Alinhamento de seqüências com rearranjos

Augusto Fernandes Vellozo
a.vellozo@yahoo.com.br

24 de abril de 2007

Alinhamento

- Utilizado na comparação de seqüências
- Um bom alinhamento mostra os eventos biológicos ocorridos
- Normalmente são considerados inserção, remoção e substituição
- Visualização típica de um alinhamento de $CAGCACTGTTC \times CAGCGATGC$:

```
CAGCACTGTT-C
||||   | | |
CAGC---GATGC
```

- *Match*, *mismatch* e *gap*

Definição de alinhamento

Definição 1 (Alinhamento de s e t) *Sejam $s = \Sigma^*$ e $t = \Sigma^*$ duas seqüências de comprimentos n e m, respectivamente. Um alinhamento de s e t é uma matriz $A_{2\times r}$, tal que:*

- $r \geq m$, $r \geq n$, $r \leq m + n$,
- *para todo j tal que $0 \leq j \leq r - 1$, se $A[0, j] = A[1, j]$ então $A[0, j] \neq -$ e*
- *existem duas subseqüências $S = (i_1, i_2, \ldots, i_n)$ e $T = (j_1, j_2, \ldots, j_m)$ dos índices das colunas de A, tais que:*
  - $s = A[0, i_1]A[0, i_2] \ldots A[0, i_n]$,
  - $t = A[1, j_1]A[1, j_2] \ldots A[1, j_m]$,
  - $A[0, i] = -$ *para toda coluna i de A, tal que $i \notin S$, e*
  - $A[1, j] = -$ *para todo coluna j de A, tal que $j \notin T$.*

Exemplo de alinhamento

Alinhamento das seqüências
$s = AGCGTATCCAGT$ e $t = AGTATCACGGAT$.

$$A = \begin{bmatrix} A & G & \mathbf{C} & \mathbf{G} & T & - & A & T & C & \mathbf{C} & A & G & - & T \\ A & G & \mathbf{T} & \mathbf{A} & T & C & A & - & C & \mathbf{G} & - & G & A & T \end{bmatrix}$$

4 *gaps* e 3 *mismatches*

Exemplo de alinhamento

Alinhamento das seqüências
$s = AGCGTATCCAGT$ e $t = AGTATCACGGAT$.

$$A = \begin{bmatrix} A & G & \mathbf{C} & \mathbf{G} & T & - & A & T & C & \mathbf{C} & A & G & - & T \\ A & G & \mathbf{T} & \mathbf{A} & T & C & A & - & C & \mathbf{G} & - & G & A & T \end{bmatrix}$$

4 *gaps* e 3 *mismatches*

$$A = \begin{bmatrix} A & G & C & G & T & A & T & C & C & A & - & G & - & - & T \\ A & G & - & - & T & A & T & C & - & A & C & G & G & A & T \end{bmatrix}$$

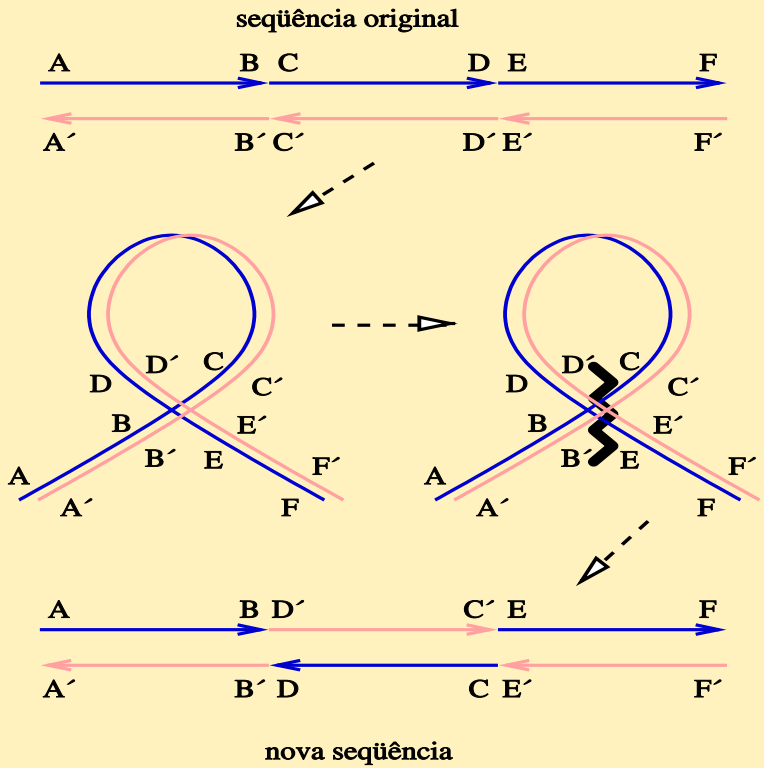
6 *gaps* e 0 *mismatches*
Qual é melhor?

Sistemas de pontuação

- Muito utilizados: *gap* linear e *gap* afim
- Função $\varphi : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \to \mathbb{R}$, determina que a pontuação de cada coluna k de um alinhamento A é igual a $\varphi(A[0, k], A[1, k])$
- Cada coluna está associada a um evento de inserção, remoção ou substituição.
- A pontuação do alinhamento é a soma das pontuações das colunas
- *Gap* afim: pontuação extra na primeira coluna de um trecho de inserção ou remoção
- Queremos um alinhamento com pontuação máxima (alinhamento ótimo)

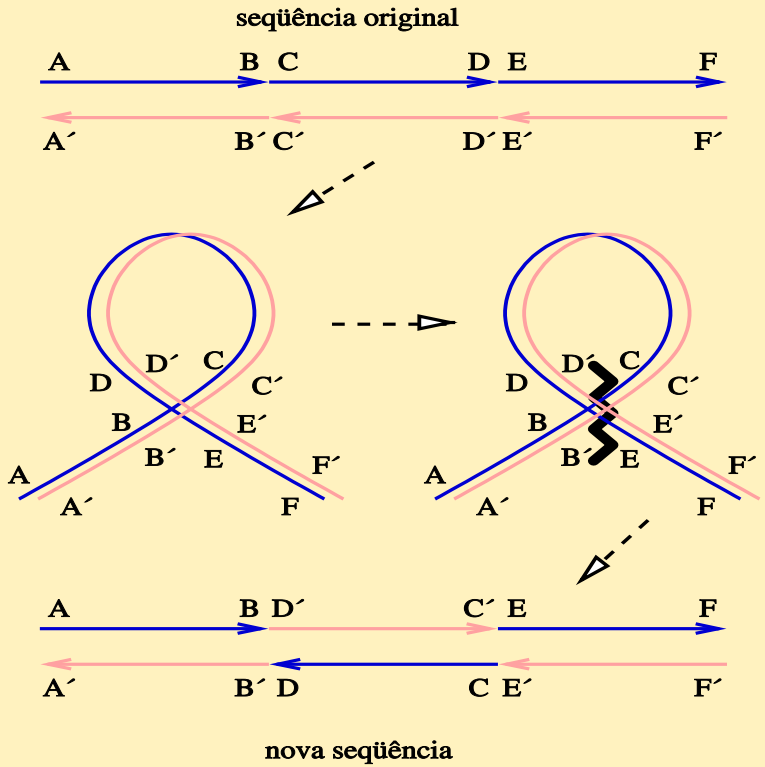
Inversão

- O evento de inversão é comum em seqüências de DNA



Inversão

- O evento de inversão é comum em seqüências de DNA



- Vamos considerar apenas inversões não sobrepostas

Duplicação

- Um fragmento de DNA de uma seqüência é copiado e inserido na própria seqüência, gerando uma repetição na seqüência
- Estima-se que mais de 80% dos genomas de planta são compostos por repetições
- São dois os tipos de duplicações que consideraremos que ocorrem: duplicações encadeadas (em *tandem*) ou transposições
- Exemplo de duplicação com transposição:
  AA**CTGGT**$GGAC$**CTGGT**$TCAG$
- Exemplo de duplicação em *tandem*: seqüência da *Pseudomonas aeruginosa PA01* das posições 98902 a 99067:
  TGGCTGTGGCTG**TGGCTG**TGGCTG**TGGCTG**TGGCTG
  TGGCTGTGGCTG**TGGCTG**TGGCTG**TGGCTG**

Algoritmos

- Considerando os eventos pontuais (substituição, inserção e remoção) há um algoritmo clássico que obtém um alinhamento ótimo em tempo $O(n^2)$.
- Considerando também o rearranjo da inversão não sobreposta tínhamos algoritmos com tempo $O(n^4)$.
- Considerando também o rearranjo da duplicação em *tandem* tínhamos algoritmos com tempo $O(n^4)$ e espaço $O(n^3)$.

