Grafos e sequenciação de genomas

Como os grafos e os algoritmos sobre grafos podem ajudar na montagem de leituras (fragmentos) de genomas

Parcialmente adaptado do curso “Bioinformatics Algorithms”, P. Pevzner et al
How Do We Assemble Genomes? (Graph Algorithms)

Sequenciação de genomas

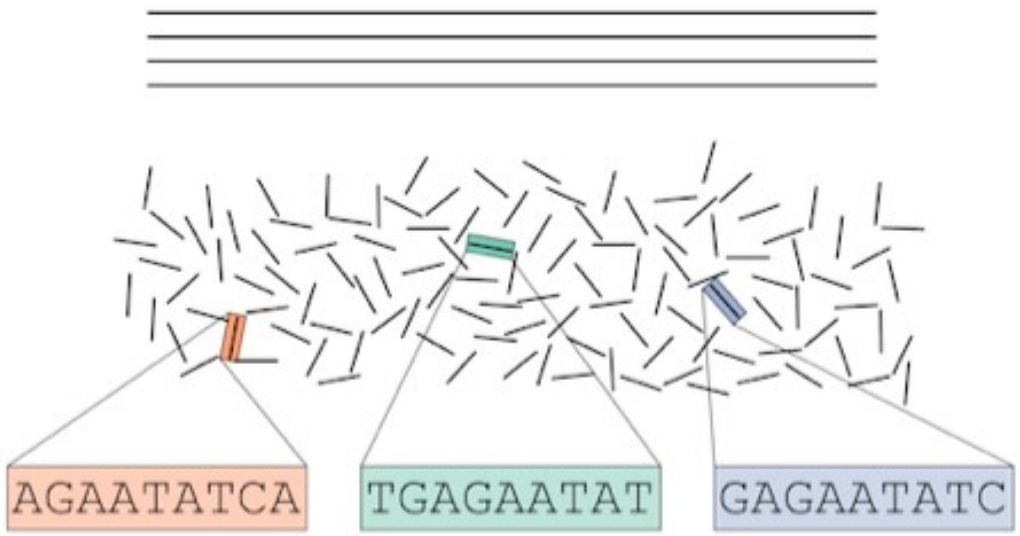
- **Sequenciar um genoma** não é como ler um livro pois não existem tecnologias que permitam sequenciar um genoma do início até ao final
- As técnicas de sequenciação criam conjuntos de leituras (*reads*) de pequenos **fragmentos** do DNA com um máximo de algumas centenas de bases
- É necessário haver algoritmos e ferramentas computacionais para **montar estes fragmentos** reconstituindo o genoma original que tem tipicamente centenas ou milhares de milhões de bases

Sequenciação de genomas

Múltiplas cópias do genoma

Fragmentação do genoma

Sequenciação dos fragmentos (reads)



Montagem (assembly) do genoma a partir dos fragmentos (problema computacional)

Montagem do genoma

- Localização dos segmentos não é conhecida pelo que a montagem tem que ser realizada pela **sobreposição** dos fragmentos (tal como na montagem de um puzzle)
- A **montagem** (*assembly*) do genoma a partir dos fragmentos constitui um problema computacional de **elevada complexidade**, dado a pequena dimensão dos fragmentos quando comparada com a grande dimensão dos genomas completos
- Até aos anos 1990, pensava-se ser impossível montar genomas completos de organismos mais complexos; mesmo hoje, os genomas de maior dimensão estão ainda fora do alcance dos programas mais modernos de montagem de genomas

Montagem do genoma

- Outros fatores tornam o problema real ainda mais difícil:
  - As moléculas de DNA têm duas **cadeias complementares** e não há forma de saber de qual delas é proveniente cada fragmento
  - Os sequenciadores têm uma dada taxa de **erro**; estes tornam o processo de sobreposição mais difícil pois em muitos casos não é perfeito
  - Há regiões do genoma que são difíceis ou impossíveis de sequenciar, não sendo cobertas por nenhum fragmento
- Numa primeira fase dos nossos algoritmos, assumiremos que estes problemas não existem para desenvolver algoritmos para **casos ideais** de cobertura total do genoma e inexistência de erros, assumindo que todas as leituras são feitas numa única cadeia

Composição de uma string em *k-mers*

- Uma definição útil nos algoritmos que se seguem é a da **composição** de uma string (sequência) *s* em segmentos de tamanho *k* (**k-mers**) – $comp_k(s)$
- Esta é definida como a coleção de todas as sub-sequências de *s*, de tamanho *k*, incluindo repetições
- Dado que, na sequenciação, não temos ideia da ordem dos fragmentos, vamos ordenar os segmentos por ordem lexicográfica, i.e. pela ordem "alfabética" que apareceriam num dicionário

