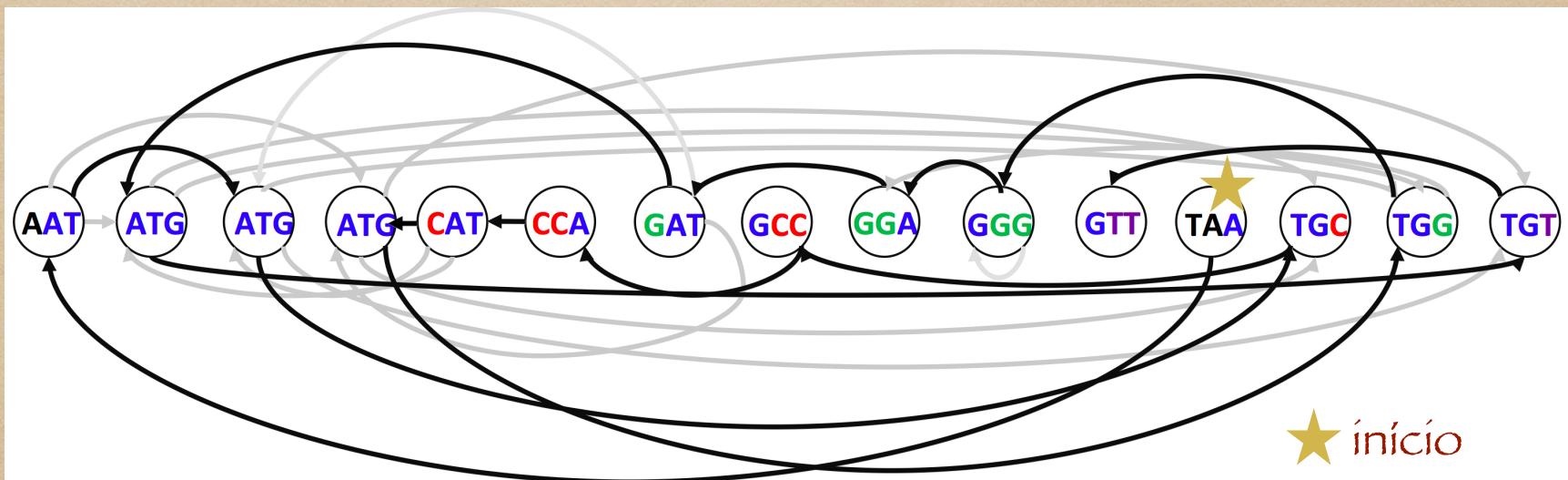


O genoma original pode ainda ser reconstruído a partir do **Genoma path**.



Se construirmos o grafo a partir da **Composition** (TA₁ATG₂GCC₃CAT₄TGG₅GAT₆GTT₇)

O problema da reconstrução de strings - parte 2



A figura acima mostra o caminho que se deve seguir para reconstruir o genoma original TAATGCCATGGGATGTT.

Pergunta: "Outros" genomas podem ser reconstruídos a partir do grafo acima?

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)

O problema da reconstrução de strings - parte 2

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)

Exemplo

Input: A collection Patterns of k -mers.

Output: The overlap graph Overlap(Patterns), in the form of an adjacency list.

Exemplo de $k=5$ **Input:**

AGGCA

ATGCG

CATGC

GCATG

GGCAT

Exemplo de $k=5$ **Output** (grafo representado por lista de adjacência):

AGGCA -> GGCAT

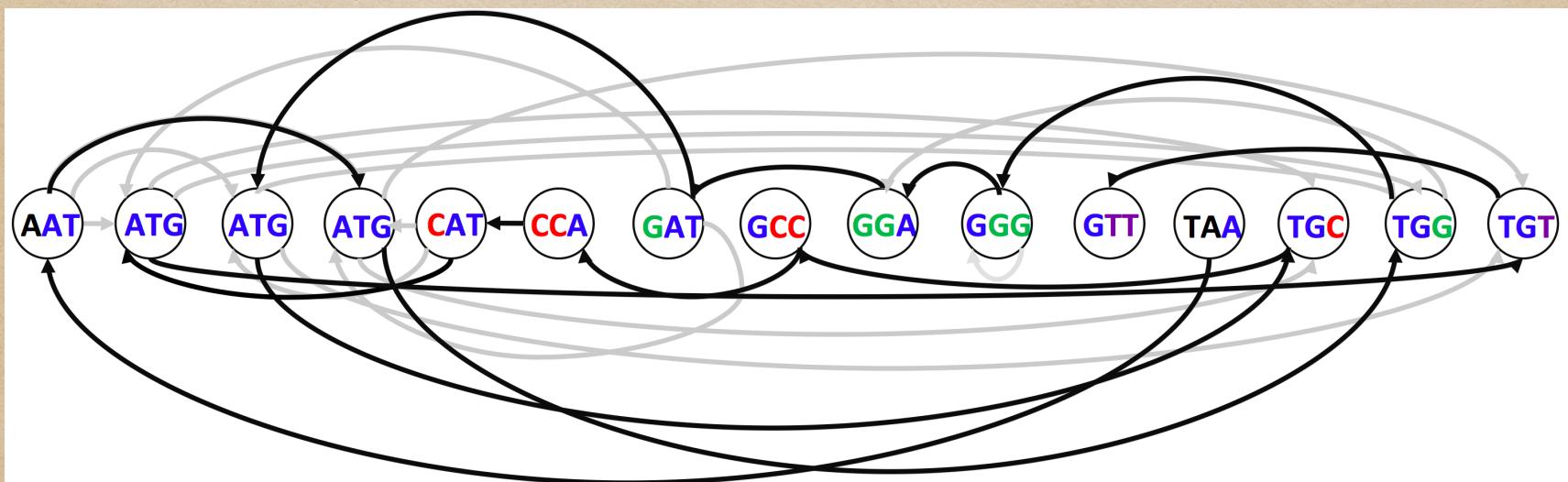
CATGC -> ATGCG

GCATG -> CATGC

GGCAT -> GCATG

O problema da reconstrução de strings

Para resolver o problema da remontagem, procuramos um caminho no grafo sobreposto (overlap graph) que visita cada nó exatamente uma vez (caminho hamiltoniano)



Dois caminhos hamiltonianos encontrados:

TA₁ATG₂CC₃ATGGG₄ATG₅T

TA₁ATG₂GG₃ATG₄CC₅ATG₆T

O problema da reconstrução de strings - parte 2

Hamiltonian Path Problem: *Construct a Hamiltonian path in a graph.*

Input: A directed graph.

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

O problema com o algoritmo acima é que ele é NP Completo ou NPC. Sendo assim, não existe um algoritmo eficiente para resolvê-lo.



Ao invés de procurar um caminho hamiltoniano em um grafo, um novo grafo foi desenvolvido por De Bruijn. Tal grafo pode representar melhor uma composição k-mer.

