Universidade Federal de Uberlândia
Programa de Pós-Graduação em Ciência da Computação
Análise de Algoritmos
Profª Dra. Márcia Aparecida Fernandes

Alinhamento de Sequências

Ângelo Travizan

Laura Marquez

Leonard Vieira

Alinhamento de sequências

- Objetivo: O problema de alinhamento de sequências consiste em comparar duas ou mais sequências de forma a observar seu nível de similaridade.

  - Recuperação de informação: dada uma chave, buscar em um dicionário por palavras que são semelhantes à chave.

  - Biologia molecular: compara duas sequências de DNA e verifica se são semelhantes.

Alinhamento de sequências

- Uma sequência de DNA ou sequência genética é uma série de letras representando a estrutura primária de uma molécula ou cadeia de DNA, real ou hipotética, com a capacidade de carregar informações;

- As letras possíveis são A, C, G e T, representando os quatro nucleotídeos (subunidades) de uma cadeia de DNA – as bases adenina, citosina, guanina, timina, covalentemente ligadas a uma "coluna vertebral" de fósforo.

Alinhamento de sequências

- Um alinhamento de duas sequências de caracteres X e Y, é obtido inserindo espaços (gap's) nas sequências e então colocando-as uma sobre a outra de modo que cada caractere ou espaço esteja emparelhado a um único caractere (ou a um espaço) da outra cadeia.
- Sequências:

X = AAACTGCACAATCTTAATGCCCTTTTTAT

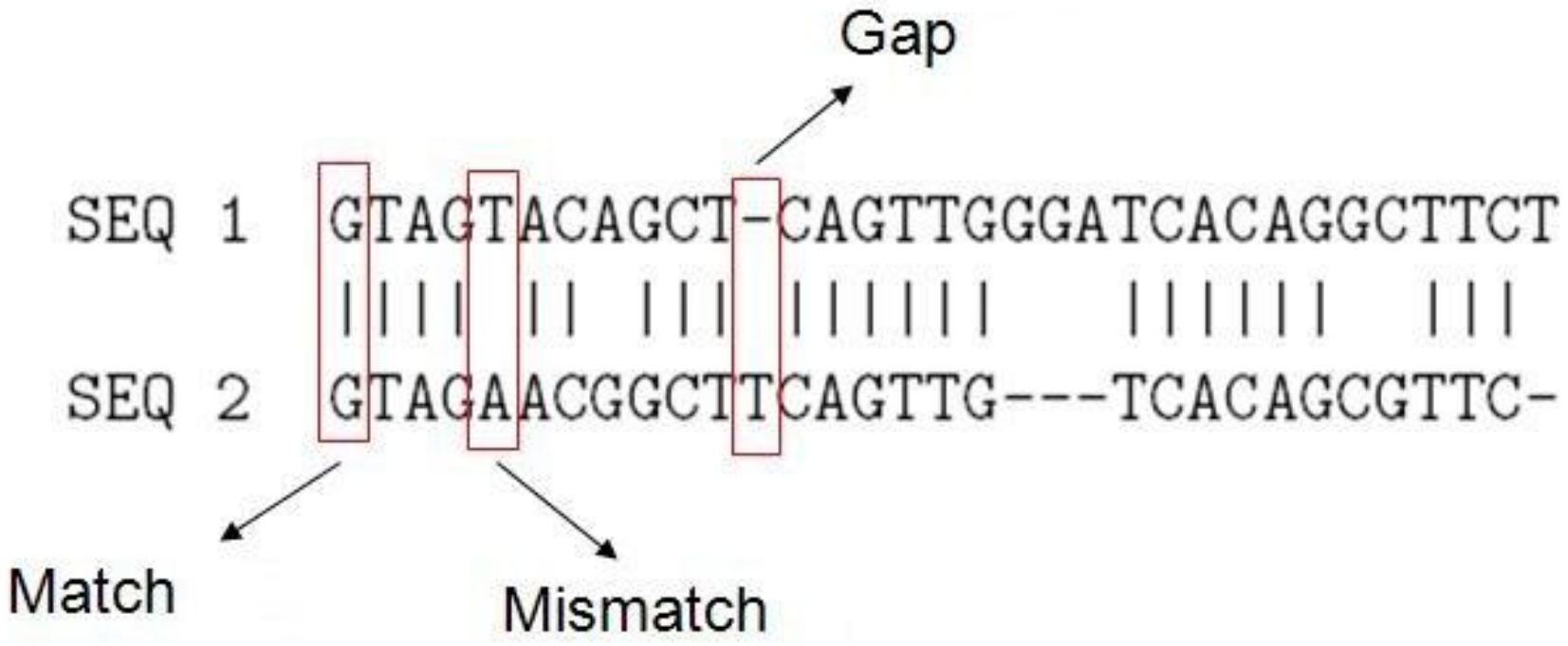
Y = GCGGATCAACTTATTCCATCTCTT

- Alinhamento:

X' = AAACTGCA-CAATCTTAATGCC - - CTTTTTAT

Y' = - - GC-GGATCAA-CTT - ATTCCATCTCTT - -

Alinhamento de sequências



Cálculo do Score de alinhamento

- Score de alinhamento: Soma dos valores associados a cada posição, de acordo com o grau de similaridade entre os elementos correspondentes.

Ex.: match = + 1 (good) (bom)

mismatch = -1 (bad) (ruim)

gap = -2 (worse) (pior)

G A – C G G A T T A G

G A T C G G A A T A G

Score = 9 * 1 + 1 * (-1) + 1 * (-2) = **6**

Alinhamento de sequências

- Para alinhar duas sequências de nucleotídeos é necessário definir quais os parâmetros que serão utilizados para calcular o score do alinhamento;

- O melhor alinhamento será aquele com maior score.

Problema

Alinhar duas cadeias de caracteres de tal maneira que elas tenham maior similaridade:

- Entrada: Duas cadeias de caracteres;
- Saída: Alinhamento das cadeias, possivelmente incluindo gap’s.

Passos da Programação Dinâmica

- Podemos perceber que a programação dinâmica pode resolver este problema, visto que devemos verificar todos os subproblemas (possibilidades) e escolher o melhor alinhamento possível entre sequências.

- Passo I – Subestrutura Ótima: Solução ótima do problema contém soluções ótimas dos subproblemas;
- Passo II – Expressão Recursiva: Especificar expressão recursiva para cálculo do custo da solução ótima;
- Passo III – Algoritmo força bruta (recursivo): Especificar o algoritmo para calcular o custo da solução ótima.

Passo 1 - Subestrutura Ótima

- Dado o problema, alinhar duas cadeias de caracteres de tal maneira que elas tenham maior semelhança, queremos descobrir onde está o **maior score** de alinhamento, pois pretendemos maximizar a similaridade do alinhamento.

- Então, para isso, há três alternativas possíveis para resolver este problema:

I – $(m, n) \in M$

II – a m-ésima posição de $X \notin M$

III – a n-ésima posição de $Y \notin M$

*onde M é o alinhamento

Passo 1 - Subestrutura Ótima

- Caso ocorra o caso I, tem-se:

  $OPT(m,n) = \alpha_{XmYn} + OPT(m -1, n -1)$

- Caso ocorra o caso II, “paga-se” o custo de um intervalo desde a m-ésima posição de X que não foi encontrada e alinhar X_1, X_2, ..., X_{m-1} bem como Y_1, Y_2, ..., Y_n. Desta forma, teremos:

  $OPT(m, n) = \delta + OPT(m -1, n)$

- Caso ocorra o caso III, será semelhante ao caso II, porém: $OPT(m, n) = \delta + OPT(m, n -1)$

Passo 2 – Expressão Recursiva

$$OPT(i,j) \begin{cases} i * \delta & \text{se } j=0 \\ j * \delta & \text{se } i=0 \\ MAX\ [\alpha_{XiYj} + OPT(i-1, j-1),\ \delta + OPT(i-1, j),\ \delta + OPT(i, j-1)] \end{cases}$$

δ – gap (custo pelo espaço)

α_{XiYj} – custo de emparelhar Xi e Yj

i e j – são comprimentos

*Queremos encontrar a máxima similaridade do alinhamento.