Problema da reconstrução de uma string a partir da sua composição

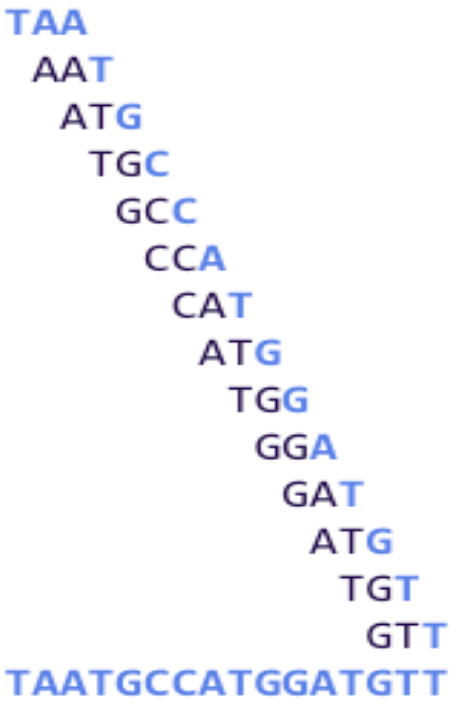
- Em sequenciação, não sabemos a sequência original, mas conhecemos a sua composição em fragmentos
- Assim, um problema mais útil é o problema inverso: **reconstruir a string original dada a sua composição em k-mers**
- Entradas do problema
  - O valor de *k*; uma coleção *p* de k-mers
- Saída do problema
  - Uma string *s* com composição de k-mers igual a *p* (se existir)

Problema da reconstrução de uma string a partir da sua composição

- A solução mais natural para este problema será considerar um dos fragmentos para começar e estendê-lo (numa das duas direções) com fragmentos que tenham sobreposição de k-1 posições
- Vamos testar esta estratégia com o exemplo seguinte:
  AAT ATG GTT TAA TGT
- Teste agora com o exemplo:
  AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG G TT TAA TGC TGG TGT
- Será este um algoritmo viável ?

Problema da reconstrução de uma string da sua composição

Correto: usa todos os fragmentos



Incorreto: não usa todos os fragmentos

As repetições de (k-1)-mers levam a que haja alternativas na escolha do próximo fragmento
Escolhas erradas em certos pontos levam a um resultado final incorreto … **a resolução do problema é mais complexa** … e implica “back-tracking” nas escolhas !

As repetições tornam o problema da montagem do genoma mais complexo

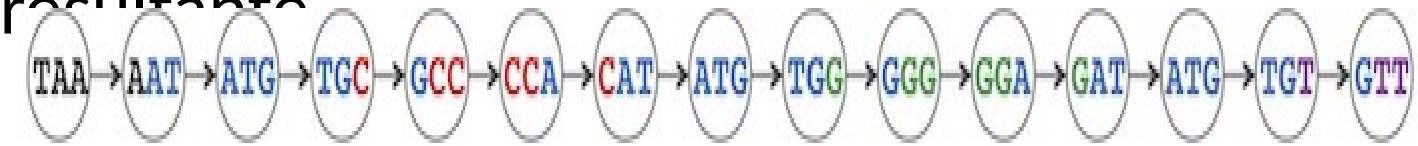
Uma ilustração do problema das repetições na montagem do genoma pode ser encontrada no puzzle “triazzle”, onde padrões repetidos em várias zonas do puzzle tornam complexa a tarefa de definir o local correto de cada peça !

Problema da reconstrução de uma string: definições

- Definições necessárias:
  - **Prefixo$_{k-1}$** de uma sequência de tamanho k: primeiros k-1 elementos da sequência
  - **Sufixo$_{k-1}$** de uma sequência de tamanho k: últimos k-1 elementos da sequência
- Dadas estas definições podemos ligar duas sequências S1 e S2 como consecutivas se o **sufixo$_{k-1}$ de S1 é igual ao prefixo$_{k-1}$ de S2**
  - Exemplo: TAA -> AAT dado que prefixo(AAT) = sufixo (TAA) = AA

Problema da reconstrução de uma string: grafo de sobreposições

- Dado um conjunto de sequências (ou fragmentos) podemos criar o respetivo **grafo de sobreposições** da seguinte forma:
  - **Nós** correspondem às sequências dos diversos segmentos
  - **Arcos** são definidos entre nós A -> B, se sufixo_{k-1} (A) = prefixo_{k-1} (B)
- Assim, reconstrução da string original corresponde a um **caminho no grafo de sobreposições**
- Necessário transformar o caminho na string resultante



