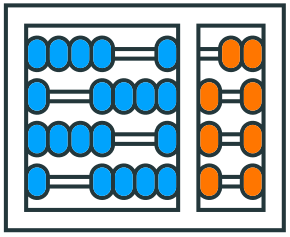


Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

Rearranjos de Genomas com Genes Replicados

MO640 - Biologia Computacional / MC668 - Bioinformática

Gabriel Siqueira

2024

Instituto de Computação

Roteiro

Rearranjos em Permutações

Representação dos Genomas

Aproximação

Partição de Strings

Strings não Balanceadas

Representações com Mais Informações Biológicas

Rearranjos em Permutações

Rearranjos em Permutações

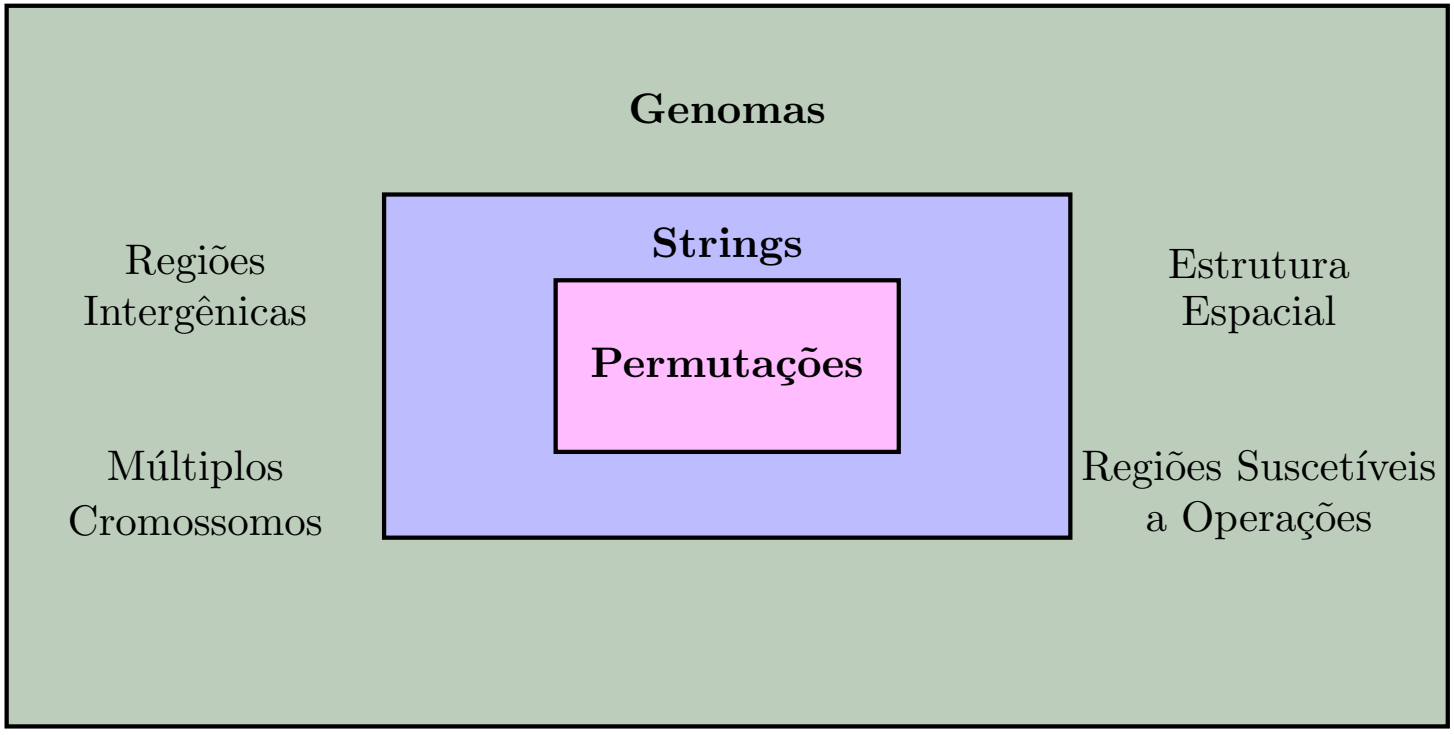
- Representação do genoma considerando blocos conservados de genes.
- O objetivo é transformar um genoma em outro com operações de rearranjos (reversão, transposição, transposição reversa, fissão, fusão, translocação, inserção, remoção, duplicação, etc).
- Podemos considerar ou não a orientação dos genes.
- Com apenas um cromossomo e sem genes repetidos uma permutação π pode ser usada para representar um genoma.
- Nesse caso em particular, o problema pode ser visto como um problema de ordenação.

Rearranjos em Permutações

- Considerando breakpoints e strips existe uma algoritmo com fator de aproximação 2 para o problema de ordenação de permutações sem sinais por reversões.
- Incluindo informações sobre pares ordenados e hurdles, existe um algoritmo exato para o problema de ordenação de permutações com sinais por reversões.

