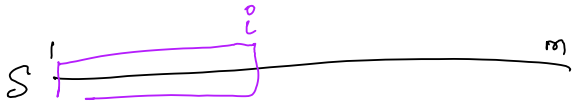
Global Alignment (using DP)

Friday, January 29, 2021 10:33 AM



$\begin{bmatrix} 0 \le i \le m \\ 0 \le j \le n \end{bmatrix}$ $m+1$ $\times$ $n+1$

Use Dynamic Programming.

Definitions:

Prefix i of $S = a_1 a_2 \dots a_i$

$\rightarrow$ let $T(i,j)$ $\leftarrow$ the optimal score for aligning $\begin{pmatrix} S_1: a_1 a_2 \dots a_i \\ S_2: b_1 b_2 \dots b_j \end{pmatrix}$

$T(m,n)$ $\leftarrow$ the opt. score for aligning $\begin{pmatrix} S_1: a_1 a_2 \dots a_m \\ S_2: b_1 b_2 \dots b_n \end{pmatrix}$

↙ Answer

Initialization:

$T(0,0) = 0$

$T(i,0) = i \times g$

$T(0,j) = j \times g$

$\begin{pmatrix} a_1 a_2 \dots a_i \\ | \; | \\ - \; - \end{pmatrix}$ i deletions $\equiv i \times g$

$\begin{pmatrix} - \; - \dots - \\ b_1 \; b_2 \dots b_j \end{pmatrix} \equiv j \times g$

Recurrence:

$i>0,\ j>0$

$$T(i,j) = \max \begin{cases} T(i-1,j-1) + \delta(a_i, b_j) \\ T(i-1,j) + g \\ T(i,j-1) + g \end{cases}$$

An alignment bet^n $S_1[1 \dots i]$ $S_2[1 \dots j]$ must end in only one of these three possibilities

Substitution case: $\begin{pmatrix} a_1 \; a_2 \dots & a_{i-1} \\ b_1 \; b_2 \dots & b_{j-1} \end{pmatrix} \begin{bmatrix} a_i \\ \text{or} \\ - \end{bmatrix} \begin{bmatrix} - \\ \text{or} \\ b_j \end{bmatrix}$

Deletion case: $\begin{pmatrix} a_1 \; a_2 \dots & a_{i-1} \\ b_1 & b_j \end{pmatrix} \begin{matrix} a_i \\ - \end{matrix}$

Insertion case: $\begin{pmatrix} a_1 \; a_2 \dots & a_i \\ b_1 \; b_2 \dots & b_{j-1} \end{pmatrix} \begin{matrix} - \\ b_j \end{matrix}$

Output:

$T(m,n)$

↑ This is the optimal **global** alignment score for aligning S_1 & S_2.

Algorithm by: Needleman-Wunsch, 1970

```
int δ(a_i, b_j)
{ if (a_i == b_j)
    return ma_i;
  (else)
    return mi_i;
}
```

Needleman-Wunsch algorithm, 1970

Monday, February 1, 2021 10:05 AM

Initialization:

$T(0,0) = 0$

$T(i,0) = i \times g$

$T(0,j) = j \times g$

Recurrence:

$(0 \le i \le m)$
$(0 \le j \le n)$

$$T(i,j) = \max \begin{cases} T(i-1,j-1) + \delta(a_i, b_j) \\ T(i-1,j) + g \\ T(i,j-1) + g \end{cases}$$