Circuitos Hamiltonianos

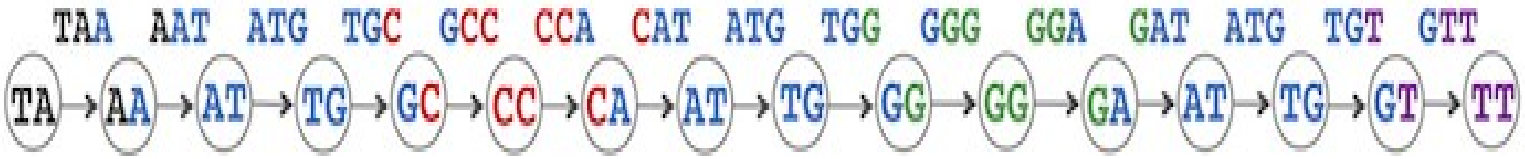
- Num grafo orientado, um **Circuito Hamiltoniano** é um **caminho que passa por todos os nós do grafo exatamente uma vez**
- O **problema da reconstrução de uma string a partir da sua composição em k-mers** pode ser formulado como a **procura de um circuito Hamiltoniano no grafo de sobreposições** (com repetições) da forma definida anteriormente
- A partir dum caminho Hamiltoniano a sequência que gerou os k-mers representados no grafo pode ser determinada facilmente !

Circuitos Hamiltonianos

- É fácil verificar se um caminho é Hamiltoniano … mas …
- O problema de procura de circuitos é um problema de elevada complexidade, categorizado como **NP-completo**, o que significa que não há algoritmos eficientes para determinar a solução ótima. Assim, quando o nº de nós do grafo aumenta, o problema torna-se impossível de resolver em tempo útil.
- Uma alternativa óbvia, ainda que não escalável para grafos de grandes dimensões, é realizar uma **procura exaustiva**
- Esta é baseada numa travessia em profundidade do grafo, com *backtracking* quando se atinge um nó a partir do qual não é possível adicionar um sucessor ainda não visitado nesse caminho

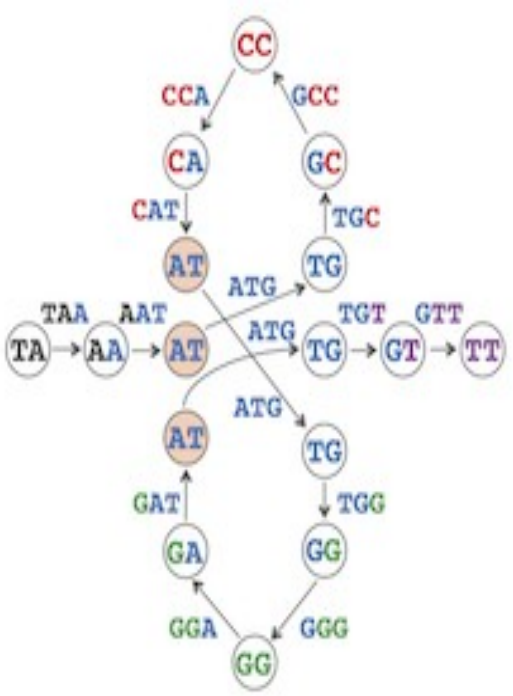
Representação alternativa: grafos de DeBruijn

- Abordando problemas semelhantes, o matemático **DeBruijn** propôs uma representação distinta do problema ainda usando grafos
- A ideia passa por representar os fragmentos (k-mers) não como nós do grafo mas antes como **arcos**, sendo os **nós** sequências de tamanho k-1 correspondendo a **prefixos/ sufixos** destes fragmentos
- O exemplo abaixo ilustra a abordagem para o conjunto de fragmentos anterior:

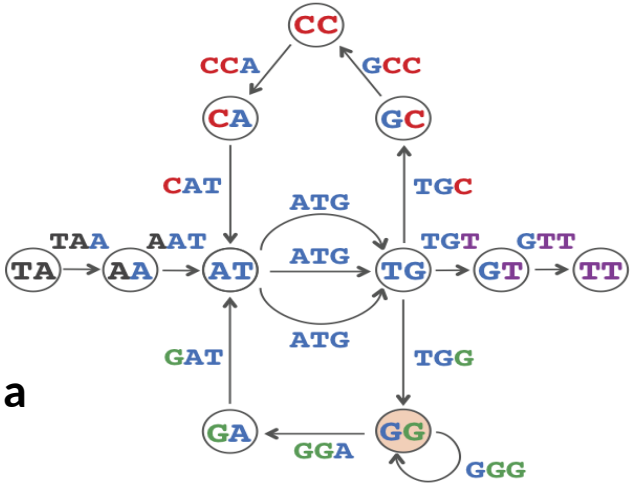


Representação alternativa do problema em grafos

- No exemplo anterior, o grafo completo é representado abaixo:



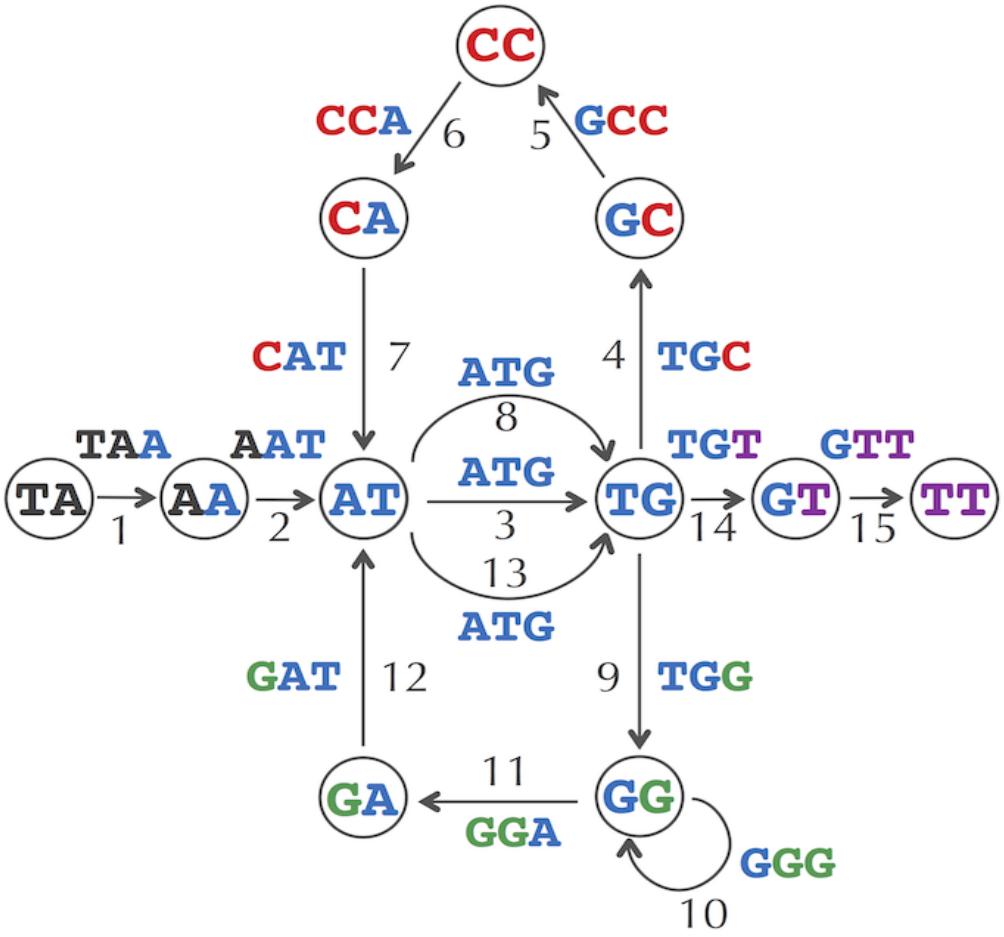
Juntando nós com a mesma sequência



Circuitos Eulerianos

- Num grafo orientado, um **Circuito Euleriano** é um **caminho que passa por todos os arcos do grafo exatamente uma vez**
- Da mesma forma, um **Ciclo Euleriano** passa por todos os arcos do grafo exatamente uma vez, regressando ao nó de partida
- O **problema da reconstrução de uma string a partir da sua composição em k-mers** pode ser formulado como a **procura de um circuito Euleriano no grafo de DeBruijn** anterior
- Temos, assim, duas formas diferentes, com dois grafos diferentes de resolver o mesmo problema. Ganhamos alguma coisa com isso?
- Para responder a esta questão temos que perceber como podemos identificar circuitos/ ciclos Eulerianos em grafos e qual a complexidade destes algoritmos