Representação dos Genomas

Representação dos Genomas



Representação dos Genomas

- Vamos agora assumir que os genomas podem ter genes repetidos e que os conjuntos de genes dos dois genomas podem ser diferentes.
- Strings podem representar genes repetidos.
- Assim como em permutações, podemos usar sinais para representar as orientações dos genes.
- Vamos considerar que as strings estão estendidas com a adição de caracteres no começo e no final, tal que cada um desses caracteres apresente apenas uma cópia.
- A ocorrência de um caractere é a quantidade de vezem em que esse caractere aparece.

$$S = \overbrace{+0 \quad +1 \quad -2 \quad -2 \quad -1 \quad -2 \quad +3 \quad +\infty}$$

Representação dos Genomas

- Um valor importante para nós é a ocorrência máxima em uma string, que é o maior valor de ocorrência considerando todos os caracteres.
- Dizemos que duas strings são balanceadas se elas possuem o mesmo conjunto de caracteres e a ocorrência de cada caractere é a mesma.
- Ao comparar strings balanceadas podemos usar apenas operações conservativas.
- Se tivermos strings não balanceadas devemos usar operações não conservativas e nesse caso normalmente assumimos que devemos adicionar ou remover apenas os caracteres necessários.

$$S = \overbrace{+0 \quad +1 \quad -2 \quad -2 \quad -1 \quad -2 \quad +3 \quad +\infty}$$

$$P = \overbrace{+0 \quad +2 \quad +2 \quad -1 \quad -2 \quad +3 \quad +1 \quad +\infty}$$

Representação dos Genomas

- Um valor importante para nós é a ocorrência máxima em uma string, que é o maior valor de ocorrência considerando todos os caracteres.
- Dizemos que duas strings são balanceadas se elas possuem o mesmo conjunto de caracteres e a ocorrência de cada caractere é a mesma.
- Ao comparar strings balanceadas podemos usar apenas operações conservativas.
- Se tivermos strings não balanceadas devemos usar operações não conservativas e nesse caso normalmente assumimos que devemos adicionar ou remover apenas os caracteres necessários.

$$S = \overbrace{+0 \quad +4 \quad -1 \quad -2 \quad -2 \quad -1 \quad -4 \quad +\infty}$$

$$P = \overbrace{+0 \quad +1 \quad +2 \quad -2 \quad +3 \quad +2 \quad +4 \quad +\infty}$$

Problemas de Rearranjos

- Considerando o evento de reversão e strings balanceadas temos os problemas:
  - Distância de Reversão em strings balanceadas sem sinais.
  - Distância de Reversão em strings balanceadas com sinais.
- Ambos são problemas $\mathcal{NP}$-Difíceis.

$$
\begin{aligned}
S = &\ (+0\ \ -2\ \ -2\ \ -1\ \ +3\ \ -2\ \ -4\ \ +\infty) \\
&\ (+0\ \ +2\ \ -3\ \ +1\ \ +2\ \ +2\ \ -4\ \ +\infty) \\
&\ (+0\ \ +2\ \ -1\ \ +3\ \ +2\ \ +2\ \ -4\ \ +\infty) \\
P = &\ (+0\ \ +2\ \ -1\ \ +3\ \ +2\ \ +2\ \ +4\ \ +\infty)
\end{aligned}
$$