Passo 3 – Algoritmo utilizando força bruta (recursivo)

1 - Alinhamento_recursivo(X,m,Y,n)

2 - If m = 0

3 - then return n*δ

4 - If n = 0

5 - then return m*δ

6 - A = α_{XiYj} + Alinhamento_recursivo(X,m-1,Y,n-1)

7 - B = δ + Alinhamento_recursivo(X,m-1,Y,n)

8 - C = δ + Alinhamento_recursivo(X,m,Y,n-1)

9- return max (A,B,C)

Passo 3.1 - Análise do algoritmo alinhamento_recursivo

- A complexidade para o algoritmo alinhamento_recursivo(X, i, Y, j) baseado na expressão recursiva é:

$$T(m,n) = T(m-1, n-1) + T(m, n-1), + T(m-1, n) + \Theta(1)$$

$$T(m,n) \geq 3T(m-1,n-1) + \Theta(1)$$

$$T(m,n) = \Omega(3^{\min(m,n)})$$

- Ordem Exponencial

Passo 3.2 - Algoritmo Utilizando Programação Dinâmica

- Alinhamento_PD(X,Y,δ)

1 - Initialize A[i,0] = i*δ for i = 0, ..., m

2 - Initialize A[0,j] = j*δ for j = 1, ..., n

3 - for i = 1, ..., m

4 - for j = 1, ..., n

5 - A[i,j] = max (α_{XiYj} + A[i -1, j -1], δ + A[i -1, j], δ + A[i, j -1])

6 - end for

7 - end for

8 - return A[m,n]

Passo 3.3 – Análise do algoritmo Alinhamento_pd(X,Y, δ)

- A complexidade para o algoritmo Alinhamento_pd é Θ(m*n), pois é o tempo de preencher a Matriz A;
- Este custo é expresso nas linhas de 3 à 5 do algoritmo Alinhamento_pd(X,Y,δ). As demais linhas tem tempo constante (Θ(1));
- O espaço ocupado é Θ(m*n), pois o tamanho da matriz é (m+1) * (n+1);

- Tempo de execução Θ(m*n).

Algoritmo para mostrar a solução ótima

Algorithm 1 Solução Ótima

```
1:  função ALINHAMENTO(X,Y)
2:      n ← comp(X)
3:      m ← comp(Y)
4:      C(0,0).valor = 0
5:      C(0,0).direcao = " "
6:      para i ← 1 até m faça
7:          C(i,0).valor ← δ · i
8:          C(i,0).direcao ← " ↑ "
9:      fim para
10:     para j ← 1 até n faça
11:         C(0,j).valor ← δ · j
12:         C(0,j).direcao ← " ← "
13:     fim para
14:     para i ← 1 até m faça
15:         para j ← 1 até n faça
16:             aux1 ← α_ij + C(i − 1, j − 1)
17:             aux2 ← δ + C(i, j − 1)
18:             aux3 ← δ + C(i − 1, j)
19:             se aux1 > aux2 e aux1 > aux3 então
20:                 C(i,j).valor ← aux1
21:                 C(i,j).direcao = "↖"
22:             senão
23:                 se aux2 > aux3 então
24:                     C(i,j).valor ← aux2
25:                     C(i,j).direcao = "←"
26:                 senão
27:                     C(i,j).valor ← aux3
28:                     C(i,j).direcao = "↑"
29:                 fim se
30:             fim se
31:         fim para
32:     fim para
33:     retorna C(m,n)
34: fim função
```

- Em alguns algoritmos pode-se passar o Matche e o Mismatche como parâmetros da função principal. No livro da Eva Tardos ela leva em consideração apenas as sequências X e Y.