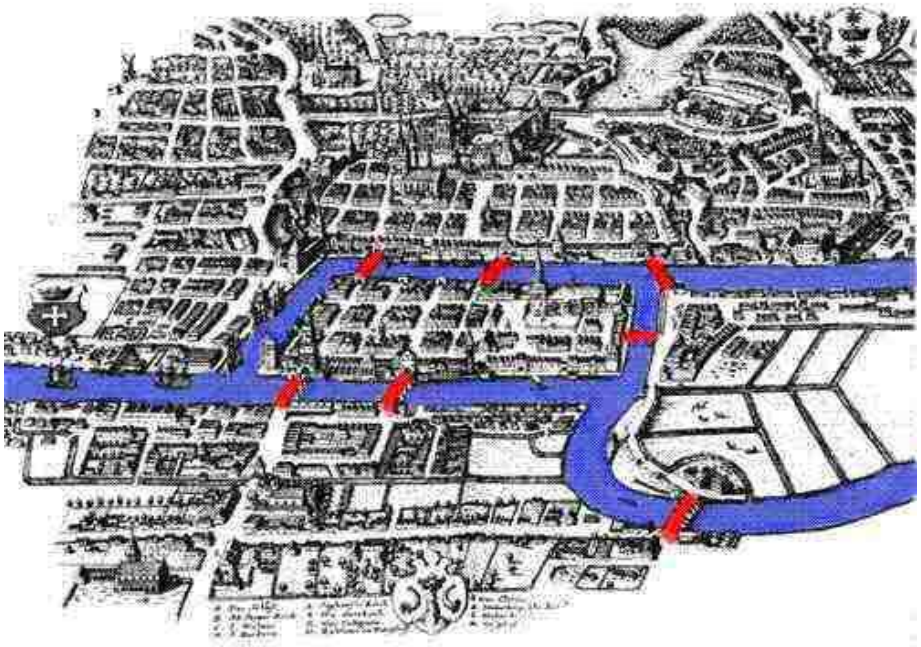
Circuitos Eulerianos para problema de reconstrução: exemplo



Caminho prossegue pela ordem numerada na figura

O início da teoria dos grafos

Descobrir um caminho que passe 1 vez em cada ponte, Leonhard Euler, 1735



pontes de Königsberg

Teorema de Euler

- Um grafo orientado é **balanceado** se para cada vértice o nº de arcos que "chegam" é igual ao nº de arcos que "partem":

  grau entrada (v)= grau saída(v)

- **Teorema**: *Um grafo conectado tem um ciclo Euleriano (é chamado grafo Euleriano) sse cada um dos seus vértices é balanceado.*

Num grafo **conectado**, todos os pares de nós estão ligados entre si por caminhos

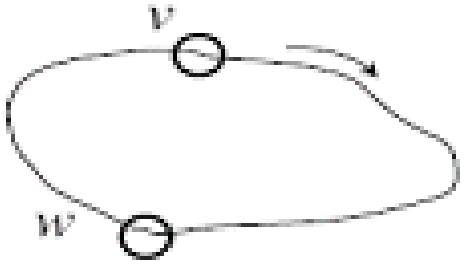
Será que há um teorema semelhante para grafos não orientados ?

Algoritmo para a construção de ciclos Eulerianos

Passo 1

Começar com um vértice *v* e formar um ciclo arbitrário usando arcos ainda não visitados até atingir um “beco sem saída”, i.e. um nó sem sucessores atingíveis por arcos não usados.

Como o grafo é Euleriano paramos necessariamente em *v*.

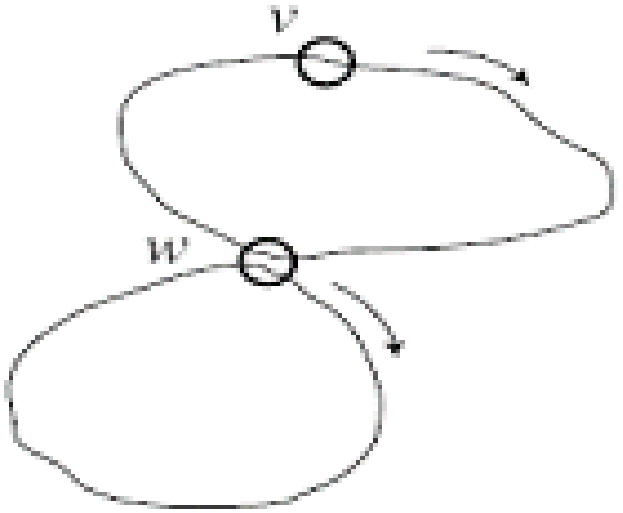


(a)

Algoritmo para Construção de um ciclo Euleriano (cont.)

Passo 2

Se ciclo do passo 1 ainda não é Euleriano (i.e. não contém todos os arcos), contém pelo menos um vertice *w*, com arcos não usados. Repetir o passo 1, usando *w* como ponto de partida. Este passo terminará em *w*.

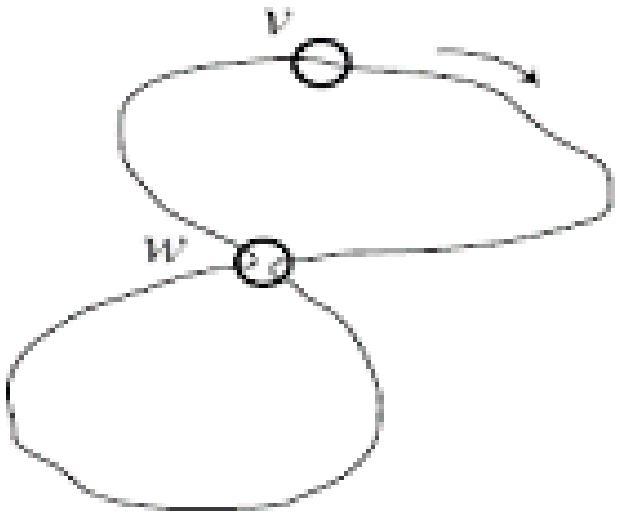


(b)

Algoritmo para Construção de um ciclo Euleriano (cont.)

