DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Alineación de secuencias
Análisis y Diseño de Algoritmos

Aplicaciones

Aplicaciones

- Bioinformática:
  Alineación de secuencias de ADN
- Ingeniería del Software:
  Detección de clones (fragmentos de código duplicado)
- Procesamiento del lenguaje natural:
  Reconocimiento de voz, correctores ortográficos…

Alineación de secuencias

Problema

Dadas dos secuencias, $X = (x_1\ x_2\ \ldots\ x_m)$ e $Y = (y_1\ y_2\ \ldots\ y_n)$, encontrar la forma de alinearlas con un coste mínimo.

¿Cómo medimos ese coste?

- δ [gap penalty], cuando en una cadena no aparece un símbolo que sí está en la otra.
- α_{pq} [mismatch penalty], cuando en la cadena X aparece el símbolo p pero en la cadena Y aparece q.

Alineación de secuencias

Ejemplos

C	T	G	A	C	C	T	A	C	C	T
C	C	T	G	A	C	T	A	C	A	T

$\alpha_{TC} + \alpha_{GT} + \alpha_{AG} + 2\alpha_{CA}$

-	C	T	G	A	C	C	T	A	C	C	T
C	C	T	G	A	C	-	T	A	C	A	T

$2\delta + \alpha_{CA}$

Alineación de secuencias

Formalización

- Una alineación M es un conjunto de pares (x_i,y_j) tal que cada uno de los elementos x_i e y_j aparece como mucho en un par y no se produce ningún cruce.
- Los pares (x_i,y_j) y $(x_{i'},y_{j'})$ se cruzan si $i<i'$ pero $j>j'$.

- El coste de la alineación M, por tanto, viene dado por:

$$\text{coste}(M)= \underbrace{\sum_{(x_i,y_j)\in M} \alpha_{x_i y_j}}_{\text{errores}} + \underbrace{\sum_{x_i \notin M} \delta + \sum_{y_j \notin M} \delta}_{\text{huecos}}$$

Alineación de secuencias

Ejemplo

CTACCG vs. TACATG

x_1	x_2	x_3	x_4	x_5		x_6
C	T	A	C	C	-	G
-	T	A	C	A	T	G
	y_1	y_2	y_3	y_4	y_5	y_6

Alineación M = { (x_2,y_1), (x_3,y_2), (x_4,y_3), (x_5,y_4), (x_6,y_6) }

Coste de la alineación: coste(M) = $2\delta + \alpha_{CA}$

Definición recursiva

OPT(i, j): Coste mínimo de alineación de las secuencias $X_i=(x_1\ x_2\ \dots\ x_i)$ e $Y_j=(y_1\ y_2\ \dots\ y_j)$.

- Caso 1: (x_i,y_j) está en la mejor alineación M_{OPT}
  Hay que pagar el coste de emparejar x_i con y_j, a lo que habrá que añadir el coste de alinear las secuencias $X_{i-1}=(x_1\ x_2\ \dots\ x_{i-1})$ e $Y_{j-1}=(y_1\ y_2\ \dots\ y_{j-1})$.
- Caso 2a: M_{OPT} deja x_i sin emparejar
  Hay que pagar una penalización δ más el coste de alinear las cadenas $X_{i-1}=(\ x_1\ x_2\ \dots\ x_{i-1})$ e $Y_j=(y_1\ y_2\ \dots\ y_j)$.
- Caso 2b: M_{OPT} deja y_j sin emparejar
  Penalización δ más el coste de alinear las cadenas $X_i=(\ x_1\ x_2\ \dots\ x_i)$ e $Y_{j-1}=(y_1\ y_2\ \dots\ y_{j-1})$.