Resumo - Aula 2

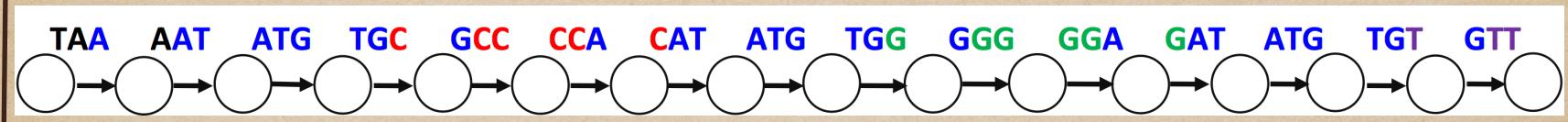
- ◆ Grafos de De Bruijn
- ◆ Teorema de Euler
- ◆ Do Teorema de Euler para um Algoritmo

Grafos de De Bruijn

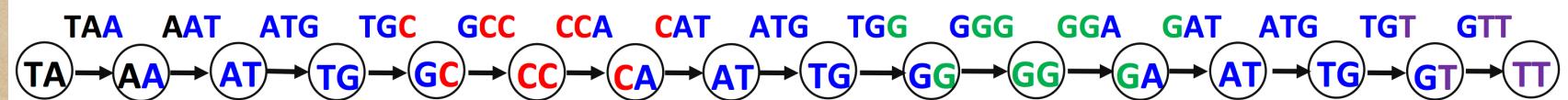
Seja a representação do genoma TAATGCCATGGGATGTT como uma seqüência de suas 3-mers:

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

Ao invés de colocar os 3-mers nos nós, coloquemos nas arestas:



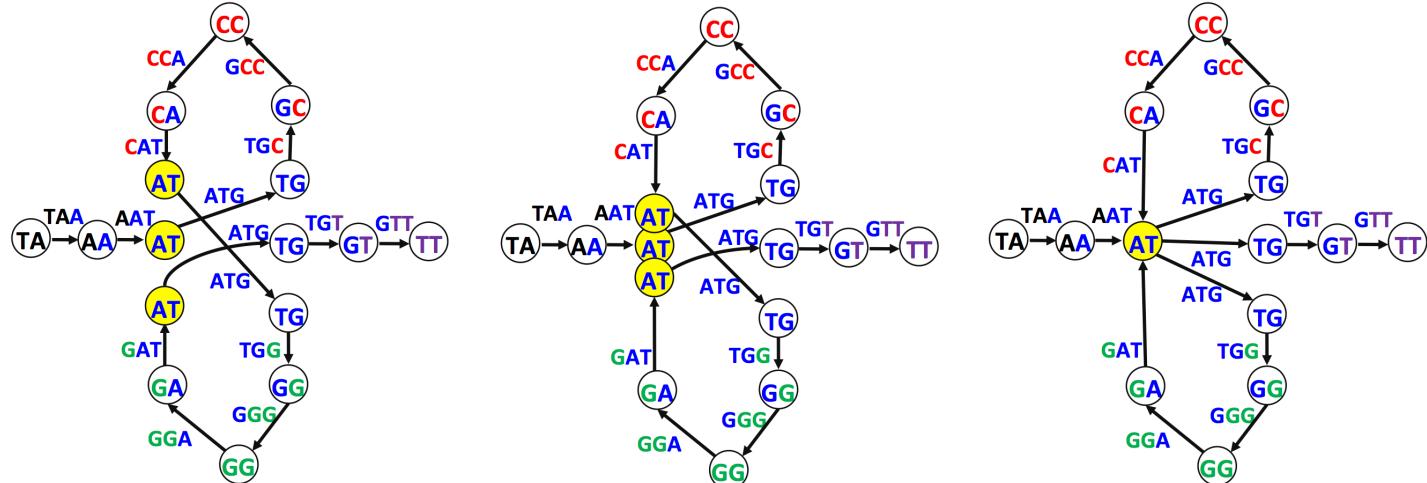
Acrescentemos os prefixos e sufixos nos vértices, i.e., 16 nós como 2-mers:



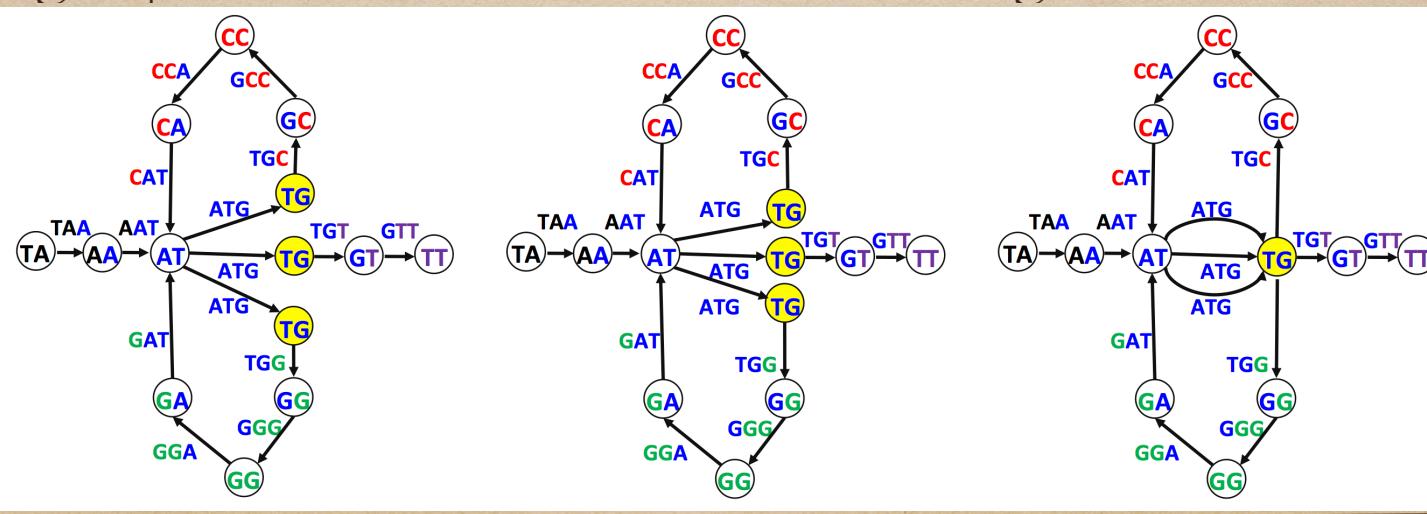
O novo grafo gerado pela função PathGraph_k(TAATGCCATGGGATGTT)

Grafos de De Bruijn

Aparentemente nada de novo. Agora juntemos os **nós idênticos AT**.



Agora juntemos os **nós idênticos TG**. Temos um grafo com 12 nós.



Grafos de De Bruijn

De Bruijn Graph from a String Problem: Construct the de Bruijn graph of a string.

Input: $\text{PathGraph}_k(\text{Text})$. Where k is an integer and Text is a string.

Output: $\text{DeBruijn}_k(\text{PathGraph}_k(\text{Text}))$.



O $\text{DeBruijn}_k(\text{Text})$ é o $\text{PathGraph}_k(\text{Text})$ onde os nós idênticos são “colados” ou juntados.

Pergunta: Se tivermos o grafo $\text{DeBruijn}_k(\text{Text})$ sem fornecer o Text , seria possível reconstruir Text ?