Alinhamento

Inicialização da 1ª linha e 1ª coluna da matriz C.

$S_{i,0} = w * i$

$S_{0,j} = w * j$

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44
G	-4											
G	-8											
A	-12											
T	-16											
C	-20											
G	-24											
A	-28											

$s(a_i b_j) = +5$ if $a_i = b_j$ (match score)
$s(a_i b_j) = -3$ if $a_i \neq b_j$ (mismatch score)
$w = -4$ (gap)

- X = GAATTCAGTTA; m = 11
- Y = GGATCGA; n = 7
- m+1 linhas, n+1 colunas.

Preenchimento da Matriz

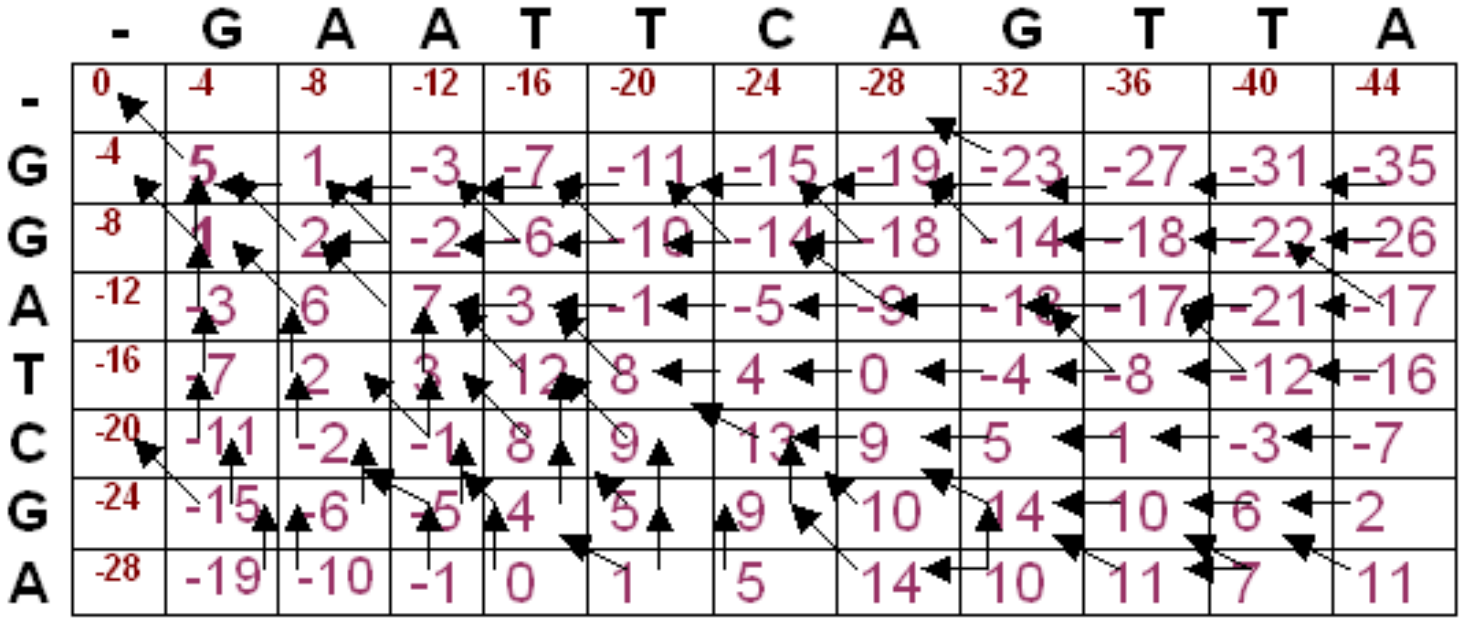
$S_{1,1} = MAX[S_{0,0} + 5, S_{1,0} - 4, S_{0,1} - 4] = MAX[5, -8, -8]$

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44
G	-4	5										
G	-8											
A	-12											
T	-16											
C	-20											
G	-24											
A	-28											

$s(a_i b_j) = +5$ if $a_i = b_j$ (match score)
$s(a_i b_j) = -3$ if $a_i \neq b_j$ (mismatch score)
$w = -4$ (gap)

- X = GAATTCAGTTA; m = 11
- Y = GGATCGA; n = 7
- m+1 linhas, n+1 colunas.

