Alineación de secuencias

Alineación de secuencias

Entrada:

- Dos cadenas $X = \{x_1, x_2, \ldots, x_M\}$, $Y = \{y_1, y_2, \ldots, y_N\}$ de algún alfabeto determinado (*G, T, C, A* por ejemplo)
- Una penalidad positiva α por insertar un vacío (gap)
- Una penalidad positiva β por desigualdad de caracteres

Salida: Alineación (secuencia de inserción de vacíos en *X*, *Y*) que minimice la penalidad total.

Ejemplo: Si X = *GATACA*, Y= *AACT*, α = 1, y β= *1.5*

G A T A C A
– A – A C T

con penalidad total de 1+1+1.5=3.5

Solución mediante programación dinámica

¿Qué forma debería tener la solución óptima de un subproblema?

Dados los primeros *i* elementos de *X* y los *j* primeros elementos de *Y* la solución óptima para ese subproblema consistiría en elegir una de las siguientes opciones:

i primeros elementos de *X* + vacíos	?
j primeros elementos de *Y* + vacíos	?

1. Incluir el elemento *i+1* de *X* y el *j+1* de *Y*, sumar la penalidad resultante a la mínima posible obtenida con los previos *i* elementos de *X* y los previos *j* elementos de Y
2. Incluir el elemento *i+1* de *X* y un vacío en *Y*, sumar la penalidad resultante a la mínima posible obtenida con los previos *i* elementos de *X* y los previos *j-1* elementos de Y
3. Incluir un vacío en *X* y el elemento *j+1* de *Y*, sumar la penalidad resultante a la mínima posible obtenida con los previos *i-1* elementos de *X* y los previos *j* elementos de Y

Solución mediante programación dinámica

```
for i = 0 to M:
    P_{i,0} = i*α
for j = 0 to N:
    P_{0,j} = j*α
for i = 1 to M
    for j = 1 to N:
```

$$P_{i,j} = \text{MIN}\begin{cases} P_{i-1,j-1} + \beta \text{ if } X_i \neq Y_j, \text{ else } P_{i-1,j-1} \\ P_{i,j-1} + \alpha \\ P_{i-1,j} + \alpha \end{cases}$$

Ejemplo: para $X = ATA$, Y = TAC, $\alpha = 1$, y $\beta = 3$

j Caracteres de Y

i Caracteres de X

			T	A	C
		0	1	2	3
	0	0	1	2	3
A	1	1	2	1	2
T	2	2	1	2	3
A	3	3	2	1	**2**

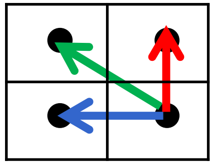
¿Cuál debe ser la solución para los casos base?, es decir, para $P_{i,0}$ y $P_{0,j}$?

$i * \alpha$, $j * \alpha$ respectivamente

¿Cuál es la eficiencia de este algoritmo? O(MN)

Backtracking para obtener la subsecuencia

```
if Xi = Yj:
   Xi + Yj
else:
   MIN(Pi-1,j-1, Pi,j-1 → Yj + gap en X, Pi-1,j → Xi + gap en Y)
```



Si i = 0 ó j = 0 se ‘rellenan’ los espacios sobrantes de la cadena que ya se terminó con vacíos

Caracteres de Y

Caracteres de X i

			T	A	C
		0	1	2	3
	0	0	1	2	3
A	1	1	2	1	2
T	2	2	1	2	3
A	3	3	2	1	**2**

_	T	A	C
A	T	A	_