Grafos de De Bruijn - representação como matriz de adjacência

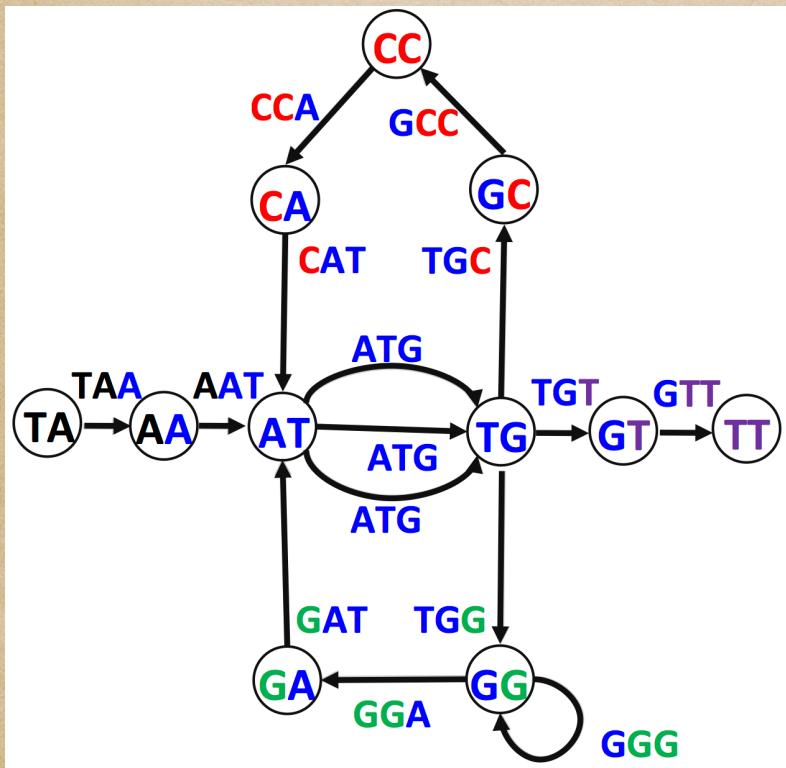
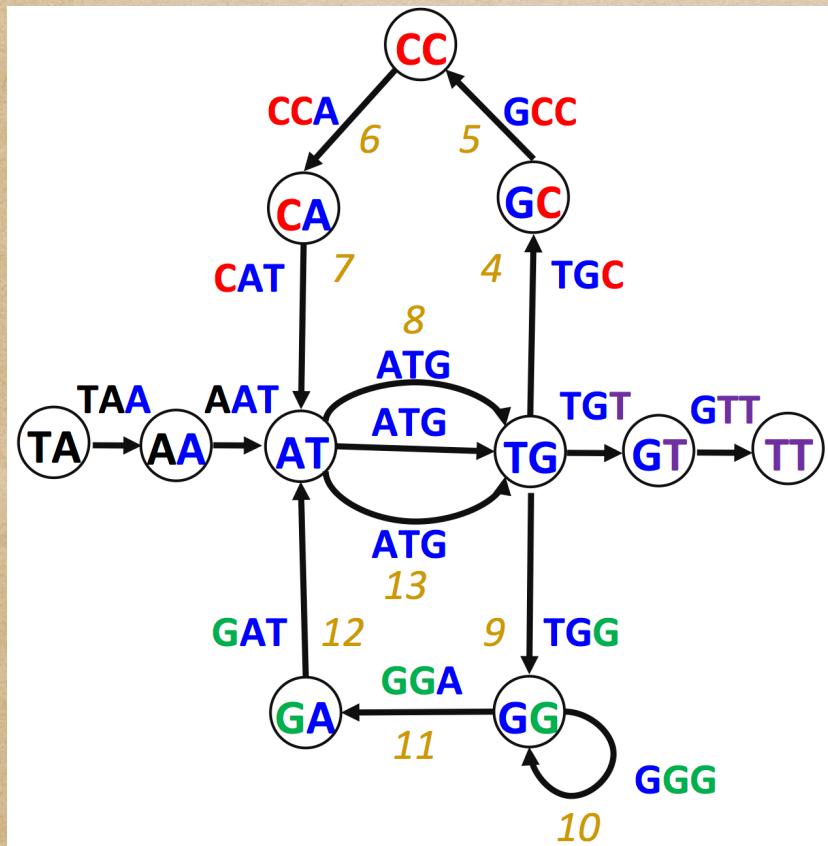


Figura 1 - DeBruijn₃(TAATGCCATGGGATGTT)

Grafos de De Bruijn - Caminhada

Resolver o problema de reconstruir a string utilizando um Grafo de DeBruijn consiste em caminhar o grafo de forma a visitar cada aresta ou arco apenas uma vez: **Caminho Euleriano**.



TAATGCCATGGGATGTT

Figura 2 - DeBruijn₃(TAATGCCATGGGATGTT)

Grafos de De Bruijn

Eulerian Path Problem: Construct an Eulerian path in graph.

Input: A directed graph.

Output: A path visiting every edge in the graph exactly once (if it exists).

Pergunta: Pode-se construir o grafo DeBruijn_k(Text) sem saber qual Text(o) que o gerou, mas sabendo-se a composição k-mer?

A Figura 3 abaixo representa o grafo da composição 3-mer da string TAATG**CC**ATG**GG**ATGT.

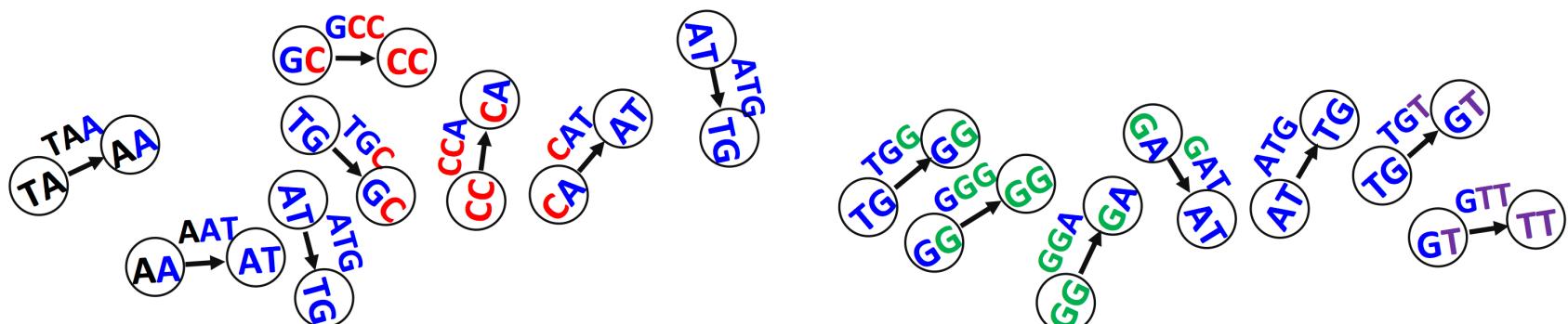
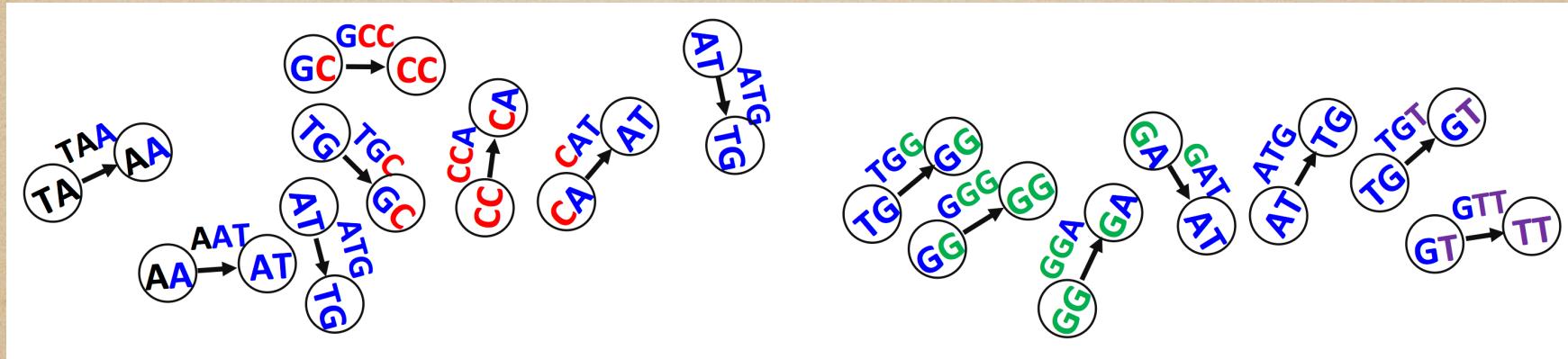


