

**Instituto de  
Computação**

UNIVERSIDADE ESTADUAL DE CAMPINAS



# Montagem de Fragmentos

MO640 - Biologia Computacional / MC668 - Bioinformática

---

Zanoni Dias

2021

Instituto de Computação

Montagem de Fragmentos

Representação de Sobreposição de Fragmentos

Caminhos e Supersequências

Algoritmo para Shortest Common Superstring

Montagem de Fragmentos em Grafos de Sobreposições Acíclicos

Problemas com Repetições

Montagem de Fragmentos usando Grafos de  $k$ -Mers

Phred, Phrap e Consed

CAP3

# Montagem de Fragmentos

---

# Montagem de Fragmentos

- A tecnologia padrão de sequenciamento não permite obter fragmentos de DNA maiores que 1000 pares de bases.
- É possível obter fragmentos de DNA (um pouco) maiores, mas a um custo proibitivo.
- Na prática, muitas vezes precisamos obter a sequência de organismos de milhões de pares de bases.
- Montagem de fragmentos é a tarefa de, dado um conjunto de fragmentos, reconstruir a sequência que originou os fragmentos (sequência alvo), com base nas sobreposições dos fragmentos.
- Montagem de fragmentos pode ser revolido com estratégias convencionais de alinhamento múltiplo de sequências?
  - Não! Apesar de parecidos, os problemas tem diferenças importantes e usam técnicas distintas para obter soluções.

# Principais Dificuldades

- Erros de sequenciamento.
- Orientação desconhecida dos fragmentos.
- Falta de cobertura da sequência original.
- Tamanho desconhecido da sequência original.
- Regiões repetidas na sequência original.
- Sequências quiméricas.
- Contaminação pelo vetor de sequenciamento.

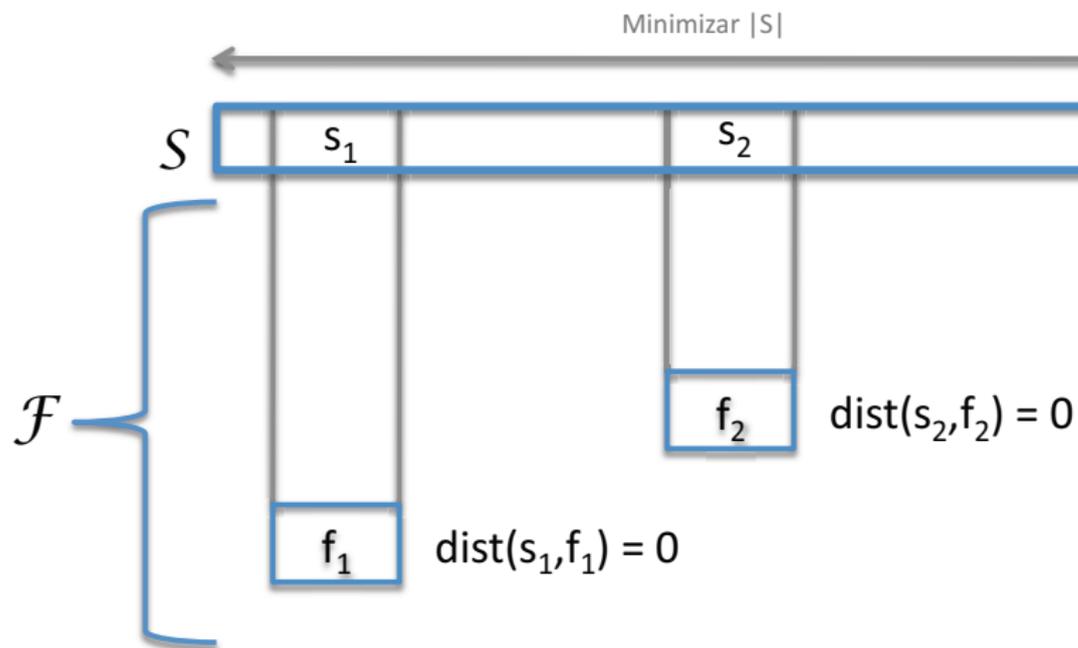
# Modelos para Montagem de Fragmentos

- Modelos mais comuns:
  - *Shortest Common Superstring (SCS)*.
  - *Reconstruction*.
  - *Multicontig*.
- Todos estes modelos supõem que os fragmentos não possuem contaminações ou quimeras.

# Shortest Common Superstring

- Dada uma coleção  $\mathcal{F}$  de fragmentos, obter a menor sequência possível  $S$ , tal que para todo  $f \in \mathcal{F}$ ,  $S$  é uma supersequência de  $f$ .
- Modelo essencialmente teórico, sem suporte a maioria dos problemas práticos.
- Pode não produzir a sequência original, devido a dificuldade de lidar com longos trechos repetidos.
- $SCS \in NP$ -Completo.

# Shortest Common Superstring



# Reconstruction

- Dada uma coleção  $\mathcal{F}$  de fragmentos e uma tolerância de erro  $\epsilon$  ( $0 \leq \epsilon \leq 1$ ), obter a menor sequência possível  $S$ , tal que para todo  $f \in \mathcal{F}$ , temos:

$$\min(\text{dist}_s(f, S), \text{dist}_s(\bar{f}, S)) \leq \epsilon|f|$$

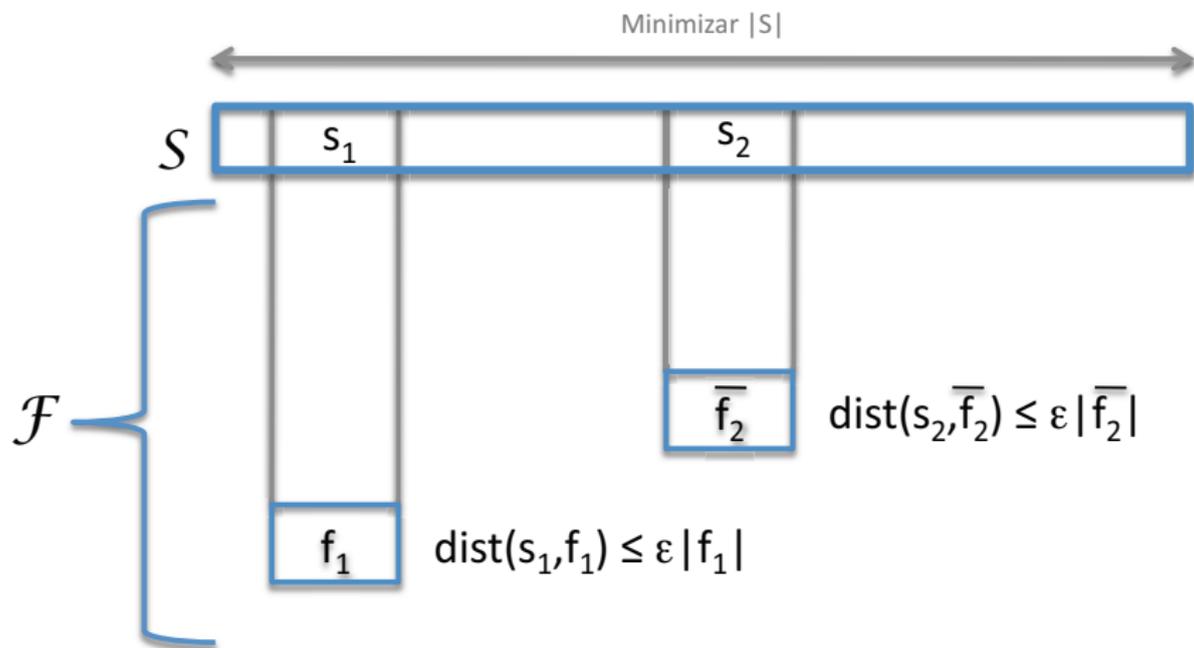
onde  $\bar{f}$  é o complemento reverso de  $f$  e  $\text{dist}_s$  é definida como:

$$\text{dist}_s(a, b) = \min_{s \in S(b)} \text{dist}(a, s)$$

onde  $S(b)$  é o conjunto das subsequências de  $b$ .

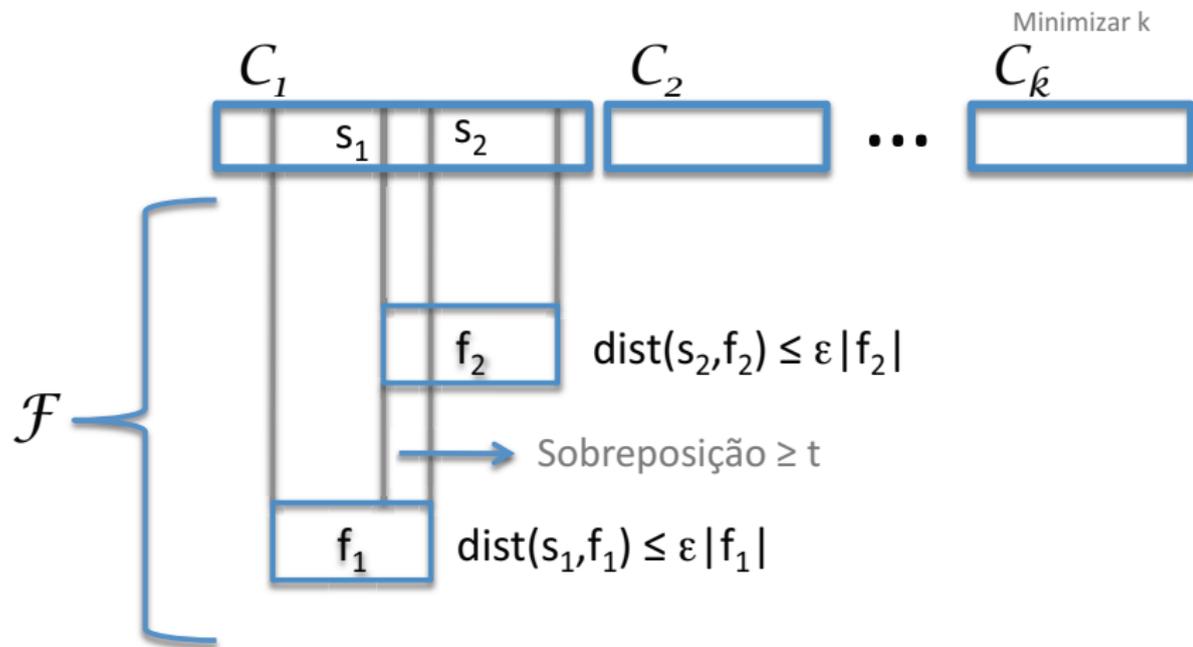
- *Reconstruction* é uma generalização de SCS.
- *Reconstruction*  $\in$  NP-Completo.

