Programación III

Ricardo Wehbe

UADE

21 de septiembre de 2021

Programa

1. Repaso de la clase anterior
   - Introducción a la programación dinámica
   - El problema del cambio
   - El problema de la mochila 0-1
2. Programación dinámica. Algunos ejemplos
   - El algoritmo de Floyd-Warshall
   - Subsecuencia común más larga
   - Subsecuencia creciente más larga
3. Ejercicios propuestos

Those who cannot remember the past are condemned to repeat it.

(Aquellos que no pueden recordar el pasado están condenados a repetirlo.)

George Santayana, *The Faces of Human Progress*

Introducción a la programación dinámica

Introducción a la programación dinámica

- La técnica de *programación dinámica* se basa en la resolución de problemas a través de su descomposición en subproblemas más pequeños del mismo tipo.

Introducción a la programación dinámica

- La técnica de *programación dinámica* se basa en la resolución de problemas a través de su descomposición en subproblemas más pequeños del mismo tipo.
- ¿Pero no es esto lo que ya hemos hecho con *divide & conquer*?

Introducción a la programación dinámica

- La técnica de *programación dinámica* se basa en la resolución de problemas a través de su descomposición en subproblemas más pequeños del mismo tipo.
- ¿Pero no es esto lo que ya hemos hecho con *divide & conquer*?
- En esencia, sí. La diferencia es que cuando los subproblemas no son independientes, la eficiencia de *divide & conquer* puede no ser aceptable.

Introducción a la programación dinámica

- La técnica de *programación dinámica* se basa en la resolución de problemas a través de su descomposición en subproblemas más pequeños del mismo tipo.
- ¿Pero no es esto lo que ya hemos hecho con *divide & conquer*?
- En esencia, sí. La diferencia es que cuando los subproblemas no son independientes, la eficiencia de *divide & conquer* puede no ser aceptable.
- En el caso de la programación dinámica, los subproblemas no son independientes. Hay subproblemas comunes, cuya resolución se almacena para su reutilización.

Algunas consideraciones sobre programación dinámica

Algunas consideraciones sobre programación dinámica

- Como en el caso de *divide & conquer*, esta técnica se aplica a problemas cuya solución óptima es una combinación de soluciones óptimas a subproblemas.

Algunas consideraciones sobre programación dinámica

- Como en el caso de *divide & conquer*, esta técnica se aplica a problemas cuya solución óptima es una combinación de soluciones óptimas a subproblemas.
- Esto se llama a veces *principio de optimalidad*. Si este principio se aplica al problema que queremos resolver, la programación dinámica o *divide & conquer* son dos buenos candidatos.

Algunas consideraciones sobre programación dinámica

- Como en el caso de *divide & conquer*, esta técnica se aplica a problemas cuya solución óptima es una combinación de soluciones óptimas a subproblemas.
- Esto se llama a veces *principio de optimalidad*. Si este principio se aplica al problema que queremos resolver, la programación dinámica o *divide & conquer* son dos buenos candidatos.
- Si los subproblemas tienen elementos en común, la programación dinámica es la técnica más prometedora.

Algunas consideraciones sobre programación dinámica

- Como en el caso de *divide & conquer*, esta técnica se aplica a problemas cuya solución óptima es una combinación de soluciones óptimas a subproblemas.
- Esto se llama a veces *principio de optimalidad*. Si este principio se aplica al problema que queremos resolver, la programación dinámica o *divide & conquer* son dos buenos candidatos.
- Si los subproblemas tienen elementos en común, la programación dinámica es la técnica más prometedora.
- Esta técnica se aplica a menudo a problemas de optimización, aunque esto no es excluyente.

Nueva visita al problema del cambio

Nueva visita al problema del cambio

- Tenemos n denominaciones y queremos pagar v con la mínima cantidad de monedas.

Nueva visita al problema del cambio

- Tenemos n denominaciones y queremos pagar v con la mínima cantidad de monedas.
- Soluciones parciales: pagar cantidades menores con la misma cantidad de denominaciones y con menos denominaciones.

Nueva visita al problema del cambio

- Tenemos n denominaciones y queremos pagar v con la mínima cantidad de monedas.
- Soluciones parciales: pagar cantidades menores con la misma cantidad de denominaciones y con menos denominaciones.
- Considere primero el siguiente ejemplo.

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$. Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1									
4									
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0								
4									
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1							
4									
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4									
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3					
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1				
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2			
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3		
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3	4	
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3	4	2
6									

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3	4	2
6	0	1	2	3	1	2			

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3	4	2
6	0	1	2	3	1	2	1		

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$. Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3	4	2
6	0	1	2	3	1	2	1	2	

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$.
Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
4	0	1	2	3	1	2	3	4	2
6	0	1	2	3	1	2	1	2	**2**

El problema del cambio. Un ejemplo de juguete

Supongamos que tenemos $D = (6, 4, 1)$ y debemos pagar $V = 8$. Descomponemos el problema en pagos de cifras menores ($j \leq V$) con un número menor de denominaciones ($i \leq |D|$.)

	0	1	2	3	4	5	6	7	8
1	**0**	1	2	3	4	5	6	7	8
4	**0**	1	2	3	**1**	2	3	4	**2**
6	0	1	2	3	1	2	1	2	**2**

La construcción de la tabla. Prolegómenos

La construcción de la tabla. Prolegómenos

- La solución para pagar $j \leq V$ con $i \leq n$ será el mínimo entre
  - pagar j con $i - 1$ denominaciones, y
  - pagar $j - d_i$ con i denominaciones más uno.

La construcción de la tabla. Prolegómenos

- La solución para pagar $j \leq V$ con $i \leq n$ será el mínimo entre
  - pagar j con $i - 1$ denominaciones, y
  - pagar $j - d_i$ con i denominaciones más uno.
- Es decir, si uso una moneda d_i debo sumar uno más la cantidad de monedas que necesito para pagar $j - d_i$ con i denominaciones; si no, considero simplemente la cantidad de monedas que necesito para pagar j con $i - 1$ denominaciones (no uso la moneda d_i.)

La construcción de la tabla. Prolegómenos

- La solución para pagar $j \leq V$ con $i \leq n$ será el mínimo entre
  - pagar j con $i - 1$ denominaciones, y
  - pagar $j - d_i$ con i denominaciones más uno.
- Es decir, si uso una moneda d_i debo sumar uno más la cantidad de monedas que necesito para pagar $j - d_i$ con i denominaciones; si no, considero simplemente la cantidad de monedas que necesito para pagar j con $i - 1$ denominaciones (no uso la moneda d_i.)
- Esto continúa ...