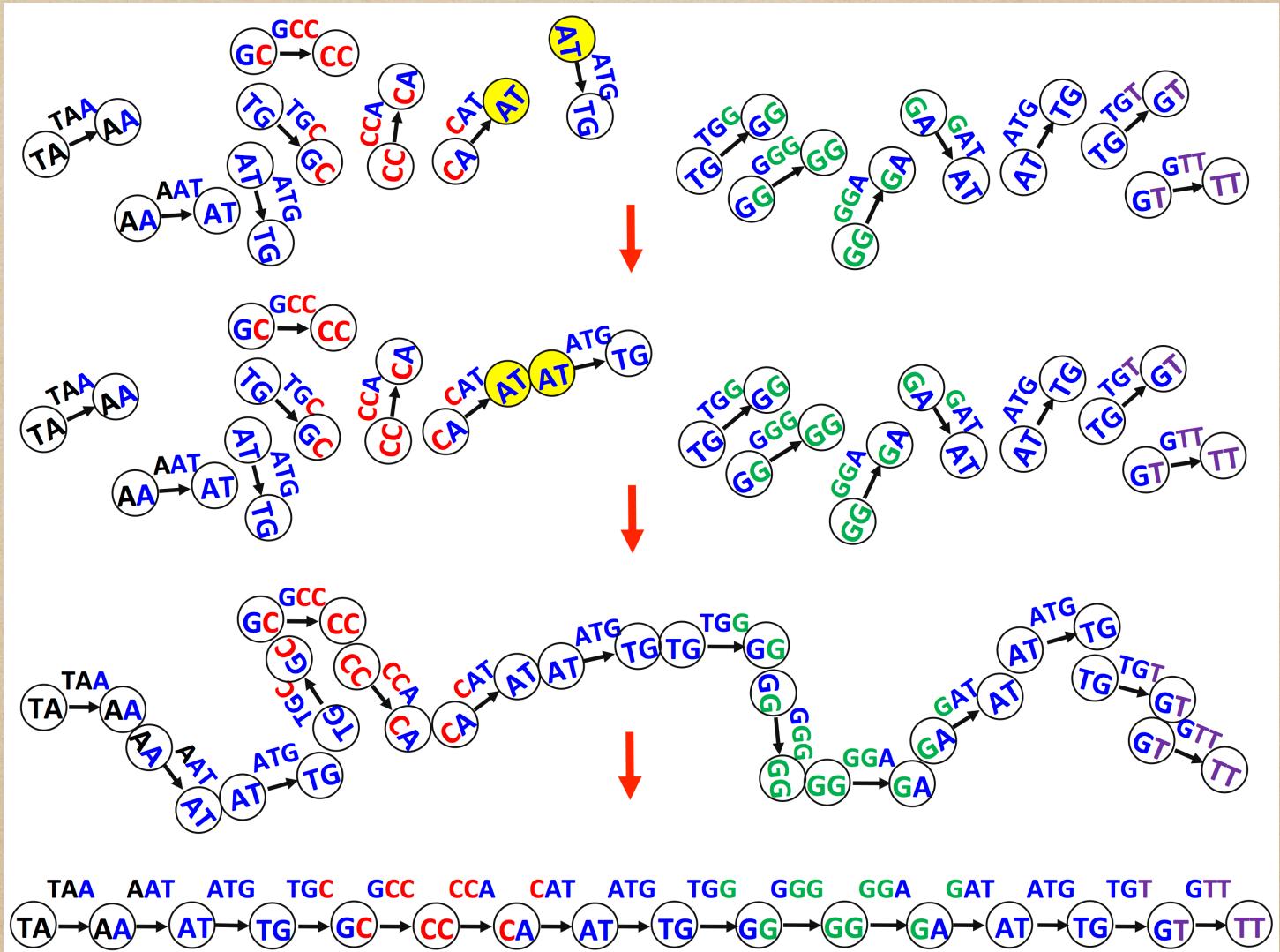
Figura 3 - Resultado de *CompositionGraph*₃(TAATGCCATGGGATGTT)

Grafos de De Bruijn



Cole os vértices idênticos de $\text{CompositionGraph}_3(\text{TAATGCCCCATGGGGATGTT})$. Como o grafo resultante pode ser comparado ao grafo $\text{DeBruijn}_3(\text{TAATGCCCCATGGGGATGTT})$?

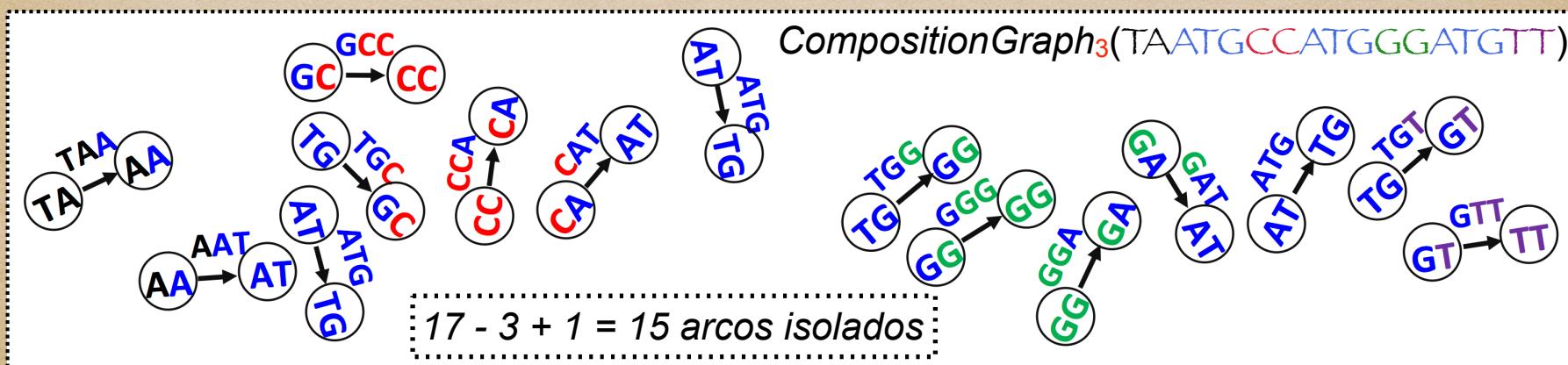
Grafos de De Bruijn



Grafos de De Bruijn

Formalmente:

Para uma string Text qualquer, $\text{CompositionGraph}_k(\text{Text})$ é o grafo que consiste de $|\text{Text}| - k + 1$ arcos isolados. Cada arco é nomeado por um k-mer de Text. Cada arco nomeado por um k-mer conecta dois nós. O primeiro nó é nomeado pelo prefixo e, o segundo nó, pelo sufixo do k-mer.



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).

Teorema de Euler

A Figura 4 abaixo apresenta dois grados: um desbalanceado (a direita) e um balanceado (a esquerda). Um grafo é balanceado quando todos os vértices possuem grau de entrada igual ao grau de saída.