Algoritmos

Desenvolvemos algoritmos para obter um alinhamento ótimo que contempla uma das seguintes situações:

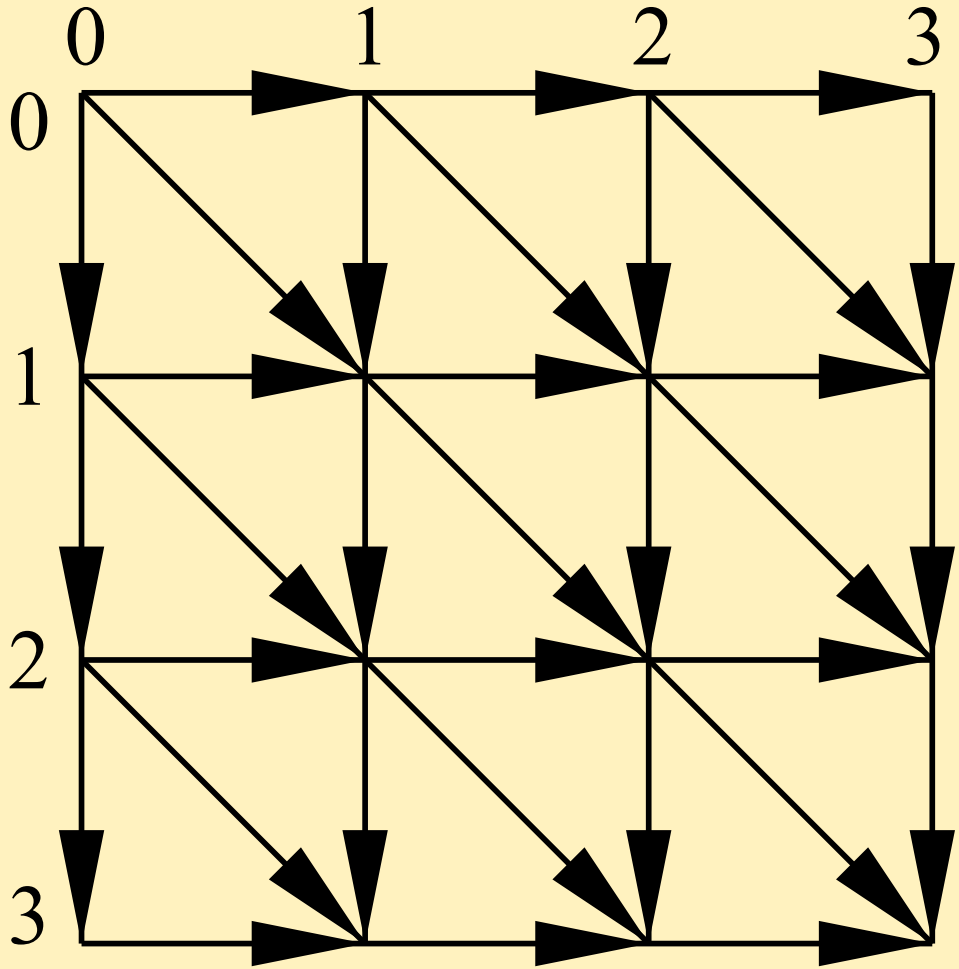
1. Inversão não sobreposta para sistemas de pontuação mais gerais $(O(n^3 \log n))$
2. Inversão não sobreposta para sistemas de pontuação com valores constantes e inteiros $(O(n^3))$
3. Duplicação com transposição $(O(n^3))$
4. Duplicação em *tandem* $(O(n^3))$

Grafo de edição de s e t

O grafo de edição de s e t é o grafo orientado com pesos nas arestas $G = (V, E, \omega)$, onde:

1. $V = \{(i, j) | 0 \leq i \leq n, 0 \leq j \leq m\}$.
2. $E = E_H \cup E_D \cup E_V$, tal que:
   - $E_H = \{((i, j-1), (i, j)) | 0 \leq i \leq n, 0 < j \leq m\}$ é o conjunto das arestas horizontais de G;
   - $E_D = \{((i-1, j-1), (i, j)) | 0 < i \leq n, 0 < j \leq m\}$ é o conjunto das arestas diagonais de G;
   - $E_V = \{((i-1, j), (i, j)) | 0 < i \leq n, 0 \leq j \leq m\}$ é o conjunto das arestas verticais de G.
3. A função $\omega : E \longrightarrow \mathbb{R} \cup \{-\infty\}$ associa a cada aresta $e \in E$ o seu peso $\omega(e)$.

Exemplo de grafo de edição



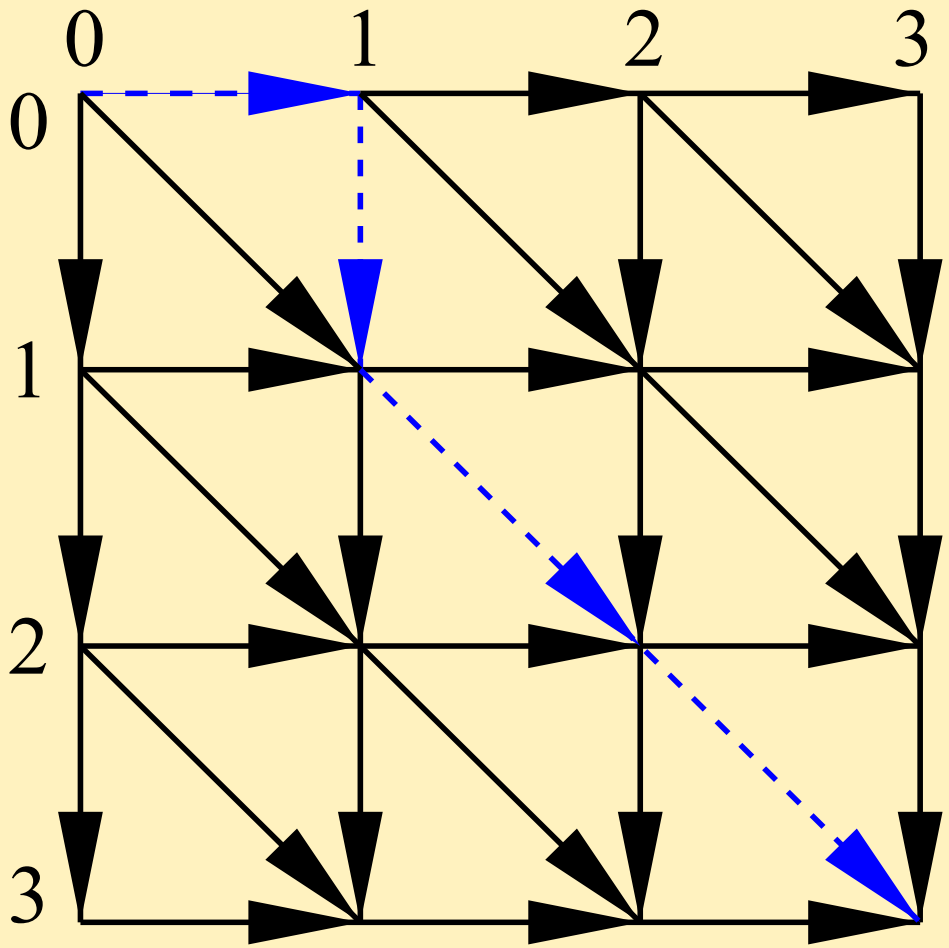
Neste exemplo não são mostrados os pesos das arestas

Grafo de edição

- $\epsilon_V^{(i,j)} = ((i-1,j),(i,j))$ corresponde à remoção da letra $s[i]$
- $\epsilon_H^{(i,j)} = ((i,j-1),(i,j))$ corresponde à inserção da letra $t[j]$
- $\epsilon_D^{(i,j)} = ((i-1,j-1),(i,j))$ corresponde à substituição de $s[i]$ por $t[j]$
- Sejam $u = (i,j)$ e $v = (i',j')$ dois vértices de G. Iremos considerar que um caminho p de u a v é ótimo se ele tiver peso máximo entre todos os caminhos de u a v.
- Dizemos que $\omega(u,v)$ é o peso de um caminho ótimo de u a v. Se não houver um caminho de u a v então $\omega(u,v) = -\infty$.
- Existe uma relação um-para-um entre um caminho em G e um alinhamento de um trecho de s contra um trecho de t.
- Cada caminho de $(0,0)$ a (i,j) em G corresponde a um alinhamento de $s[1..i]$ contra $t[1..j]$.

Exemplo

Exemplo de um caminho num grafo de edição.



Grafo de edição estendido

- Um grafo de edição estendido de s e t é um grafo de edição de s e t com algumas arestas a mais. Chamaremos estas arestas de arestas estendidas.

  O conjunto das arestas estendidas é o conjunto:

  $E_X = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m} E_X^{i,j}$, tal que:

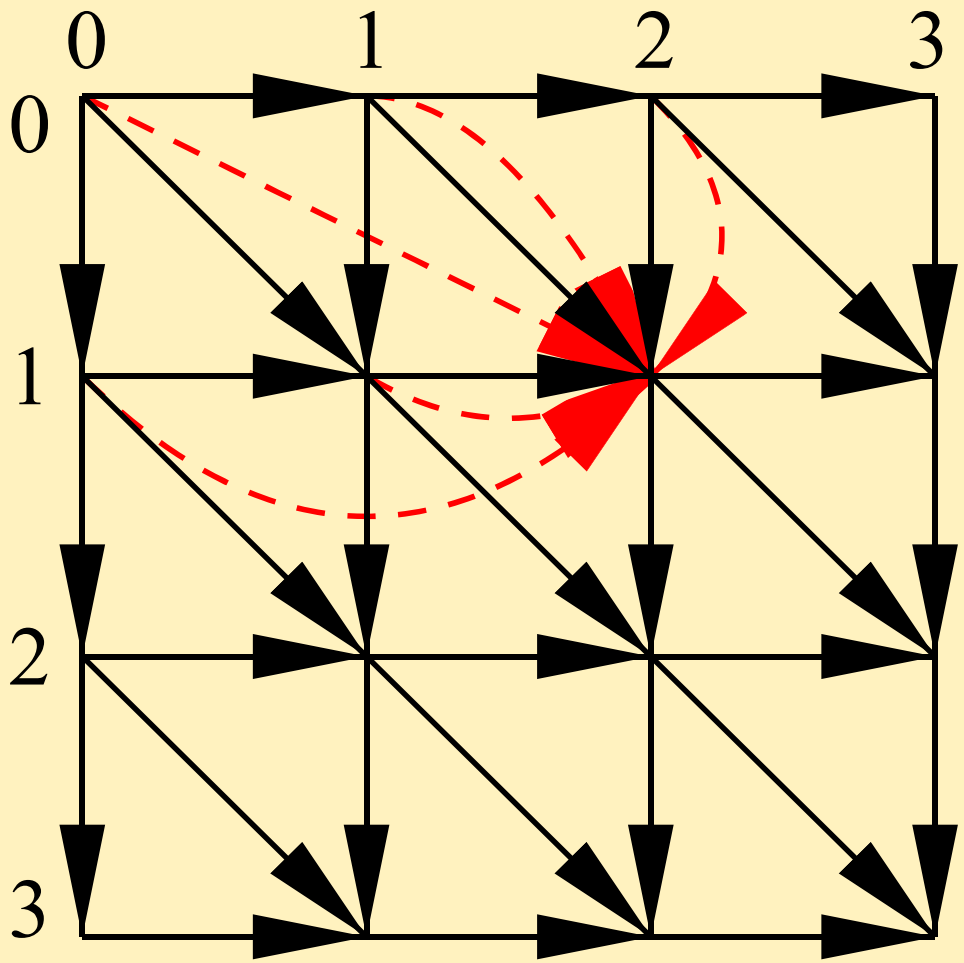
  - $E_X^{i,j} = \{((i', j'), (i, j)) \mid 0 \leq i' \leq i \leq n, 0 \leq j' \leq j \leq m$ e $(i', j') \neq (i, j)\}$ e
  - $E_X \cap \{E_H \cup E_D \cup E_V\} = \emptyset$.