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2									
4									
7									

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2	0								
4									
7									

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2	0	∞							
4									
7									

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2	0	∞	1	∞	2	∞	3	∞	4
4									
7									

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2	0	∞	1	∞	2	∞	3	∞	4
4	0	∞	1	∞					
7									

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2	0	∞	1	∞	2	∞	3	∞	4
4	0	∞	1	∞	1	∞	2	∞	2
7									

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2	0	∞	1	∞	2	∞	3	∞	4
4	0	∞	1	∞	1	∞	2	∞	2
7	0	∞	1	∞	1	∞	2		

¿Qué formula estamos usando?

$$C[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } j = 0 \\ +\infty & \text{si } i = 1 \text{ y } d_i > j \\ 1 + C[i, j - d_i] & \text{if } i = 1 \text{ y } d_i \leq j \\ C[i-1, j] & \text{si } i > 1 \text{ and } d_i > j \\ \min(1 + C[i, j - d_i], C[i-1, j]) & \text{sino} \end{cases}$$

	0	1	2	3	4	5	6	7	8
2	0	∞	1	∞	2	∞	3	∞	4
4	0	∞	1	∞	1	∞	2	∞	2
7	0	∞	1	∞	1	∞	2	1	2

1. Repaso de la clase anterior
   - Introducción a la programación dinámica
   - El problema del cambio
   - El problema de la mochila 0-1
2. Programación dinámica. Algunos ejemplos
   - El algoritmo de Floyd-Warshall
   - Subsecuencia común más larga
   - Subsecuencia creciente más larga
3. Ejercicios propuestos

El problema de la mochila 0-1

El problema de la mochila 0-1

- Tenemos n objetos y una mochila. Para $i \in \{1, \ldots, n\}$, cada objeto $O[i]$ tiene un peso positivo p_i y un valor positivo v_i.

El problema de la mochila 0-1

- Tenemos n objetos y una mochila. Para $i \in \{1, \ldots, n\}$, cada objeto $O[i]$ tiene un peso positivo p_i y un valor positivo v_i.
- La mochila tiene una capacidad máxima de P. Si se le carga más peso, se desfonda.

El problema de la mochila 0-1

- Tenemos n objetos y una mochila. Para $i \in \{1, \ldots, n\}$, cada objeto $O[i]$ tiene un peso positivo p_i y un valor positivo v_i.
- La mochila tiene una capacidad máxima de P. Si se le carga más peso, se desfonda.
- Los objetos no pueden ser fraccionados. O se los incluye completos en la mochila o se los deja de lado.

El problema de la mochila 0-1

- Tenemos n objetos y una mochila. Para $i \in \{1, \ldots, n\}$, cada objeto $O[i]$ tiene un peso positivo p_i y un valor positivo v_i.
- La mochila tiene una capacidad máxima de P. Si se le carga más peso, se desfonda.
- Los objetos no pueden ser fraccionados. O se los incluye completos en la mochila o se los deja de lado.
- El objetivo es maximizar el valor cargado en la mochila respetando el límite de capacidad impuesto.

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3										
2										
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0							
2										
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3						
2										
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2										
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0								
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8							
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8					
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11				
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6										
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11				
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11	11	11		
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11	11	11	13	
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11	11	11	13	13
7										

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11	11	11	13	13
7	0	0	8	8	8	11	11			

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11	11	11	13	13
7	0	0	8	8	8	11	11	11		

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11	11	11	13	13
7	0	0	8	8	8	11	11	11	13	

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	0	0	0	3	3	3	3	3	3	3
2	0	0	8	8	8	11	11	11	11	11
6	0	0	8	8	8	11	11	11	13	13
7	0	0	8	8	8	11	11	11	13	**15**

El problema de la mochila 0-1. Un ejemplo

Supongamos que tenemos cuatro objetos de pesos $peso = (3, 2, 6, 7)$ con valores $valor = (3, 8, 5, 7)$. El peso máximo es $P = 9$. Descomponemos el problema como se explicó:

	0	1	2	3	4	5	6	7	8	9
3	**0**	0	0	3	3	3	3	3	3	3
2	0	0	**8**	8	8	11	11	11	11	11
6	0	0	**8**	8	8	11	11	11	13	13
7	0	0	8	8	8	11	11	11	13	**15**

La construcción de la tabla

La construcción de la tabla

- Luego del ejemplo, debería ser relativamente fácil construir la tabla. Recuerde que tenemos n objetos y que la carga máxima que puede llevar la mochila es P.

La construcción de la tabla

- Luego del ejemplo, debería ser relativamente fácil construir la tabla. Recuerde que tenemos n objetos y que la carga máxima que puede llevar la mochila es P.
- Recuerde también que, por conveniencia, el índice i se mueve entre 1 y n, mientras que el índice j se mueve entre 0 y P. Las fórmulas de recurrencia son:

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1,j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1,j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j],\ V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)											
(6, 5)											
(4, 4)											
(5, 3)											

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0				
(6, 5)											
(4, 4)											
(5, 3)											

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0	8	8	8	8
(6, 5)											
(4, 4)											
(5, 3)											

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0	8	8	8	8
(6, 5)	0	0	0	0	0	0					
(4, 4)											
(5, 3)											

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0	8	8	8	8
(6, 5)	0	0	0	0	0	0	5	8	8	8	8
(4, 4)											
(5, 3)											

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j],\ V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0	8	8	8	8
(6, 5)	0	0	0	0	0	0	5	8	8	8	8
(4, 4)	0	0	0	0							
(5, 3)											

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0	8	8	8	8
(6, 5)	0	0	0	0	0	0	5	8	8	8	8
(4, 4)	0	0	0	0	4	4	5	8	8	8	9
(5, 3)											

¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0	8	8	8	8
(6, 5)	0	0	0	0	0	0	5	8	8	8	8
(4, 4)	0	0	0	0	4	4	5	8	8	8	9
(5, 3)	0	0	0	0	4						

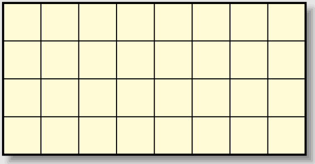
¿Qué formula estamos usando?