Definición recursiva

OPT(i, j): Coste mínimo de alineación de las secuencias $X_i=(x_1\ x_2\ \dots\ x_i)$ e $Y_j=(y_1\ y_2\ \dots\ y_j)$.

$$OPT(i,j)=\begin{cases} j\delta & \text{si } i=0 \\ \min\begin{cases}\alpha_{x_i y_j}+OPT(i-1,j-1)\\ \delta+OPT(i-1,j)\\ \delta+OPT(i,j-1)\end{cases} & \text{en otro caso} \\ i\delta & \text{si } j=0 \end{cases}$$

Implementación

Algoritmo basado en programación dinámica

```
SequenceAlignmentDP ( X, Y, δ, α)
{
   for i = 0 to X.length
      M[0, i] = i*δ;
   for j = 0 to Y.length
      M[j, 0] = j*δ;

   for i = 1 to X.length
      for j = 1 to Y.length
         M[i, j] = min ( α[X[i]][Y[j]] + M[i-1][j-1],
                         δ + M[i-1][j],
                         δ + M[i][j-1]);
   return M[X.length][Y.length];
}
```


Implementación

Algoritmo basado en programación dinámica

Eficiencia

- Tiempo: $\Theta(|X||Y|)$
- Espacio: $\Theta(|X||Y|)$

Aplicaciones

- Reconocimiento de voz con DTW, |X|<10, |Y|<100, OK.
- Biología computacional: |X|=|Y|=100000
  - 10^{10} operaciones, OK
  - Array con 10^{10} entradas > 10 GB !!!

Optimización

¿Podemos evitar que nuestro algoritmo sea cuadrático con respecto al espacio de almacenamiento que requiere?

Es fácil si sólo nos interesa el coste de la alineación:

Calculamos el valor OPT(i,·) a partir de OPT(i-1,·).

Pero ya no podremos alinear las secuencias, ya que no disponemos de la matriz M…

Optimización

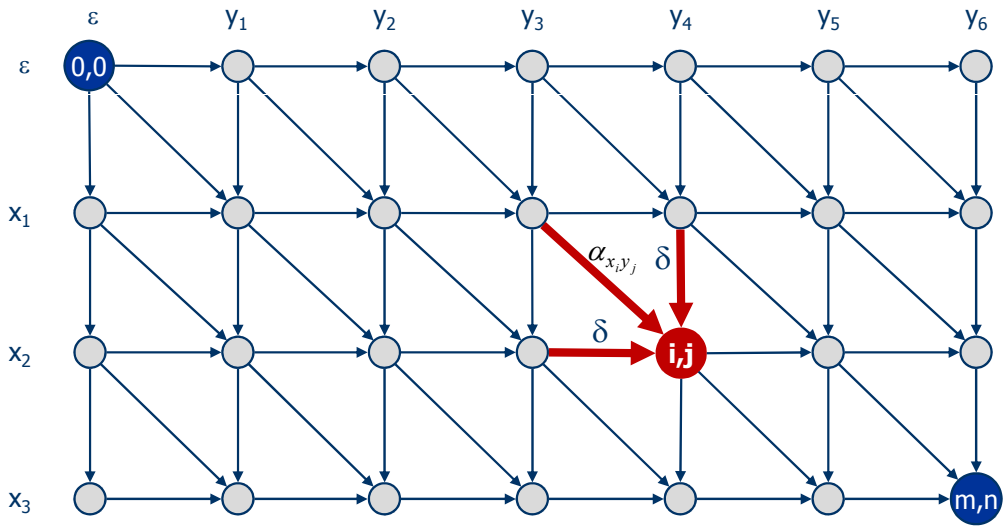
¿Podemos obtener la alineación óptima de dos secuencias con un algoritmo de orden cuadrático en tiempo pero lineal en espacio?

Sí, si combinamos nuestro algoritmo basado en programación dinámica con la técnica "divide y vencerás". [Hirschberg 1975].