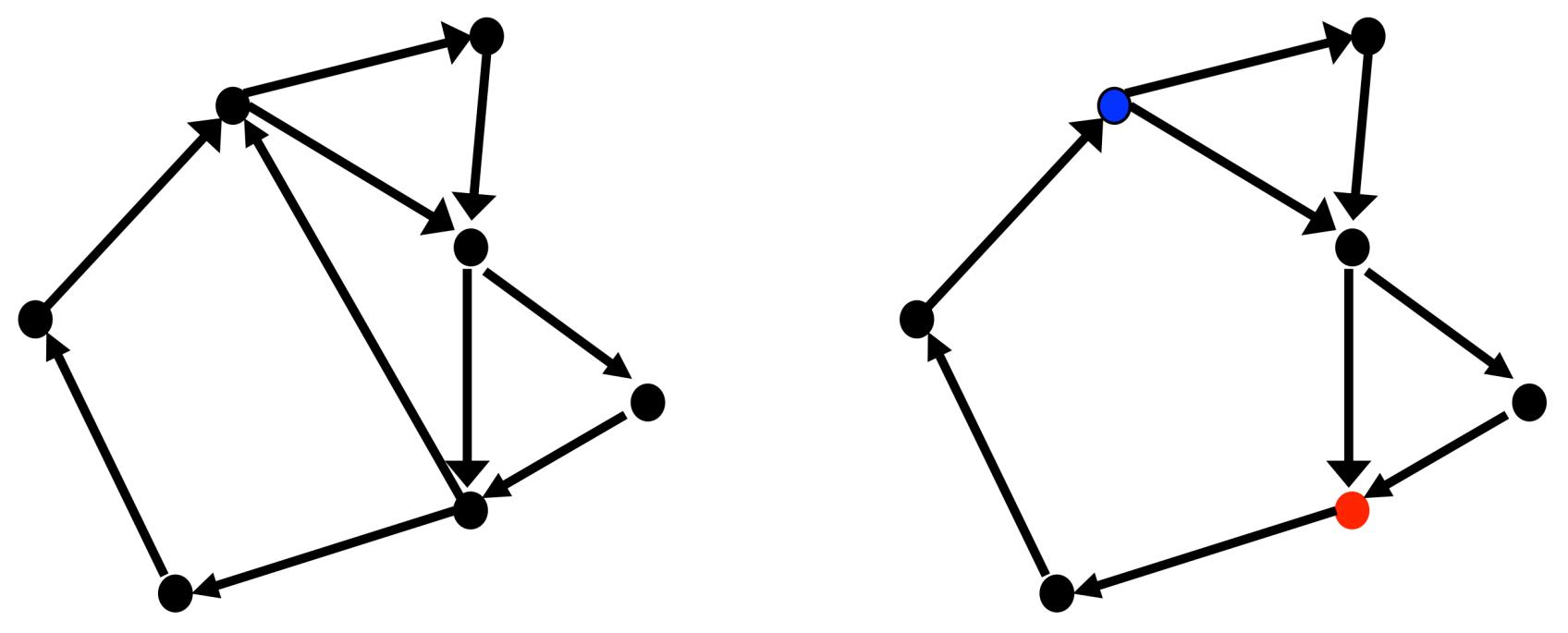


Figura 4 - Grafo balanceado (a esquerda) e desbalanceado (a direita). O Teorema de Euler estabelece que um grafo é euleriano se é balanceado ou se o vértice de partida tem grau de saída ímpar e o vértice de chegada tem grau de entrada ímpar.

Teorema de Euler

A Figura 5 abaixo mostra que um grafo desbalanceado e desconexo não atende ao Teorema de Euler. Sendo assim, nem todo grafo balanceado é também Euleriano.

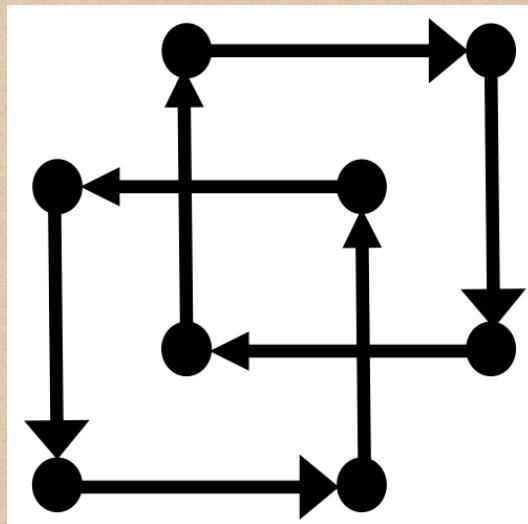
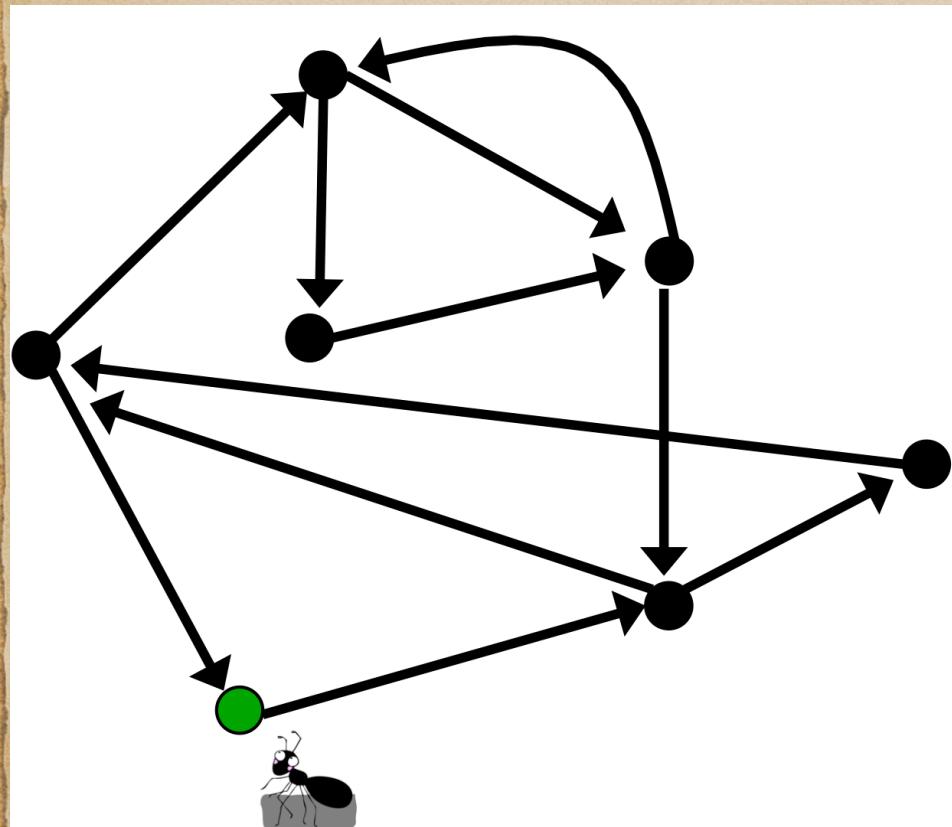


Figura 5 - Grafo balanceado e desconexo.

Teorema de Euler: "Um grafo G conexo possui caminho euleriano se e somente se ele é balanceado ou tem exatamente dois vértices de grau ímpar"

Encontrando um Caminho Euleriano

A Saga de Léo - a formiga andarilha



A formiga Léo, começa sua caminhada no vértice verde.

