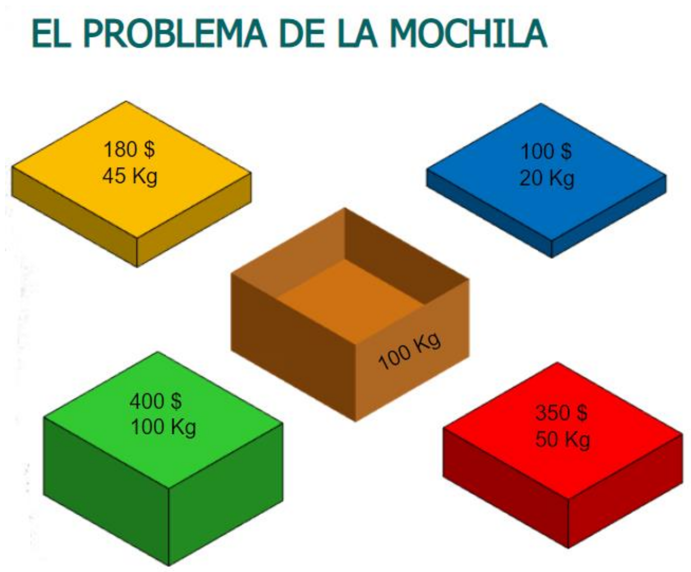


Complejidad Algorítmica

Unidad 2: Algoritmos voraces, programación dinámica y problemas P-NP

Módulo 12: Programación Dinámica - Casos de Uso

Complejidad Algorítmica

Semana 12

Objetivos

- Examinar casos de uso de la Programación Dinámica

MÓDULO 12: Programación Dinámica - Casos de Uso

Contenido

1. Aplicaciones de la Programación Dinámica
    - **Caso de la Mochila**
    - Caso del Cambio Mínimo de Monedas

Preguntas / Conclusiones

1. Aplicaciones de la Programación Dinámica

¿Qué problemas solucionamos con DP?

Existen muchos problemas que se resuelven utilizando un enfoque de programación dinámica para encontrar la solución óptima.

A continuación, conoceremos los siguientes casos de uso:

- **El problema de la Mochila.**
- El problema del Cambio Mínimo de Monedas.

6. Programación Dinámica

¿Cómo resolver problemas de programación dinámica?

Cuando se trata de encontrar la solución al problema utilizando la programación dinámica, a continuación se detallan algunos pasos que debe considerar seguir:

1) Reconocer el problema de DP
2) Identificar las variables del problema
3) Expresar la relación de recurrencia
4) Identificar el caso base
5) Decidir el enfoque iterativo o recursivo para resolver el problema
6) Añadir memorización

1. Aplicaciones de la Programación Dinámica

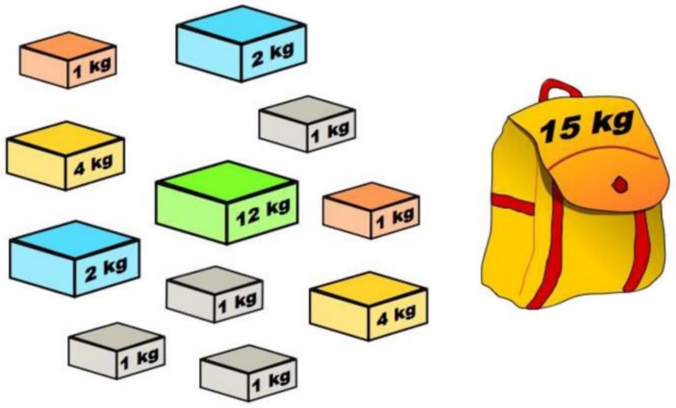
El problema de la Mochila

- El problema de la mochila es el ejemplo perfecto de un **algoritmo de programación dinámica** y la pregunta más frecuente en una entrevista técnica de empresas basadas en productos.