# Reconstruction



- Dada uma coleção  $\mathcal{F}$  de fragmentos, um inteiro  $t \geq 0$  e uma tolerância de erro  $\epsilon$  ( $0 \leq \epsilon \leq 1$ ), obter uma partição de  $\mathcal{F}$  em um número mínimo de subcoleções,  $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k\}$ , tal que cada  $\mathcal{C}_i$  (com  $1 \leq i \leq k$ ) forma um *contig* com sobreposição mínima  $t$  entre os fragmentos e taxa de erro  $\epsilon$  de cada fragmento em relação ao consenso do *contig*.
- Neste caso, cada *contig* representa uma sequência consenso para um subconjunto dos fragmentos.
- *Multicontig*  $\in$  NP-Completo.

# Multicontig



# Calculando o Progresso da Montagem

- Seja:
  - $n$ : número de fragmentos.
  - $f$ : tamanho médio dos fragmentos.
  - $T$ : tamanho da sequência alvo a ser montada.
  - $t$ : sobreposição mínima entre dois fragmentos para montagem.
- A cobertura média ( $c$ ) da sequência alvo pode ser calculada como:

$$c = \frac{nf}{T}$$

- O número esperado de subsequências contíguas montadas com sobreposição mínima  $t$  é dado por:

$$p = ne^{-\frac{n(f-t)}{T}}$$

- O número esperado de subsequências contíguas montadas por pelo menos 2 fragmentos, com sobreposição mínima  $t$  é dado por:

$$p' = ne^{-\frac{n(f-t)}{T}} - ne^{-\frac{2n(f-t)}{T}}$$

## Calculando a Cobertura da Sequência Alvo

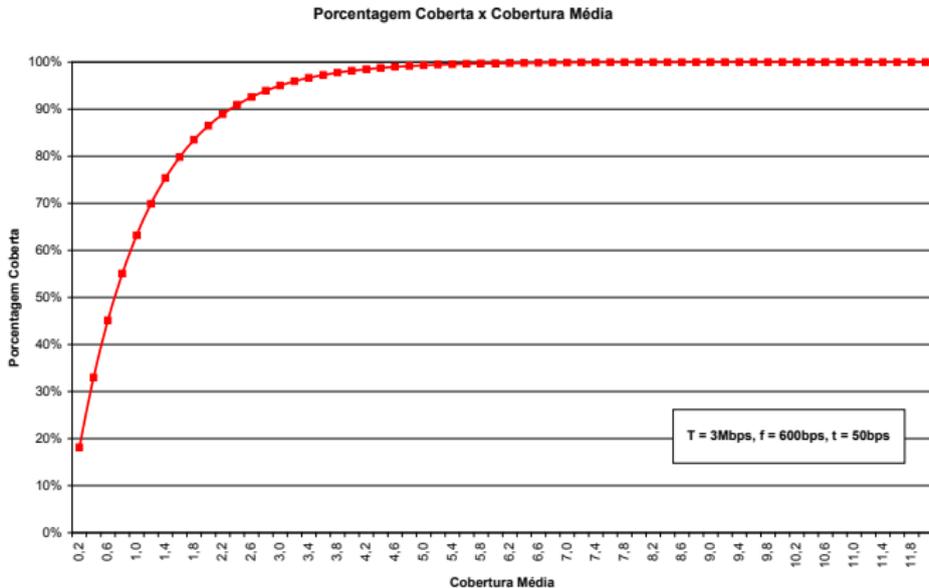
- A fração da sequência alvo coberta por exatamente  $k$  fragmentos é dado por:

$$r_k = \frac{e^{-c} c^k}{k!}$$

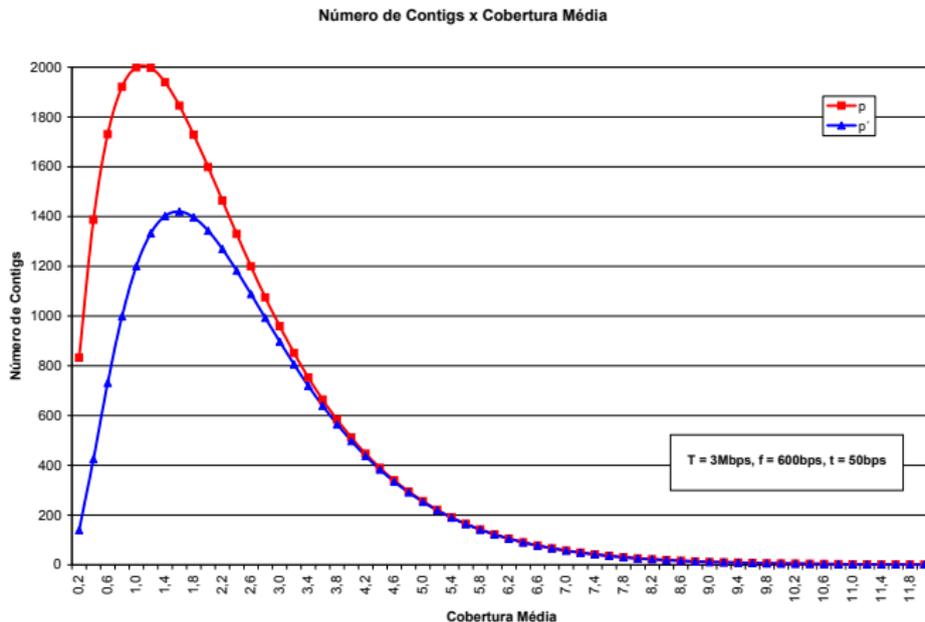
- A fração da sequência alvo coberta por pelo menos um fragmento é dado por:

$$r = 1 - \left(1 - \frac{f}{T}\right)^n$$

# Calculando a Cobertura da Sequência Alvo



# Calculando a Cobertura da Sequência Alvo



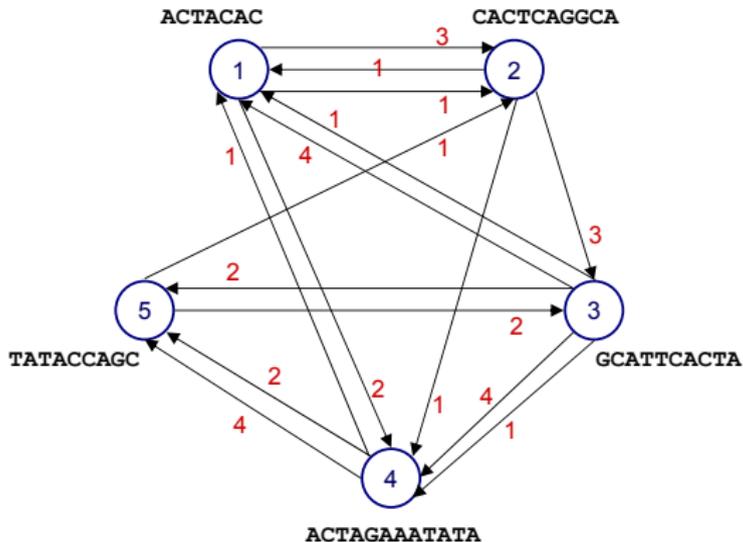
# Representação de Sobreposição de Fragmentos

---

# Representação de Sobreposição de Fragmentos

- Seja  $\mathcal{F}$  uma coleção de fragmentos de tal forma que nenhum fragmento esteja completamente contido em outro fragmento.
- O Multigrafo de Sobreposição  $\mathcal{OM}(\mathcal{F})$  (ou *Overlap Multigraph*) de uma coleção de fragmentos de sequências  $\mathcal{F}$  é um multigrafo orientado e ponderado.
- O conjunto de vértices  $V$  representa cada um dos fragmentos  $f \in \mathcal{F}$ .
- Uma aresta entre os vértices  $a$  e  $b$  ( $a \neq b$ ), com peso  $t \geq 0$ , existe se o sufixo do fragmento representado por  $a$ , com  $t$  caracteres, é um prefixo do fragmento representado por  $b$ .
- Por definição,  $\mathcal{OM}(\mathcal{F})$  não admite autolaços.
- Podem existir múltiplas arestas entre dois vértices.
- Existe pelo menos uma aresta entre todo par de vértices (com  $t = 0$ ).

# Multigrafo de Sobreposição - $\mathcal{OM}(\mathcal{F})$



Sobreposição mínima:  $t = 1$

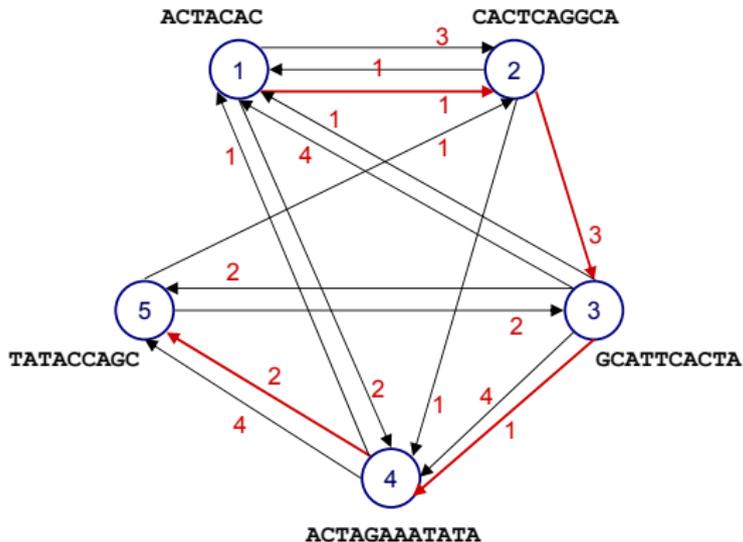
# Caminhos e Supersequências

---

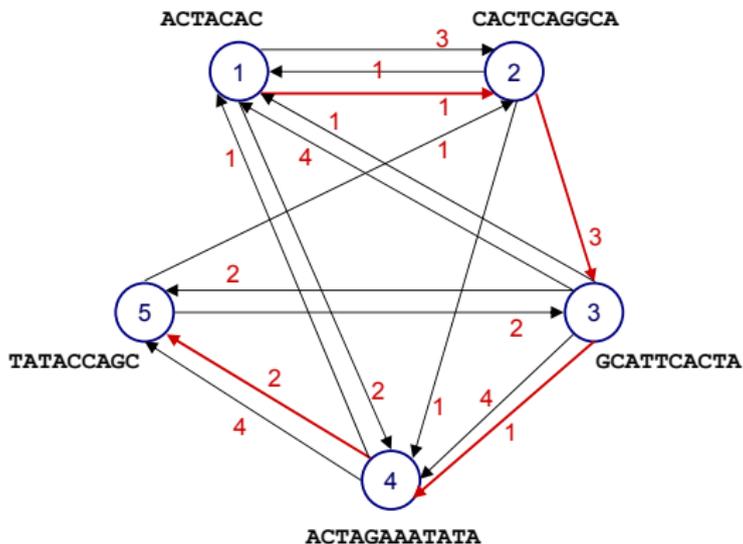
# Caminhos e Supersequências

- Caminhos no Multigrafo de Sobreposição  $\mathcal{OM}(\mathcal{F})$  representam supersequências envolvendo os fragmentos representados pelos vértices do caminho.
- Seja:
  - $P$ : um caminho em  $\mathcal{OM}(\mathcal{F})$ .
  - $w(P)$ : a soma dos pesos de todas as arestas de  $P$ .
  - $\mathcal{F}(P)$ : o conjunto de fragmentos representados pelos vértices de  $P$ .
  - $||\mathcal{F}(P)||$ : a soma dos tamanhos de todos os fragmentos de  $\mathcal{F}(P)$ .
  - $S(P)$ : a sequência consenso originada por  $P$ .
- A seguinte relação é verdadeira:
  - $||\mathcal{F}(P)|| = w(P) + |S(P)|$
- Obter uma SCS para a coleção  $\mathcal{F}$ , é equivalente a encontrar um caminho de peso máximo que passe por todos os vértices de  $\mathcal{OM}(\mathcal{F})$ .
- Logo, uma solução para SCS pode ser obtida através de um Caminho Hamiltoniano Máximo no multigrafo  $\mathcal{OM}(\mathcal{F})$ .