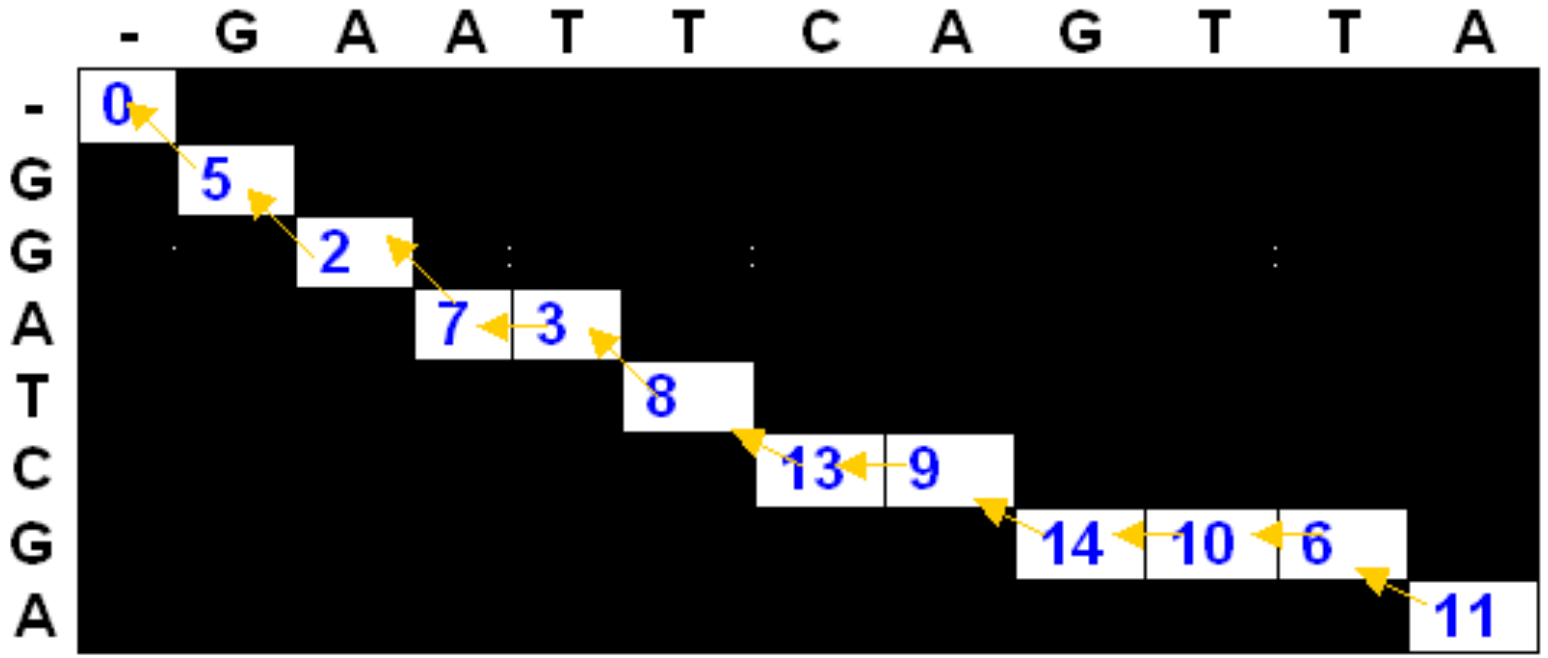
Matriz preenchida



$s(a_i b_j) = +5$ if $a_i = b_j$ (match score)
$s(a_i b_j) = -3$ if $a_i \neq b_j$ (mismatch score)
$w = -4$ (gap)

- X = GAATTCAGTTA; m = 11
- Y = GGATCGA; n = 7
- m+1 linhas, n+1 colunas.