$$d_R(S, P) = 3$$

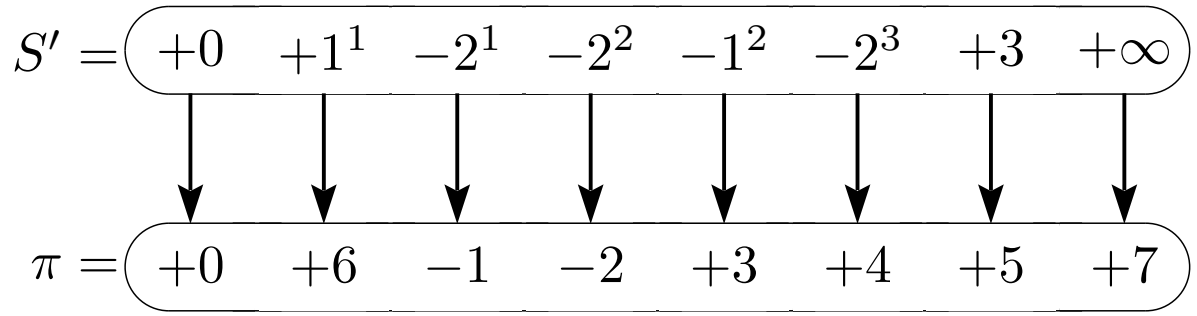
Aproximação

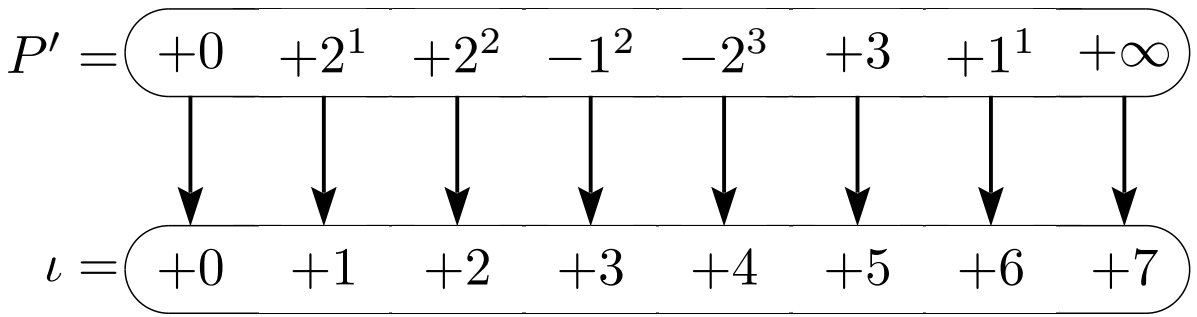
Aproximação

- Para obter uma aproximação, precisamos de um limitante inferior e de um limitante superior.
- Para permutações e a operação de reversão temos:
  - Limitante inferior: número minimo de operações necessárias para remover todos os breakpoints ($\frac{b(\pi)}{2}$).
  - Limitante superior: sequência de operação removendo 1 breakpoint em média.
  - Considerando esse limitantes temos uma aproximação com fator 2.

Mapeamento

- Vamos mapear uma string em uma permutação e gerar uma aproximação com os limitantes inferiores e superiores para permutações.





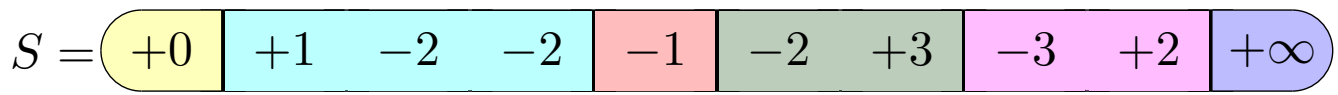
Mapeamento

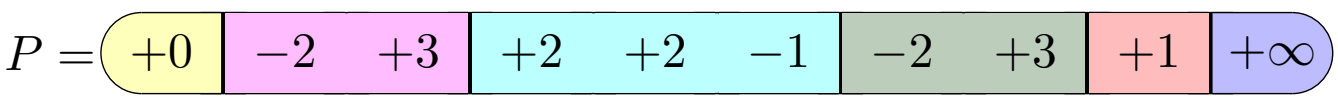
- Para adaptar os limitantes queremos um mapeamento que minimiza o número de breakpoints.
- Seja $b_{min}(S)$ a quantidade de breakpoints desse mapeamento.
- Serão necessárias pelo menos $\frac{b_{min}(S)}{2}$ reversões para transformar S em P.
- Usando esse mapeamento para encontrar as reversões podemos ordenar com $b_{min}(S)$ breakpoints em média e garantir uma aproximação com fator de 2.
- Como gerar esse mapeamento que minimiza o número de breakpoints?

Partição de Strings

Partição de Strings

- Generalização da ideia de breakpoints.
- Vamos encontrar um conjunto de substrings que podem formar as strings S e P se concatenadas com duas ordens.
- As substrings podem ser invertidas (os sinais são invertidos em strings com sinais)
- O objetivo é encontrar o menor conjunto possível.
- Podemos assumir que no conjunto de substrings não temos duas substrings consecutivas com as mesmas orientações nas duas ordens (pois nesse caso poderíamos substituí-las por uma única substring).





Partição de Strings

- Um mapeamento é compatível com uma partição se ele respeita as substrings encontradas.
- O número de breakpoints em uma permutação gerada por um mapeamento é igual ao tamanho de uma partição compatível com ele menos 1 (as substrings da partição correspondem as strips dessa permutação).

$S =$ | $+0$ | $+1^1$ -2^1 -2^2 | -1^2 | -2^3 $+3^1$ | -3^2 $+2^4$ | $+\infty$ |

$P =$ | $+0$ | -2^4 $+3^2$ | $+2^2$ $+2^1$ -1^1 | -2^3 $+3^1$ | $+1^2$ | $+\infty$ |

Aproximação com Partição de Strings

- Se pudermos encontrar a partição de tamanho minimo podemos manter a aproximação 2 para os problemas de distância de reversão.
- Infelizmente encontrar uma partição de tamanho minimo é um problema $\mathcal{NP}$-Difícil.
- Portanto vamos buscar uma aproximação ℓ para o problema de partição.
- Usando uma partição obtida por essa aproximação podemos garantir uma 2ℓ aproximação para os problemas de distância de reversão (o limitante inferior é $\frac{b_{min}(S)}{2}$ e o limitante superior é $\ell b_{min}(S)$).