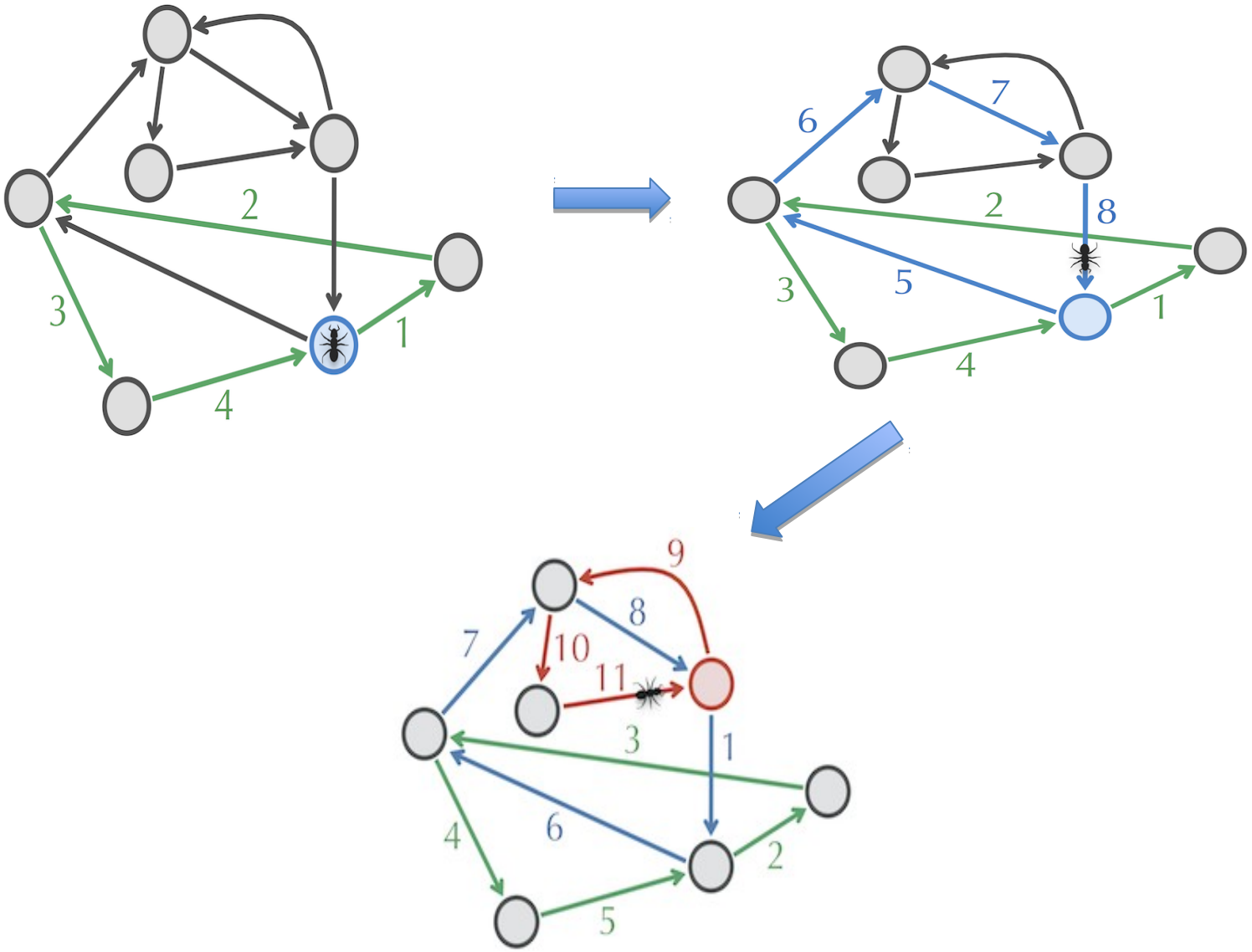
Passo 3

Combinar ciclos dos passos 1 e 2 num único. Iterar passo 2 até “cobrir” todos os arcos



(c)