Alineación de secuencias

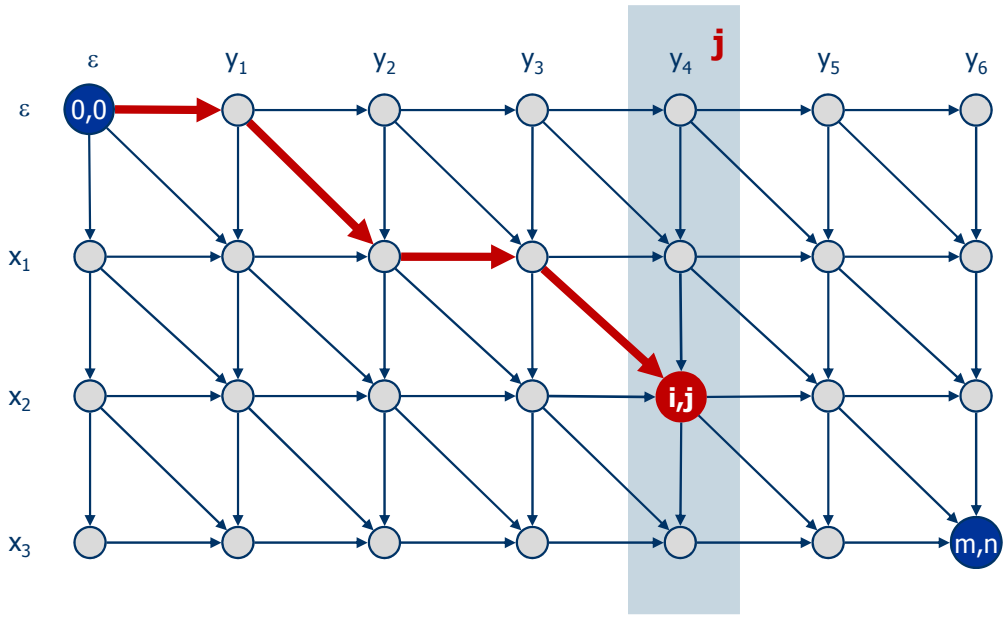
Sea f(i,j) el camino más corto de (0,0) a (i,j).

- Observación: f(i,j) = OPT(i,j), m=|X|, n=|Y|



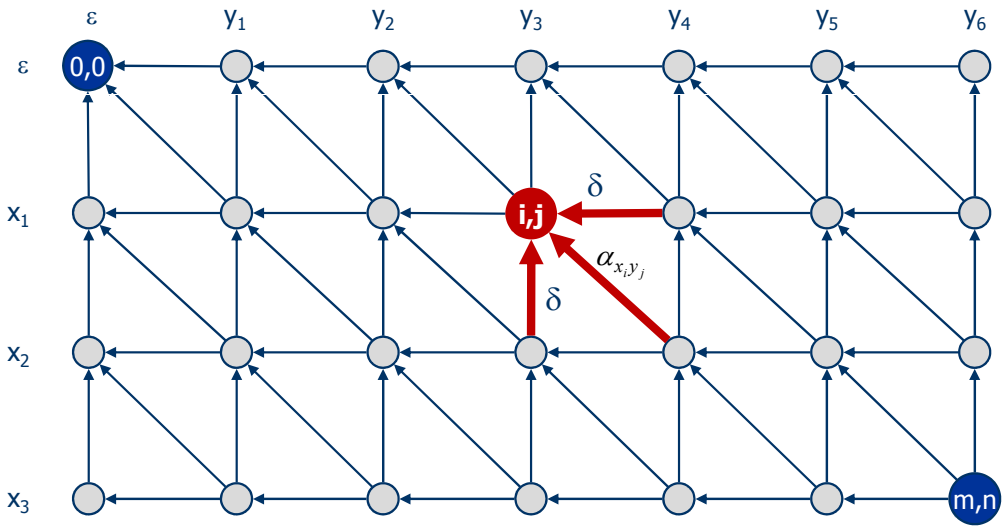
Alineación de secuencias

"Forward alignment": Podemos calcular **f(·,j)** para cualquier j en tiempo O(mn) y espacio O(m+n).