- Consideraremos que um caminho ótimo em um grafo de edição estendido é um caminho de peso máximo.

- Uma aresta estendida representa um evento com mais de um símbolo (duplicação ou inversão)

Exemplo

Exemplo de arestas estendidas que chegam em $(1, 2)$.



Matriz de pesos

Seja G um grafo de edição de s e t. Sejam i e i' tais que $0 \leq i' \leq i \leq n$ e $i - i' = n'$. Seja $W_G^{i',i}$ a matriz $(m+1) \times (m+1)$ tal que $W_G^{i',i}[j', j] = \omega((i', j'), (i, j))$.

- Um algoritmo simples que calcula $W_G^{i',i}$ leva tempo $O(m^2 n')$
- Jeanette Schmidt desenvolveu um algoritmo incremental que constrói uma estrutura de árvores que armazenam os valores de $W_G^{i',i}$ em tempo $O(mn' \log m)$. Porém o tempo de acesso ao valor de $W_G^{i',i}[j', j]$ é $O(\log m)$.
- A matriz $W_G^{i',i}$ é uma matriz de monge inversa triangular superior.
- Obter os máximos de cada coluna de $W_G^{i',i}$ leva tempo $O(m)$.

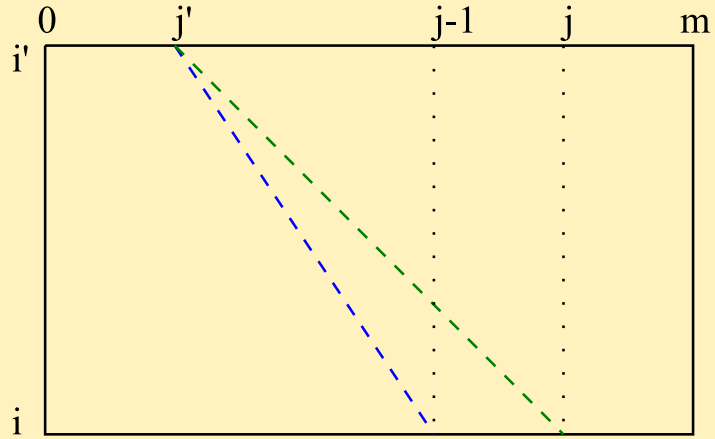
Funções hDif_G, vDif_G e dDif_G

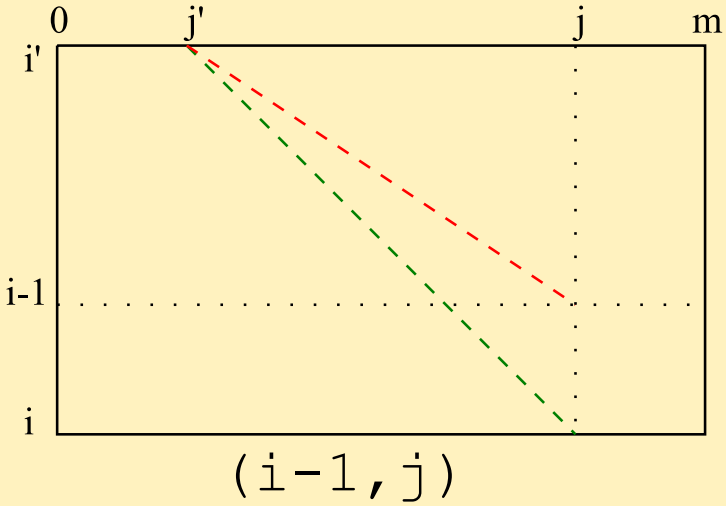
Dado um grafo de edição $G = (V, E, \omega)$ de s e t, definimos as funções $\text{hDif}_G : V \times V \to \mathbb{R} \cup \{-\infty\}$, $\text{vDif}_G : V \times V \to \mathbb{R} \cup \{-\infty\}$ e $\text{dDif}_G : V \times V \to \mathbb{R} \cup \{-\infty\}$ da seguinte forma:

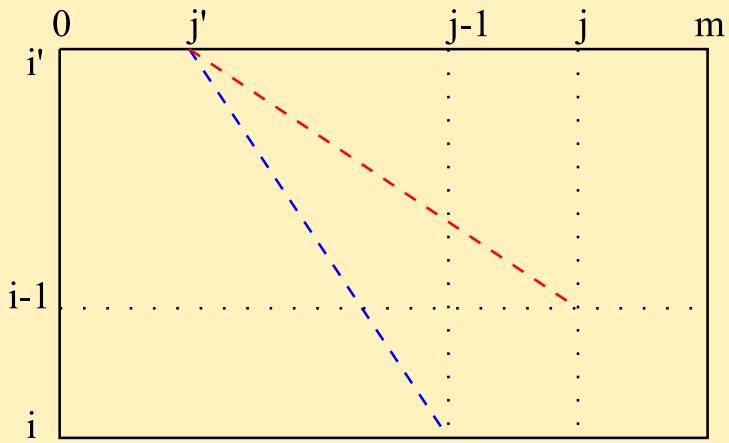
- $\text{hDif}_G((i', j'), (i, j)) = \omega((i', j'), (i, j)) - \omega((i', j'), (i, j-1))$, se $j' < j$ e $i' \leq i$
- $\text{vDif}_G((i', j'), (i, j)) = \omega((i', j'), (i-1, j)) - \omega((i', j'), (i, j))$, se $j' \leq j$ e $i' < i$
- $\text{dDif}_G((i', j'), (i, j)) = \omega((i', j'), (i-1, j)) - \omega((i', j'), (i, j-1))$, se $j' < j$ e $i' < i$
- $\text{hDif}_G((i', j'), (i, j)) = -\infty$, se $j' \geq j$ ou $i' > i$
- $\text{vDif}_G((i', j'), (i, j)) = -\infty$, se $j' > j$ ou $i' \geq i$
- $\text{dDif}_G((i', j'), (i, j)) = -\infty$, se $j' \geq j$ ou $i' \geq i$

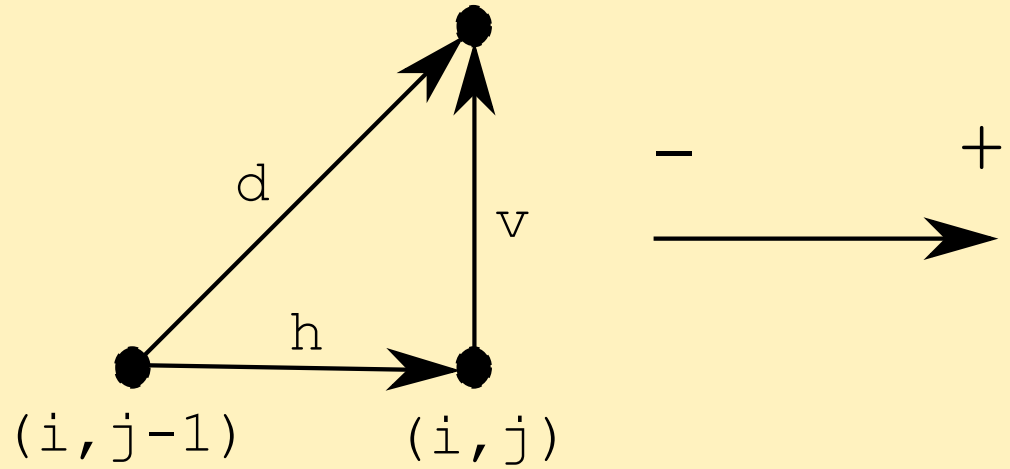
hDif_G, vDif_G e dDif_G

Baseado no fato que $W_G^{i',i}$ é uma matriz de monge, temos que, fixados i, i' e j, os valores de $\text{hDif}_G((i',j'),(i,j))$, $\text{vDif}_G((i',j'),(i,j))$ e $\text{dDif}_G((i',j'),(i,j))$ são **não decrescentes**.