Exemplo



Circuitos Eulerianos

- O problema de procura de circuitos Eulerianos é de **complexidade bastante inferior** ao seu homólogo para circuitos Hamiltonianos, podendo ser resolvido para grafos de grandes dimensões usando o algoritmo anterior (uma implementação eficiente garante tempos lineares)
- Os programas de montagem de genomas só recentemente começaram a usar esta estratégia; os primeiros programas (e.g. os usados no projeto do Genoma Humano) usavam grafos de sobreposição e caminhos Hamiltonianos / PCV
- Atualmente, muitos dos programas para montagem de genomas a partir de dados de sequenciação são baseados em grafos de DeBruijn e circuitos Eulerianos

Teorema Euler: Extensão

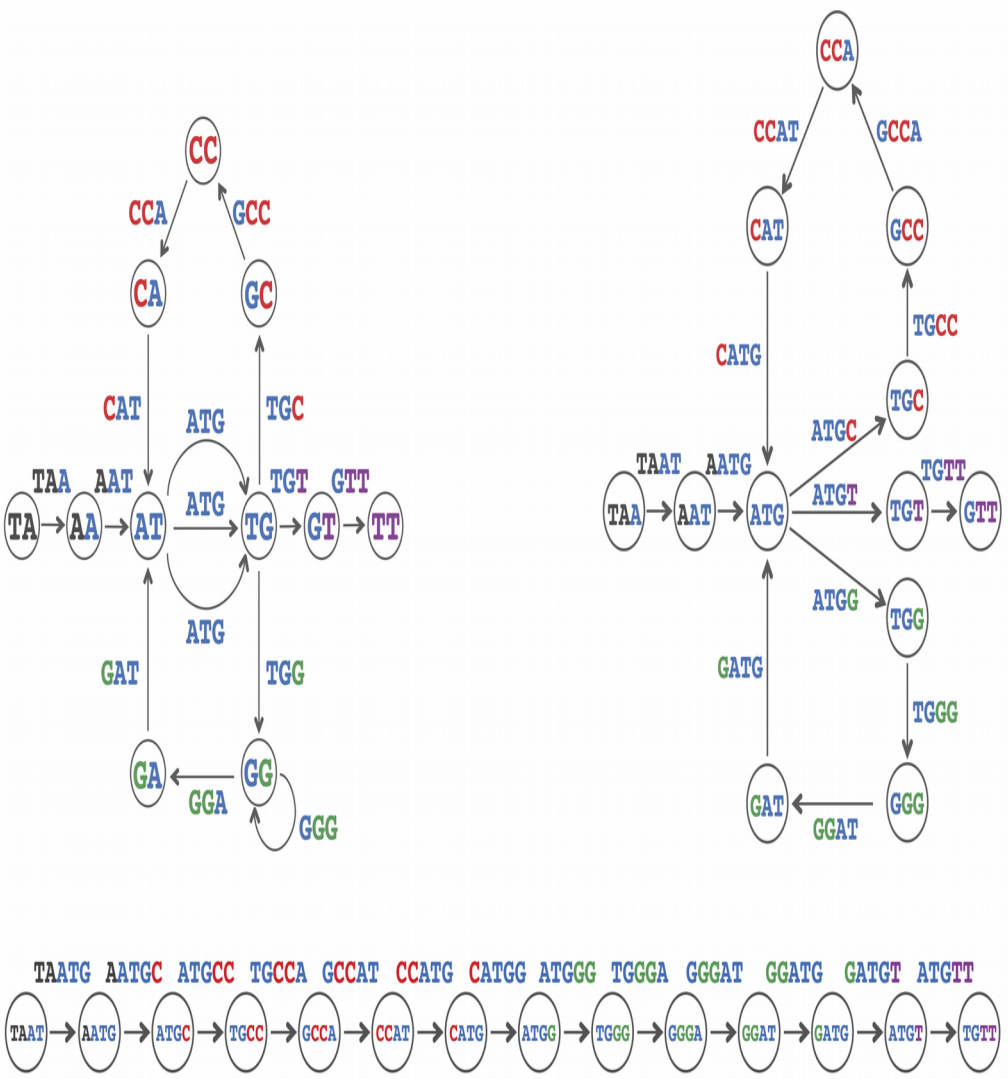
- **Teorema**: *Um grafo conectado tem um* ***caminho Euleriano*** *sse contém no máximo dois vértices semi-balanceados (diferença entre grau de entrada e saída é 1 e -1 respetivamente) e todos os outros são balanceados.*

- **Algoritmo**: *(a) unir dois vértices semi-balanceados – passamos a ter grafo euleriano; (b) determinar ciclo euleriano no novo grafo; (c) remover do ciclo o arco inserido*

E no caso dos grafos não orientados ?