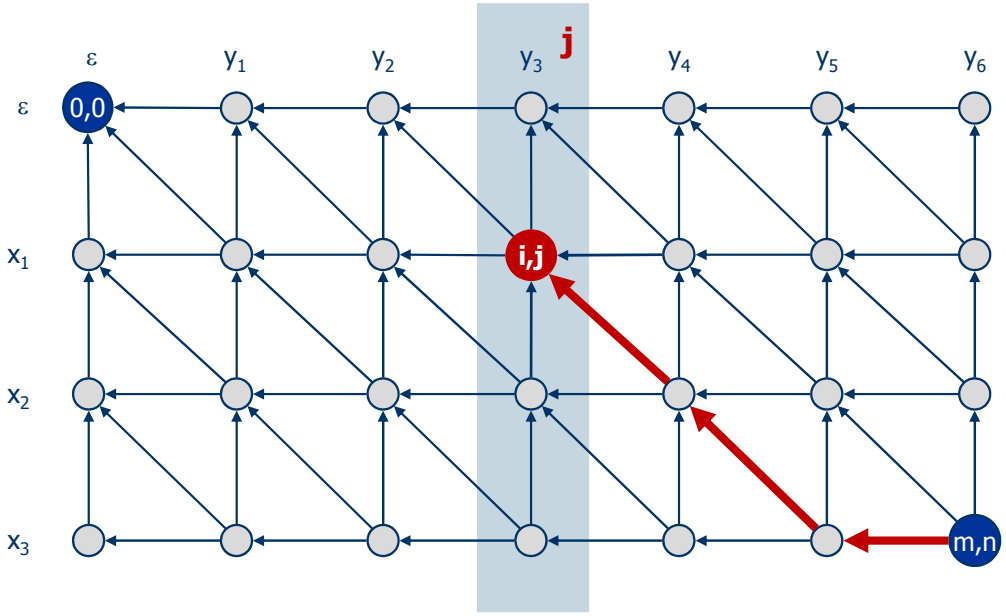
Alineación de secuencias

Sea g(i,j) el camino más corto de (m,n) a (i,j):
Se puede calcular invirtiendo los arcos del grafo anterior…



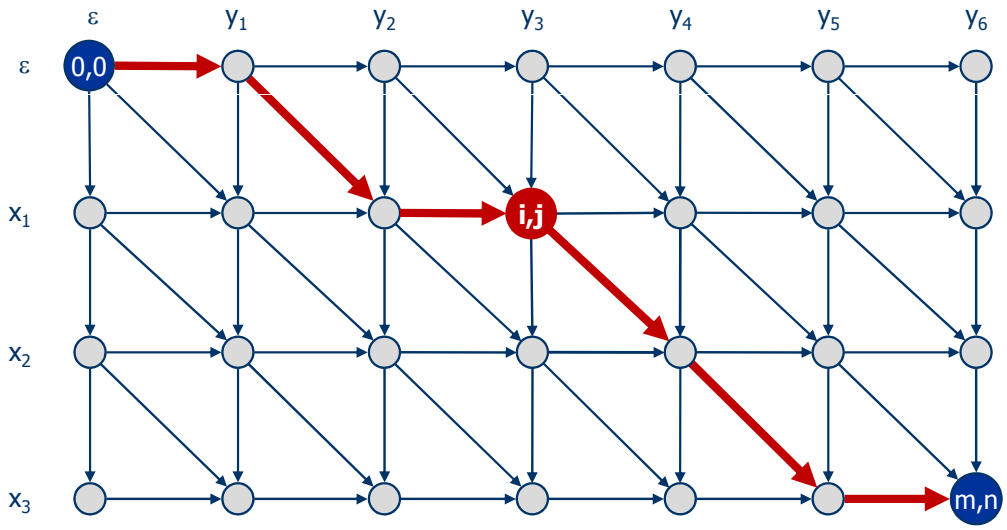
Alineación de secuencias

Backward alignment: Podemos calcular **g(·,j)**
para cualquier j en tiempo O(mn) y espacio O(m+n).