Proposição 1

Proposição 1 *Dados um grafo de edição G de s e t e dois vértices, (i', j') e (i, j), de G tais que $i' < i$ e $j' < j$, então podemos dizer que*

1. $\omega((i', j'), (i, j)) = \omega((i', j'), (i, j-1)) + \omega(\epsilon_H^{(i,j)}) \iff$
   (a) $vDif_G((i', j'), (i, j-1)) \leq \omega(\epsilon_H^{(i,j)}) - \omega(\epsilon_D^{(i,j)})$ *e*
   (b) $dDif_G((i', j'), (i, j)) \leq \omega(\epsilon_H^{(i,j)}) - \omega(\epsilon_V^{(i,j)})$
2. $\omega((i', j'), (i, j)) = \omega((i', j'), (i-1, j-1)) + \omega(\epsilon_D^{(i,j)}) \iff$
   (a) $hDif_G((i', j'), (i-1, j)) \leq \omega(\epsilon_D^{(i,j)}) - \omega(\epsilon_V^{(i,j)})$ *e*
   (b) $vDif_G((i', j'), (i, j-1)) \geq \omega(\epsilon_H^{(i,j)}) - \omega(\epsilon_D^{(i,j)})$
3. $\omega((i', j'), (i, j)) = \omega((i', j'), (i-1, j)) + \omega(\epsilon_V^{(i,j)}) \iff$
   (a) $hDif_G((i', j'), (i-1, j)) \geq \omega(\epsilon_D^{(i,j)}) - \omega(\epsilon_V^{(i,j)})$ *e*
   (b) $dDif_G((i', j'), (i, j)) \geq \omega(\epsilon_H^{(i,j)}) - \omega(\epsilon_V^{(i,j)})$

Existem j_1 e j_2

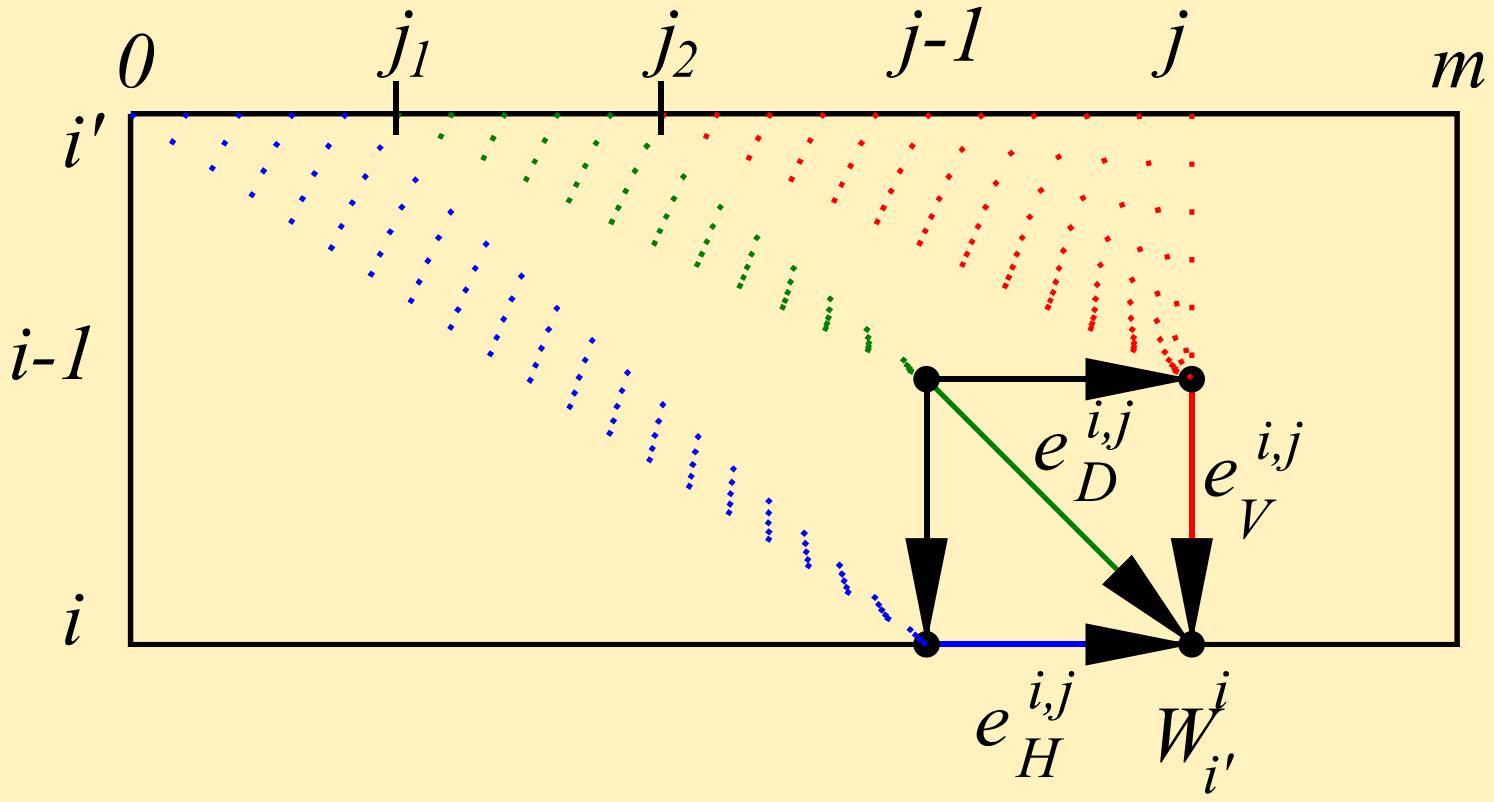
Lema 1 *Dados i', i e j tais que $0 \leq i' < i \leq n$, $0 \leq j \leq m$ então existem j_1 e j_2 tais que $0 \leq j_1 \leq j_2 \leq j$ e*

$$\omega((i',j'),(i,j)) = \begin{cases} \omega((i',j'),(i,j-1)) + \omega(\epsilon_H^{(i,j)}) & \forall\ j' \mid 0 \leq j' < j_1 \\ \omega((i',j'),(i-1,j-1)) + \omega(\epsilon_D^{(i,j)}) & \forall\ j' \mid j_1 \leq j' < j_2 \\ \omega((i',j'),(i-1,j)) + \omega(\epsilon_V^{(i,j)}) & \forall\ j' \mid j_2 \leq j' \leq j \end{cases}$$

Os índices j_1 e j_2 agrupam os caminhos ótimos que chegam em (i,j) de acordo com a utilização das arestas $\epsilon_H^{(i,j)}$, $\epsilon_D^{(i,j)}$ e $\epsilon_V^{(i,j)}$.
Este Lema se baseia na Proposição 1 e no fato dos valores $\text{hDif}_G((i',j'),(i,j))$, $\text{vDif}_G((i',j'),(i,j))$ e $\text{dDif}_G((i',j'),(i,j))$ serem não decrescentes.

Existem j_1 e j_2

A idéia do Lema 1 é que podemos agrupar os índices j' em 3 grupos de acordo com o uso das arestas horizontal, vertical e diagonal pelos caminhos ótimos que chegam em (i, j).
A única aresta que obrigatoriamente é utilizada é a aresta vertical.



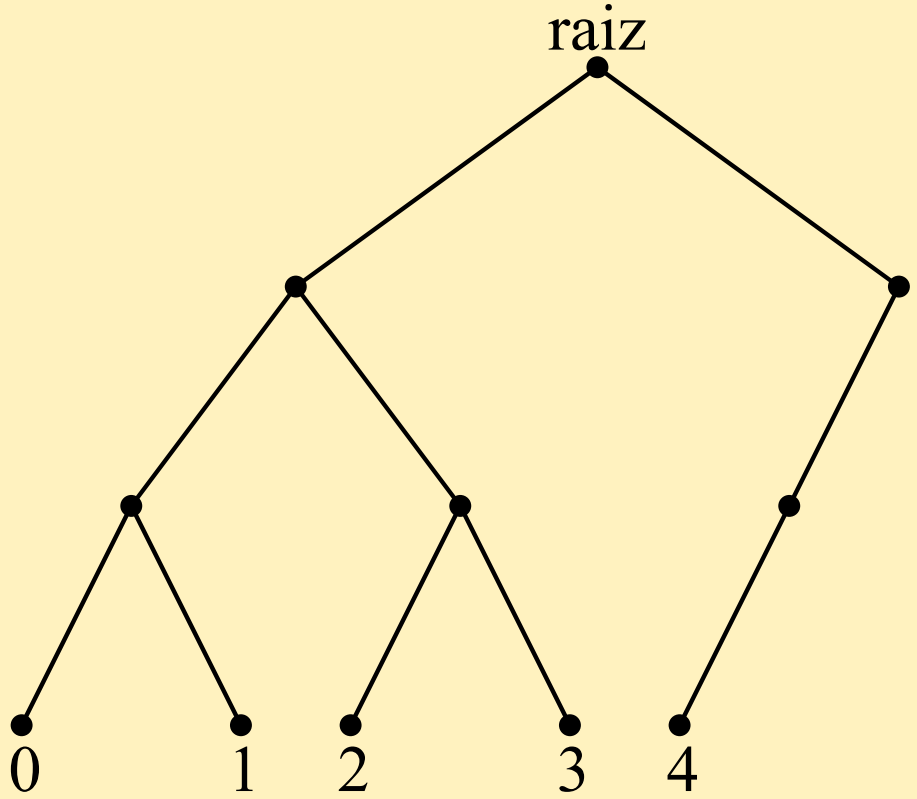
Árvore binária

Sejam (i, j) um vértice de um grafo de edição G e $i' \leq i$. Definimos a árvore binária $B_G^{i',i,j}$ da seguinte forma:

- $B_G^{i',i,j}$ possui pesos nas arestas.
- $B_G^{i',i,j}$ tem $j+1$ folhas, tal que todas têm profundidade $\lceil \log_2(j+1) \rceil$, são rotuladas seqüencialmente de 0 a j e o peso do caminho da raiz até a folha j' é igual à $\omega((i', j'), (i, j))$.
- A cada vértice v de $B_G^{i',i,j}$ associamos os seguintes atributos:
  - esq_v: o filho à esquerda de v e dir_v: o filho à direita de v,
  - h_v: o comprimento do caminho de v até uma folha,
  - p_v: o peso do caminho da raiz até v e
  - pe_v: o peso do caminho de v até a folha mais a direita da subárvore esquerda de v, se esta subárvore for completa; e $-\infty$ se esta subárvore for incompleta.