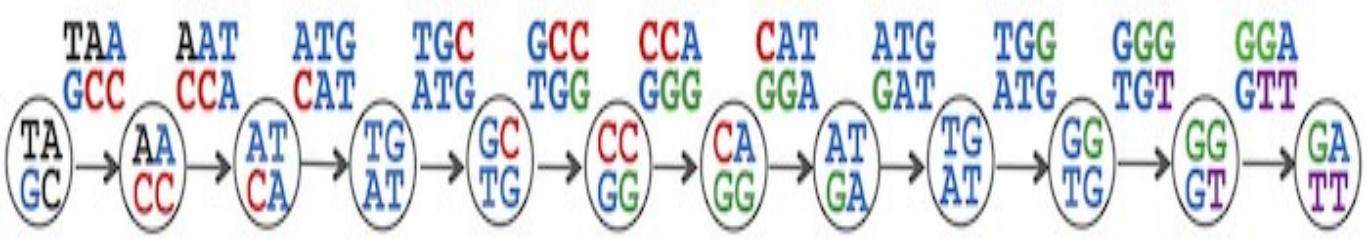
Grafos de DeBruijn no mundo real da sequenciação

- Em casos reais, quanto maior for o tamanho das leituras do sequenciador menor é o grau médio dos nós e mais fácil é definir o caminho Euleriano (ver exemplo de 3 grafos para a mesma sequência original com diferentes tamanhos de fragmentos)



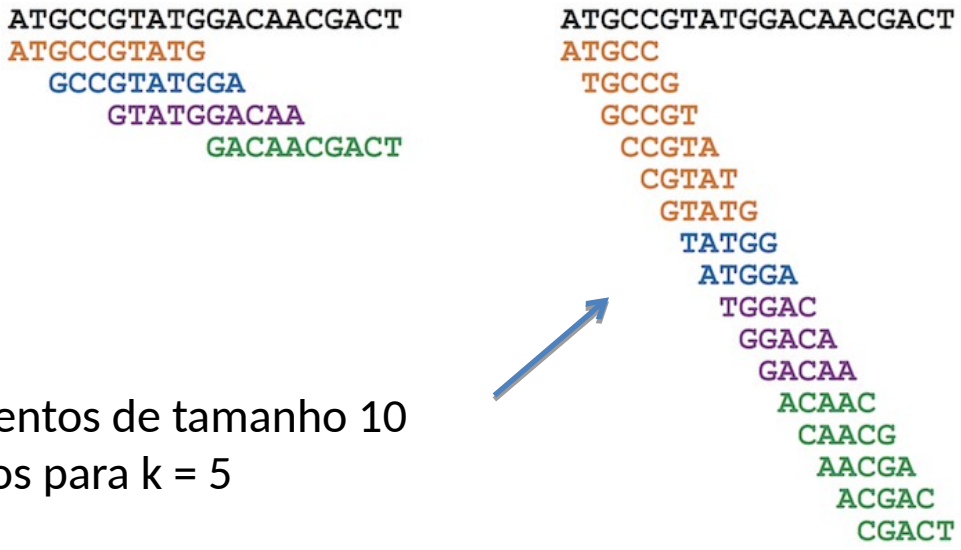
Grafos de DeBruijn no mundo real da sequenciação

- **Repetições** no genoma causam problemas, especialmente se forem maiores que o tamanho das leituras
- Uma forma de ultrapassar este problema são as **leituras emparelhadas (*paired reads*)**, onde temos k nucléotidos medidos, um gap fixo de d nuceótidos e outros k nucleótidos medidos
- Algoritmos semelhantes aos vistos atrás com grafos de DeBruijn podem ser usados para estas leituras emparelhadas



Problemas no mundo real da sequenciação

- Numa sequenciação, cada fragmento do genoma é sequenciado um dado nº de vezes, chamado de **cobertura**
- No entanto, este é um valor médio havendo fragmentos que não aparecem
- Uma forma de ligar com este problema é trabalhar nos algoritmos com valores de k mais pequenos que o tamanho dos fragmentos



Fragmentos de tamanho 10 partidos para k = 5

Problemas no mundo real da sequenciação: contigs

- Mesmo com a solução anterior, continuarão a existir gaps, levando os grafos de DeBruijn a terem arcos em falta
- Assim, programas de montagem de genomas, numa primeira fase, montam partes contíguas do genoma (designadas por **contigs**) com o menor número e a maior extensão possível
- Numa sequenciação procariótica, é típico termos centenas de contigs como resultado com tamanhos na ordem dos milhares até às centenas de milhares de nucleótidos

Problemas no mundo real da sequenciação: erros de leitura

- A existência de **erros** de leitura em muitos dos fragmentos lidos torna mais difícil a tarefa da montagem
- Alguns erros mais simples (e.g. troca de um nucleótido) podem ser identificados nos grafos de DeBruijn e removidos por algoritmos especializados
- No entanto, é sempre complexo saber quais os caminhos no grafo que são erros e os que são corretos
- A existência de repetições não exatas no genoma torna este problema ainda mais complexo