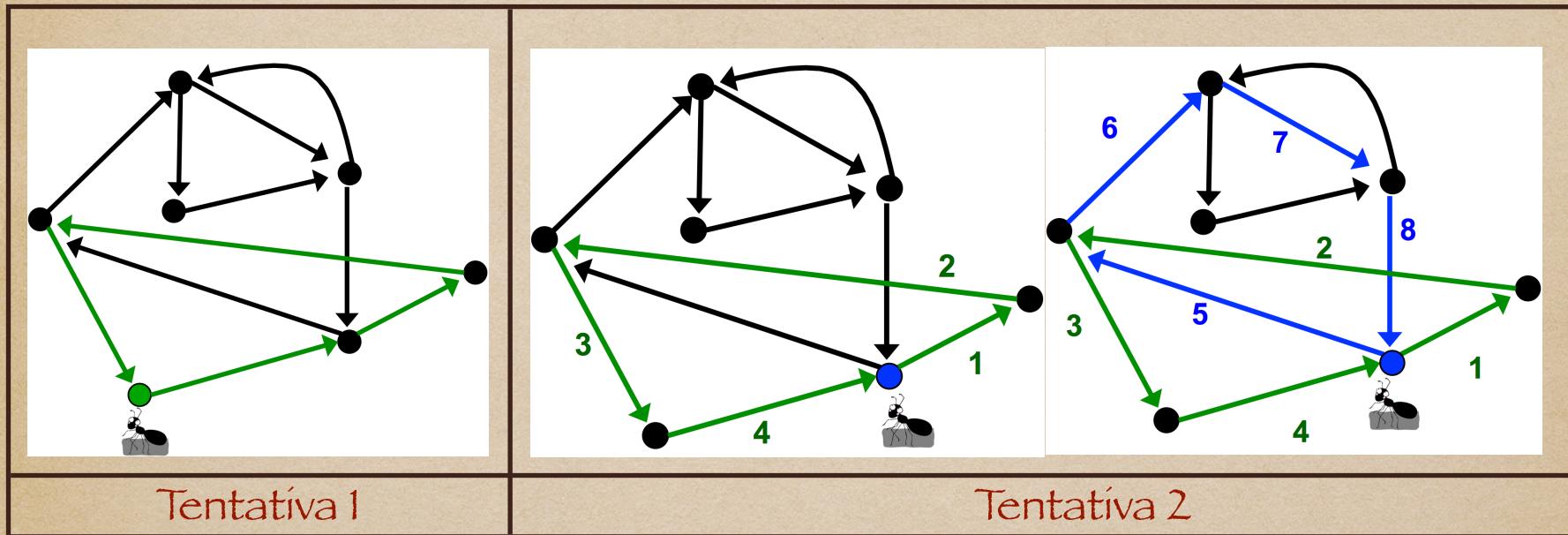
Se Léo fosse um gênio ou extremamente sortudo, ele atravessaria cada arco ou aresta apenas uma vez.

No entanto, experiências com formiga tem mostrado que geralmente elas ficam presas antes de encontrarem o caminho euleriano.

Pergunta: Em que nós Léo poderia ficar preso? Isto é, não teria uma aresta que não tenha andado para poder sair?

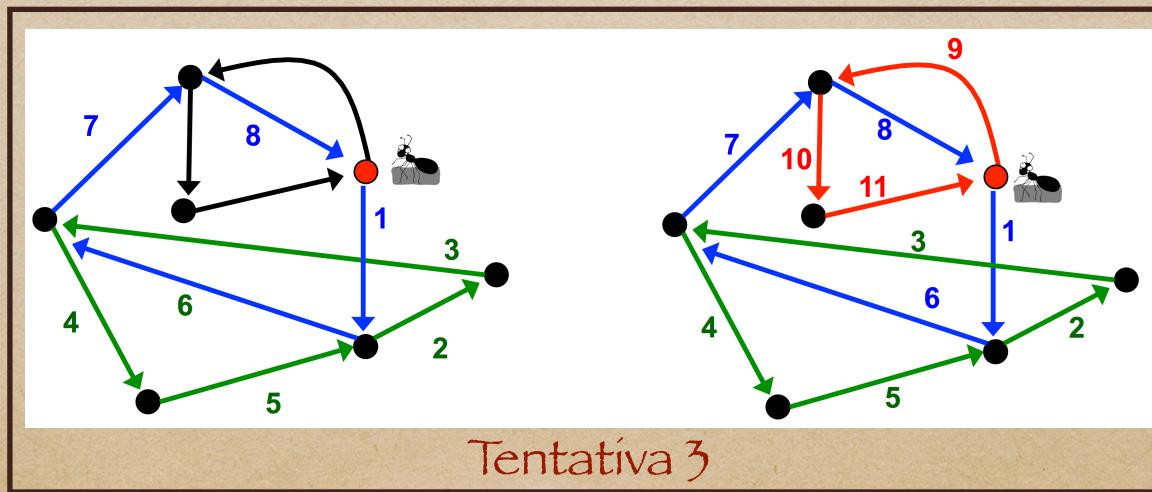
Encontrando um Caminho Euleriano

A Saga de Léo - a formiga andarilha



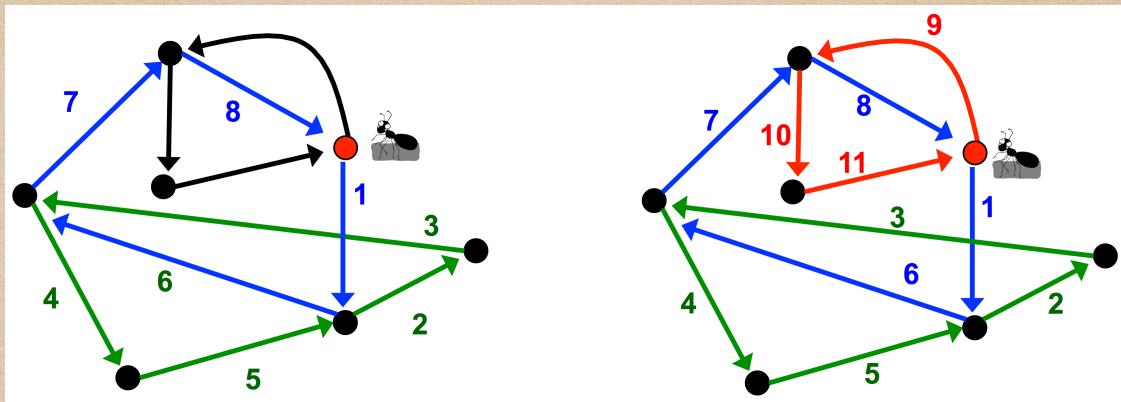
Tentativa 1

Tentativa 2



Tentativa 3

Do Teorema para um Algoritmo



EulerianCycle(**Grafo**)

gere um caminho **Ciclo** visitando arestas randomicamente no **Grafo**

Enquanto existir arestas não visitadas no **Grafo** **faça**

1. selecione um **novo nó inicial** em **Ciclo** que ainda tenha arestas não visitadas
2. gere um novo caminho **Ciclo1** atravessando **Ciclo** (iniciando em **novo nó inicial**) e então gere randomicamente uma nova travessia