# Caminhos Hamiltonianos no Multigrafo de Sobreposição



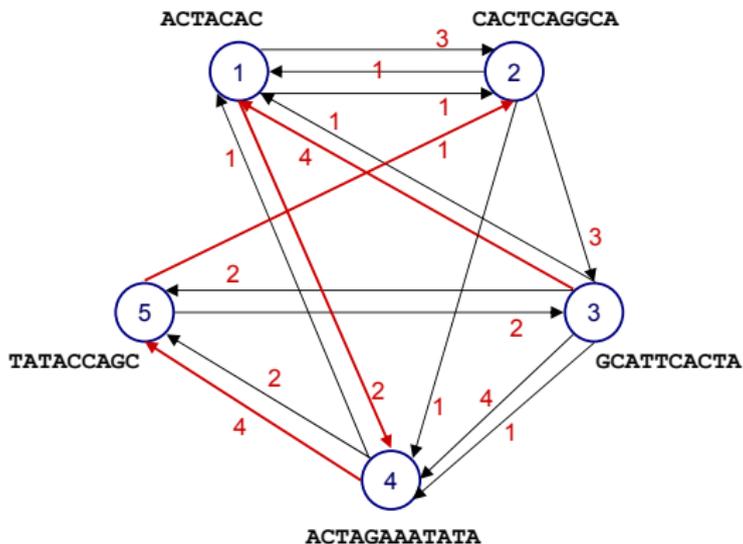
# Caminhos Hamiltonianos no Multigrafo de Sobreposição



ACTACACACTCAGGCAATTCACTACTAGAAATATATACCAGC

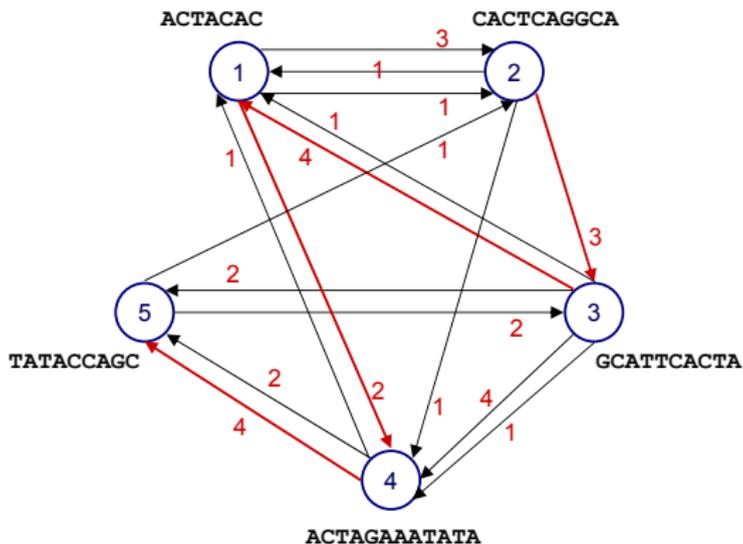


# Caminhos Hamiltonianos no Multigrafo de Sobreposição

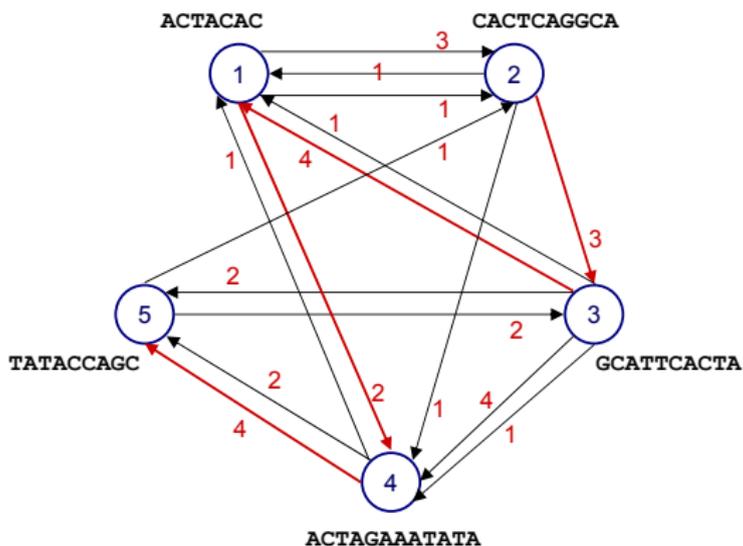


GCATTC**ACTACACT**AGAA**TATA**CCAG**CACTCAGGCA**

# Caminhos Hamiltonianos no Multigrafo de Sobreposição

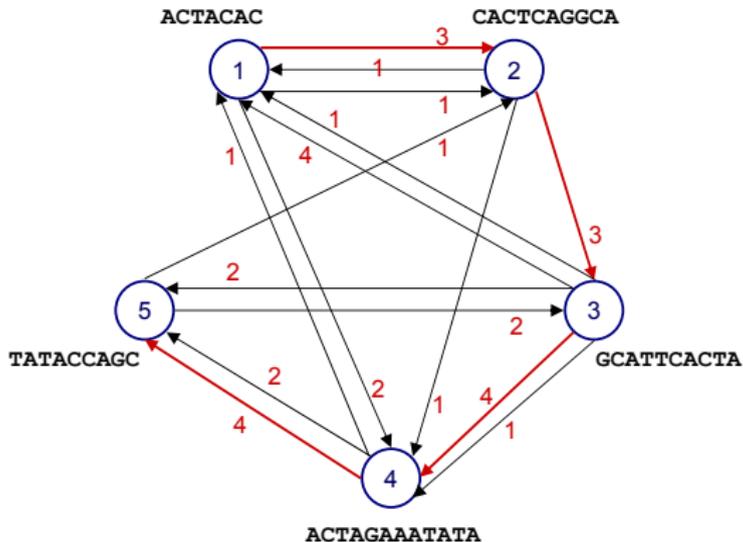


# Caminhos Hamiltonianos no Multigrafo de Sobreposição

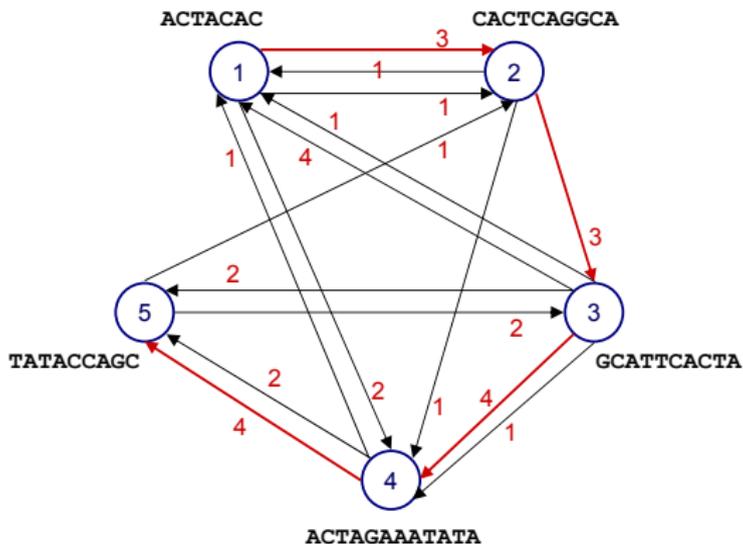


CACTCAGGCAATTCAC TA ACTAGAAATATA TATACCAGC

# Caminhos Hamiltonianos no Multigrafo de Sobreposição



# Caminhos Hamiltonianos no Multigrafo de Sobreposição



ACTACACTCAGGCATTCAC TAGAAATATACCAGC

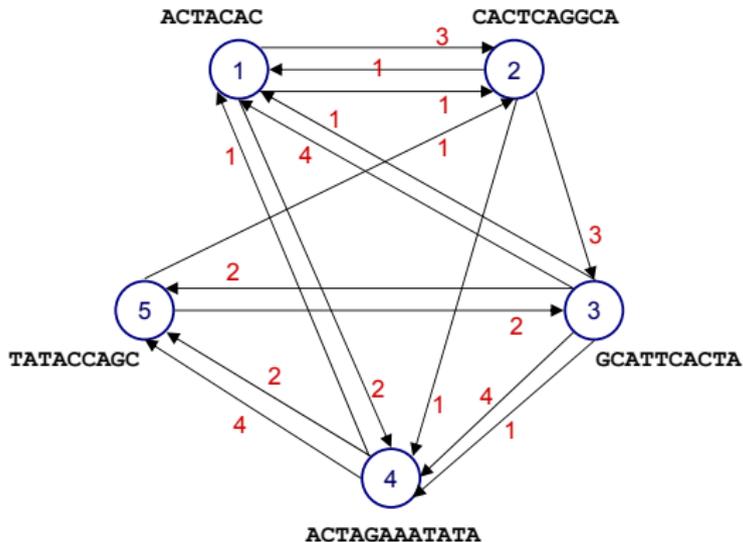
# Algoritmo para Shortest Common Superstring

---

# Algoritmo Guloso para Shortest Common Superstring

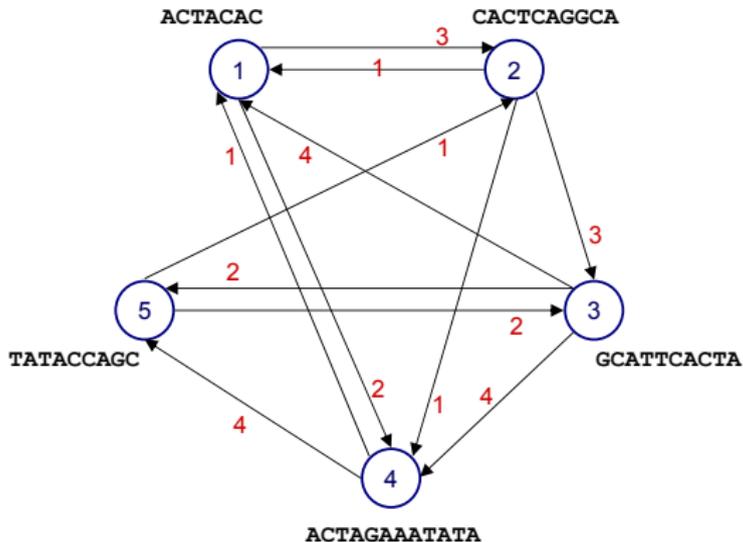
- Neste caso podemos trabalhar com o Grafo de Sobreposição  $OG(\mathcal{F})$  (*Overlap Graph*), que pode ser obtido a partir de  $OM(\mathcal{F})$  mantendo-se apenas a aresta mais pesada entre cada par de vértices.
- Algoritmos gulosos fazem escolhas locais ótimas.
- Para tentar maximizar o peso do caminho a ser montado, o algoritmo, a cada passo, escolhe a aresta válida mais pesada de  $OG(\mathcal{F})$ .
- Uma aresta é dita válida se a inclusão dela na solução corrente respeita as seguintes condições:
  - Duas arestas não podem sair de um mesmo vértice.
  - Duas arestas não podem chegar em um mesmo vértice.
  - Nenhum ciclo pode ser formado.
- O algoritmo termina quando o caminho  $P$  contiver todos os vértices de  $OG(\mathcal{F})$ .