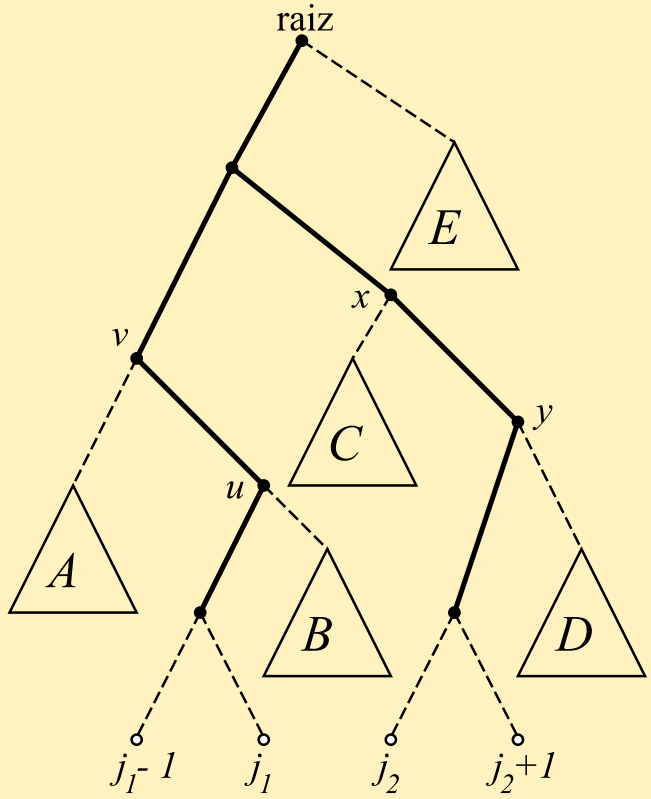
Exemplo de $B_G^{i',i,j}$

Fixado um j, a topologia das árvores $B_G^{i',i,j}$ são iguais. Exemplo de árvore binária $B_G^{i',i,4}$.



Construção de $B_G^{i',i,j}$

Utilizando o Lema 1 podemos construir a árvore $B_G^{i',i,j}$ a partir de pedaços das árvores $B_G^{i',i,j-1}$,$B_G^{i',i-1,j-1}$ e $B_G^{i',i-1,j}$ em $O(\log m)$.



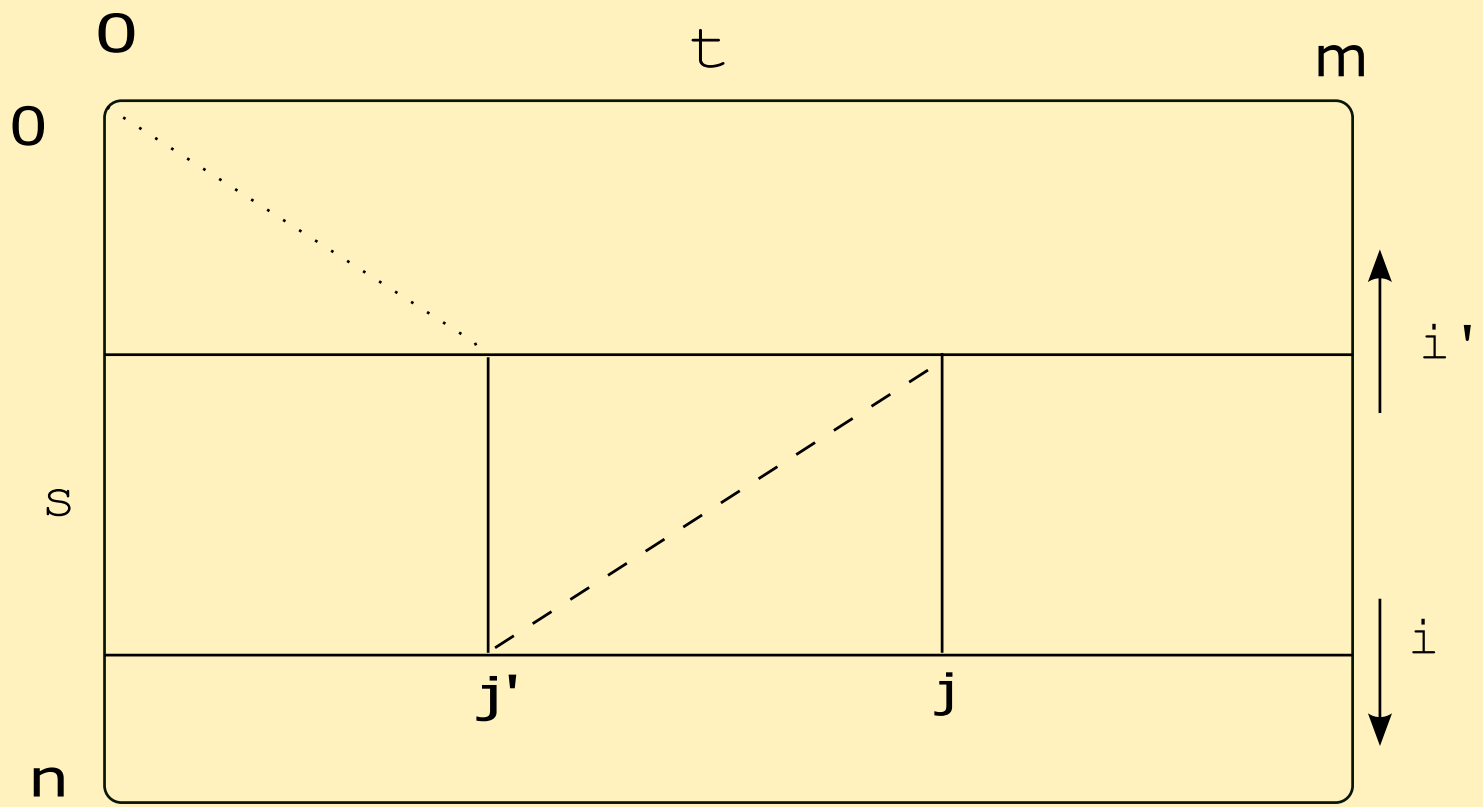
Algoritmo $n^3 \log n$

Algoritmo $n^3 \log n$ para alinhamento com inversões não sobrepostas

1. Para i de 0 descendo até $|s|$ faça
   (a) $\forall j$ obtém $B[i, j]$ das arestas não estendidas
   (b) Para i' de i descendo até 0 faça
      i. Constrói as árvores $B_{\overline{G}}^{n-i,n-i',j}$ para $W_{\overline{G}}^{n-i,n-i'}$
      ii. Define $A_i^{i'}[j', j] = W_{\overline{G}}^{n-i,n-i'}[j', j] + B[i', j'] + \omega_{inv}$
      iii. $maxTotMonotonica(A_i^{i'}, MaxCol_i^{i'})$
      iv. $\forall$ façaj $B[i, j] \leftarrow \max(B[i, j], A[MaxCol_i^{i'}[j], j])$

Como as linhas i e iii são executadas em tempo $O(m \log m)$ o algoritmo executa em tempo $O(n^2 m \log m)$.

Algoritmo $n^3 \log n$



Definições para o algoritmo n^3

$BL_G^{i',i,j}$: lista ordenada cujo primeiro elemento é 0 e os demais são os j' onde $\text{hDif}_G((i',j'),(i,j)) \neq \text{hDif}_G((i',j'-1),(i,j))$. $|BL_G^{i',i,j}|$ constante.

Pesos de caminhos ótimos de $(0,j')$ a (n,j)

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9
j'=0	-4	-2	-2	-2	-3	-2	0	-1	-2	-3
j'=1	-	-4	-2	-2	-3	-1	1	0	-1	-2
j'=2	-	-	-4	-4	-2	0	2	1	0	-1
j'=3	-	-	-	-4	-2	0	2	1	0	0
j'=4	-	-	-	-	-4	-2	0	-1	-2	-1
j'=5	-	-	-	-	-	-4	-2	-2	-2	0
j'=6	-	-	-	-	-	-	-4	-2	-2	0
j'=7	-	-	-	-	-	-	-	-4	-4	-2
j'=8	-	-	-	-	-	-	-	-	-4	-2
j'=9	-	-	-	-	-	-	-	-	-	-4

hDif((0,j'),(n,j))

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9
j'=0	-	2	0	0	-1	1	2	-1	-1	-1
j'=1	-	-	**2**	0	-1	**2**	2	-1	-1	-1
j'=2	-	-	-	0	**2**	2	2	-1	-1	-1
j'=3	-	-	-	-	2	2	2	-1	-1	**0**
j'=4	-	-	-	-	-	2	2	-1	-1	**1**
j'=5	-	-	-	-	-	-	2	**0**	**0**	**2**
j'=6	-	-	-	-	-	-	-	**2**	0	2
j'=7	-	-	-	-	-	-	-	-	0	2
j'=8	-	-	-	-	-	-	-	-	-	2
j'=9	-	-	-	-	-	-	-	-	-	-

BLj

s=AATG		j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9
t=TTCATGACG	0	-	0	0	0	0	0	0	0	0	0
Peso do *gap* = -1	1	-	-	1	-	2	1	-	5	5	3
Peso do *mismatch* = -1	2	-	-	-	-	-	-	-	6	-	4
Peso do *match* = +1	3	-	-	-	-	-	-	-	-	-	5

Figura 1: Os elementos de $\text{hDif}((0,j'),(n,j))$ em negrito são *Borderline points*. Os j' onde eles ocorrem estão no vetor BL_j e são os elementos de $BL_G^{0,n,j}$, onde G é um grafo de edição de $AATG \times TTCATGACG$.