$$V[i,j] = \begin{cases} 0 & \text{si } i = 1 \text{ y } peso[i] > j \\ valor[i] & \text{si } i = 1 \text{ y } peso[i] \leq j \\ V[i\text{-}1, j] & \text{si } i > 1 \text{ y } peso[i] > j \\ \max(V[i\text{-}1, j], V[i\text{-}1, j\text{-}peso[i]] + valor[i]) & \text{si } i > 1 \text{ y } peso[i] \leq j \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10
(7, 8)	0	0	0	0	0	0	0	8	8	8	8
(6, 5)	0	0	0	0	0	0	5	8	8	8	8
(4, 4)	0	0	0	0	4	4	5	8	8	8	9
(5, 3)	0	0	0	0	4	4	5	8	8	8	9

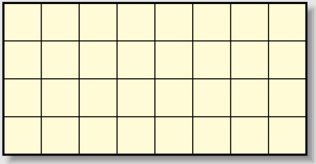
Algunos comentarios

Problema del cambio



$C[i, j] = \mathtt{min}(C[i\text{-}1, j],\ C[i, j\text{-}d[i]] + 1)$

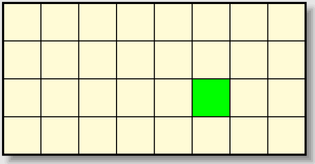
Problema de la mochila



$C[i, j] = \mathtt{max}(C[i\text{-}1, j],\ C[i\text{-}1, j\text{-}p[i]] + v[i])$

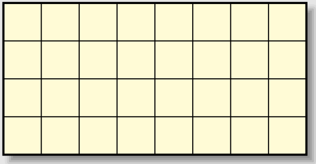
Algunos comentarios

Problema del cambio



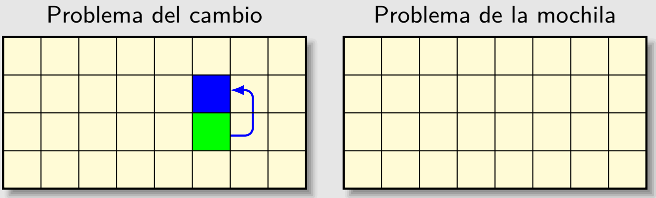
$C[i, j] = \mathtt{min}(C[i\text{-}1, j], C[i, j\text{-}d[i]] + 1)$

Problema de la mochila



$C[i, j] = \mathtt{max}(C[i\text{-}1, j], C[i\text{-}1, j\text{-}p[i]] + v[i])$

Algunos comentarios

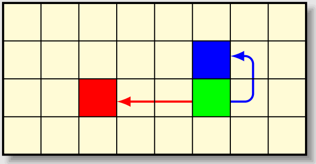


$C[i, j] = \texttt{min}(C[i\text{-}1, j],\ C[i, j\text{-}d[i]] + 1)$

$C[i, j] = \texttt{max}(C[i\text{-}1, j],\ C[i\text{-}1, j\text{-}p[i]] + v[i])$

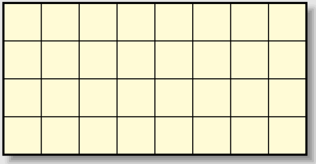
Algunos comentarios

Problema del cambio



$C[i,j] = \mathtt{min}(C[i\text{-}1,j], C[i,j\text{-}d[i]] + 1)$

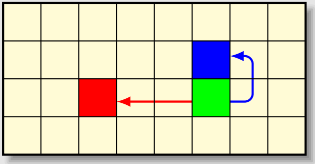
Problema de la mochila



$C[i,j] = \mathtt{max}(C[i\text{-}1,j], C[i\text{-}1,j\text{-}p[i]] + v[i])$

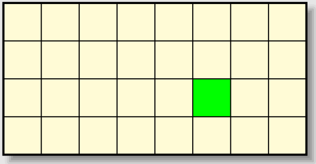
Algunos comentarios

Problema del cambio



$C[i, j] = \mathtt{min}(C[i\text{-}1, j], C[i, j\text{-}d[i]] + 1)$

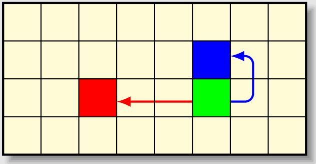
Problema de la mochila



$C[i, j] = \mathtt{max}(C[i\text{-}1, j], C[i\text{-}1, j\text{-}p[i]] + v[i])$

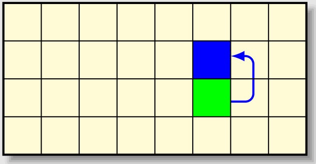
Algunos comentarios

Problema del cambio



$C[i,j] = \texttt{min}(C[i\text{-}1,j], C[i,j\text{-}d[i]] + 1)$

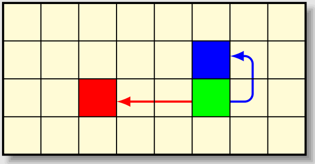
Problema de la mochila



$C[i,j] = \texttt{max}(C[i\text{-}1,j], C[i\text{-}1,j\text{-}p[i]] + v[i])$

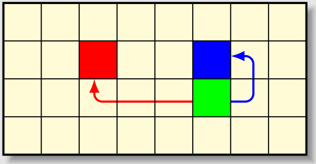
Algunos comentarios

Problema del cambio



$C[i,j] = \mathtt{min}(C[i\text{-}1,j], C[i,j\text{-}d[i]] + 1)$

Problema de la mochila



$C[i,j] = \mathtt{max}(C[i\text{-}1,j], C[i\text{-}1,j\text{-}p[i]] + v[i])$

El algoritmo de Floyd-Warshall

El algoritmo de Floyd-Warshall

- Este algoritmo encuentra el mínimo camino entre dos nodos cualesquiera de un grafo dirigido si este camino existe.

El algoritmo de Floyd-Warshall

- Este algoritmo encuentra el mínimo camino entre dos nodos cualesquiera de un grafo dirigido si este camino existe.
- En términos más técnicos, el algoritmo encuentra el cierre transitivo de un grafo dirigido.

El algoritmo de Floyd-Warshall

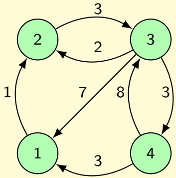
- Este algoritmo encuentra el mínimo camino entre dos nodos cualesquiera de un grafo dirigido si este camino existe.
- En términos más técnicos, el algoritmo encuentra el cierre transitivo de un grafo dirigido.
- Si se aplica el algoritmo sobre un grafo $G = (V, A)$, se obtiene un grafo $G' = (V, A')$ donde, si existe un camino mínimo de costo c en G entre dos nodos x y y, entonces existe una arista $(x, y) \in A'$ de costo c.