- Como datos tenemos **las ganancias** y **los pesos de N artículos**, y debemos colocar estos artículos en una mochila con la capacidad 'W', por tanto, debemos encontrar (seleccionar) la cantidad de artículos que sea menor o igual a la capacidad de la mochila.

PLANTEAMIENTO DEL PROBLEMA

Dada una bolsa con capacidad **W**, y una lista de artículos junto con sus pesos y ganancias asociadas con ellos.
La tarea es llenar la mochila de manera eficiente, de modo que se logre el máximo beneficio.



1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

EJEMPLO:

Capacidad de la mochila: 11kg
Nro. Productos: 5

Producto	1	2	3	4	5
Pesos	1	2	5	6	7
Precios	1	6	18	22	28

Donde:

Capacidad de la Mochila = W
Lista de pesos : wt = []
Lista de precios : pr = []
No. De productos = N

LIMITE DE PESOS

$w_1=1$	$v_1=1$
$w_2=2$	$v_2=6$
$w_3=5$	$v_3=18$
$w_4=6$	$v_4=22$
$w_5=7$	$v_5=28$

j = w = pesos →

i ↓ N = productos

i \ j	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1 1	1	1	1	1	1	1	1	1	1	1
2	0	1	6	1 + 6 7	7	7	7	7	7	7	7	7
3	0	1	6	7	7	18	18 +1 19	18+6 24	18 + 7 25	25	25	25
4	0	1	6	7	7	18	22	22+1 24	22+6 28	22+7 29	29	22+18 40
5	0	1	6	7	7	18	22	28	28+1 29	28+6 34	28+1+6 35	40

1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

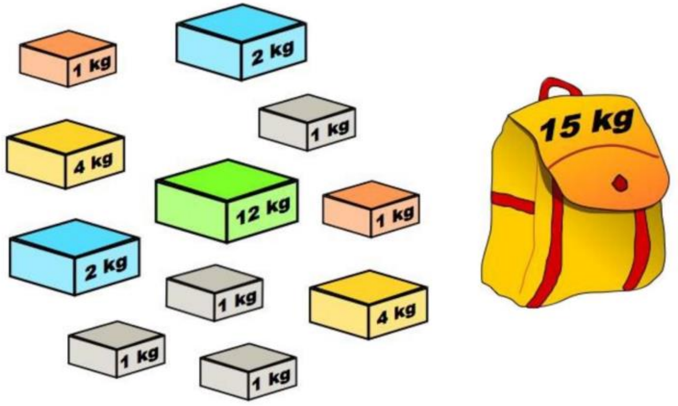
PASOS A SEGUIR

1. Crear una tabla **dp[ ][ ]** y considerar todos los pesos posibles de 1 a W como columnas y pesos posibles a elegir como filas.

2. El estado /celda **dp[i][j]** en la tabla representa la ganancia máxima alcanzable si **'j'** es la capacidad de la mochila y los primeros elementos **'i'** se incluyen en la matriz peso/artículo.
   Por lo tanto, la última celda representará el estado de respuesta.

3. Solo se pueden incluir artículos si su peso es inferior a la capacidad de la mochila.

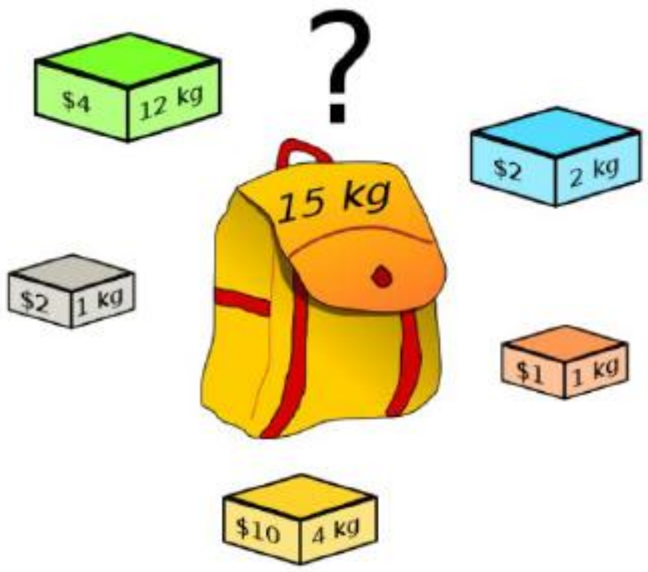
4. Existen dos posibilidades para
   la condición en la que puede completar todas las columnas que tienen 'peso> peso [i-1]'.