Definições para o algoritmo n^3

$$\mathsf{out}((i',j'),(i,j)) = \omega((0,0),(i',j')) + \overline{\omega}((n-i,j'),(n-i',j))$$

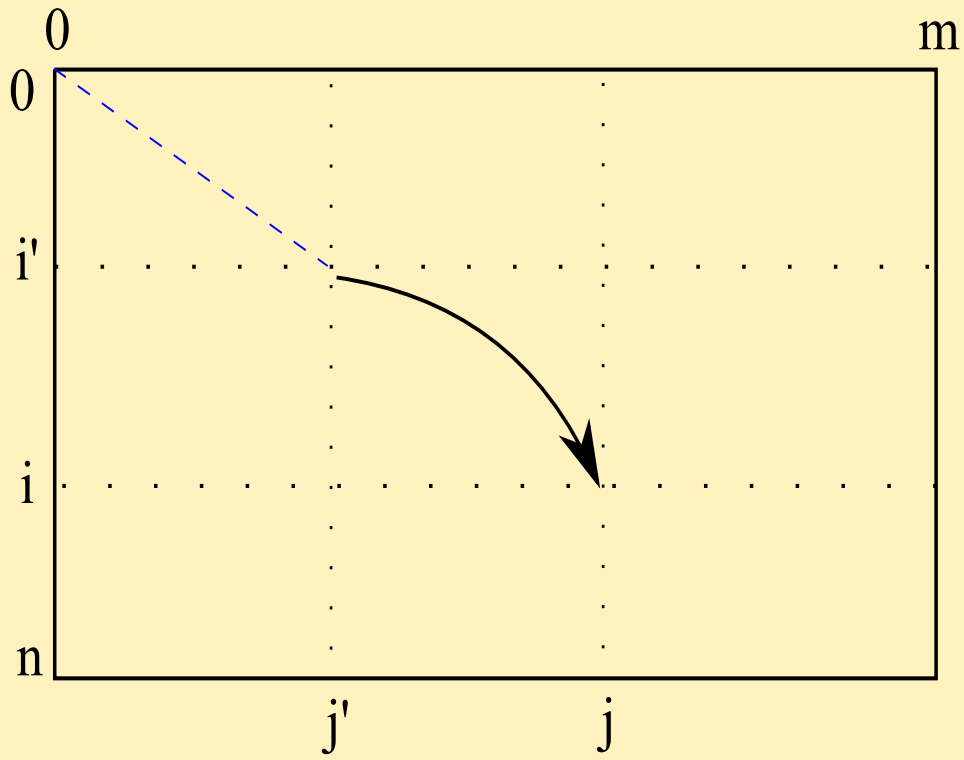


Figura 2: Representação de um caminho de $(0,0)$ a (i,j) num grafo de edição estendido com peso $\mathsf{out}((i',j'),(i,j)) + \omega_{inv}$. A linha tracejada representa um caminho ótimo de $(0,0)$ a (i',j') e a seta a aresta estendida $\epsilon^{(i,j)}_{(i',j')}$.

Definições para o algoritmo n^3

Dadas as linhas i' e i de um grafo de edição G tais que $i' \leq i$ definimos:

- CL_j é a lista ordenada com os j_1 tais que não existe outra coluna j_2 de G tal que $j_1 < j_2 \leq j$ e $\mathsf{out}((i', j_2), (i, j)) \geq \mathsf{out}((i', j_1), (i, j))$.
  - j sempre está em CL_j
  - se $j' < j$ e $j' \notin CL_j$ então $j' \notin CL_{j+1}$
- Função $\Delta\mathsf{out}_G$: Dado um grafo de edição estendido $G = (V, E, \omega)$ de s e t, a função parcial $\Delta\mathsf{out}_G : V \times V \to \mathbb{R}$ é definida por
  1. $\Delta\mathsf{out}_G((i', j_2), (i, j)) = \mathsf{out}((i', j_2), (i, j)) - \mathsf{out}((i', j_1), (i, j))$, se $i' \leq i$, $j_2 \in CL_j$ e j_2 não é o primeiro elemento de CL_j, onde j_1 é o antecessor de j_2 em CL_j;
  2. $\Delta\mathsf{out}_G((i', j_1), (i, j)) = \mathsf{out}((i', j_1), (i, j))$, se $i' \leq i$ e j_1 é o primeiro elemento de CL_j.

Esboço de $obtemMaxOut(BL)$

1. Para j de 0 até m faça $> i$ e i' estão fixos
   (a) Insere j na lista de candidatos CL
   (b) $\Delta\text{out}_G[j] \leftarrow \text{out}((i', j), (i, j)) - \text{out}((i', j-1), (i, j))$
   (c) Para α de 0 até $|BL[j]| - 1$ faça $> BL[j, \alpha] = BL_{\overline{G}}^{n-i, n-i', j}[\alpha]$
      i. $j' \leftarrow CL.find(BL[j, \alpha])$
      ii. $\Delta\text{out}_G[j'] \leftarrow +\text{hDif}_G((i', BL[j, \alpha]), (i, j))$
      iii. se $\alpha > 0$, $\Delta\text{out}_G[j'] \leftarrow -\text{hDif}_G((i', BL[j, \alpha - 1]), (i, j))$
      iv. Enqto $(\Delta\text{out}_G[j'] \geq 0)$ e $(j' \neq CL.first())$
         A. $j'' \leftarrow CL.previous(j')$
         B. $CL.remove(j'')$
         C. $\Delta\text{out}_G[j'] \leftarrow \Delta\text{out}_G[j'] + \Delta\text{out}_G[j'']$
   (d) $maxOut[j] \leftarrow \Delta\text{out}_G[CL.first()]$
2. devolve maxOut

Esboço do alg n^3

1. Para i de 0 descendo até $|s|$ faça
   (a) $\forall j$ obtém $B[i,j]$ das arestas não estendidas
   (b) Para i' de i descendo até 0 faça
      i. Constrói $BL > BL[j,\alpha] = BL_{\overline{G}}^{n-i,n-i',j}[\alpha]$
      ii. $MaxOut \leftarrow obtemMaxOut(BL)$
      iii. $\forall\ j\ B[i',j] \leftarrow \max(B[i,j], MaxOut[j] + \omega_{inv})$

De acordo com trabalho feito por Jeanette Schmidt a construção de BL pode ser feita em tempo $O(m)$. Como $obtemMaxOut(BL)$ é executado em tempo $O(m)$, o algoritmo é executado em tempo $O(n^2m)$.

Alinhamento com duplicações

- Os trechos envolvidos na duplicação (repetição e seqüência original) podem ter sofrido alterações após a duplicação
  AA**CTGGT**$GGAC$**CTCGT**$TCAG$

- Pontuação para uma duplicação:

  - $\mathsf{dup}(t, j', j) = \omega_u(t[j' \,..\, j]) +$
    $\max(\omega_L(t[j' \,..\, j], s), \omega_L(t[j' \,..\, j], t[1 \,..\, j'-1]t[j+1 \,..\, m]))$,
  - $\mathsf{dup}(s, i', i) = \omega_x(s[i' \,..\, i]) +$
    $\max(\omega_L(s[i' \,..\, i], t), \omega_L(s[i' \,..\, i], s[1 \,..\, i'-1]s[i+1 \,..\, n]))$,

  onde $\omega_L(s', t')$ é a pontuação do alinhamento ótimo **sem** duplicações de s' com qualquer fator de t'.

Alinhamento com duplicações

- Os trechos envolvidos na duplicação (repetição e seqüência original) podem ter sofrido alterações após a duplicação
  $AA\mathbf{CTGGT}GGAC\mathbf{CTCGT}TCAG$

- Pontuação para uma duplicação:

  - $\mathsf{dup}(t, j', j) = \omega_u(t[j' \,..\, j]) +$
    $\max(\omega_L(t[j' \,..\, j], s), \omega_L(t[j' \,..\, j], t[1 \,..\, j'-1]t[j+1 \,..\, m]))$,
  - $\mathsf{dup}(s, i', i) = \omega_x(s[i' \,..\, i]) +$
    $\max(\omega_L(s[i' \,..\, i], t), \omega_L(s[i' \,..\, i], s[1 \,..\, i'-1]s[i+1 \,..\, n]))$,

  onde $\omega_L(s', t')$ é a pontuação do alinhamento ótimo **sem** duplicações de s' com qualquer fator de t'.

- Problema dos ciclos

Alinhamento com duplicações

- Pontuação das arestas estendidas
  - $\omega(\epsilon^{(i,j)}_{(i',j')}) = \mathsf{dup}(s, i'+1, i)$, se $i' \neq i$ e $j' = j$,
  - $\omega(\epsilon^{(i,j)}_{(i',j')}) = \mathsf{dup}(t, j'+1, j)$, se $i' = i$ e $j' \neq j$,
  - $\omega(\epsilon^{(i,j)}_{(i',j')}) = -\infty$ se $i' \neq i$ e $j' \neq j$.
- Vamos considerar somente as arestas estendidas horizontais e arestas estendidas verticais
- Para cada vértice consideraremos $O(n)$ arestas estendidas
- Algoritmo simples calcula o peso de cada aresta estendida em tempo $O(n^2)$
- Algoritmo ingênuo calcula o alinhamento com duplicações em tempo $O(n^5)$. Calcularemos em $O(n^3)$.