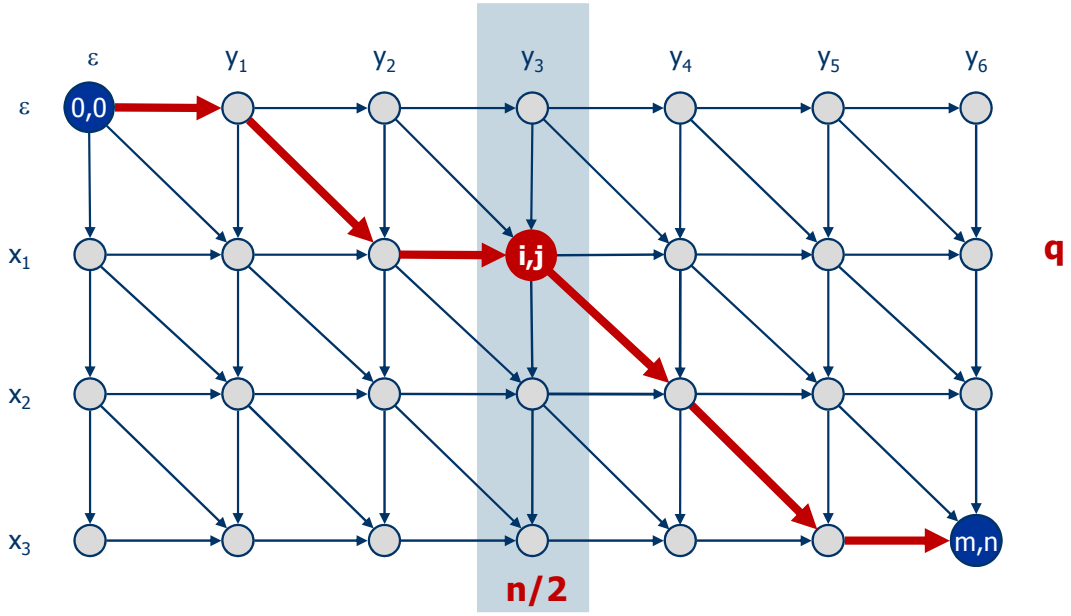
Alineación de secuencias

El coste del camino mínimo
que pasa por (i,j) es **f(i,j) + g(i,j)**



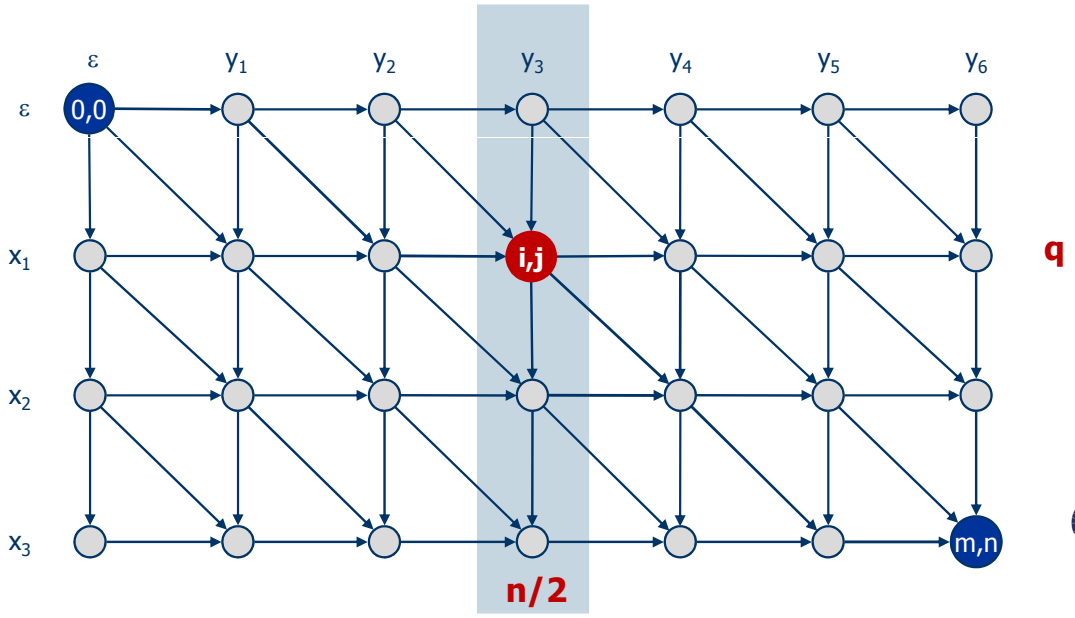
Alineación de secuencias

Sea q el índice que minimiza f(q, n/2) + g(q, n/2):
El camino más corto de (0,0) a (m,n) pasa por **(q,n/2)**.