# Multigrafo de Sobreposição - $\mathcal{OM}(\mathcal{F})$



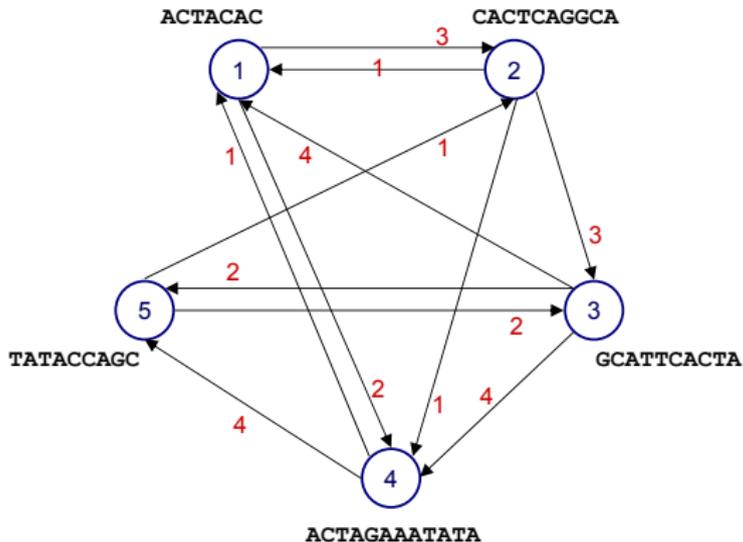
Sobreposição mínima:  $t = 1$

# Grafo de Sobreposição - $OG(\mathcal{F})$



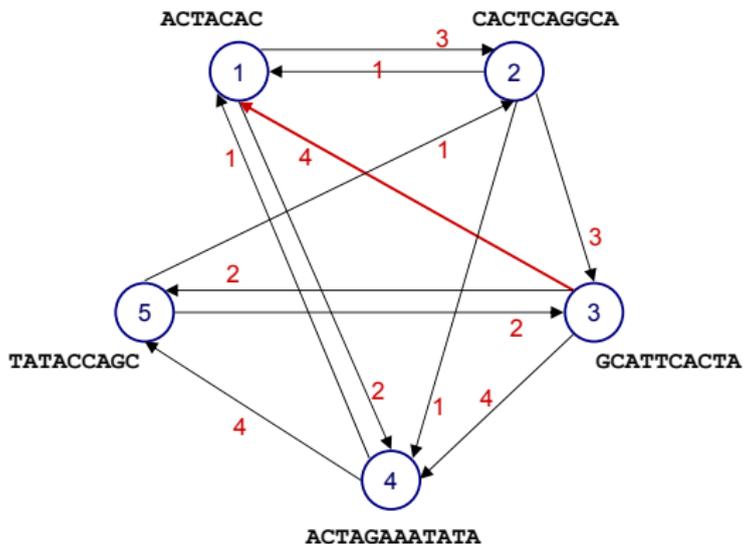
Sobreposição mínima:  $t = 1$

# Algoritmo Guloso para Shortest Common Superstring



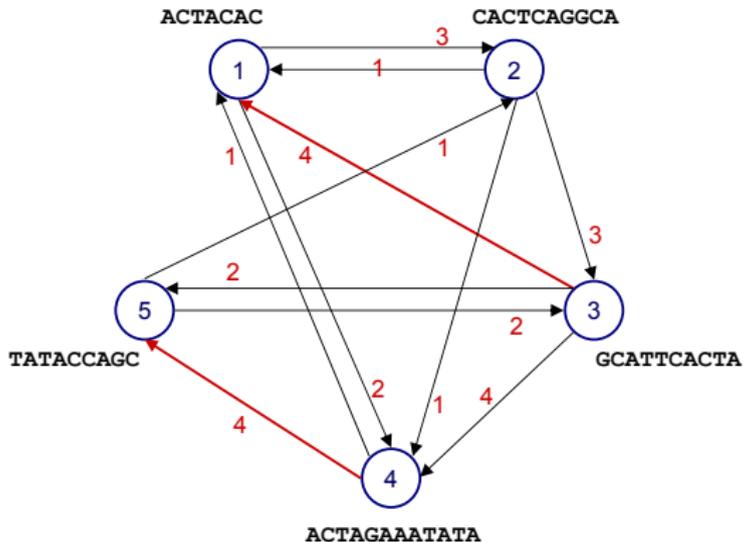
ACTACAC | CACTCAGGCA | GCATTCACTA | ACTAGAAATATA | TATACCAGC

# Algoritmo Guloso para Shortest Common Superstring



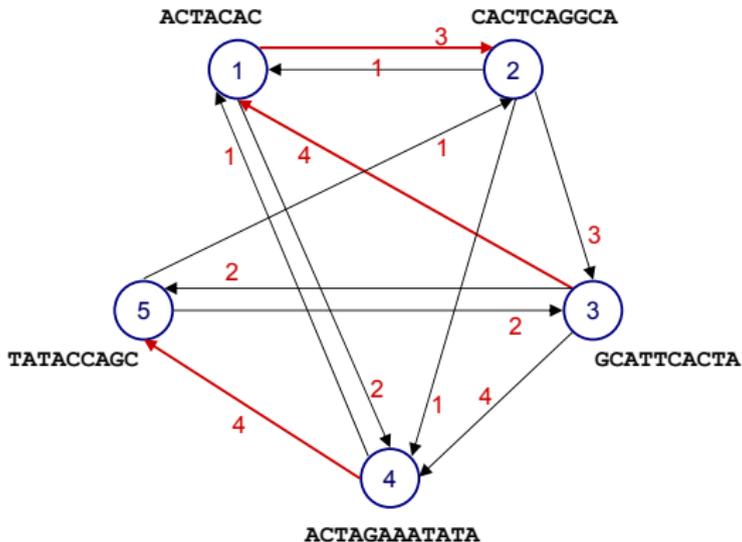
GCATTC**ACT**ACAC | CACTCAGGCA | ACTAGAAATATA | TATACCAGC

# Algoritmo Guloso para Shortest Common Superstring



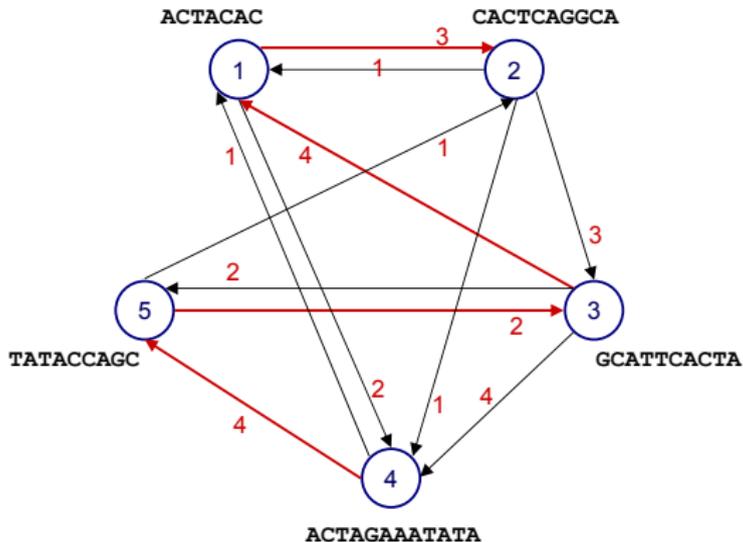
ACTAGAAA**TATACCAGC** | GCATTCA**CTACAC** | CACTCAGGCA

# Algoritmo Guloso para Shortest Common Superstring



ACTAGAAA**TATACCAGC** | GCATTC**ACTACAC**TCAGGCA

# Algoritmo Guloso para Shortest Common Superstring



ACTAGAAATATACCAGCATTCACTACACTCAGGCA

# Algoritmo Guloso para Shortest Common Superstring

- Complexidade:
  1. Construir o grafo  $\mathcal{OG}(\mathcal{F})$ :
    - Usando comparação par a par:  $O(\sum_{i=1}^n \sum_{j=1}^n f_i f_j + n^2)$   
 $= O(\|\mathcal{F}\|^2 + n^2)$ .
    - Usando árvores de prefixos:  $O(\|\mathcal{F}\| + n^2)$ .
  2. Ordenar as arestas de  $\mathcal{OG}(\mathcal{F})$  em função do peso:
    - Usando heapsort:  $O(n^2 \log n)$ .
    - Usando counting sort:  $O(n^2 + \|\mathcal{F}\|)$ .
  3. Para toda aresta, testar se ela é válida:
    - Usando conjuntos disjuntos:  $O(n^2 \alpha(n))$ .
    - Usando vetores auxiliares para armazenar os vértices iniciais e os vértices finais dos caminhos que passam por cada vértice:  $O(n^2)$ .
  4. Para toda aresta válida, expandir um caminho:
    - Usando conjuntos disjuntos:  $O(n \alpha(n))$ .
    - Usando vetores auxiliares para armazenar os vértices iniciais e os vértices finais dos caminhos que passam por cada vértice:  $O(n^2)$ .
  5. Dado o Caminho Hamiltoniano  $P$  construir a sequência  $S(P)$ :  
 $O(\|\mathcal{F}\|)$ .
    - Total:  $O(\|\mathcal{F}\| + n^2)$ .