Ciclo <- **Ciclo1**

retorne Ciclo

Tentativa 3

Círculo Euleriano a partir de um Caminho Euleriano

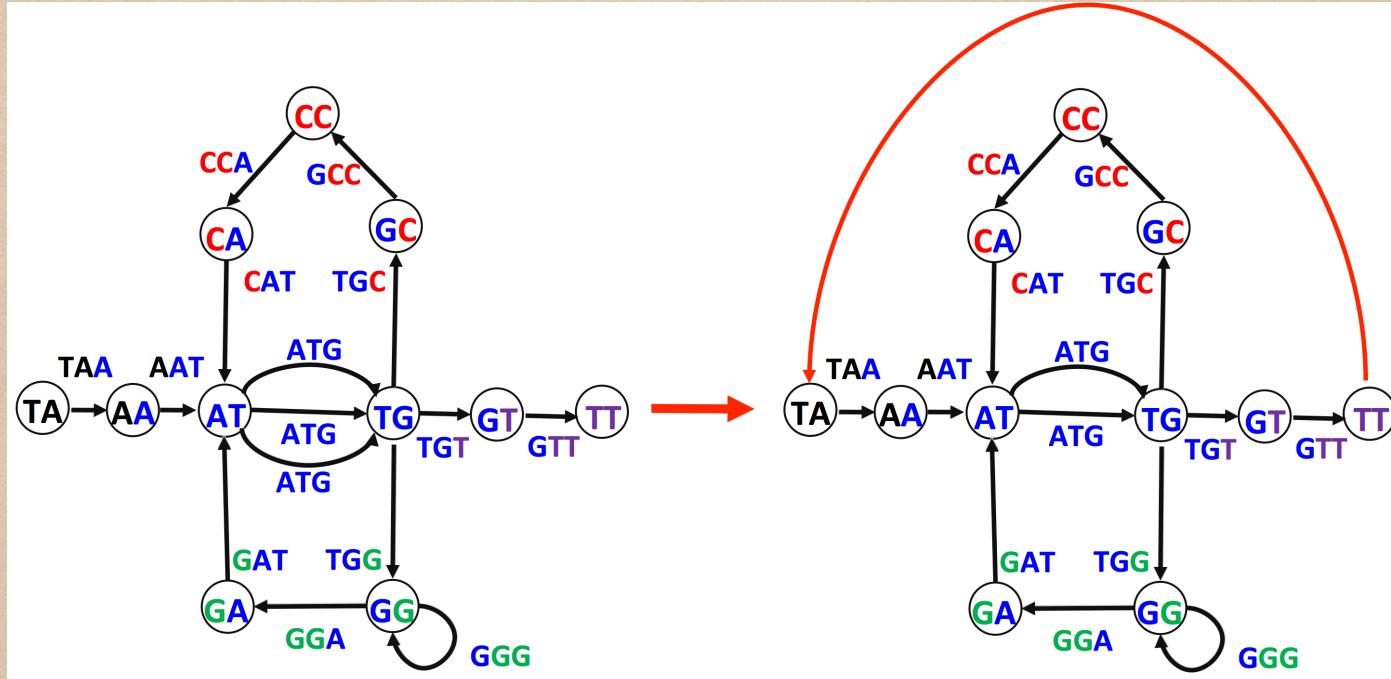
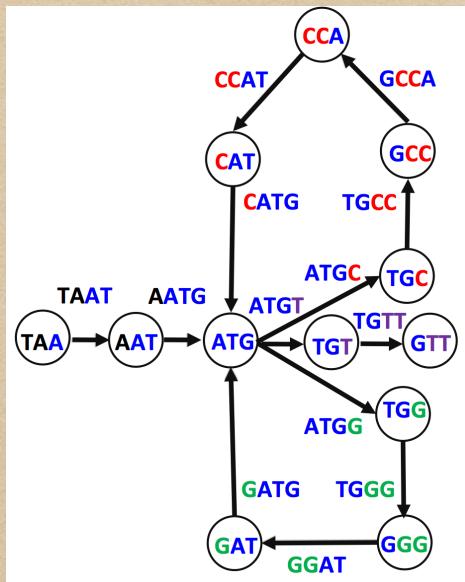


Figura 5 - Transformando um Caminho Euleriano em um Círculo Euleriano: Basta adicionar uma aresta entre o último nós e o primeiro.

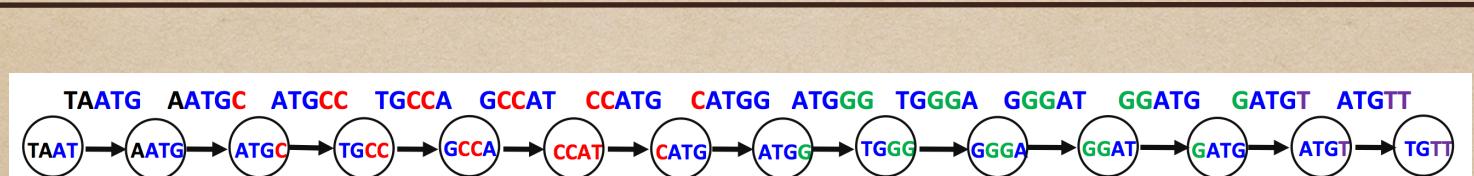
Resumo - Aula 3

- ◆ Grafos de De Bruijn pareado

Quanto maior o k, menos “emaranhado”



DeBruijn₄(TAATGCCATGGGATGTT)
K = 4



DeBruijn₅(TAATGCCATGGGATGTT)
k= 5

Quanto maior o K (tamanho da leitura), menos repetição

Maior leitura precisa com tecnologia atual = 300 nucleotídeos

k=300 é insuficiente para evitar repetição no menor genoma existente.

Solução Virtual dos Biólogos

- Criar uma leitura k -mer + d + k -mer, onde tem-s um k -mer, uma região desconhecida de tamanho d e outro k -mer.
- O primeiro k -mer é conhecido, a região d é desconhecida e o segundo k -mer é conhecido.
- Aumenta-se virtualmente o tamanho da leitura, diminuindo a probabilidade de repetições. Diminuindo a probabilidade do grafo de De Bruijn gerar “emaranhados”.

Solução Virtual

Dada uma cadeia de caractere Text, um (k, d) -mer é um par de k -mers no Text separados pela distância d . A notação $(\text{Padrão1} \mid \text{Padrão2})$ representa um (k, d) -mer.

Por exemplo, $(\text{ATG} \mid \text{GGG})$ é um $(3, 4)$ -mer pareado de TAATGCCATGGGATGTT .

A composição (k, d) -mer de Text é denotada por $\text{ComposicaoPareada}_{k,d}(\text{Text})$. Tal composição é a coleção de todos os (k, d) -mers de Text (incluindo os (k, d) -mers repetidos).

Exemplo: $\text{ComposicaoPareada}_{3,1}(\text{TAATGCCATGGGATGTT})$:

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

Na ordem do dicionário:

$(\text{AAT} \mid \text{CCA}), (\text{ATG} \mid \text{CAT}), (\text{ATG} \mid \text{GAT}), (\text{CAT} \mid \text{GGA}), (\text{CCA} \mid \text{GGG}), (\text{GCC} \mid \text{TGG}),$
 $(\text{GGA} \mid \text{GTT}), (\text{GGG} \mid \text{TGT}), (\text{TAA} \mid \text{GCC}), (\text{TGC} \mid \text{ATG}), (\text{TGG} \mid \text{ATG})$

Solução Virtual

Exercício: Gere a composição (3,1)-mer de TA^{ATGCCATGGGATGTT} na ordem do dicionário. Include repetições e retorne a composição como uma linha.

Observe que a composição de TA^{ATGCCATGGGATGTT} é diferente da Composição Pareada_{3,1}, TA^{ATGCCATGGGATGTT}.

Embora TA^{ATGCCATGGGATGTT} e TA^{ATGGGATGCCATGTT} tenham a mesma Composição 3-mer, Eles tem diferentes composições (3,1)-mer. Assim, podemos diferenciar ambas cadeias, resolvendo o problema do múltiplo caminho Euleriano gerado pelo grafo de De Bruijn

Pode-se reconstruir uma cadeia a partir da Composição (k,d)-mer? Pode-se adaptar o grafo de De Bruijn para esse propósito?

String Reconstruction from Read-Pairs Problem: Reconstruct a string from its (k,d)-mer composition.

Input: A collection of paired k-mers *PairedReads* and an integer *d*.

Output: A string *Text* with (k,d)-mer composition equal to *PairedReads* (if such a string exists).