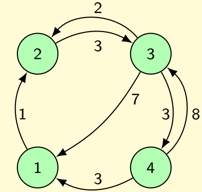
El algoritmo de Floyd-Warshall

- Este algoritmo encuentra el mínimo camino entre dos nodos cualesquiera de un grafo dirigido si este camino existe.
- En términos más técnicos, el algoritmo encuentra el cierre transitivo de un grafo dirigido.
- Si se aplica el algoritmo sobre un grafo $G = (V, A)$, se obtiene un grafo $G' = (V, A')$ donde, si existe un camino mínimo de costo c en G entre dos nodos x y y, entonces existe una arista $(x, y) \in A'$ de costo c.
- El grafo $G = (V, A)$ se representa como una matriz $M \in \mathbb{R}^{n\times n}$ donde $n = |V|$. Cada elemento $M[i, j]$ contiene el costo de la arista (i, j) o ∞ si la arista no existe.

El algoritmo de Floyd-Warshall. Un ejemplo

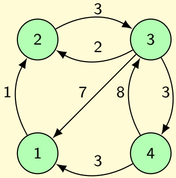
$$G = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & \infty & 8 & 0 \end{bmatrix}$$



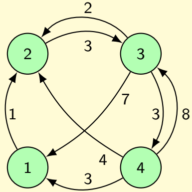


El algoritmo de Floyd-Warshall. Un ejemplo

$$G = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & \infty & 8 & 0 \end{bmatrix} \quad A_0 = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 8 & 0 \end{bmatrix}$$

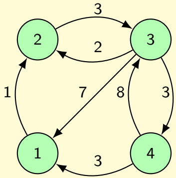


$$A_0[4,2] = \min(\underbrace{G[4,2]}_{\infty}, \underbrace{G[4,1] + G[1,2]}_{3+1=4})$$



El algoritmo de Floyd-Warshall. Un ejemplo

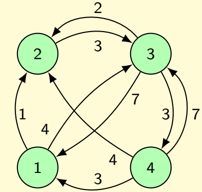
$$G = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & \infty & 8 & 0 \end{bmatrix} \quad A_0 = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 8 & 0 \end{bmatrix}$$



$$A_1 = \begin{bmatrix} 0 & 1 & 4 & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 7 & 0 \end{bmatrix}$$

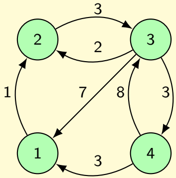
$$A_1[1,3] = \min(\underbrace{A_0[1,3]}_{\infty}, \underbrace{A_0[1,2] + A_0[2,3]}_{1+3=4})$$

$$A_1[4,3] = \min(\underbrace{A_0[4,3]}_{8}, \underbrace{A_0[4,2] + A_0[2,3]}_{4+3=7})$$



El algoritmo de Floyd-Warshall. Un ejemplo

$$G = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & \infty & 8 & 0 \end{bmatrix} \qquad A_0 = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 8 & 0 \end{bmatrix}$$

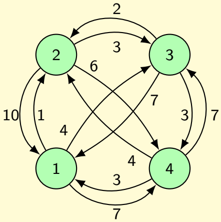


$$A_1 = \begin{bmatrix} 0 & 1 & 4 & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 7 & 0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0 & 1 & 4 & 7 \\ 10 & 0 & 3 & 6 \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 7 & 0 \end{bmatrix}$$

$$A_2[1,4] = \min(\underbrace{A_1[1,4]}_{\infty}, \underbrace{A_1[1,3] + A_1[3,4]}_{4+3=7})$$

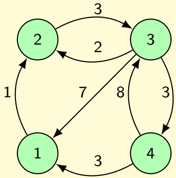
$$A_2[2,1] = \min(\underbrace{A_1[2,1]}_{\infty}, \underbrace{A_1[2,3] + A_1[3,1]}_{3+7=10})$$

$$A_2[2,4] = \min(\underbrace{A_1[2,3]}_{\infty}, \underbrace{A_1[3,4] + A_1[2,3]}_{3+3=6})$$



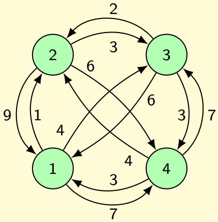
El algoritmo de Floyd-Warshall. Un ejemplo

$$G = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & \infty & 8 & 0 \end{bmatrix} \qquad A_0 = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 8 & 0 \end{bmatrix}$$



$$A_1 = \begin{bmatrix} 0 & 1 & 4 & \infty \\ \infty & 0 & 3 & \infty \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 7 & 0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0 & 1 & 4 & 7 \\ 10 & 0 & 3 & 6 \\ 7 & 2 & 0 & 3 \\ 3 & 4 & 7 & 0 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 1 & 4 & 7 \\ 9 & 0 & 3 & 6 \\ 6 & 2 & 0 & 3 \\ 3 & 4 & 7 & 0 \end{bmatrix}$$



$$A_3[2,1] = \min(\underbrace{A_2[2,1]}_{10}, \underbrace{A_2[2,4] + A_2[4,1]}_{6+3=9})$$

$$A_3[3,1] = \min(\underbrace{A_2[3,1]}_{7}, \underbrace{A_2[3,4] + A_2[4,1]}_{3+3=6})$$

El algoritmo de Floyd-Warshall

```
Algoritmo Floyd-Warshall (real [n][n]G, M)
    A = initializeMatrix(A) : [1..n, 1..n, 1..n]          // Matrices A_k
    for (k = 0; k < n; k++) {
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                if (k == 0) {
                    A[k, i, j] = min(G[i, j], G[i, k]+G[k, j])
                } else {
                    A[k, i, j] = min(A[k-1, i, j], A[k-1, i, k]+A[k-1, k, j])
                }
            }
        }
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            M[i, j] = A[n-1, i, j]
        }
    }
    return M
```