Algoritmo Guloso

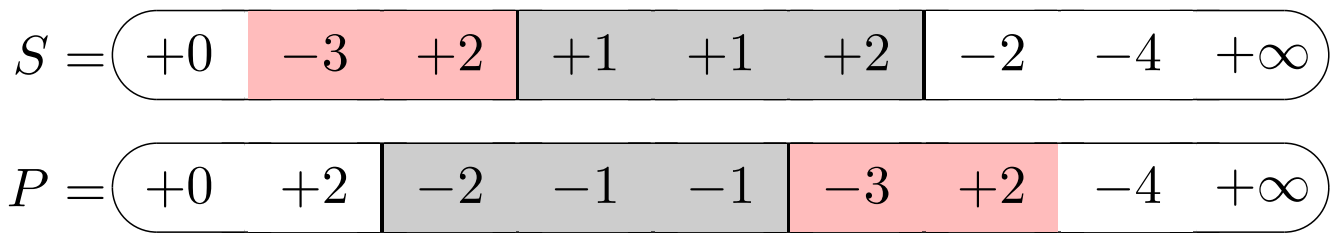
- Começaremos com uma estratégia gulosa pegando sempre a maior substring comum entre as strings S e P.
- A cada iteração essa substring deve ser encontrada entre os caracteres não selecionados.
- Como encontrar a maior substring comum?
- A melhor prova de aproximação para esse algoritmo um fator $O(n^{0.69})$, para strings de tamanho n.
- O melhor algoritmo dependente de n tem um fator de aproximação $O(\log n \log^* n)$.

$$S = (+0 \quad -3 \quad +2 \quad +1 \quad +1 \quad +2 \quad -2 \quad -4 \quad +\infty)$$

$$P = (+0 \quad +2 \quad -2 \quad -1 \quad -1 \quad -3 \quad +2 \quad -4 \quad +\infty)$$

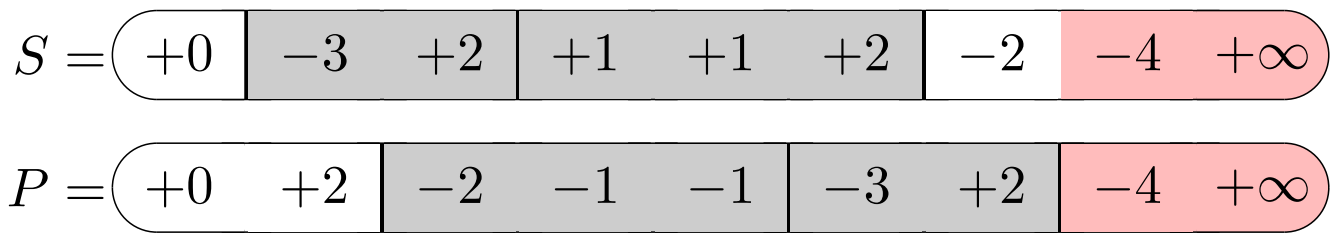
Algoritmo Guloso

- Começaremos com uma estratégia gulosa pegando sempre a maior substring comum entre as strings S e P.
- A cada iteração essa substring deve ser encontrada entre os caracteres não selecionados.
- Como encontrar a maior substring comum?
- A melhor prova de aproximação para esse algoritmo um fator $O(n^{0.69})$, para strings de tamanho n.
- O melhor algoritmo dependente de n tem um fator de aproximação $O(\log n \log^* n)$.



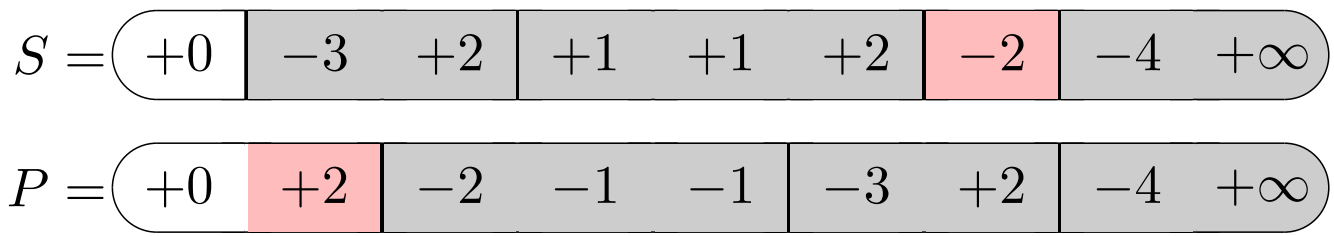
Algoritmo Guloso

- Começaremos com uma estratégia gulosa pegando sempre a maior substring comum entre as strings S e P.
- A cada iteração essa substring deve ser encontrada entre os caracteres não selecionados.
- Como encontrar a maior substring comum?
- A melhor prova de aproximação para esse algoritmo um fator $O(n^{0.69})$, para strings de tamanho n.
- O melhor algoritmo dependente de n tem um fator de aproximação $O(\log n \log^* n)$.