# Algoritmo Guloso para Shortest Common Superstring

- Algoritmo proposto independentemente por Jorma Tarhio e Esko Ukkonen (1988) e Jonathan Turner (1989).
- Avrim Blum, Tao Jiang, Ming Li, John Tromp e Mihalis Yannakakis (1994) provaram que o algoritmo guloso é um algoritmo de aproximação com fator 4.
- Haim Kaplan e Nira Shafrir (2005) provaram que o algoritmo guloso é um algoritmo de aproximação com fator 3.5.

## Conjectura

*O algoritmo guloso para SCS é um algoritmo de aproximação com fator 2.*

# Algoritmos de Aproximação para SCS

- Avrim Blum, Tao Jiang, Ming Li, John Tromp e Mihalis Yannakakis (1994) apresentaram um algoritmo de aproximação com fator 3.
- Shang-Hua Teng e Frances Yao (1993) apresentaram um algoritmo de aproximação com fator  $2 + 8/9$ .
- Artur Czumaj, Leszek Gasieniec, Marek Piotrow e Wojciech Rytter (1994) apresentaram um algoritmo de aproximação com fator  $2 + 5/6$ .
- Chris Armen e Clifford Stein (1995) apresentaram um algoritmo de aproximação com fator  $2 + 3/4$ .
- Chris Armen e Clifford Stein (1996) apresentaram um algoritmo de aproximação com fator  $2 + 2/3$ .
- Elizabeth Sweedyk (1999) apresentou um algoritmo de aproximação com fator  $2 + 1/2$ .

## Exercício

*Mostre como adaptar o algoritmo guloso para Shortest Common Superstring para lidar com erros de sequenciamento.*

## Exercício

*Mostre como adaptar o algoritmo guloso para Shortest Common Superstring para lidar com orientação desconhecida dos fragmentos.*

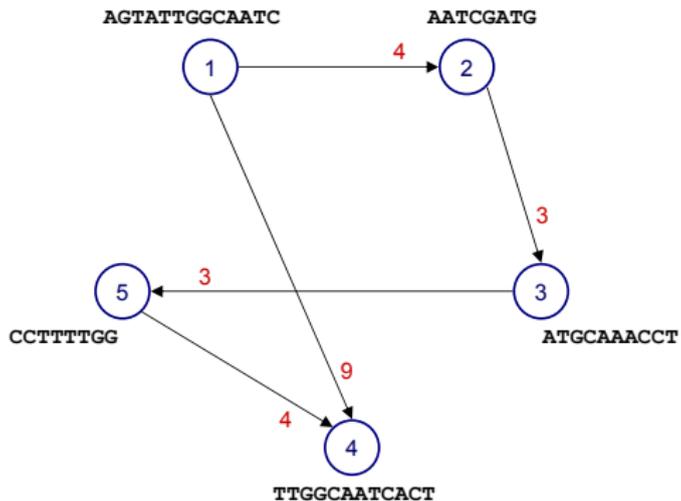
# Montagem de Fragmentos em Grafos de Sobreposições Acíclicos

---

# Montagem de Fragmentos - Grafos de Sobreposições Acíclicos

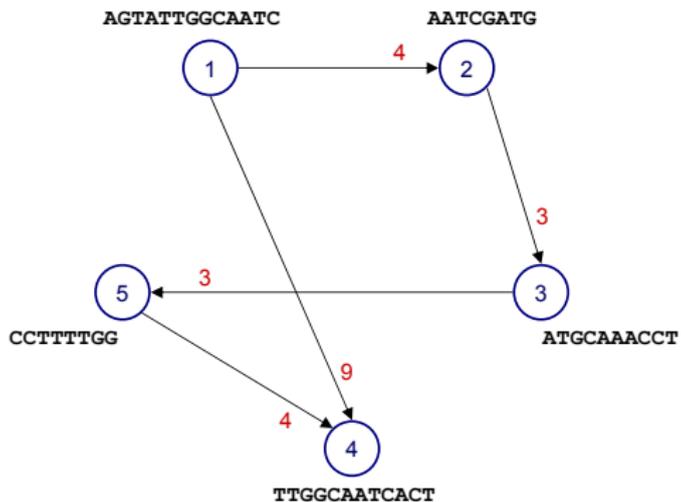
- Seja  $\mathcal{F}$  uma coleção de fragmentos tal que nenhum fragmento esteja completamente contido em outro.
- Considere o grafo  $\mathcal{OG}(\mathcal{F}, t)$ , que pode ser construído a partir de  $\mathcal{OG}(\mathcal{F})$  removendo as arestas de peso menor do que  $t$ .
- Se  $\mathcal{OG}(\mathcal{F}, t)$  possui um ciclo orientado, então existe uma repetição de tamanho maior ou igual a  $t$  na sequência original ( $S$ ). Note que o contrário não é necessariamente verdade.
- Se a sequência original ( $S$ ) for totalmente coberta por um único *contig*, com sobreposição mínima  $t$  entre os fragmentos, e sem nenhuma repetição de tamanho maior ou igual a  $t$ , então o grafo  $\mathcal{OG}(\mathcal{F}, t)$  é acíclico, existe um único Caminho Hamiltoniano ( $P$ ) em  $\mathcal{OG}(\mathcal{F}, t)$  e  $S = S(P)$ .
- Neste caso, o Caminho Hamiltoniano em  $\mathcal{OG}(\mathcal{F}, t)$ , pode ser obtido através de uma ordenação topológica, em tempo  $O(n^2)$ .

# Algoritmo Guloso



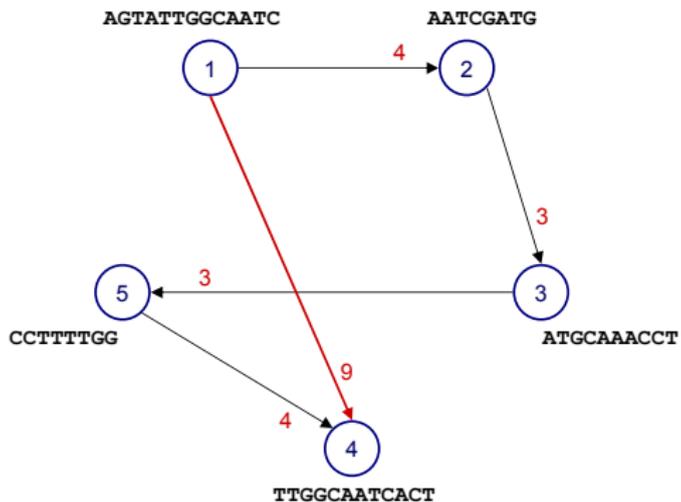
Sobreposição mínima:  $t = 3$

# Algoritmo Guloso



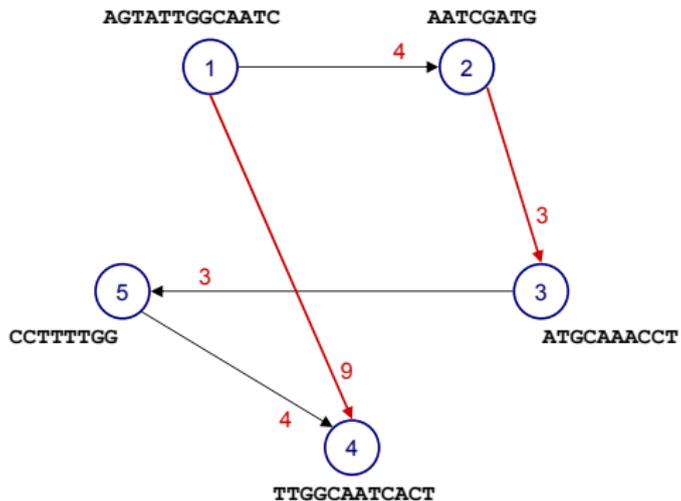
AGTATTGGCAATC | AATCGATG | ATGCAAACCT | TTGGCAATCACT | CCTTTTGG

# Algoritmo Guloso



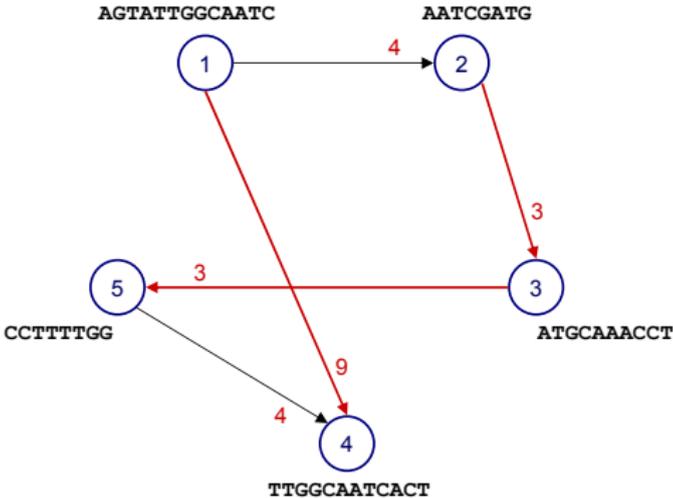
AGTATTGGCAATCACT | AATCGATG | ATGCAAACCT | CCTTTTGG

# Algoritmo Guloso



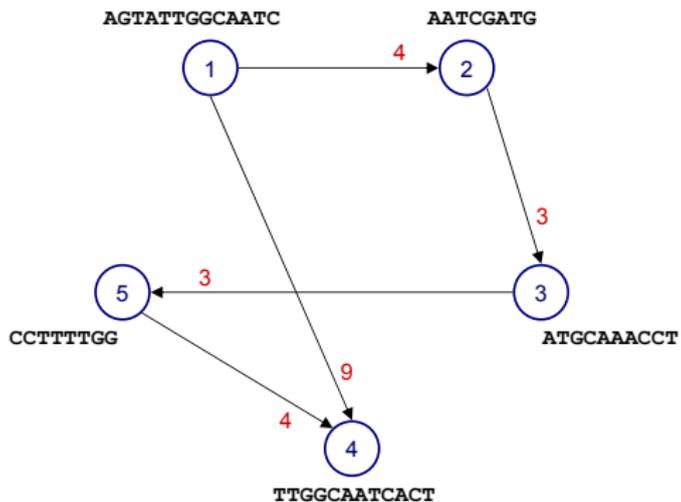
AGTATTGGCAATCACT | AATCGATGCAAACCT | CCTTTTGG