1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

EJEMPLO #1: El problema de la Mochila



Estado Inicial: mochila vacía

Estado Final: cualquier combinación de objetos en la mochila
Operadores: meter o sacar objetos de la mochila
Heurística: $max \sum_i Valor_i$ o $max \sum_i \frac{Valor_i}{Peso_i}$

1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

EJEMPLO #1: El problema de la Mochila

Capacidad de la mochila = 8kg
Artículos = 6

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	**Artículo**	**Dimensión (Kg)**	**Ganancia (€)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2									
2	B	2	5									
3	C	4	6									
4	D	5	10									
5	E	7	13									
6	F	8	16									

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #0: El estado de la mochila esta vacía y la ganancia es 0.
Evaluamos 6 etapas porque solo hay 6 objetos (de A-F, existe solo 1 und. x cada objeto).
Los colocamos en la tabla en orden creciente en dimensión con su respectiva ganancia.

1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	Artículo	Dimensión (Kg)	Ganancia (€)	0	1	2	3	4	5	6	7	8
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #1: Introducimos el objeto A en el casillero de 1kg con la ganancia asociada igual a 2.
Como no hay otro objeto a introducir perteneciente a etapas previas, copiamos esta casilla en las restantes (de la casilla 2 a la de 8kg).

Etapa #2:

- Introducimos el objeto B en el casillero de 2kg con la ganancia asociada igual a 5, manteniendo los valores de las casilla 0-1 de la etapa #1.
- En el siguiente casillero, verificamos si podemos introducir uno o mas objetos que totalicen 3kg => A + B cumplen la condición con una ganancia de 2+5 = 7.
- Colocamos 7 en el casillero 3(kg) haciendo referencia a los objetos A+B y copiamos este valor al resto de casillas porque ya no hay mas objetos que introducir.

1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	Artículo	Dimensión (Kg)	Ganancia (€)	0	1	2	3	4	5	6	7	8
0	Estado inicial			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #3: No introducimos el objeto C en el casillero de 4kg con la ganancia asociada igual a 6 porque la ganancia de la etapa #2 para el casillero 4 fue de 7 (mayor a 6, por tanto, lo dejamos en 7).

Evaluamos que otros objetos podemos introducir en los siguientes casilleros mayores a 4kg:

- En la casilla de 5kg podemos colocar los objetos A y C (5Kg) que totalizan una ganancia 8 (mayor a 7 de la etapa anterior)
- En la casilla de 6kg podemos colocar los objetos B y C (6kg) que totalizan una ganancia de 11 (mayor a 7 de la etapa anterior)
- En la casilla de 7kg podemos colocar los objetos A, B y C (7kg) que totalizan una ganancia de 13 (mayor a 7 de la etapa anterior)
- En la casilla de 8kg ya no podemos colocar mas objetos, por tanto copiamos el resultado de la casilla de 7kg.

1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	Artículo	Dimensión (Kg)	Ganancia (€)	0	1	2	3	4	5	6	7	8
0	Estado inicial			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C
4	D	5	10	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #4: Introducimos el objeto D en el casillero de 5kg con la ganancia asociada igual a 10, manteniendo los valores de las casilla 0-4 de la etapa #3.