El algoritmo de Floyd-Warshall

```
     Algoritmo Floyd-Warshall (real [n][n]G, M)
         A = initializeMatrix(A) : [1..n, 1..n, 1..n]          // Matrices A_k
Θ(n)     for (k = 0; k < n; k++) {
Θ(n)         for (i = 0; i < n; i++) {
Θ(n)             for (j = 0; j < n; j++) {
                     if (k == 0) {
                         A[k, i, j] = min(G[i, j], G[i, k]+G[k, j])
                     } else {
                         A[k, i, j] = min(A[k-1, i, j], A[k-1, i, k]+A[k-1, k, j])
                     }
                 }
             }
         }
Θ(n)     for (i = 0; i < n; i++) {
Θ(n)         for (j = 0; j < n; j++) {
                 M[i, j] = A[n-1, i, j]
             }
         }
         return M
```

$$\Theta(n^3) + \Theta(n^2) = \Theta(n^3)$$

La clase *Cubo*

```
public class Cubo {
    private int Size;
    private int[][][] Cubo;
    private int[] Etiqs;
    public Cubo (int tam) {                        // El constructor de la clase
        this.Size = tam {
        this.Cubo = new int[tam+1][tam][tam];
        this.Cubo = new int[tam];
    }
    public void setCubo(int x, int y, int z, int v) {
        this.Cubo[x][y][z] = v;
    }
    public int getCubo(int x, int y, int z) {
        return this.Cubo[x][y][z] = v;
    }
    public void setEtiqs(int x, int v) {
        this.Etiqs[x] = v;
    }
    public int getEtiqs(int x) {
        return this.Etiqs[x];
    }
    public int getSize() {
        return this.Size;
    }
}
```

El problema de la subsecuencia común más larga (I)

El problema de la subsecuencia común más larga (I)

- Supongamos que se tiene una secuencia $X = (x_1 x_2 \dots x_m)$. Se dice que $Z = (z_1 z_2 \dots z_k)$ es una *subsecuencia* de X (siendo $k \leq m$) si existe una secuencia creciente $(i_1 i_2 \dots i_k)$ de índices de X tales que para todo $j = 1, 2, \dots, k$ se tiene $x_{i_j} = z_j$.

El problema de la subsecuencia común más larga (I)

- Supongamos que se tiene una secuencia $X = (x_1 x_2 \ldots x_m)$. Se dice que $Z = (z_1 z_2 \ldots z_k)$ es una *subsecuencia* de X (siendo $k \leq m$) si existe una secuencia creciente $(i_1 i_2 \ldots i_k)$ de índices de X tales que para todo $j = 1, 2, \ldots, k$ se tiene $x_{i_j} = z_j$.
- Ejemplo: $Z = BCDBCBCB$ es una subsecuencia de $X = (ABCDABCCABDBDACB)$ con secuencia de índices $(2, 3, 4, 6, 7, 10, 15, 16)$, como se muestra a continuación.

A	B	C	D	A	B	C	C	A	B	D	B	D	A	C	B
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	B	C	D		B	C			B					C	B

El problema de la subsecuencia común más larga (I)

- Supongamos que se tiene una secuencia $X = (x_1 x_2 \ldots x_m)$. Se dice que $Z = (z_1 z_2 \ldots z_k)$ es una *subsecuencia* de X (siendo $k \leq m$) si existe una secuencia creciente $(i_1 i_2 \ldots i_k)$ de índices de X tales que para todo $j = 1, 2, \ldots, k$ se tiene $x_{i_j} = z_j$.
- Ejemplo: $Z = BCDBCBCB$ es una subsecuencia de $X = (ABCDABCCABDBDACB)$ con secuencia de índices $(2, 3, 4, 6, 7, 10, 15, 16)$, como se muestra a continuación.

A	B	C	D	A	B	C	C	A	B	D	B	D	A	C	B
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	B	C	D		B	C			B					C	B

- Dadas dos secuencias X y Y, se dice que Z es una *subsecuencia común* de X y Y si es una subsecuencia de ambas.

El problema de la subsecuencia común más larga (II)

El problema de la subsecuencia común más larga (II)

- Dadas dos secuencias X y Y, se desea encontrar la subsecuencia común Z de máxima longitud.

El problema de la subsecuencia común más larga (II)

- Dadas dos secuencias X y Y, se desea encontrar la subsecuencia común Z de máxima longitud.
- Por ejemplo, consideremos las secuencias $X = (ABCBDA)$ y $Y = (ABACADAC)$. La subsecuencia común más larga tiene longitud 5:

(A B C B D A)

(A B A C A D A C)

El problema de la subsecuencia común más larga (II)

- Dadas dos secuencias X y Y, se desea encontrar la subsecuencia común Z de máxima longitud.
- Por ejemplo, consideremos las secuencias $X = (ABCBDA)$ y $Y = (ABACADAC)$. La subsecuencia común más larga tiene longitud 5:

(A B C B D A)

(A B A C A D A C)

- La solución de este problema consiste en la descomposición en subproblemas con la misma estructura, tomando los primeros i caracteres de la primera secuencia y los primeros j caracteres de la segunda.

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	$\emptyset$	A	B	A	C	A	D	A	C
$\emptyset$									
A									
B									
C									
B									
D									
A									

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	$\emptyset$	A	B	A	C	A	D	A	C
$\emptyset$	0	0	0	0	0	0	0	0	0
A	0								
B	0								
C	0								
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1							
B	0								
C	0								
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	$\emptyset$	A	B	A	C	A	D	A	C
$\emptyset$	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0								
C	0								
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1							
C	0								
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1	2						
C	0								
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1	2	2	2	2	2	2	2
C	0								
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1	2	2	2	2	2	2	2
C	0	1	2	2					
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1	2	2	2	2	2	2	2
C	0	1	2	2	3				
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1	2	2	2	2	2	2	2
C	0	1	2	2	3	3	3	3	3
B	0								
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1	2	2	2	2	2	2	2
C	0	1	2	2	3	3	3	3	3
B	0	1	2	2	3	3	3	3	3
D	0								
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

	∅	A	B	A	C	A	D	A	C
∅	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1
B	0	1	2	2	2	2	2	2	2
C	0	1	2	2	3	3	3	3	3
B	0	1	2	2	3	3	3	3	3
D	0	1	2	2	3	3			
A	0								

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

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

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

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

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

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

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

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

Un ejemplo de subsecuencia común más larga

Tenemos las secuencias $X = (ABCBDA)$ y $X = (ABACADAC)$.

Descomponemos el problema como se explicó:

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

La construcción de la tabla

La construcción de la tabla

- Tenemos dos secuencias $X[1..n]$ y $Y[1..m]$. El elemento $M[i,j]$ nos da la longitud de la subsecuencia común más larga entre $X[1..i]$ y los $Y[1..j]$.

La construcción de la tabla

- Tenemos dos secuencias $X[1..n]$ y $Y[1..m]$. El elemento $M[i,j]$ nos da la longitud de la subsecuencia común más larga entre $X[1..i]$ y los $Y[1..j]$.
- Si una de las secuencias es vacía, la máxima subsecuencia común es vacía. Por lo tanto $M[0,j] = M[i,0] = 0$ para todo $i \in \{1,\ldots,n\}$, $j \in \{1,\ldots,m\}$.