# Algoritmo Guloso



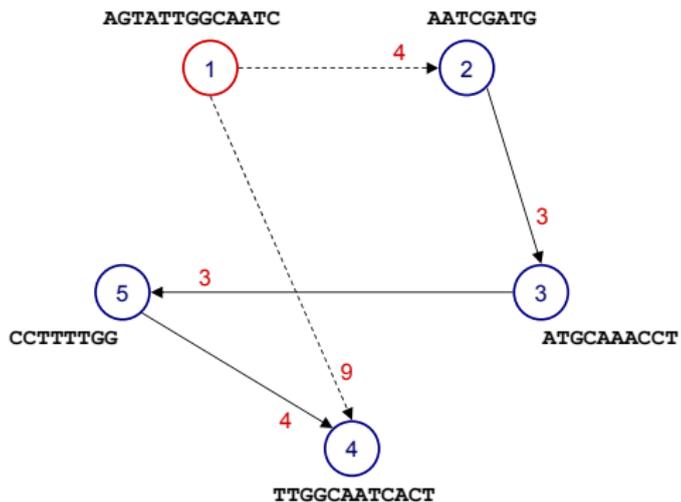
AGTATTGGCAATCACT | AATCGATGCAAACCTTTTGG

# Ordenação Topológica



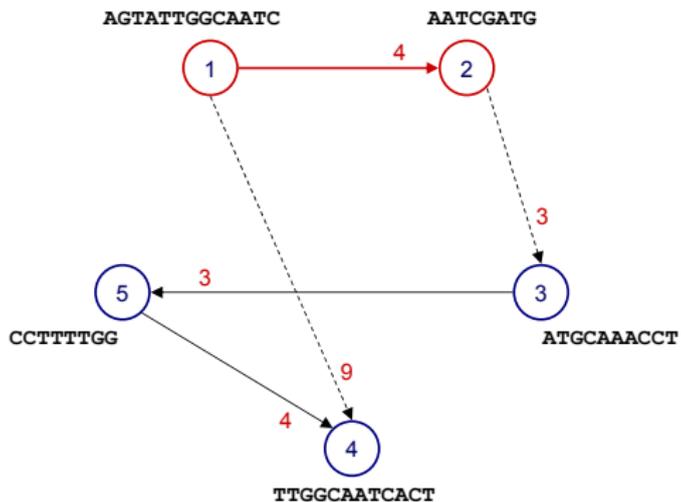
Sobreposição mínima:  $t = 3$

# Ordenação Topológica



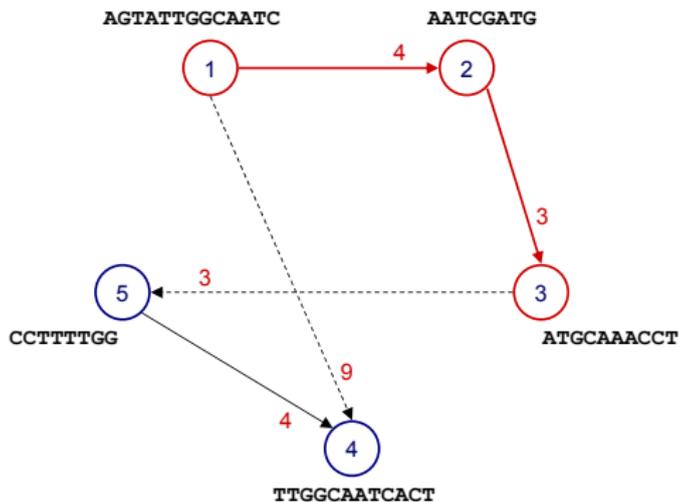
AGTATTGGCAATC

# Ordenação Topológica



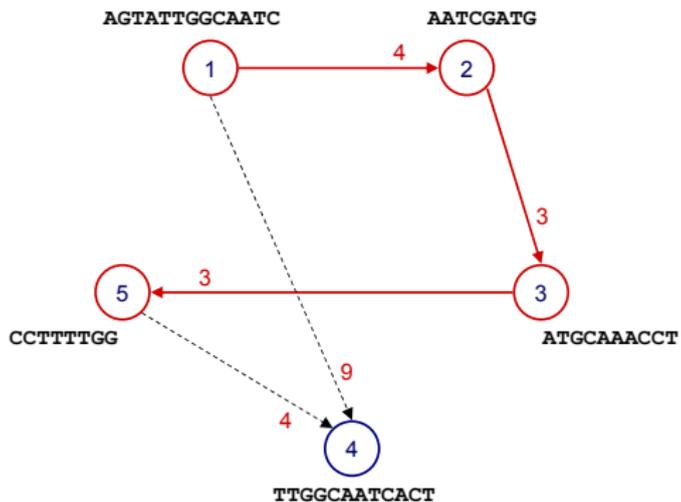
AGTATTGGCAATCGATG

# Ordenação Topológica



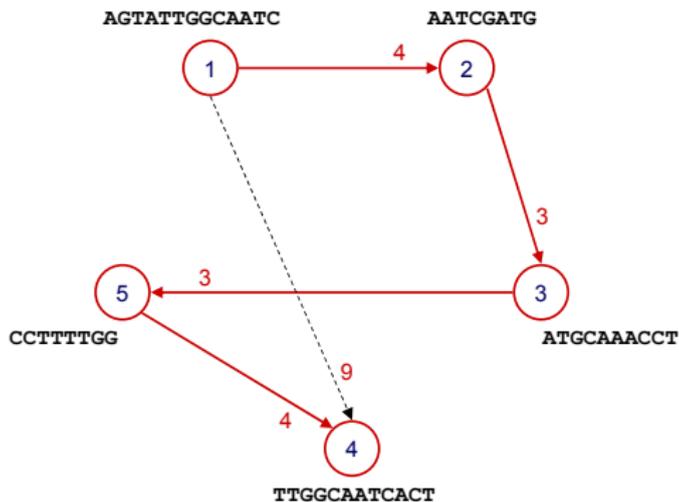
AGTATTGGCAATCGATGCAAACCT

# Ordenação Topológica



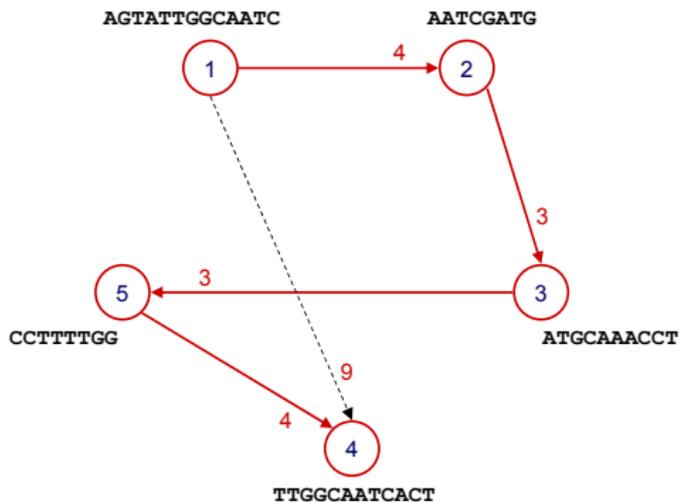
AGTATTGG**CAATCGATG**CAAACCTTTTGG

# Ordenação Topológica



AGTATTGGCA**AATCGATG**CAAA**CCTTTTGG**CAATCACT

# Ordenação Topológica

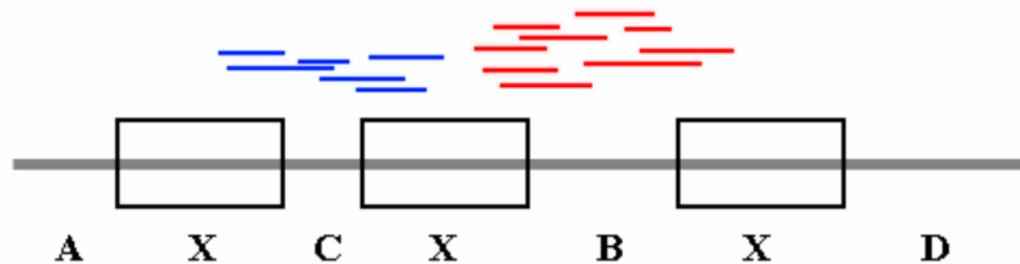
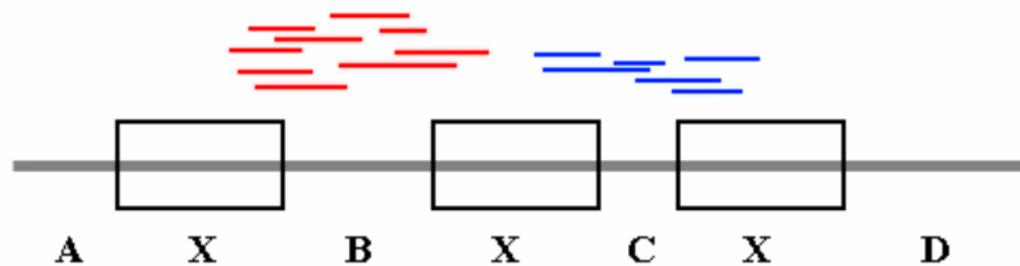