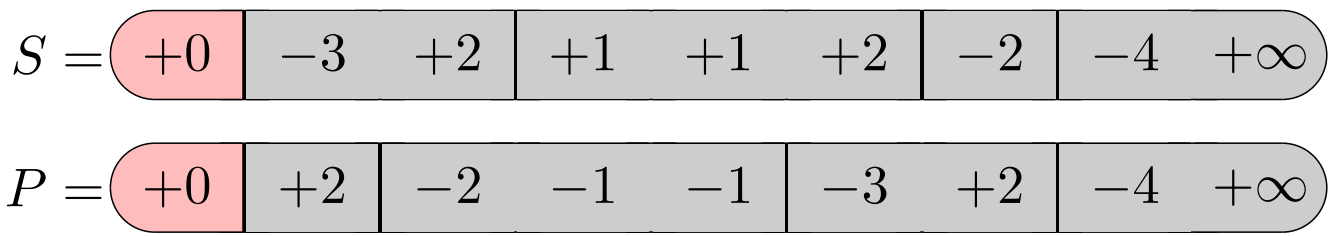
Algoritmo Guloso

- Começaremos com uma estratégia gulosa pegando sempre a maior substring comum entre as strings S e P.
- A cada iteração essa substring deve ser encontrada entre os caracteres não selecionados.
- Como encontrar a maior substring comum?
- A melhor prova de aproximação para esse algoritmo um fator $O(n^{0.69})$, para strings de tamanho n.
- O melhor algoritmo dependente de n tem um fator de aproximação $O(\log n \log^* n)$.



Algoritmo Guloso

- Começaremos com uma estratégia gulosa pegando sempre a maior substring comum entre as strings S e P.
- A cada iteração essa substring deve ser encontrada entre os caracteres não selecionados.
- Como encontrar a maior substring comum?
- A melhor prova de aproximação para esse algoritmo um fator $O(n^{0.69})$, para strings de tamanho n.
- O melhor algoritmo dependente de n tem um fator de aproximação $O(\log n \log^* n)$.



Algoritmo Guloso

- Começaremos com uma estratégia gulosa pegando sempre a maior substring comum entre as strings S e P.
- A cada iteração essa substring deve ser encontrada entre os caracteres não selecionados.
- Como encontrar a maior substring comum?
- A melhor prova de aproximação para esse algoritmo um fator $O(n^{0.69})$, para strings de tamanho n.
- O melhor algoritmo dependente de n tem um fator de aproximação $O(\log n \log^* n)$.

$$S = \boxed{+0 \mid -3 \; +2 \mid +1 \; +1 \; +2 \mid -2 \mid -4 \; +\infty}$$

$$P = \boxed{+0 \mid +2 \mid -2 \; -1 \; -1 \mid -3 \; +2 \mid -4 \; +\infty}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Existe um algoritmo que garante um fator de aproximação $2k$ para o problema de partição (k é a ocorrência máxima de um caractere nas strings).
- Usando uma partição obtida por essa aproximação podemos garantir uma $4k$ aproximação para os problemas de distância de reversão.
- O algoritmo se baseia na ideia de adicionar breakpoints que estejam conectados com breakpoints necessários para a partição.

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \overbrace{+0 \quad -2 \quad -1 \quad -2 \quad +3 \quad -2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \overbrace{+0 \quad -2 \quad +3 \quad +1 \quad +2 \quad -2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0-2-1, -1\ -2, \ +3\ -2, \\ +0\ -2\ +3, +3\ +1, +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = (+0 \quad -2 \quad -1 \quad -2 \quad +3 \quad -2 \quad +3 \quad +\infty)$$

$$\mathcal{H} = (+0 \quad -2 \quad +3 \quad +1 \quad +2 \quad -2 \quad +3 \quad +\infty)$$

$$T_{min} = \{+0-2-1, -1\ -2,\ +3\ -2,\ +0\ -2\ +3, +3\ +1, +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = (+0 \quad -2 \quad -1 \mid -2 \quad +3 \quad -2 \quad +3 \quad +\infty)$$

$$\mathcal{H} = (+0 \quad -2 \quad +3 \quad +1 \quad +2 \quad -2 \quad +3 \quad +\infty)$$

$$T_{min} = \{+0-2-1, -1\ -2,\ +3\ -2,\ +0\ -2\ +3, +3\ +1, +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2 \quad -1 \mid -2 \quad +3 \quad -2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3 \quad +1 \quad +2 \quad -2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0-2-1,\ +3\ -2,\ +0\ -2\ +3,\ +3\ +1,\ +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2 \quad -1 \mid -2 \quad +3 \mid -2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3 \quad +1 \quad +2 \quad -2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0-2-1,\ +3\ -2,\ +0\ -2\ +3,\ +3\ +1,\ +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2 \quad -1 \mid -2 \quad +3 \mid -2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3 \quad +1 \quad +2 \quad -2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 - 2 - 1,\ +0\ -2\ +3,\ +3\ +1,\ +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2 \quad -1 \mid -2 \quad +3 \mid -2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3 \mid +1 \quad +2 \quad -2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0-2-1,\ +0\ -2\ +3,\ +3\ +1,\ +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2 \quad -1 \mid -2 \quad +3 \mid -2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3 \mid +1 \quad +2 \quad -2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 - 2 - 1,\ +0\ -2\ +3,\ +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2 \quad -1 \mid -2 \quad +3 \mid -2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3 \mid +1 \quad +2 \mid -2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 - 2 - 1,\ +0\ -2\ +3,\ +2\ -2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2 \quad -1} \,|\, \boxed{-2 \quad +3} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3} \,|\, \boxed{+1 \quad +2} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 - 2 - 1,\ +0\ -2\ +3\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2} \,|\, \boxed{-1} \,|\, \boxed{-2 \quad +3} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3} \,|\, \boxed{+1 \quad +2} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 - 2 - 1,\ +0\ -2\ +3\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2} \,|\, \boxed{-1} \,|\, \boxed{-2 \quad +3} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3} \,|\, \boxed{+1 \quad +2} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 \ -2 \ +3, +1 \ +2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2} \boxed{-1} \boxed{-2 \quad +3} \boxed{-2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3} \boxed{+1} \boxed{+2} \boxed{-2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 \ -2 \ +3, +1 \ +2\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2} \; \boxed{-1} \; \boxed{-2 \quad +3} \; \boxed{-2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2 \quad +3} \; \boxed{+1} \; \boxed{+2} \; \boxed{-2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 \;\; -2 \;\; +3\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \boxed{+0 \quad -2} \,|\, \boxed{-1} \,|\, \boxed{-2 \quad +3} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2} \,|\, \boxed{+3} \,|\, \boxed{+1} \,|\, \boxed{+2} \,|\, \boxed{-2 \quad +3 \quad +\infty}$$

