Viaje por el rio

Viaje por el rio

Entrada: Tabla de costos (valores positivos) para viajar entre *N* embarcaderos en un rio que solo se puede navegar a favor de la corriente

Salida: Menor costo para viajar desde el primer hasta el último embarcadero

* La idea es que puede ocurrir que un viaje entre *i* y *j* salga más barato haciendo escala en *k* embarcaderos que yendo directamente.

Ejemplo: Con *N* = 4

Solución: A-C-D con costo total 3

	A	B	C	D
A	0	1	2	4
B		0	3	5
C			0	1
D				0

Solución mediante programación dinámica

Para resolver mediante programación dinámica ¿Qué forma debería tener la solución óptima de un subproblema?

Para viajar entre dos embarcaderos *i, j* ($1 \leq i < j \leq N$) la solución óptima para ese subproblema consistiría en elegir una de las siguientes opciones:

Viajar directamente de *i* a *j* con costo $C_{i,j}$

ó

Viajar de *k* a *j*, y sumar el costo $C_{k,j}$ a la solución óptima de viajar de i a k (para todo $i < k < j$)

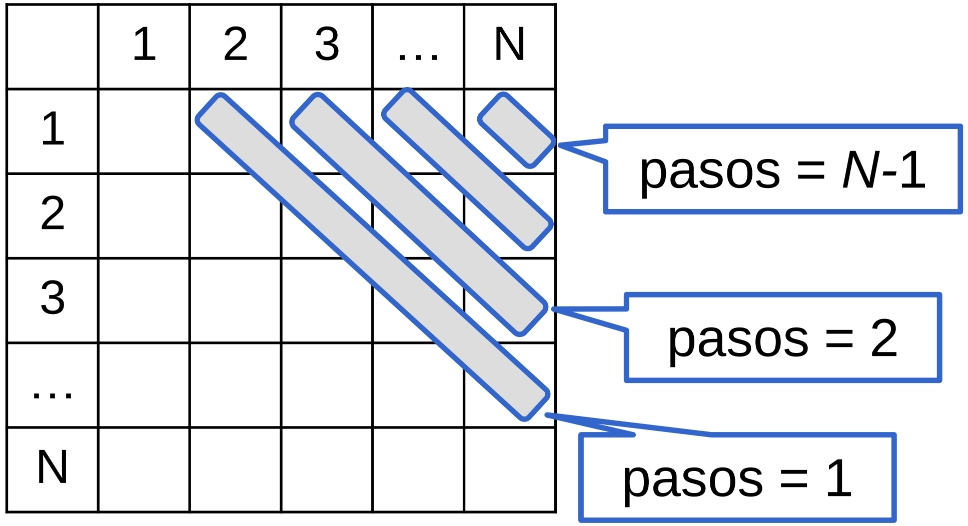
Solución mediante programación dinámica

Finalmente con esta relación podemos esquematizar una solución mediante programación dinámica, sin embargo debemos preguntarnos: ¿Cuáles son los subproblemas más pequeños y cómo van incrementándose sistemáticamente el tamaño de los subproblemas siguientes?

Los subproblemas más pequeños son cuando hay un solo "paso" entre i y j, es decir, cuando $j-i=1$, los subproblemas que le siguen son cuando hay dos pasos, es decir, cuando $j-i = 2$; y así sucesivamente hasta que el subproblema mayor (el original) es cuando hay $N-1$ pasos.

Solución mediante programación dinámica

```
for pasos = 1 to N-1:
   for i = 1 to N-pasos:
      j = i+pasos
      menor = costo_{i,j}
      for k = i+1 to j-1:
         menor = min(menor, M_{k,j} + M_{i,k})
      M_{i,j} = menor
print(M_{1,N})
```



Ejemplo:

Costo

	A	B	C	D
A	0	1	2	4
B	∞	0	3	5
C	∞	∞	0	1
D	∞	∞	∞	0

M

	A	B	C	D
A		1	2	**3**
B			3	4
C				1
D				

¿Cuál es la eficiencia de este algoritmo? $O(N^3)$, mejor que $O(2^{N-2})$ de búsqueda exhaustiva