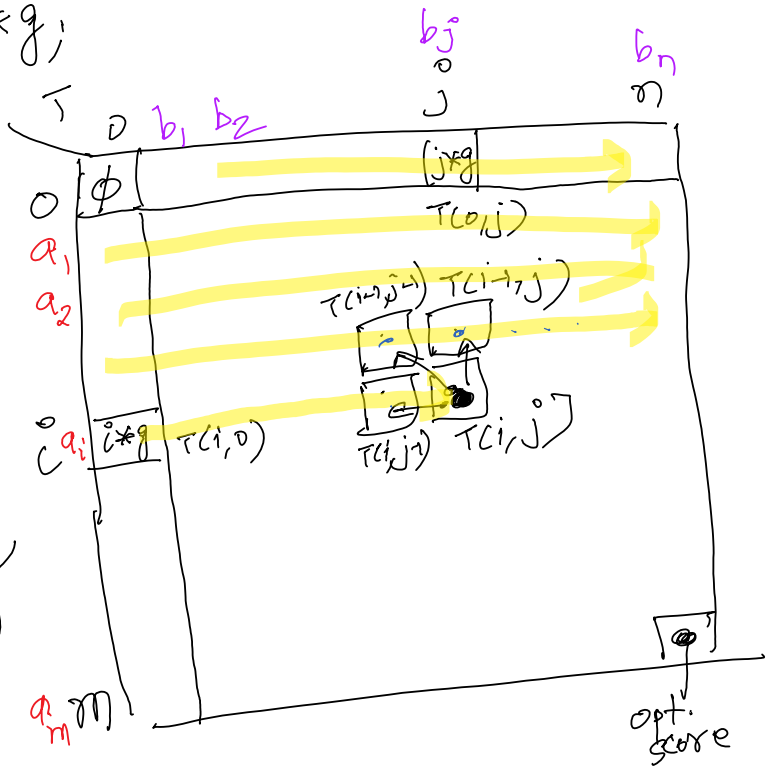
$a_i = S_1[i-1]$
$b_j = S_2[j-1]$

Init:

```
int T[m+1][n+1];
T[0][0] = 0
for (i = 0 to m)  T[i][0] = i*g;
for (j = 0 to n)  T[0][j] = j*g;
```

Main Recurrence:

```
// row major traversal
for (i = 1 to m)   // for each row
    for (j = 1 to n)   // for each col. in row i
        T[i][j] = max { T[i-1][j-1] + δ(a_i, b_j)
                      { T[i-1][j] + g
                      { T[i][j-1] + g
Output T[m][n];
```



Complexity:

Time: $O(mn)$

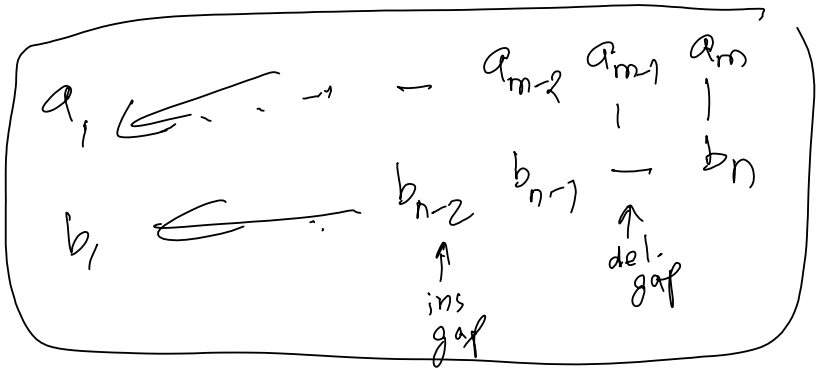
Space: $O(mn + m + n)$

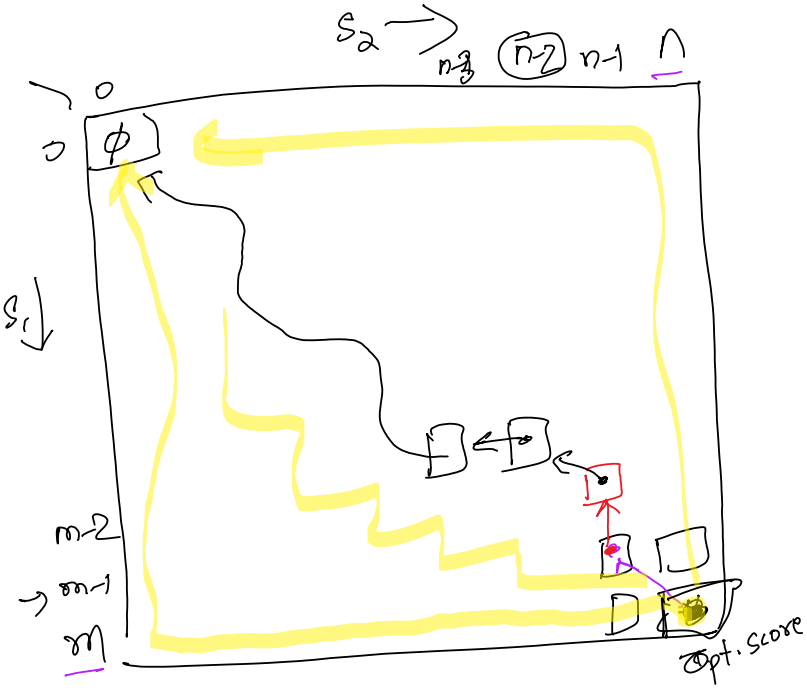
Global Alignment (Needleman-Wunsch algorithm)

Monday, February 1, 2021 10:05 AM

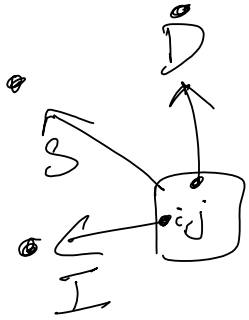
Optimal Path Computation:

$$T(i,j) = \max \begin{cases} T(i-1,j-1) + \delta(a_i, b_j) \\ T(i-1,j) + g \\ T(i,j-1) + g \end{cases}$$





Time = $O(m+n)$

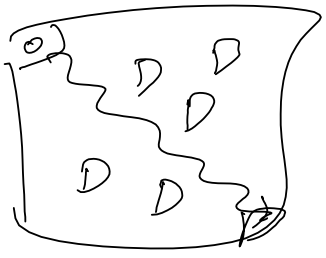


Forward Computation:

$O(mn)$

Reverse / Retrace Computation:

$O(m+n)$

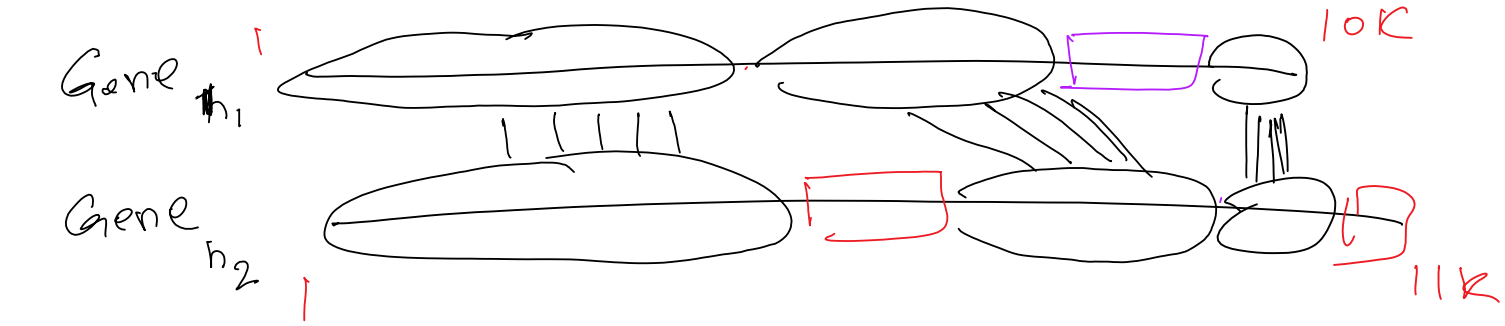


s_1: 0 2 5 10 2 3 4 15

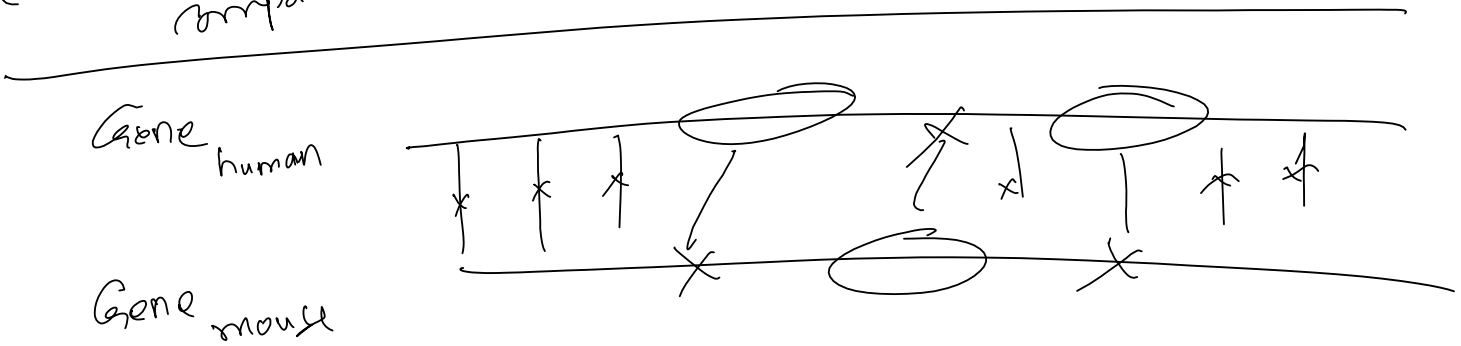
s_2: 1 5 10 3 8 7 15

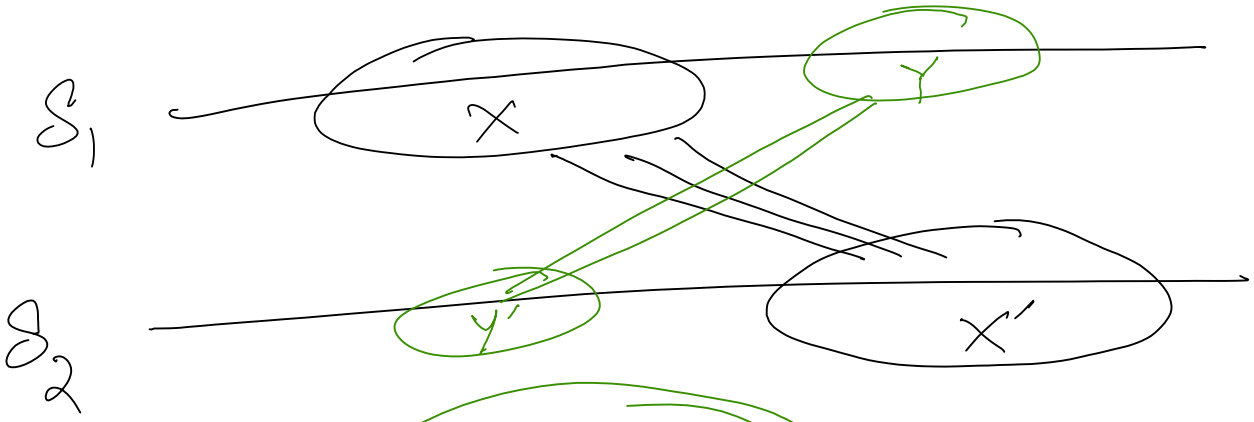
Global Alignment (Needleman-Wunsch algorithm)

Monday, February 1, 2021 10:05 AM



(intra-organism comparison)





need another way to model the alignment → local

Global vs. Local alignments

Wednesday, February 3, 2021 11:04 AM

Examples:

Ex: 1

s_1 : a a c c g t a t t g c a g a t a t a t a c a c g

s_2 : a a g c g t a t c g c a g a t a - - - c a c g

deletion gaps

⇒ global alignment will work just fine for this

Ex: 2

s_1 : a a c c g t a t t g c a g g t a t a t a c a c g

s_2 : a a g c g a t a c a c g g t a t c g c a

(global alignment won't be a good choice here)

local alignment