La construcción de la tabla

- Tenemos dos secuencias $X[1..n]$ y $Y[1..m]$. El elemento $M[i, j]$ nos da la longitud de la subsecuencia común más larga entre $X[1..i]$ y los $Y[1..j]$.
- Si una de las secuencias es vacía, la máxima subsecuencia común es vacía. Por lo tanto $M[0, j] = M[i, 0] = 0$ para todo $i \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$.
- Si $X[i] = Y[j]$, podemos pensar que esto equivale a tomar el valor $M[i-1, j-1]$ y sumarle 1.

La construcción de la tabla

- Tenemos dos secuencias $X[1..n]$ y $Y[1..m]$. El elemento $M[i,j]$ nos da la longitud de la subsecuencia común más larga entre $X[1..i]$ y los $Y[1..j]$.
- Si una de las secuencias es vacía, la máxima subsecuencia común es vacía. Por lo tanto $M[0,j] = M[i,0] = 0$ para todo $i \in \{1,\ldots,n\}$, $j \in \{1,\ldots,m\}$.
- Si $X[i] = Y[j]$, podemos pensar que esto equivale a tomar el valor $M[i-1,j-1]$ y sumarle 1.
- Si no, tomamos el máximo valor entre $M[i-1,j]$ y $M[i,j-1]$.

La construcción de la tabla

- Tenemos dos secuencias $X[1..n]$ y $Y[1..m]$. El elemento $M[i,j]$ nos da la longitud de la subsecuencia común más larga entre $X[1..i]$ y los $Y[1..j]$.
- Si una de las secuencias es vacía, la máxima subsecuencia común es vacía. Por lo tanto $M[0,j] = M[i,0] = 0$ para todo $i \in \{1,\ldots,n\}$, $j \in \{1,\ldots,m\}$.
- Si $X[i] = Y[j]$, podemos pensar que esto equivale a tomar el valor $M[i-1,j-1]$ y sumarle 1.
- Si no, tomamos el máximo valor entre $M[i-1,j]$ y $M[i,j-1]$.
- Entonces:

$$M[i,j] = \begin{cases} 0 & \text{si } i = 0 \text{ o } j = 0 \\ M[i-1,j-1]+1 & \text{si } i > 0, j > 0 \text{ y } X[i] = Y[j] \\ \max(M[i-1,j], M[i,j-1]) & \text{sino} \end{cases}$$

El algoritmo para la subsecuencia común más larga

```
Algoritmo LCS (string X, Y., int v)
    A = initializeMatrix : [X.length], [Y.length]
    for (i = 0; i < X.length; i++) {
        for (j = 0; i < Y.length; j++) {
            if (i == 0 || j == 0) {
                M[i, j] = 0
            } else {
                if X[i] = Y[j]
                    M[i, j] = M[i − 1, j − 1] + 1
                } else {
                    M[i, j] = max(M[i − 1, j], M[i, j − 1])
                }
            }
        }
    }
    return M[n, m]
```

El algoritmo para la subsecuencia común más larga

$\Theta(n)$

$\Theta(m)$

```
Algoritmo LCS (string X, Y., int v)
    A = initializeMatrix : [X.length], [Y.length]
    for (i = 0; i < X.length; i++) {
        for (j = 0; i < Y.length; j++) {
            if (i == 0 || j == 0) {
                M[i, j] = 0
            } else {
                if X[i] = Y[j]
                    M[i, j] = M[i - 1, j - 1] + 1
                } else {
                    M[i, j] = max(M[i - 1, j], M[i, j - 1])
                }
            }
        }
    }
    return M[n, m]
```

$\Theta(m \cdot n)$

Subsecuencia creciente más larga

Subsecuencia creciente más larga

- Dada una secuencia $X = (x_1 x_2 \ldots x_n)$, una subsecuencia $Z = (z_1 z_2 \ldots z_k)$ de X es una *subsecuencia creciente* de X si para todo $i, j \in \{1, \ldots, k\}$, $i < j$ implica que $z_i \leq z_j$.

Subsecuencia creciente más larga

- Dada una secuencia $X = (x_1 x_2 \ldots x_n)$, una subsecuencia $Z = (z_1 z_2 \ldots z_k)$ de X es una *subsecuencia creciente* de X si para todo $i, j \in \{1, \ldots, k\}$, $i < j$ implica que $z_i \leq z_j$.
- Por ejemplo, si $X = (0, 4, 12, 2, 10, 6, 9, 13, 3, 11, 7, 15)$, entonces $Z = (2, 6, 9)$ es una subsecuencia creciente de X.

Subsecuencia creciente más larga

- Dada una secuencia $X = (x_1 x_2 \ldots x_n)$, una subsecuencia $Z = (z_1 z_2 \ldots z_k)$ de X es una *subsecuencia creciente* de X si para todo $i, j \in \{1, \ldots, k\}$, $i < j$ implica que $z_i \leq z_j$.
- Por ejemplo, si $X = (0, 4, 12, 2, 10, 6, 9, 13, 3, 11, 7, 15)$, entonces $Z = (2, 6, 9)$ es una subsecuencia creciente de X.
- El problema consiste en encontrar la más larga de las subsecuencias crecientes de la secuencia .

Subsecuencia creciente más larga

- Dada una secuencia $X = (x_1 x_2 \ldots x_n)$, una subsecuencia $Z = (z_1 z_2 \ldots z_k)$ de X es una *subsecuencia creciente* de X si para todo $i, j \in \{1, \ldots, k\}$, $i < j$ implica que $z_i \leq z_j$.
- Por ejemplo, si $X = (0, 4, 12, 2, 10, 6, 9, 13, 3, 11, 7, 15)$, entonces $Z = (2, 6, 9)$ es una subsecuencia creciente de X.
- El problema consiste en encontrar la más larga de las subsecuencias crecientes de la secuencia .
- La estrategia consiste en plantear subproblemas de subsecuencias crecientes más largas de fragmentos de X:

Subsecuencia creciente más larga

- Dada una secuencia $X = (x_1 x_2 \dots x_n)$, una subsecuencia $Z = (z_1 z_2 \dots z_k)$ de X es una *subsecuencia creciente* de X si para todo $i, j \in \{1, \dots, k\}$, $i < j$ implica que $z_i \leq z_j$.
- Por ejemplo, si $X = (0, 4, 12, 2, 10, 6, 9, 13, 3, 11, 7, 15)$, entonces $Z = (2, 6, 9)$ es una subsecuencia creciente de X.
- El problema consiste en encontrar la más larga de las subsecuencias crecientes de la secuencia .
- La estrategia consiste en plantear subproblemas de subsecuencias crecientes más largas de fragmentos de X:
- Construimos una matriz $D \in \mathbb{N}^{2\times n}$ donde $D[0, j]$ tiene la longitud de la subsecuencia creciente más larga a la que el elemento $X[j]$ pertenece en el vector $X[0..j]$ y $D[1, j]$ el índice del elemento anterior en esa subsecuencia.