Prefixo, Sufixo e PathGraph do (k,d) -mer

Dado um (k,d) -mer $(a_1 \dots a_k | b_1 \dots b_k)$, definimos seus prefixo como $(k-1, d)$ -mer $(a_1 \dots a_{k-1} | b_1 \dots b_{k-1})$ e o sufixo como $(k-1, d)$ -mer $(a_2 \dots a_k | b_2 \dots b_k)$. Exemplo:

$$\text{Prefixo}((\mathbf{GAC} | \mathbf{TCA})) = (\mathbf{GA} | \mathbf{TC}) \text{ and } \text{Sufixo}((\mathbf{GAC} | \mathbf{TCA})) = (\mathbf{AC} | \mathbf{CA})$$

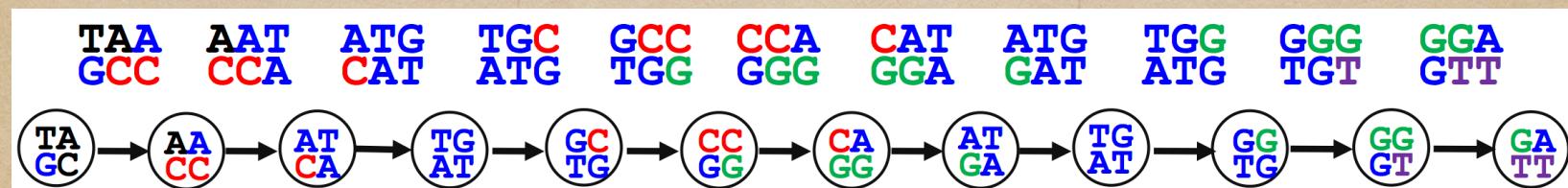
Para (k,d) -mers consecutivos, o sufixo do primeiro é igual ao prefixo do segundo. Exemplo:

Para os (k, d) -mers consecutivos $(\mathbf{TAA} | \mathbf{GCC})$ e $(\mathbf{AAT} | \mathbf{CCA})$ do texto $\mathbf{TAATGCCATGGGATGTT}$, $\text{Sufixo}((\mathbf{TAA} | \mathbf{GCC})) = \text{Prefixo}((\mathbf{AAT} | \mathbf{CCA})) = (\mathbf{AA} | \mathbf{CC})$.

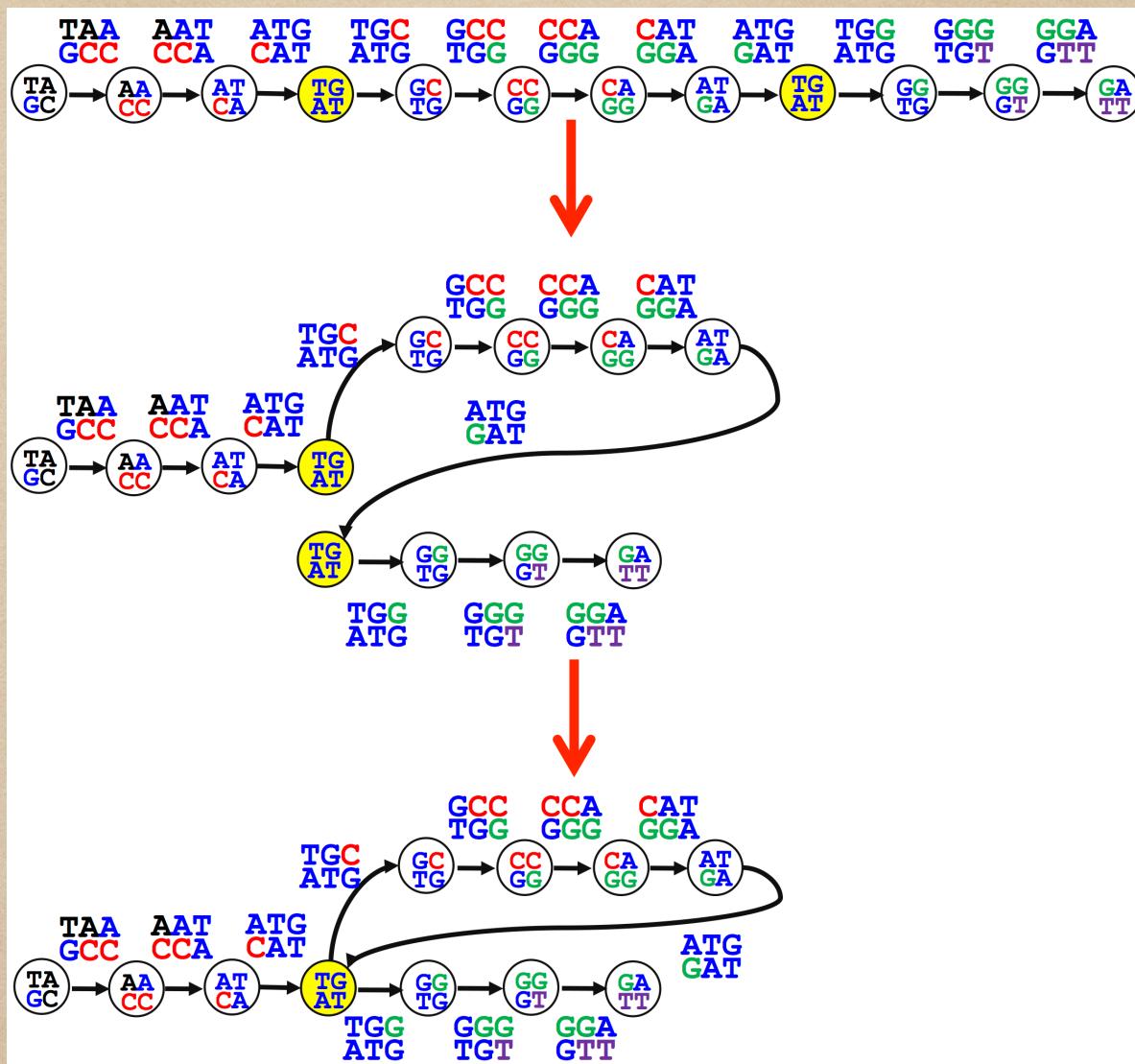
Para (k,d) -mers consecutivos, o sufixo do primeiro é igual ao prefixo do segundo. Exemplo:

Dado uma cadeia Text, constrói-se o grafo PathGraph $_{k,d}(\text{Text})$ que representa um caminho formado por $|\text{Text}| - (k + d + k) + 1$ arestas correspondentes a todos os (k, d) -mers no Text.

As arestas são nomeadas pelos (k, d) -mers and os nós iniciais e finais da aresta pelos prefixo e sufixo, respectivamente. A Figura 1 mostra o PathGraph $_{3,1}(\mathbf{TAATGCCATGGGATGTT})$.



Grafo de De Bruijn a partir do PathGraph



PathGraph_{3,1}(TAATGCCATGGGATGTT) tem 11 arestas e 12 nós. Apenas dois nós são iguais: (TG | AT).

Composição (k,d) -mer e Grafo De Bruijn

Pergunta:

Pode-se construir o grafo de De Bruijn a partir da ComposicaoPareada de um texto? Isto é, da mesma forma que se pode construir a partir do PathGraph?

CODE CHALLENGE: Solve the String Reconstruction from Read-Pairs Problem.

Input: An integer d followed by a collection of paired k -mers *PairedReads*.

Output: A string *Text* with (k, d) -mer composition equal to *PairedReads*.

Input: ComposicaoPareada_{4,2}(Text)

2

GAGA | TTGA
TCGT | GATG
CGTG | ATGT
TGGT | TGAG
GTGA | TGTT
GTGG | GTGA
TGAG | GTTG
GGTC | GAGA
GTCG | AGAT

Output: Text