Evaluamos que otros objetos podemos introducir en los siguientes casilleros mayores a 5kg:

- En la casilla de 6kg podemos colocar los objetos A y D (6Kg) que totalizan una ganancia 12 (mayor a 11 de la etapa anterior)
- En la casilla de 7kg podemos colocar los objetos B y D (7kg) que totalizan una ganancia de 15 (mayor a 13 de la etapa anterior)
- En la casilla de 8kg podemos colocar los objetos A, B y D (8kg) que totalizan una ganancia de 17 (mayor a 13 de la etapa anterior)

1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	Artículo	Dimensión (Kg)	Ganancia (€)	0	1	2	3	4	5	6	7	8
0	**Estado inicial**			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C
4	D	5	10	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D
5	E	7	13	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #5: No introducimos el objeto E en el casillero de 7kg con la ganancia asociada igual a 13 porque la ganancia de la etapa #4 para el casillero 7 fue de 15 (mayor a 13, por tanto, lo dejamos en 15).
Evaluamos que otros objetos podemos introducir en los siguientes casilleros mayores a 7kg:

- En la casilla de 8kg podemos colocar los objetos A y E que totalizan una ganancia de 15 (menor a 17 de la etapa anterior). Lo dejamos con 17.

1. Aplicaciones de la Programación Dinámica

El problema de la Mochila

1kg - 2€
2kg - 5€
4kg - 6€
5kg - 10€
7kg - 13€
8kg - 16€

				GANANCIAS EN DIMENSION (Kg.)								
Etapa	Artículo	Dimensión (Kg)	Ganancia (€)	0	1	2	3	4	5	6	7	8
0	Estado inicial			0	0	0	0	0	0	0	0	0
1	A	1	2	0	2 A	2 A	2 A	2 A	2 A	2 A	2 A	2 A
2	B	2	5	0	2 A	5 B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B	7 A+B
3	C	4	6	0	2 A	5 B	7 A+B	7 A+B	8 A+C	11 B+C	13 A+B+C	13 A+B+C
4	D	5	10	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D
5	E	7	13	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D
6	F	8	16	0	2 A	5 B	7 A+B	7 A+B	10 D	12 A+D	15 B+D	17 A+B+D

0 – 8 kg. (máximo)

Estado inicial: Mochila vacía

Etapa #6: No introducimos el objeto F en el casillero de 8kg con la ganancia asociada igual a 16 porque la ganancia de la etapa #5 para el casillero 8 fue de 17 (mayor a 16, por tanto, lo dejamos en 17).

Hemos llegado al **Estado final = Mochila llena = 8kg => Maximizando las ganancias a 17 €**

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

- Este es uno de los famosos problemas de programación dinámica que se pregunta principalmente en las entrevistas técnicas para ingresar a las principales empresas.
- El problema consiste en hacer un cambio del valor dado de centavos donde se tiene un suministro infinito de cada una de las monedas valoradas en C = {c1, c2,….cm}.

PLANTEAMIENTO DEL PROBLEMA

Dado un conjunto de denominaciones de monedas disponibles y un precio objetivo. Encuentre la cantidad mínima de monedas requeridas para pagar lo mismo.

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

PASOS A SEGUIR

1. Podemos comenzar la solución con suma = N centavos.

2. En cada iteración, encontramos las monedas mínimas requeridas dividiendo el problema original en subproblemas.

3. Consideramos una moneda de { 1, c2,…cm} y reducimos la suma repetidamente dependiendo de la moneda de la denominación que se elija.

4. Repetir el mismo proceso hasta que N se convierta en 0, y en este punto, encontramos la solución.

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

Ejemplo: Analicemos el caso de Perú que tiene las monedas de céntimos.

Ahora imaginemos que existiera una moneda más de 25 céntimos:

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

Ejemplo: Analicemos el caso de Perú que tiene las monedas de céntimos.

Ahora imaginemos que existiera una moneda más de 25 céntimos:

La solución no consiste

- En tomar siempre las monedas de mayor valor hasta llegar al número (como lo hicimos en el enfoque codicioso).