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$												
inx												

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\lvert Z \rvert$	1	1	1	1	1	1	1	1	1	1	1	1
inx	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\lvert Z \rvert$	1	2	1	1	1	1	1	1	1	1	1	1
inx	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	1	1	1	1	1	1	1	1	1
inx	-1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	1	1	1	1	1	1	1	1
inx	-1	0	1	0	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\lvert Z \rvert$	1	2	3	2	3	1	1	1	1	1	1	1
inx	-1	0	1	0	3	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	3	3	1	1	1	1	1	1
inx	-1	0	1	0	3	3	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	3	3	4	1	1	1	1	1
inx	-1	0	1	0	3	3	5	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	3	3	4	5	1	1	1	1
inx	-1	0	1	0	3	3	5	6	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	3	3	4	5	3	1	1	1
inx	-1	0	1	0	3	3	5	6	3	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\lvert Z \rvert$	1	2	3	2	3	3	4	5	3	5	1	1
inx	-1	0	1	0	3	3	5	6	3	6	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	3	3	4	5	3	5	4	1
inx	-1	0	1	0	3	3	5	6	3	6	8	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	3	3	4	5	3	5	4	6
inx	-1	0	1	0	3	3	5	6	3	6	8	9

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un primer ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	0	4	12	2	10	6	9	13	3	11	7	15
$\|Z\|$	1	2	3	2	3	3	4	5	3	5	4	6
inx	-1	0	1	0	1	1	5	6	3	6	5	8

- elemento final de la mayor subsecuencia creciente
- elementos menores o iguales al actual
- longitud de la mayor subsecuencia creciente

la mayor subsecuencia creciente es **0, 2, 6, 9, 11, 15**

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$												
inx												

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	1	1	1	1	1	1	1	1	1	1	1
inx	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	1	1	1	1	1	1	1	1	1	1
inx	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	1	1	1	1	1	1	1	1	1
inx	-1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	1	1	1	1	1	1	1	1
inx	-1	0	1	0	-1	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	3	1	1	1	1	1	1	1
inx	-1	0	1	0	3	-1	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	3	4	1	1	1	1	1	1
inx	-1	0	1	0	3	4	-1	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\lvert Z\rvert$	1	2	3	2	3	4	5	1	1	1	1	1
inx	-1	0	1	0	3	4	5	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	3	4	5	1	1	1	1	1
inx	-1	0	1	0	3	4	5	-1	-1	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\lvert Z\rvert$	1	2	3	2	3	4	5	1	4	1	1	1
inx	-1	0	1	0	3	4	5	-1	4	-1	-1	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	3	4	5	1	4	6	1	1
inx	-1	0	1	0	3	4	5	-1	4	6	-1	-1

elemento actual

elementos menores o iguales al actual

mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	3	4	5	1	4	6	3	1
inx	-1	0	1	0	3	4	5	-1	4	6	3	-1

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	3	4	5	1	4	6	3	4
inx	-1	0	1	0	3	4	5	-1	4	6	3	10

- elemento actual
- elementos menores o iguales al actual
- mayor cadena entre los elementos menores al actual

Mayor subsecuencia creciente. Un segundo ejemplo

	0	1	2	3	4	5	6	7	8	9	10	11
X	2	5	11	2	4	5	7	0	4	9	3	4
$\|Z\|$	1	2	3	2	3	4	5	1	4	6	3	9
inx	-1	0	1	0	3	4	5	-1	4	6	3	4

- elemento final de la mayor subsecuencia creciente
- elementos menores o iguales al actual
- longitud de la mayor subsecuencia creciente

la mayor subsecuencia creciente es **2, 2, 4, 5, 7, 9**

La construcción de la tabla

La construcción de la tabla

- En el caso $D[0,0]$ existe una única subsecuencia creciente, $Z[0] = X[0]$, y no hay ningún índice anterior, pues estamos en el comienzo. Por lo tanto, $D[1,0]$ no se utiliza y puede contener cualquier valor. Lo mismo si no existe ningún $k < j$ con $X[k] < X[j]$.

La construcción de la tabla

- En el caso $D[0,0]$ existe una única subsecuencia creciente, $Z[0] = X[0]$, y no hay ningún índice anterior, pues estamos en el comienzo. Por lo tanto, $D[1,0]$ no se utiliza y puede contener cualquier valor. Lo mismo si no existe ningún $k < j$ con $X[k] < X[j]$.
- En el caso $D[0,j]$ con $j > 0$, tenemos que buscar el máximo elemento $D[0,k]$, $k < j$, tal que $X[k] \leq X[j]$. Entonces, $D[0,j] = D[0,k] + 1$ y $D[1,j] = k$.

La construcción de la tabla

- En el caso $D[0,0]$ existe una única subsecuencia creciente, $Z[0] = X[0]$, y no hay ningún índice anterior, pues estamos en el comienzo. Por lo tanto, $D[1,0]$ no se utiliza y puede contener cualquier valor. Lo mismo si no existe ningún $k < j$ con $X[k] < X[j]$.
- En el caso $D[0,j]$ con $j > 0$, tenemos que buscar el máximo elemento $D[0,k]$, $k < j$, tal que $X[k] \leq X[j]$. Entonces, $D[0,j] = D[0,k] + 1$ y $D[1,j] = k$.
- En síntesis:

$$D[i,j] \leftarrow \begin{cases} 1. & \text{si } \exists k < j \text{ con } X[k] \leq X[j] : \\ & X[0,j] = \max_{k<j}\{D[1,k] \mid X[k] \leq X[j]\} + 1 \\ & X[1,j] = k \\ 2. & \text{si no existe tal valor:} \\ & D[0,j] = 1 \\ & D[1,j] = -1 \end{cases}$$