$$T_{min} = \{+0 \;\; -2 \;\; +3\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

$$\mathcal{G} = \left(+0 \quad -2 \;\middle|\; -1 \;\middle|\; -2 \quad +3 \;\middle|\; -2 \quad +3 \quad +\infty \right)$$

$$\mathcal{H} = \left(+0 \quad -2 \;\middle|\; +3 \;\middle|\; +1 \;\middle|\; +2 \;\middle|\; -2 \quad +3 \quad +\infty \right)$$

$$T_{min} = \{-2 \;\; +3\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.

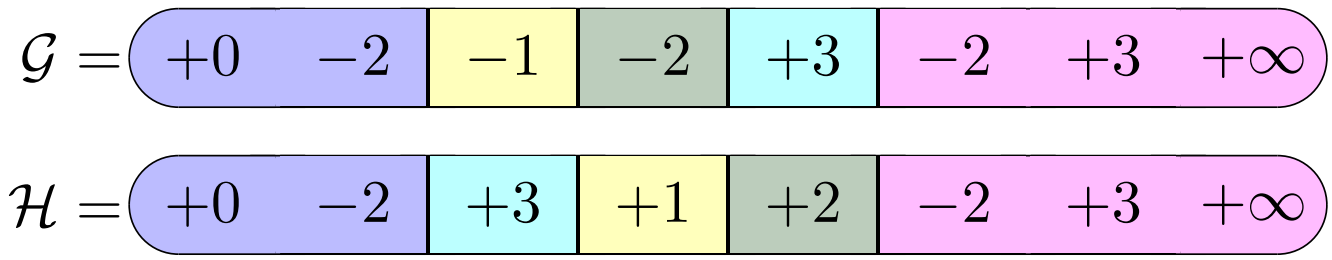
$$\mathcal{G} = \boxed{+0 \quad -2} \; \boxed{-1} \; \boxed{-2} \; \boxed{+3} \; \boxed{-2 \quad +3 \quad +\infty}$$

$$\mathcal{H} = \boxed{+0 \quad -2} \; \boxed{+3} \; \boxed{+1} \; \boxed{+2} \; \boxed{-2 \quad +3 \quad +\infty}$$

$$T_{min} = \{-2 \;\; +3\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Seja T o conjunto de todas as substrings com ocorrências distintas nas strings S e P. Temos $T_{min} = \{X \in T \,|\, Y \not\subset X, \forall Y \in T, Y \neq X\}$.
- Note que devemos ter pelo menos um breakpoint em cada substring de T_{min}.
- O algoritmo seleciona um breakpoint em uma das menores substrings de T_{min}.
- Os demais breakpoints são adicionados de forma similar com alguns cuidados para limitar o número de breakpoints ao final. O conjunto T_{min} deve ser atualizado a cada passo.



$$T_{min} = \{\}$$

Verificando Breakpoints Necessários (Aproximação de $2k$)

- Que cuidados devemos ter ao adicionar os breakpoints?
- Sabemos que é necessário um breakpoint para cada elemento de T_{min}.
- Um elemento de T_{min} pode ter mais de uma cópia nas strings.
- Ao adicionar breakpoints novos, podemos adicionar mais elementos a T_{min} que não têm breakpoints necessários.
- Quando possível, vamos garantir que cada breakpoint adicionado seja entre um par de caracteres que seja igual a um par de caracteres de um breakpoint já existente.
- Como temos no máximo $2k$ cópias de cada caractere (k em S e k em P), o número de breakpoints adicionados é limitado a $2k|T_{min}|$.
- Um outro cuidado que devemos ter é analisar as interseções entre elementos de T_{min} para lidar com casos onde dois elementos têm o mesmo breakpoint necessário.

Strings não Balanceadas

Rearranjos não Conservativos

- Quando os conjuntos de genes nos dois genomas são distintos as strings correspondentes não são mais balanceadas.
- Nesse caso os problemas de rearranjo devem envolver eventos não conservativos como inserção e remoção (indels).
- Problemas com eventos não conservativos incluem:
  - Distância de Reversão e Indel em strings sem sinais.
  - Distância de Reversão e Indel em strings com sinais.