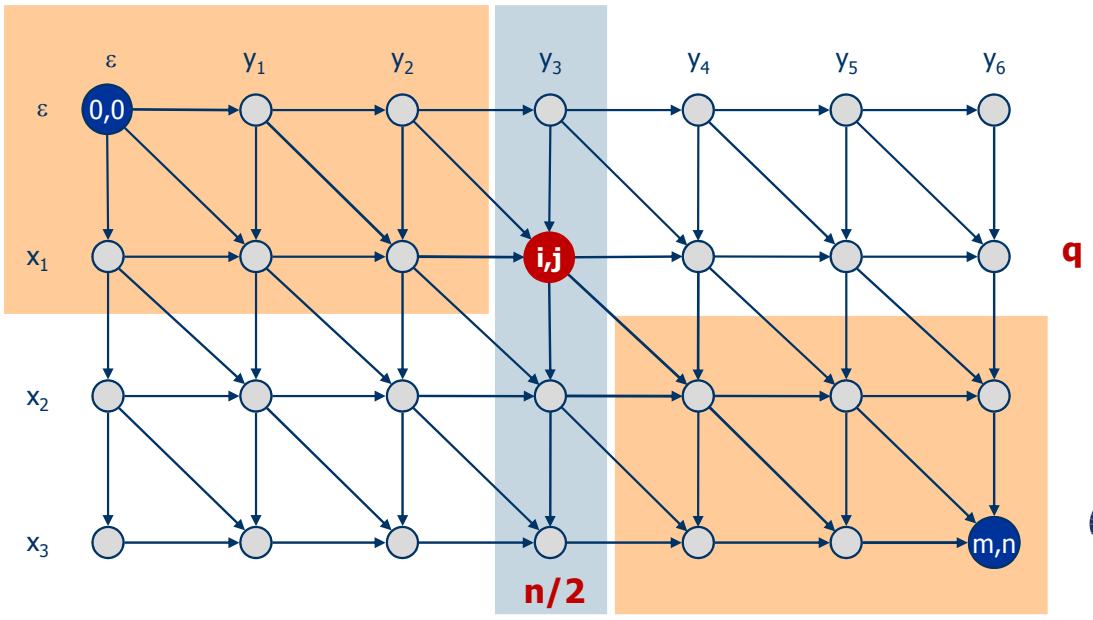
Algoritmo "divide y vencerás"

1. Encontrar el índice q que minimiza f(q,n/2)+g(q,n/2) usando programación dinámica: Alinear x_q con $y_{n/2}$



Algoritmo "divide y vencerás"

2. Calcular recursivamente la mejor alineación de los segmentos a la izquierda y a la derecha de x_q e $y_{n/2}$.



Algoritmo "divide y vencerás"

Eficiencia del algoritmo "divide y vencerás"

Sea T(m,n) el tiempo requerido por el algoritmo para alinear dos secuencias de longitud m y n:

$$T(m,n) \leq 2T(m,n/2) + O(mn) \Rightarrow T(m,n) \in O(mn\log n)$$

Este análisis no es preciso porque los subproblemas no son de tamaño (m,n/2), sino de tamaño (q, n/2) y (m-q, n/2).

Algoritmo "divide y vencerás"

Eficiencia del algoritmo "divide y vencerás"

En realidad, T(m,n) es **O(mn)**

- Tiempo O(mn) en calcular f(•, n/2) y g (•, n/2) para descubrir q.
- T(q, n/2) + T(m - q, n/2) para las llamadas recursivas:

Demostración por inducción sobre n

- Casos base: m=2 ó n=2.
- Hipótesis inductiva: T(m, n) ≤ 2cmn.

$$\begin{aligned} T(m,\ 2) &\leq cm \\ T(2,\ n) &\leq cn \\ T(m,\ n) &\leq cmn + T(q,\ n/2) + T(m-q,\ n/2) \end{aligned}$$

$$\begin{aligned} T(m,n) &\leq T(q,n/2) + T(m-q,n/2) + cmn \\ &\leq 2cqn/2 + 2c(m-q)n/2 + cmn \\ &= cqn + cmn - cqn + cmn \\ &= 2cmn \end{aligned}$$

Alineación de secuencias

Algoritmo	Tiempo	Espacio	Resultado
Programación dinámica	**O(mn)**	**O(mn)**	Coste y alineación
Forward alignment	O(mn)	O(m+n)	Coste
Backward alignment	O(mn)	O(m+n)	Coste
Divide y vencerás	**O(mn)**	**O(m+n)**	Coste y alineación