AGTATTGGCAATCGATGCAAACCTTTGGCAATCACT

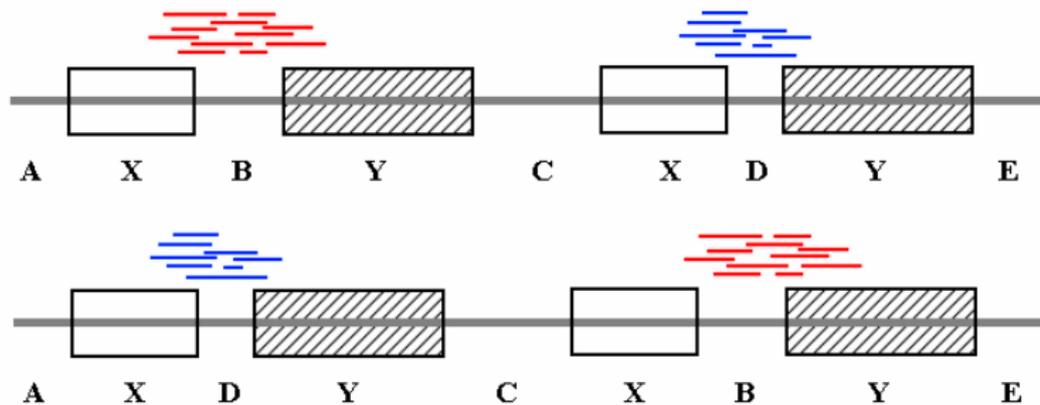
# Problemas com Repetições

---

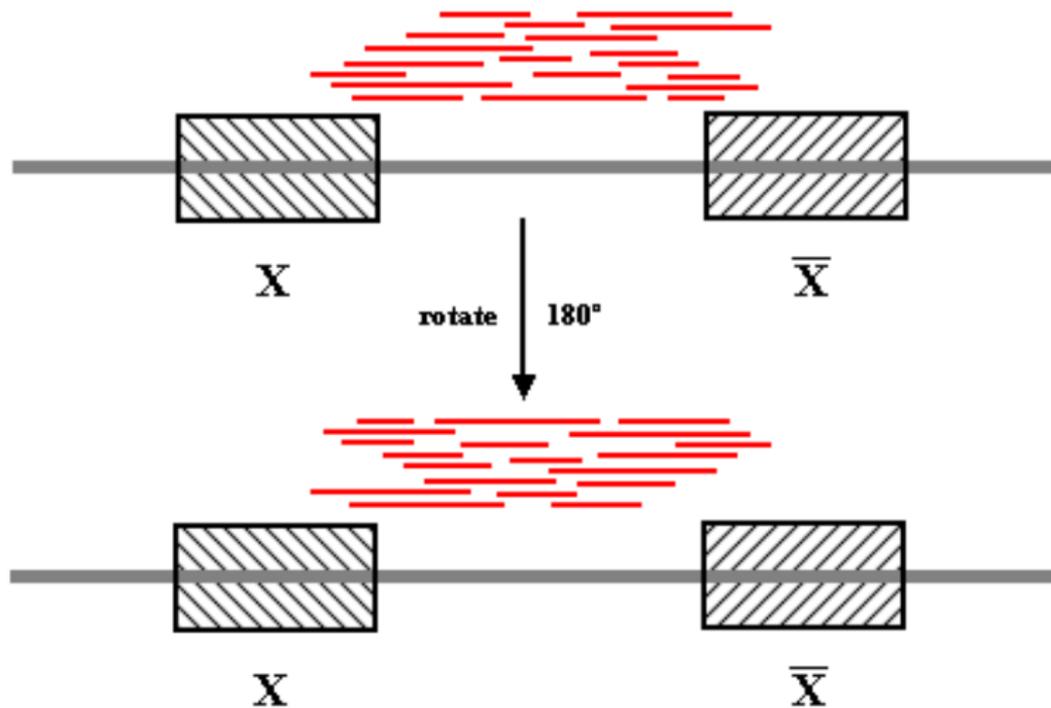
## Problemas com Repetições



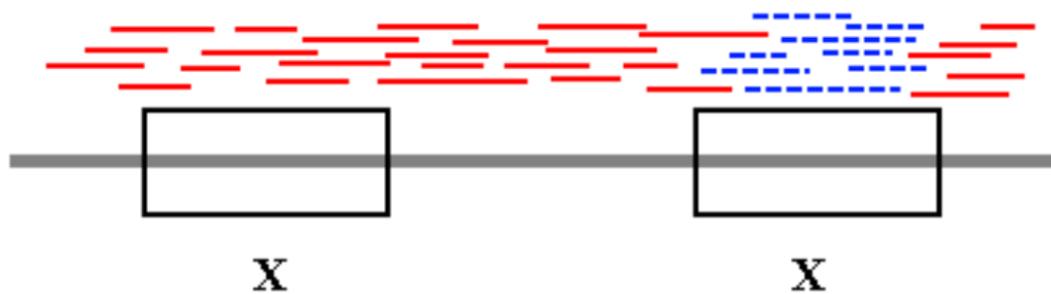
# Problemas com Repetições



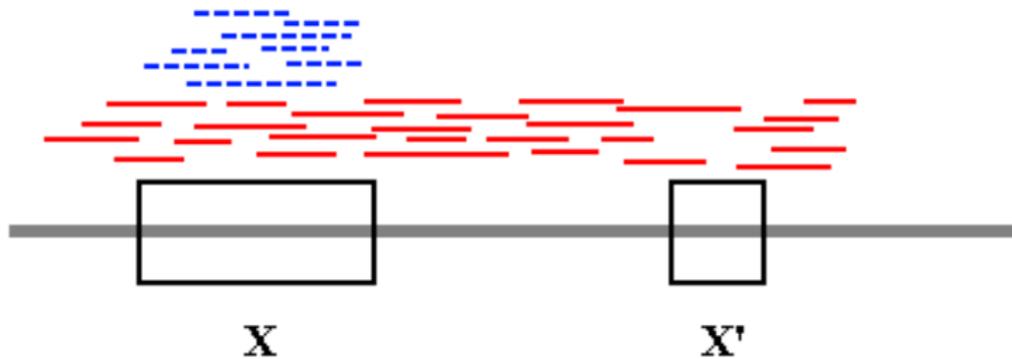
## Problemas com Repetições



## Problemas com Repetições



## Problemas com Repetições



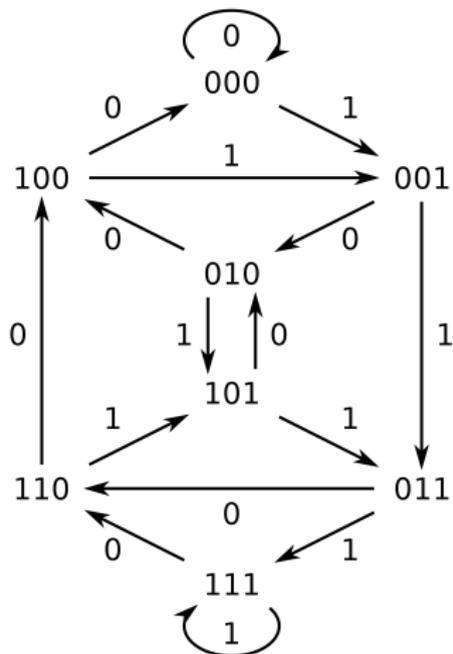
# Montagem de Fragmentos usando Grafos de $k$ -Mers

---

# Grafo de Bruijn

- Dado um alfabeto  $\mathcal{A}$  qualquer, um  $k$ -mer de  $\mathcal{A}$  é definido com uma sequência de  $k$  caracteres de  $\mathcal{A}$ .
- O grafo de Bruijn de ordem  $k$  é um grafo orientado cujos vértices são todos os  $k$ -mers de  $\mathcal{A}$  e existe uma aresta entre dois vértices  $x$  e  $y$  se e somente se os  $k - 1$  últimos caracteres de  $x$  forem iguais aos  $k - 1$  primeiros caracteres de  $y$ .
- Note que o grafo de Bruijn de ordem  $k$  possui  $|\mathcal{A}|^k$  vértices e  $|\mathcal{A}|^{k+1}$  arestas.
- O grafo de Bruijn possui algumas características interessantes. Por exemplo, ele é tanto hamiltoniano (admite circuito hamiltoniano) como euleriano (admite circuito euleriano).
- As novas técnicas de sequenciamento possuem algumas características importantes, como gerar fragmentos de tamanho fixo e (relativamente) pequenos.

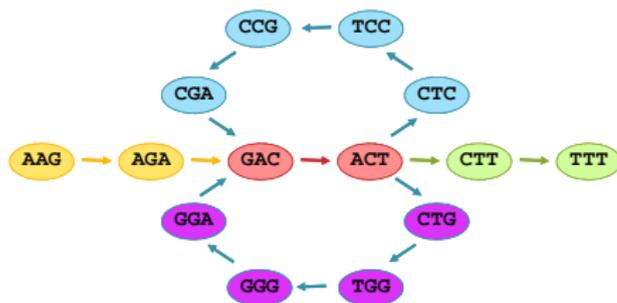
# Grafo de Bruijn



## Montagem de Fragmentos usando Grafos de $k$ -Mers

- Sendo assim, dado um conjunto de fragmentos  $\mathcal{F}$  sobre o alfabeto  $\mathcal{A}$ , todos de tamanho  $k + 1$ , podemos gerar um grafo de  $k$ -mers  $G(\mathcal{F})$ , similar ao grafo de Bruijn: para cada  $f \in \mathcal{F}$  adicionamos os vértices  $x$  e  $y$  (casos eles ainda não existam) e a aresta  $(x, y)$  tal que  $x$  é o vértice que representa o prefixo de tamanho  $k$  de  $f$  e  $y$  é o vértice que representa o sufixo de tamanho  $k$  de  $f$ .
- Neste caso, uma montagem dos fragmentos corresponde a um passeio que contenha todas as arestas deste grafo.
- Um grafo orientado admite um passeio que contem todas as arestas do grafo se e somente se o grafo reduzido (ou seja, o grafo onde todas as componentes fortemente conexas são reduzidas a um único vértice) for um grafo caminho.
- Usando a abordagem descrita acima é possível testar se um grafo possui um passeio que contem todas as arestas do grafo em tempo polinomial.

# Montagem de Fragmentos usando Grafos de $k$ -Mers



**Fragmentos:**

AAGA	TGGG	TCCG
AGAC	GGGA	CCGA
GACT	GGAC	CGAC
ACTG	ACTC	ACTT
CTGG	CTCC	CTTT

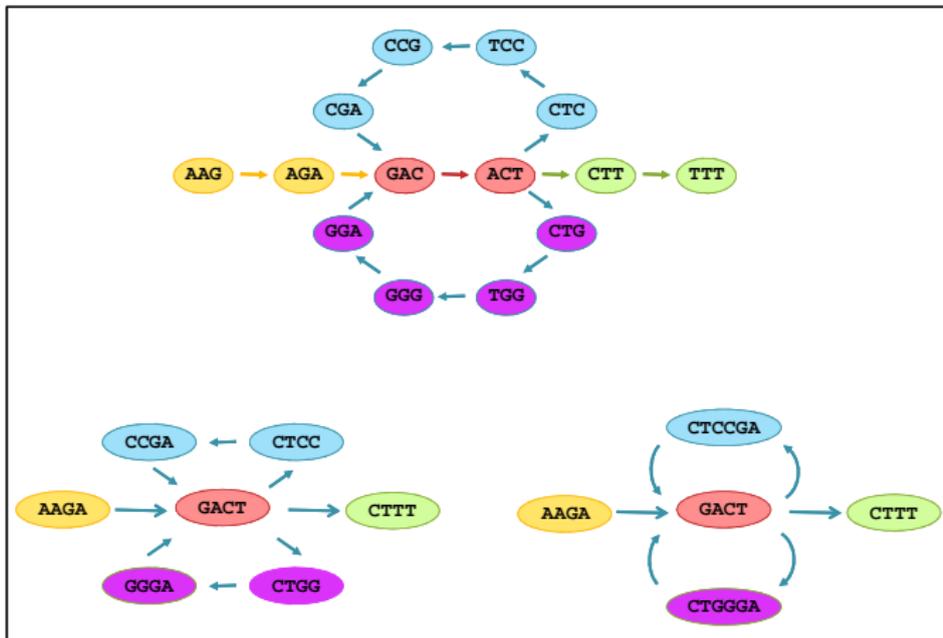
**Montagens:**

AAGACTCCGACTGGGACTTT
AAGACTGGGACTCCGACTTT

## Montagem de Fragmentos usando Grafos de $k$ -Mers

- O valor de  $k$  usado para definir o tamanho dos fragmentos tem um forte impacto na montagem:
  - Quanto menor o valor de  $k$ , maiores e mais frequentes serão os problemas com repetições.
  - Quanto maior o valor de  $k$ , mais difícil será o grafo conter um passeio que contenha todas as arestas.
- Dado um conjunto de fragmentos  $\mathcal{F}$ , tal que para todo  $f \in \mathcal{F}$  temos que  $|f| \geq k$ , podemos obter um conjunto de fragmentos  $\mathcal{F}'$ , todos de tamanho  $k$ , equivalente em termos da montagem desejada. Para isso, basta, para cada  $f \in \mathcal{F}$ , adicionar em  $\mathcal{F}'$  todas as  $|f| - k + 1$  subcadeias de  $f$  de tamanho  $k$ .
- Note que as subsequências de  $f$  irão gerar um passeio em  $G(\mathcal{F}')$ .
- Para diminuir o tamanho do grafo  $G(\mathcal{F})$ , podemos transformar todo caminho simples (sem bifurcações) em um único vértice.