Rearranjos não Conservativos

$$S = (+0 \quad +4 \quad -1 \quad -2 \quad -2 \quad -1 \quad -4 \quad +\infty)$$

$$(+0 \quad -2 \quad -2 \quad -1 \quad -4 \quad +\infty)$$

$$(+0 \quad +1 \quad +2 \quad +2 \quad -4 \quad +\infty)$$

$$(+0 \quad +1 \quad +2 \quad +2 \quad +4 \quad +\infty)$$

$$P = (+0 \quad +1 \quad +2 \quad -2 \quad +3 \quad +2 \quad +4 \quad +\infty)$$

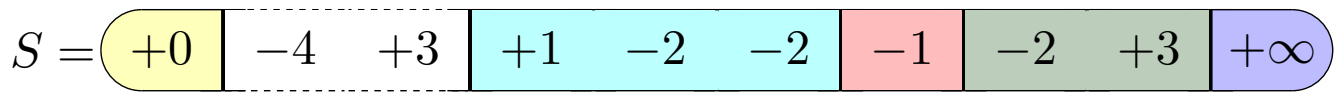
$$\boxed{d_{RI}(S, P) = 4}$$

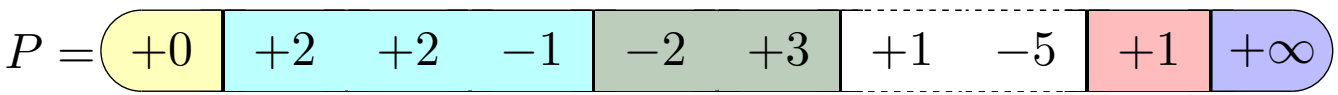
Rearranjos não Conservativos

- Quando não consideramos genes repetidos:
  - Existe um algoritmo com fator de aproximação 2 para o problema sem sinais. Esse algoritmo usa uma adaptação da ideia de breakpoints e adapta o limitante inferior para considerar a quantidade de caracteres presentes em apenas uma das strings.
  - O caso com sinais continua admitindo um algoritmo polinomial exato. Esse algoritmo tem um cuidado para juntar elementos a serem removidos ou inseridos para reduzir o número de indels necessários.
- Podemos tentar usar uma abordagem similar a que usamos para strings balanceadas mapeando os caracteres repetidos com o auxilio de uma partição.

Partição de Strings não balanceadas

- Queremos duas sequências de substrings: uma que pode formar S e uma que pode formar P.
- Agora cada sequência pode ter substrings exclusivas.
- Queremos garantir que substrings exclusivas são apenas as necessárias e a quantidade de substrings comuns é minimizada.





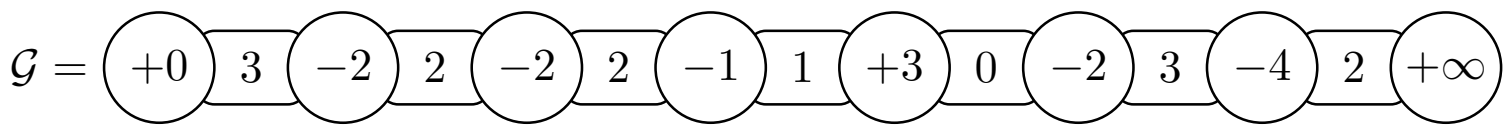
Adaptação dos Algoritmos para o Caso Balanceado

- O algoritmo guloso pode ser adaptado diretamente. Eventualmente não poderemos encontrar substrings comuns e os caracteres restantes formarão as substrings exclusivas.
- Para o algoritmo de seleção de breakpoints, devemos selecionar as substrings exclusivas antes de executar o algoritmo.
- Em ambos os casos não garantimos mais os fatores de aproximação.