En DP debemos generalizar el problema

- Las denominaciones serán $d_1, d_2, d_3, \ldots, d_k$
- Estas denominaciones estarán ordenadas, es decir:

$$d_1 < d_2 < d_3 < \ldots < d_k$$

- Así, el último ejemplo tiene:

$$d_1 = 1, \; d_2 = 5, \; d_3 = 10, \; d_4 = 20, \; d_5 = 25, \; d_6 = 50$$

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

Resolveremos este problema en 4 pasos.

<u>Pasos a seguir</u>

<u>Paso #1:</u> Describir la estructura de una solución optima

- La mejor solución al problema tiene la mejor solución a sus subproblemas.
- Por ejemplo, para 85 céntimos, el óptimo es (10,25,50):

Ambas soluciones son optimas

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

Pasos a seguir

Paso #2: Definir recursivamente el valor de una solución

- Voy a almacenar en C[p], la cantidad mínima de monedas para cambiar p centavos.
- ¿Cuál es mi caso base?

  C[0] = 0

- ¿Cómo calculamos otros C[p] para otros p?

  C[p] = min $_{i:\, d_i \leq p}$ {1+C[p - d_i]}
- Luego, para construir la solución, guardamos en S[p] la moneda escogida.

- **Ejemplo:** Si quiero calcular el mínimo de monedas para n = 85 con las monedas:

d_1 d_2 d_3 d_4 d_5 d_6

- Dijimos que: C[p] = min $_{i:\, d_i \leq p}$ {1+C[p - d_i]}

C[85] = min{1+d[84], 1+d[80], 1+d[75], 1+d[65], 1+d[60], 1+d[35]}

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

<u>Pasos a seguir</u>

<u>Paso #2:</u> Definir recursivamente el valor de una solución

- La GENERALIZACION quedaría así:

$$C[p] \begin{cases} 0, \text{ si } p = 0 \\ \min_{i:di \leq p}\{1+C[p-d_i]\} \text{ , si } p \neq 0 \end{cases}$$

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

Pasos a seguir

Paso #3: Hallar el valor de una solución optima.

d[ ] = lista de denominaciones de monedas
n = cantidad a sencillar
k = cantidad de denominaciones de monedas

```
Sencillar(d[], n)
    k ← length(d)
    C[0] ← 0                    //caso base
    for p ←1 to n
      min ← infinito
      for i ←1 to k
         if d[i] <=p then               //filtro monedas ≤ p
            if 1 + C[p-d[i]] < min then
               min ← 1 + C[p-d[i]]
               moneda ← i
      C[p] ← min                //guardo el mínimo y
      Sol[p] ← moneda           //la moneda escogida
    Retornar C y Sol
```

1. Aplicaciones de la Programación Dinámica

El problema del Cambio mínimo de Monedas

<u>Pasos a seguir</u>

<u>Paso #4:</u> Construir la solución optima.

- Ya tenemos un valor óptimo en C[p], ahora debemos construir la solución con S[]

```
Reportar(S, d, n)
   Imprimir "Número mínimo de monedas: " + C[n] + " y son: "
   While n > 0 do
      Imprimir d[Sol[n]] + ", "
      n ← n - d[Sol[n]]
```

Resolvimos el problema del cambio mínimo de monedas con la construcción de una solución optima.

1. Aplicaciones de la Programación Dinámica

CONCLUSIONES

1. La complejidad del tiempo de ejecución del problema de la mochila es **O(N*W)** donde N es el número de artículos dados y W es la capacidad de la mochila.
2. La complejidad del tiempo de ejecución del problema mínimo de cambio de moneda es **O(m*n)**, donde m es el número de monedas y n es el cambio requerido.
3. El registro tabular de los resultados simplifican realizar cálculos desde el inicio
4. Hemos aprendido a manejar y desarrollar los principios del paradigma de programación dinámica.

PREGUNTAS

Dudas y opiniones