# Montagem de Fragmentos usando Grafos de $k$ -Mers



## Montagem de Fragmentos usando Grafos de $k$ -Mers

- Como vimos anteriormente, existem várias formas diferentes de modelar o problema de montagem de fragmentos.
- Por exemplo, ao modelar o problema de montagem de fragmentos como SCS (Shortest Common Superstring) conseguimos montar um grafo de sobreposições em tempo e espaço  $O(\|\mathcal{F}\| + n^2)$ . Nesse caso desejamos encontrar um caminho hamiltoniano no grafo de sobreposições (problema NP-Difícil).
- Por outro lado, ao modelar o problema de montagem de fragmentos usando grafo de  $k$ -mers conseguimos construir o grafo de  $k$ -mers em tempo e espaço  $O(\|\mathcal{F}\| + k \times |\mathcal{A}|^k)$ . Neste caso desejamos encontrar um passeio que contenha todas as arestas do grafo de  $k$ -mers (problema que possui solução polinomial no tamanho do grafo, caso exista tal passeio).

## Exercício

*Dada uma coleção de fragmentos  $\mathcal{F}$  sobre um alfabeto  $\mathcal{A}$ , mostre como construir o grafo de  $k$ -mers em tempo e espaço  $O(\|\mathcal{F}\| + k \times |\mathcal{A}|^k)$ .*

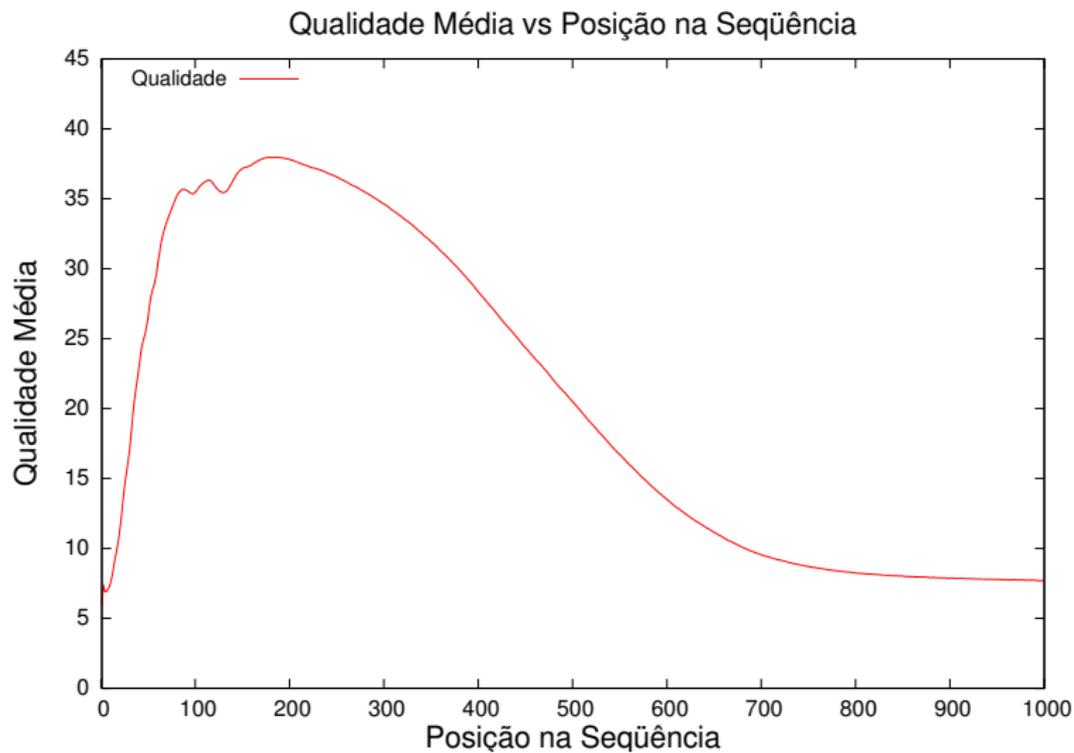
## Exercício

*Dada a coleção de fragmentos  $\mathcal{F} = \{\text{CTGCT}, \text{CTCGAC}, \text{CTCTCG}, \text{ACTGC}, \text{GCTCTC}\}$ , construa o grafo de 3-mers e determine a montagem destes fragmentos.*

## Phred, Phrap e Consed

---





- Ferramenta de *base-calling* produzida por Phil Green, Brent Ewing, LaDeana Hillier e Michael Wendl (1998).
- O método é composto por 4 fases:
  - Predição das localizações dos picos.
  - Identificação dos picos observados.
  - Comparação entre os picos previstos e observados.
  - Verificação dos picos observados que não são compatíveis com os picos previstos.

- Phred associa um valor de qualidade para cada base da sequência lida:

$$Q = -10 \times \log_{10} P_e$$

onde  $P_e$  é a probabilidade da base estar errada.

- Exemplo:
  - $Q = 10 \implies P_e = 10\%$
  - $Q = 20 \implies P_e = 1\%$
  - $Q = 30 \implies P_e = 0.1\%$
- Phred pode ser usado para remover pontas de baixas qualidades.

- Ferramenta de montagem de sequências produzida por Phil Green (1998).
- Principais características:
  - Usa a sequência inteira, não apenas os trechos de alta qualidade.
  - Usa a qualidade das sequências para obter uma montagem de alta qualidade.
  - Constrói os consensos dos contigs como um mosaico das partes de mais alta qualidade das sequências.
  - Atribui valores de qualidade para as sequências consenso.
  - Faz comparação entre as sequências usando uma variação do algoritmo de Smith-Waterman, onde as comparações são iniciadas apenas se existir um trecho idêntico de tamanho mínimo (por padrão 30), em ambas as sequências. A extensão do alinhamento é realizada usando apenas uma faixa restrita da matriz de Programação Dinâmica (por padrão, faixa de tamanho 14).

- Ferramenta de visualização e edição de montagens de sequências, com suporte a “fechamento” de montagem, desenvolvida por David Gordon, Chris Abajian e Phil Green (1998).
- Desenvolvido originalmente para dar suporte apenas ao Phrap.
- Hoje suporta uma vasta gama de montadores (que produzem arquivos no formato **ace**, lidos pelo Consed), inclusive os montadores desenvolvidos para as novas tecnologias 454 e Solexa (de sequências curtas e muitas curtas).

CAP3

---

- Ferramenta de montagem de sequências produzida por Xiaoqiu Huang e Anup Madan (1999).
- Passos principais:
  - Remoção das extremidades de baixa qualidade.
  - Identificação das sobreposição entre as sequências.
  - Remoção das falsas sobreposições.
  - Construção dos *contigs*.
  - Alinhamento múltiplo e geração da sequência consenso, considerando as somas das qualidades das bases de cada coluna.

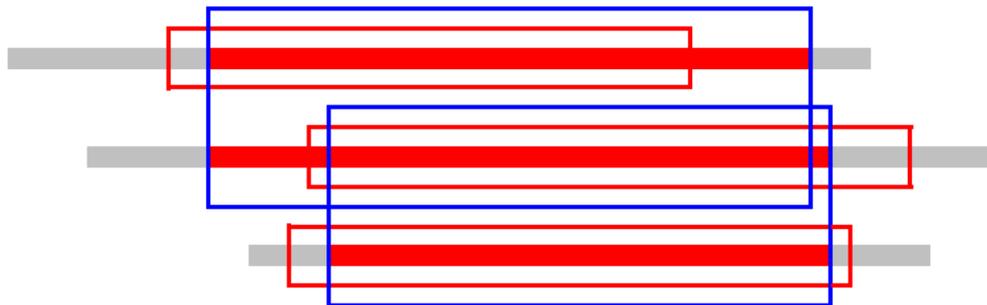
- Sobreposições identificadas em duas fases:
  1. Alinhamento local ponderado restrito a uma faixa de tamanho  $k$ :
    - $Match' = Match \times \min(q_1, q_2)$
    - $Mismatch' = Mismatch \times \min(q_1, q_2)$
    - $Gap' = Gap \times \min(q_1, q_2)$
  2. Alinhamento Global, restrito a uma faixa de tamanho  $2k$ , centralizado na posição inicial do alinhamento local ótimo calculado previamente.
- As distâncias mínimas e máximas entre cada par de sequências forward e reverse são usadas para auxiliar na montagem.

# CAP3 - Remoção de Extremidades de Baixa Qualidades



 Sequência Original	 Sequência Final	 Alta Similaridade	 Alta Qualidade
--	---	---	---

# CAP3 - Remoção de Extremidades de Baixa Qualidades



 Sequência Original	 Sequência Final	 Alta Similaridade	 Alta Qualidade
--	---	---	---

- Calcula-se a soma ponderada das qualidades de cada um dos tipos de bases presentes na coluna.
- Considera-se peso 1 para cada pontuação máxima (em cada um dos sentidos de leitura) e 0,5 para as demais qualidades.
- Para gaps, usa-se a pontuação média das bases que delimitam o bloco de gaps.
- A base de maior soma ponderada de qualidade é a escolhida para o consenso.

## CAP3 - Construção da Sequência Consenso

- A qualidade do consenso é calculada como a diferença entre a soma ponderada das qualidades da base escolhida subtraída das somas ponderadas das qualidades das demais bases daquela mesma coluna.
- Eventualmente, a base do consenso pode ter qualidade zero, indicando que as somas ponderadas das qualidades das demais bases possui soma maior ou igual a da base escolhida para o consenso.
- O CAP3 geralmente produz *contigs* mais curtos, porém de maior qualidade, quando comparados com os *contigs* gerados pelo Phrap.

# CAP3 - Construção da Sequência Consenso

Consenso	A <b>35</b>			T <b>5</b>		
	Base	Qual	Peso	Base	Qual	Peso
→	A	<b>30</b>	1	A	<b>30</b>	1
→	A	<b>20</b>	0,5	T	<b>30</b>	1
→	C	<b>10</b>	1	T	<b>20</b>	0,5
→	A	<b>20</b>	0,5	A	<b>20</b>	0,5
←	A	<b>20</b>	1	A	<b>20</b>	1
←	A	<b>10</b>	0,5	A	<b>10</b>	0,5
←	T	<b>30</b>	1	T	<b>30</b>	1
	→	←	Total	→	←	Total
A	50	25	75	40	25	65
C	10		10			0
T		30	30	40	30	70
G			0			0