Conjuntos de fatores

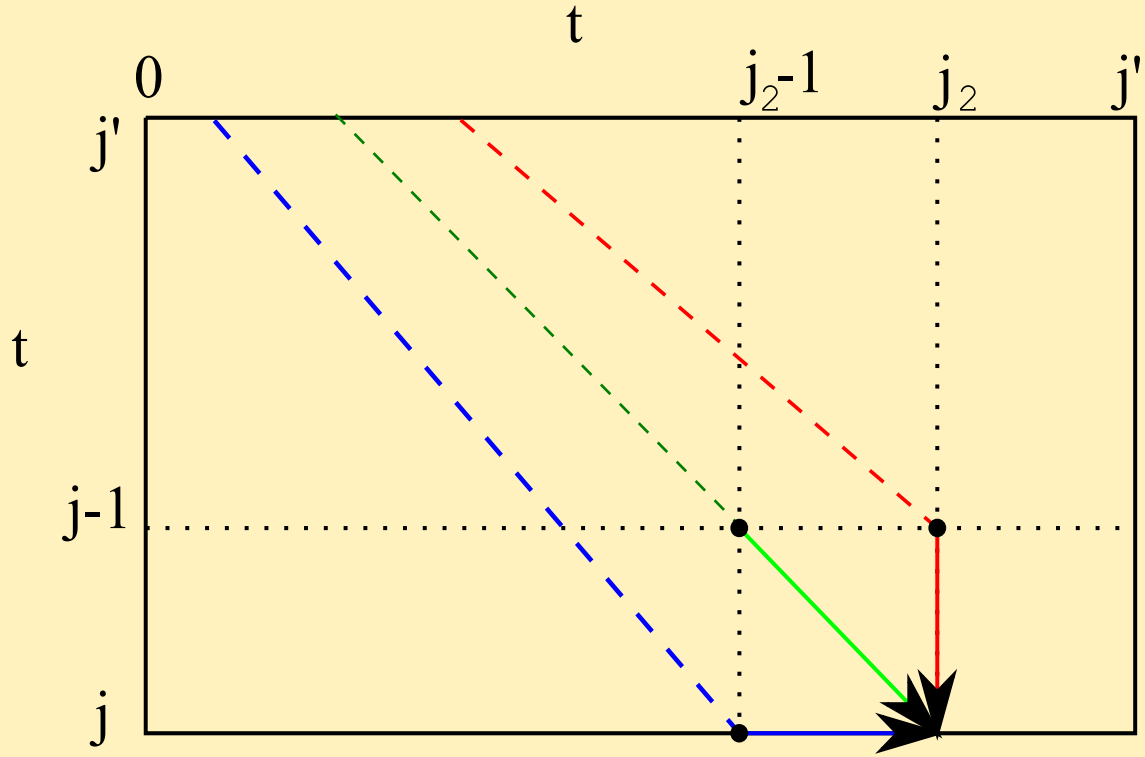
Para todos j' e j tais que $1 \leq j' \leq j \leq m$ definimos os conjuntos de fatores das seqüências s e t, $A_t^{j',j}$, $B_t^{j',j}$, $C_t^{j',j}$ e D_s da seguinte forma:

1. $A_t^{j',j} = \{t[j_1 \,..\, j_2] \mid 0 < j_1 \leq j_2 + 1 \leq j'\}$,
2. $B_t^{j',j} = \{t[j_3 \,..\, j_4] \mid j < j_3 \leq j_4 + 1 \leq |t| + 1\}$,
3. $C_t^{j',j} = \{t[j_5 \,..\, j' - 1]t[j + 1 \,..\, j_6] \mid 0 < j_5 \leq j'$ e $j \leq j_6 \leq |t|\}$ e
4. $D_s = \{s[i_1 \,..\, i_2] \mid 1 \leq i_1 \leq i_2 + 1 \leq |s| + 1\}$.

Definimos também as matrizes $\widehat{W_t^A}[j', j]$, $\widehat{W_t^B}[j', j]$, $\widehat{W_t^C}[j', j]$ e $\widehat{W_{t|s}^D}[j', j]$ que contêm o valor máximo do alinhamento ótimo sem duplicações de $t[j' + 1 \,..\, j] \times x$, onde $x \in A_t^{j'+1,j}$, $x \in B_t^{j'+1,j}$, $x \in C_t^{j'+1,j}$ e $x \in D_s$, respectivamente.

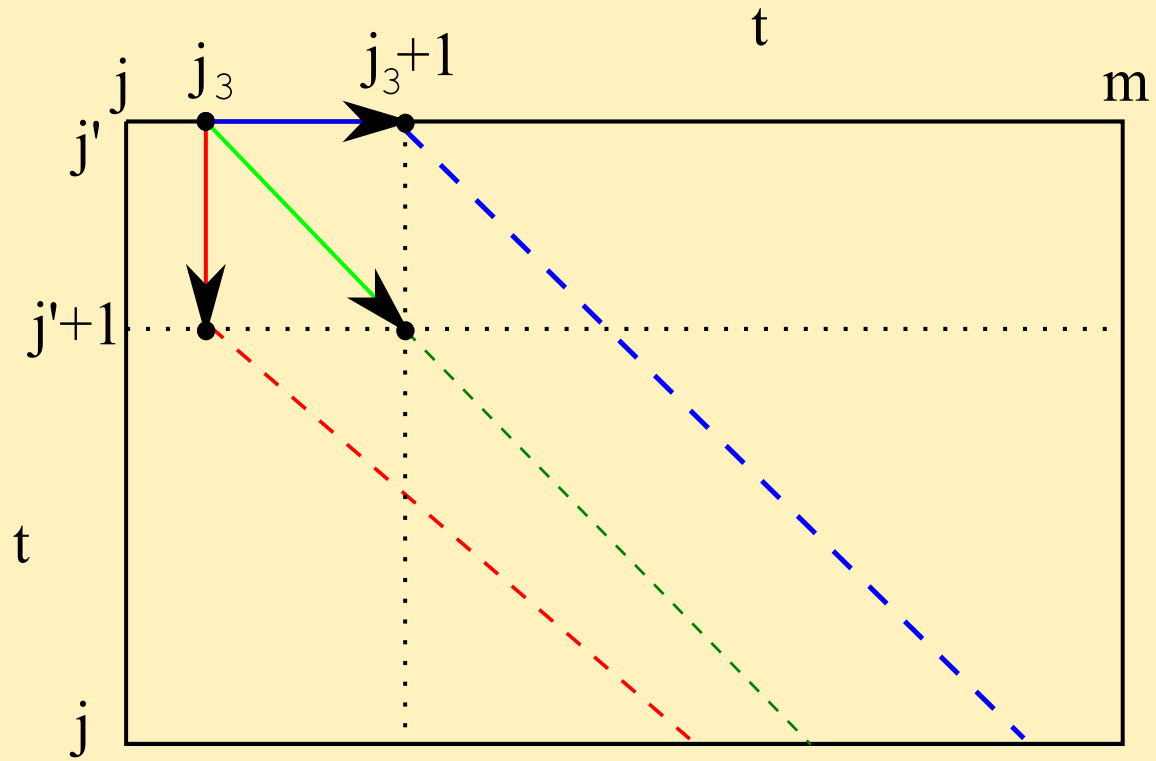
Constrói $\widehat{W_t^A}$

1. Para cada j'
   (a) Para cada j
      i. Para cada j_2
         A. $P \leftarrow$ melhor caminho da linha j' até (j_2, j)
         B. $\widehat{W_t^A}[j', j] \leftarrow \max(\widehat{W_t^A}[j', j], \omega(P))$



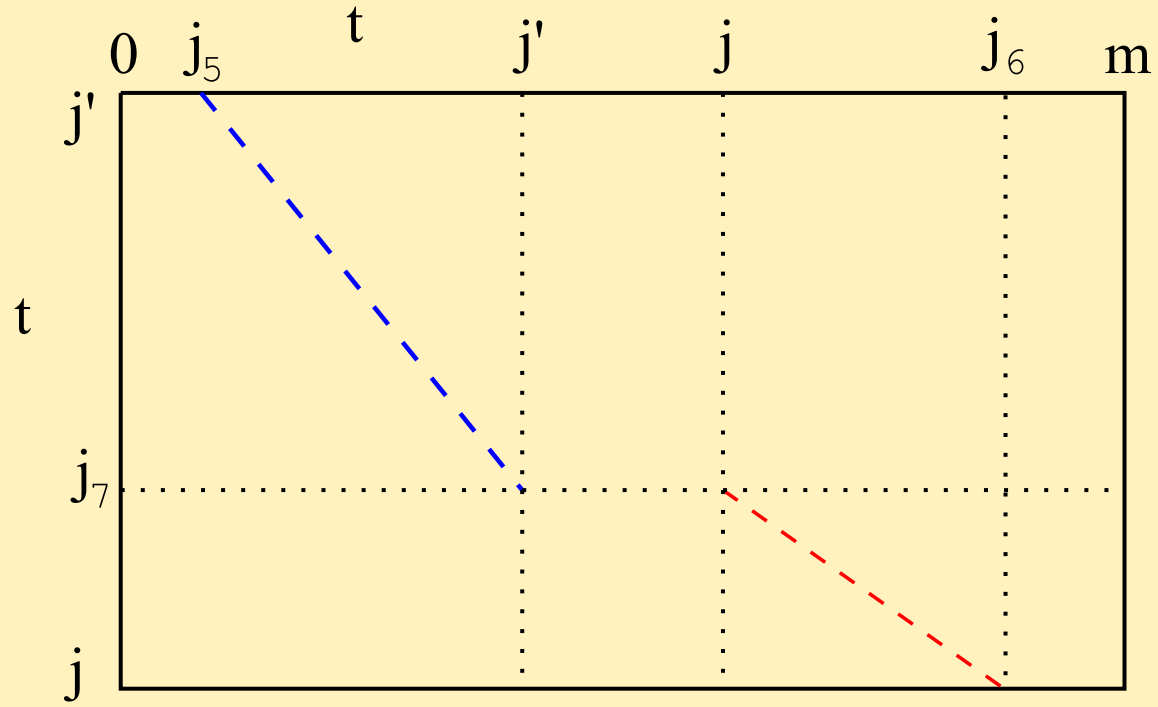
Constrói $\widehat{W_t^B}$

1. Para cada j
   (a) Para cada j'
      i. Para cada j_3
         A. $P \leftarrow$ melhor caminho de (j', j_3) até a linha j
         B. $\widehat{W_t^B}[j', j] \leftarrow \max(\widehat{W_t^B}[j', j], \omega(P))$