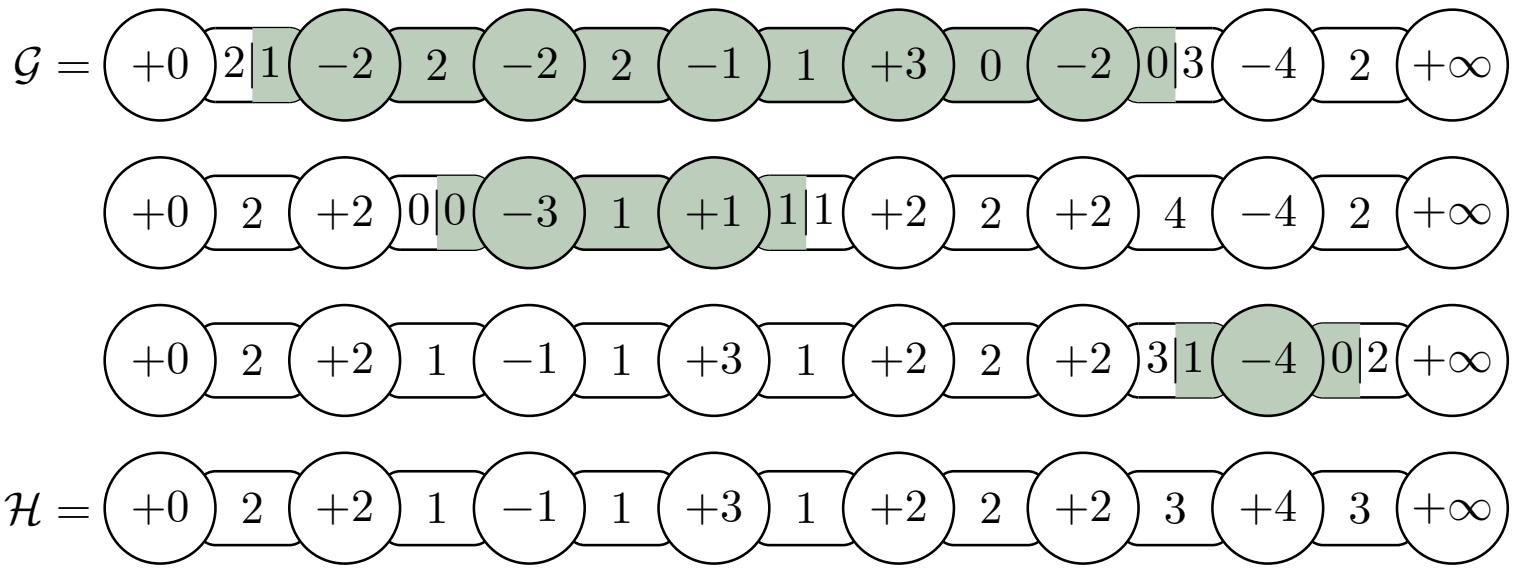
Representações com Mais Informações Biológicas

Regiões Intergênicas

- Os nucleotídeos entre genes podem ser usados para tornar os problemas de rearranjo mais biologicamente relevantes.
- Uma primeira abordagem para incluir essas informações é levar em conta o número de nucleotídeos em cada região intergênica.
- Nesse caso as operações afetam os genes e as regiões intergênicas.
- Em permutações e considerando a operação de reversão, temos um algoritmo com fator de aproximação 2 para o caso com sinais e 4 para o caso sem sinais, onde ambos os problemas pertencem a classe *NP*-Difícil.
- No caso sem sinais, o algoritmo considera uma adaptação da definição de breakpoints para incluir as regiões intergênicas.



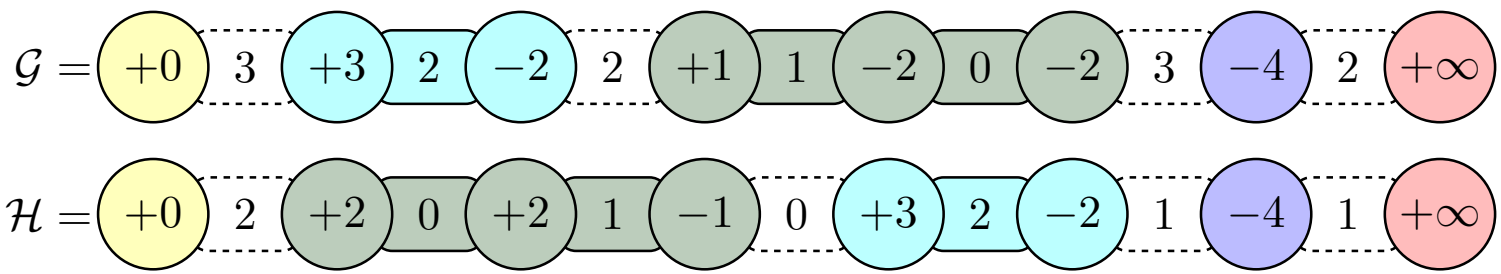
Regiões Intergênicas



$$d_R(\mathcal{G}, \mathcal{H}) = 3$$

Partições Intergênicas

- Podemos continuar usando a abordagem de partição. No lugar de substrings temos subgenomas com genes e regiões intergênicas.
- Podemos usar os mesmos algoritmos e temos as mesmas garantias de aproximação.



Múltiplos Cromossomos

- Quando consideramos múltiplos cromossomos cada genoma é representado por um conjunto de strings.
- Consideramos que as strings estão em qualquer ordem ou orientação.
- Em geral, para transformar um genoma no outro precisamos de operações que afetem múltiplos cromossomos.

$$\mathcal{G}_1 = (+3 \quad -3) \quad (+1 \quad -2 \quad -2 \quad -1) \quad (-2 \quad +3 \quad -4 \quad +2)$$

$$\mathcal{G}_2 = (+2 \quad +2 \quad -1 \quad -2 \quad +3) \quad (-2 \quad +4 \quad +1 \quad +3 \quad -3)$$

Múltiplos Cromossomos

- Novamente podemos usar partições.
- Podemos manter a aproximação de 2k para partição, mas agora utilizamos outros problemas de rearranjo.
- Por exemplo, sem repetição de genes se consideramos translocações e reversões ainda temos um algoritmo polinomial para o caso com sinais e uma aproximação com fator 2 para o caso sem sinais.