Alinhamento e cálculo do score

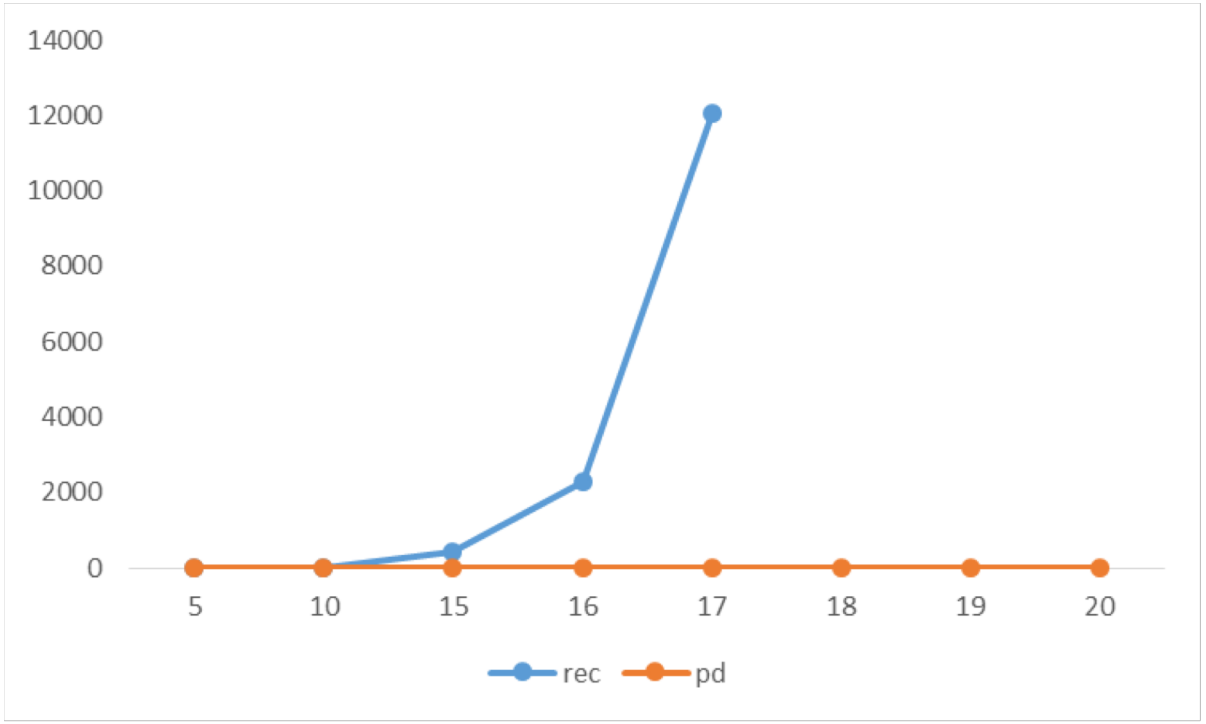


$s(a_i b_j)$ = +5 if $a_i = b_j$ (match score)
$s(a_i b_j)$ = -3 if $a_i \neq b_j$ (mismatch score)
w = -4 (gap)

```
G A A T T C A G T T A
|   |   | |   |     |
G G A - T C - G - - A
```

Score: 5 - 3 + 5 - 4 + 5 + 5 - 4 + 5 - 4 - 4 + 5 = **11**

Resultados



entrada	rec	pd
5	0,01	0,011
10	0,086	0,011
15	406,265	0,01
16	2290,402	0,011
17	12069,45	0,012
18		0,011
19		0,01
20		0,011