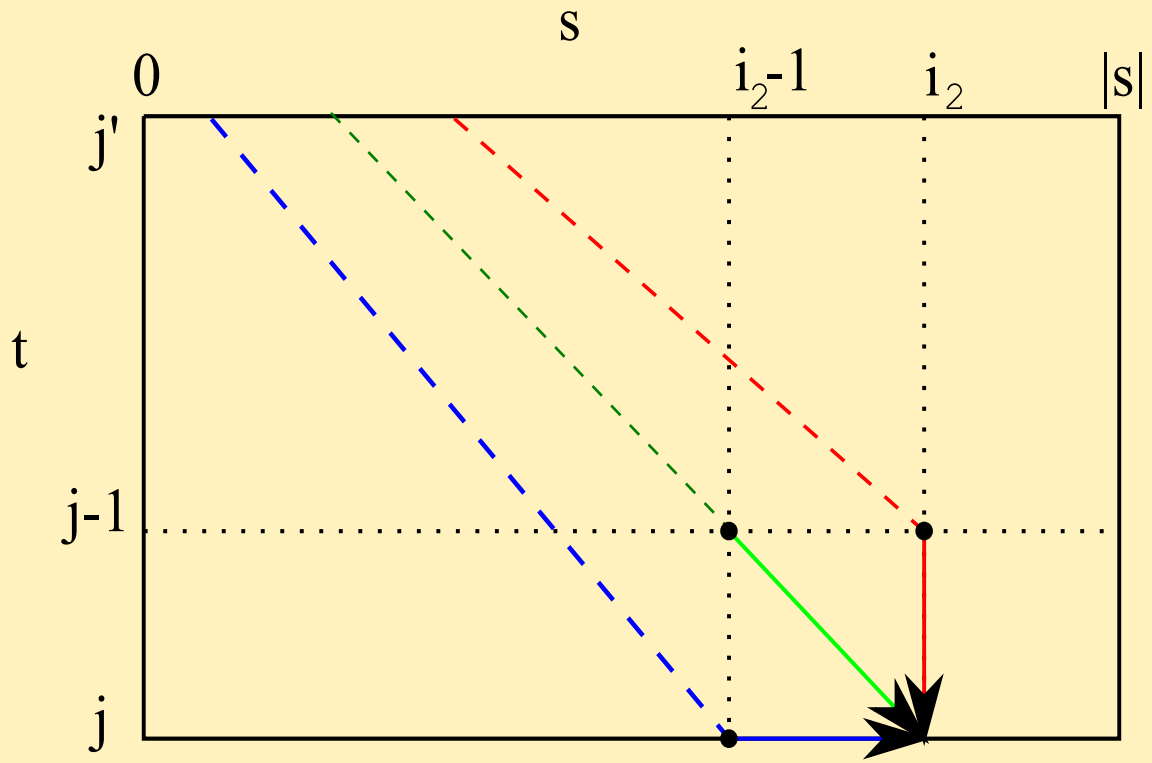
Constrói $\widehat{W_t^C}$

1. Para cada j'
   (a) Para cada j
      i. Para cada j_7
         A. $P_1 \leftarrow$ melhor caminho da linha j' até (j_7, j')
         B. $P_2 \leftarrow$ melhor caminho de (j_7, j) até a linha j
         C. $\widehat{W_t^C}[j', j] \leftarrow \max(\widehat{W_t^C}[j', j], \omega(P_1) + \omega(P_2))$



Constrói $\widehat{W^D_{t|s}}$

1. Para cada j'
   (a) Para cada j
      i. Para cada i_2
         A. $P \leftarrow$ melhor caminho da linha j' até (i_2, j)
         B. $\widehat{W^D_{t|s}}[j', j] \leftarrow \max(\widehat{W^D_{t|s}}[j', j], \omega(P))$



$\mathsf{dup}(t, j', j)$ e $\mathsf{dup}(s, i', i)$

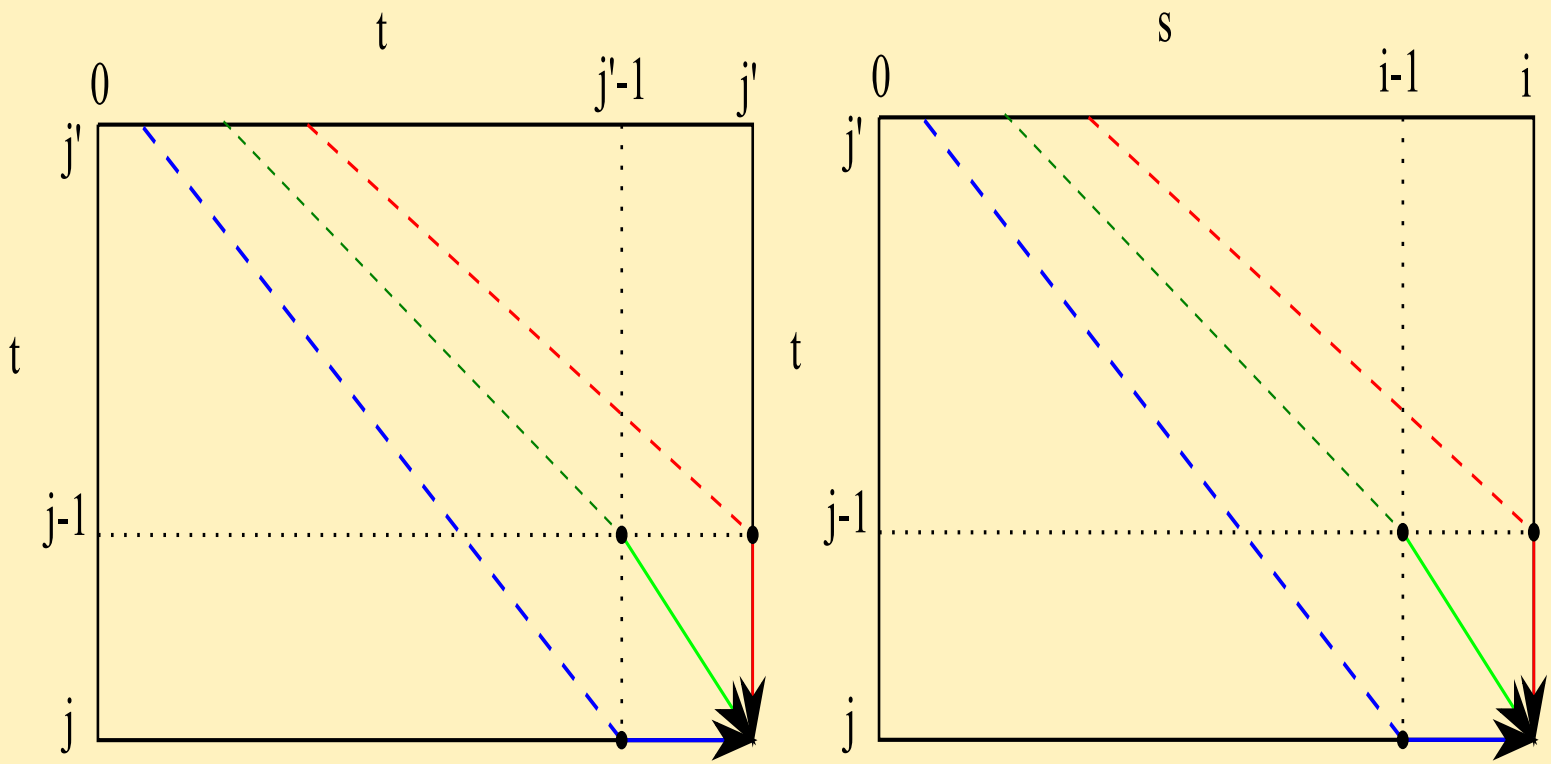
$$\mathsf{dup}(t, j', j) = \max \begin{pmatrix} \widehat{W_t^A}[j'-1, j] \\ \widehat{W_t^B}[j'-1, j] \\ \widehat{W_t^C}[j'-1, j] \\ \widehat{W_{t|s}^D}[j'-1, j] \end{pmatrix} + \omega_u(t[j' \,..\, j]).$$

$$\mathsf{dup}(s, i', i) = \max \begin{pmatrix} \widehat{W_s^A}[i'-1, i] \\ \widehat{W_s^B}[i'-1, i] \\ \widehat{W_s^C}[i'-1, i] \\ \widehat{W_{s|t}^D}[i'-1, i] \end{pmatrix} + \omega_x(s[i' \,..\, i]).$$

Portanto conseguimos construir matrizes que armazenam os pesos das arestas estendidas ($\mathsf{dup}(t, j', j)$ e $\mathsf{dup}(s, i', i)$) em tempo $O(n^3)$. Como cada vértice tem $O(n)$ arestas estendidas que chegam nele, obtemos um alinhamento ótimo com duplicações em tempo $O(n^3)$ e espaço $O(n^2)$.

Duplicações em tandem

- É um subcaso das duplicações com transposição, e o algoritmo é similar ao algoritmo visto anteriormente.
- Não tem o problema dos ciclos.



Trabalhos futuros

- Alinhamento ótimo com inversões não sobrepostas e duplicações
- Implementação e testes para os algoritmos de alinhamento com duplicações
- Pesos para abertura de *gaps* no alinhamento com inversões não sobrepostas
- Rearranjos nas repetições
  - Rearranjos como duplicação e inversões não sobrepostas
- Função ω_{inv} dependente do comprimento da inversão
  - Útil em casos onde um longo trecho tem similaridade próxima a aleatória nos alinhamentos com e sem inversão.
  - Tentar utilizar uma função do tipo $f(l) = l^{\alpha}$, onde l é o tamanho da inversão e α é uma constante.

Trabalhos futuros

- Esparsidade
  - Para certos tipos de dados podemos ter no grafo de edição as arestas verticais, horizontais e muitas diagonais com peso zero.
  - O tempo de execução depende da quantidade de arestas com peso diferente de zero.
  - Pode melhorar significativamente o tempo de execução
- Inversão com um nível de sobreposição
  - Problema surgido na comparação dos cromossomos X e Y do homem.
  - Existe uma teoria que diz que houveram 4 grandes inversões, além das substituições, inserções, remoções e pequenas inversões.

Trabalhos futuros

- Diminuir a memória utilizada pelos algoritmos propostos
  - Os algoritmos propostos utilizam memória quadrática
- Análise de tempos reais de execução dos algoritmos
  - Realizar testes com dados reais
  - Obtenção de tempos estatisticamente confiáveis
  - Testes preliminares mostraram que o algoritmo $O(n^3)$ é mais rápido que o $O(n^3 \log n)$ no alinhamento com inversões não sobrepostas.

Fim

Muito obrigado!