El algoritmo para la subsecuencia creciente más larga

```
Algoritmo LIS (int[] X, Z, int L)
    D = initializeMatrix : [2, X.length]
    for (i = 0; i < n; i++) {
        D[0, i] = 1
        D[1, i] = −1
    }
    for (i = 1; i < n; i++) {
        for (j = 0; j < i; j++) {
            if (X[j] ≤ X[i] && D[0, j] + 1 ≥ D[0, i]) {
                D[0, i] = D[0, j] + 1
                D[1, i] = j
            }
        }
    }
    inx = max_i(D[0, i])
    L = (D[0, inx])
    Z = initializeVector : [L]
    Z[L − 1] = X[inx]
    for (i = L − 2; i ≥ 0; i−−) {
        Z[i] = X[D[1, inx]]
        inx = D[1, inx]
    }
    return L, Z
```

El algoritmo para la subsecuencia creciente más larga

```
        Algoritmo LIS (int[] X, Z, int L)
            D = initializeMatrix : [2, X.length]
Θ(n)        for (i = 0; i < n; i++) {
                D[0, i] = 1
                D[1, i] = −1
            }
Θ(n)        for (i = 1; i < n; i++) {
Θ(n)            for (j = 0; j < i; j++) {
                    if (X[j] ≤ X[i] && D[0, j] + 1 ≥ D[0, i]) {
                        D[0, i] = D[0, j] + 1
                        D[1, i] = j
                    }
                }
            }
Θ(n)        inx = max_i(D[0, i])
            L = (D[0, inx])
            Z = initializeVector : [L]
            Z[L − 1] = X[inx]
Θ(n)        for (i = L − 2; i ≥ 0; i−−) {
                Z[i] = X[D[1, inx]]
                inx = D[1, inx]
            }
            return L, Z
```

$$\Theta(n^2) + \Theta(n) + \Theta(n) = \Theta(n^2)$$

Ejercicios propuestos 1

1. Analice si es posible extender el algoritmo de Dijkstra para que muestre los caminos de costo mínimo (además de los costos mínimos) utilizado una estrategia similar a la del problema de la subsecuencia creciente más larga.
2. Adapte el problema de la subsecuencia común más larga para encontrar el *substring* común más largo.
3. Dadas n funciones f_1, f_2, ..., f_n y un entero positivo M, deseamos maximizar la función

$$\sum_{i=1}^{n} f_i(x_i) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$$

con la restricción

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + \ldots + x_n = M$$

donde $f_i(0) = 0$ para $i \in \{1, \ldots, n\}$, $x_i \in \mathbb{N}$, y todas las funciones son monótonamente crecientes, es decir, $x > y \Rightarrow f_i(x) > f_i(y)$. Los valores de cada función se almacenan en n vectores de M componentes.

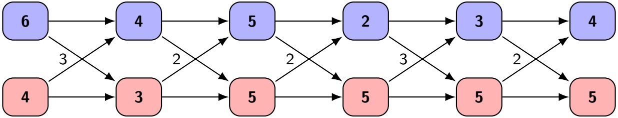
Ejercicios propuestos 2

4. **La fábrica de saxofones**. Son necesarios varias etapas para producir un saxofón: formación del cuerpo principal, curvado, perforación, adición de teclas, adición de la parte superior y and adición de la boquilla. La fábrica *Blue Note* de Fraile Muerto cuenta con dos líneas de producción. Cada línea de ensamblaje i contiene todas las estaciones (cuerpo principal, curvado, perforaciones, teclas, parte superior, boquilla). Denotamos a las estaciones $S[i,j]$ con $i \in \{1,2\}$ y $j \in \{1,\dots,6\}$. Las estaciones $S[1,j]$ y $S[2,j]$ realizan el mismo trabajo pero no en el mismo tiempo, ya que el procedimiento es a mano. Llamamos $T[i,j]$ al tiempo pasado en a estación j de la línea de ensamblaje i.
Un saxofón debe atravesar las seis etapas mencionadas hasta que está terminado. Puede hacerlo en una única línea de ensamblaje o alternar entre ambas. Cuando el instrumento pasa de la estación j a la estación $j+1$ de la misma línea de ensamblaje, esto es instantáneo; pero cuando cambia de una línea de ensamblaje a la otra, esto tiene una demora $D[j]$ con $j \in \{1,\dots,5\}$. Véase el ejemplo en la siguiente diapositiva.
Continúa en la siguiente diapositiva.

Ejercicios propuestos 3

Continuación de la diapositiva anterior

Ejemplo: supongamos que es $D = [3, 2, 2, 3, 2]$ y $D = \begin{bmatrix} 6 & 4 & 5 & 2 & 3 & 4 \\ 4 & 3 & 5 & 5 & 6 & 6 \end{bmatrix}$. Esto se muestra abajo. La línea de ensamblaje 1 es azul y la 2 es roja.



La producción de un saxofón en la línea 1 toma $6 + 4 + 5 + 2 + 3 + 4 = 24$ unidades de tiempo; Con $S[2, 1]$, $S[2, 2]$, $S[1, 3]$, $S[1, 4]$, $S[2, 5]$, $S[2, 6]$, tenemos $4 + 3 + (2) + 5 + 2 + (3) + 5 + 5 = 27$ unidades de tiempo. Los números entre paréntesis dan el tiempo para cambiar de una línea de ensamblaje a la otra.

Encuentre una estrategia de programación dinámica para minimizar el tiempo de producción de un saxofón.

Ejercicios propuestos 4

4. La *permutación* de n elementos tomados de a k se refiere al proceso de acomodar k elementos de un conjunto de n para formar una secuencia. El *coeficiente de permutaciones* $P(n, k)$ nos da el número de secuencias que podemos obtener tomando k elementos de un conjunto de n. Este coeficiente está dado por la fórmula

$$P(n,k) = \frac{n!}{(n-k)!}$$

Ejemplo: $P(4,2) = \frac{24}{2} = 12$, pues tenemos doce maneras de formar pares en un conjunto de cuatro elementos. Si $A = \{a, b, c, d\}$, podemos formar los pares (a, b), (b, a), (a, c), (c, a), (a, d), (d, a), (b, c), (c, b), (b, d), (d, b), (c, d) y (d, c).

El coeficiente $P(n, k)$ se puede computar a partir de resultados ya obtenidos para números menores:

$$P(n,k) = P(n-1,k) + k \cdot P(n-1,k-1)$$

Además tenemos los casos base: si $k > n$ entonces $P(n, k) = 0$, si $k = n$ entonces $P(n, k) = P(n, n) = n!$ y si $k = 0$ entonces $P(n, k) = P(, 0) = 1$. Con esta información, se pide definir una estrategia de programación dinámica para encontrar $P(n, k)$ dados n y k.