

# Algoritmos de programación dinámica

- Planteamiento general
- Ejemplos de aplicación
  - Problema de la mochila 0/1
  - Problema de la devolución del cambio
  - Problema de los caminos de coste mínimo
  - Problema del viajante

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- **Aplicable a problemas que cumplen**
  - Son problemas de optimización
  - La solución (óptima) se puede construir en base a una serie de decisiones sucesivas
- **Ejemplos**
  - Problema de la mochila
  - Camino de coste mínimo
  - ¿Ordenar un vector?

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- La Programación Dinámica (PD) es un punto medio entre
  - La fuerza bruta
    - Construir TODOS los candidatos a solución
  - El enfoque de avance rápido
    - Construir un único candidato a solución
    - Aplicable en problemas en que cada decisión tomada se puede mantener definitivamente
  - En general, reducción drástica del número de candidatos explorados/construidos

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- Cada candidato corresponde a una secuencia de decisiones
- Sólo se construyen aquellos candidatos que son claramente viables
- Se usa como filtro el *principio del óptimo*
  - ... si es aplicable, si no, descartar el enfoque de PD para resolver el problema

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- Principio del óptimo (PO)
  - Una secuencia óptima de decisiones tiene la propiedad de que, cualesquiera que sean su estado inicial y la primera decisión, las restantes decisiones constituyen una secuencia óptima con respecto al estado alcanzado tras la primera decisión

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- Cumplen el PO
  - Problema del camino de coste mínimo
  - Problema de la mochila 0/1
- No cumple el PO
  - Problema del camino simple de coste máximo

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- El PO se puede aplicar operativamente para derivar un algoritmo de PD
  - Problema de la mochila 0/1
- Se puede generalizar para cualquier subconjunto de decisiones

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- Se puede aplicar operativamente a decisiones intermedias
  - Enfoque hacia delante
    - La decisión  $d_i$  se hace en términos de  $d_{i+1}, \dots, d_n$
    - ¡Se calcula hacia atrás!
  - Enfoque hacia detrás
    - La decisión  $d_i$  se hace en términos de  $d_0, \dots, d_{i-1}$
    - ¡Se calcula hacia delante!
- Problema de la mochila 0/1

Algorítmica - José María Gómez Hidalgo

## Planteamiento general

- Antes de aplicar PD, comprobar que el problema cumple
  - ☑ Que es de optimización
  - ☑ Que la solución (óptima) se puede construir en base a una serie de decisiones sucesivas
  - ☑ El PO

Algorítmica - José María Gómez Hidalgo

## Ejemplos de aplicación

- Problema de la mochila 0/1
- Problema de la devolución del cambio
- Problema de los caminos de coste mínimo
- Problema del viajante
- Otros

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Problema
  - Dados  $n$  objetos, cada uno con un peso  $w_i$  y un beneficio asociado  $p_i$  ( $1 \leq i \leq n$ ), llenar una mochila de capacidad (peso máximo)  $M$ , de modo que se maximice el beneficio
  - No se pueden tomar partes de los objetos
    - La solución es un vector  $X$  de tal que  $x_i = 0, 1$

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Formalmente

MOCHILLA( $i, j, Y$ )

maximizar  $f(X) = \sum p_k \cdot X_k$

sujeito a  $\sum w_k \cdot X_k \leq Y$

con  $x_k \in \{0, 1\}$ ,  $p_k, w_k > 0$  para  $i \leq k \leq j$

=> buscamos solucionar MOCHILLA( $1, n, M$ )

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- No admite solución de avance rápido
  - $M = 7$
  - $P = [10, 7, 5]$
  - $W = [5, 4, 3]$
- Aproximación por avance rápido  $X = [1, 0, 0]$ ,  
 $f(X) = 10$
- Solución  $S = [0, 1, 1]$ ,  $f(S) = 12$

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Enfoque general para  $p_i$ ,  $w_i$  números reales
- Dos pasos
  - Cálculo del valor óptimo de la función objetivo
  - Reconstrucción de la solución

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Sea  $f_i(Y)$  el valor de la función objetivo para el problema MOCHILA(1,i,Y)
- Entonces (*enfoque hacia atrás*)  
 $f_i(Y) = \max(f_{i-1}(Y), f_{i-1}(Y - w_i) + p_i)$
- Con
  - $f_n(M)$  objetivo a calcular
  - $f_0(Y) = 0$  para todo  $Y \geq 0$
  - $f_i(Y) = -\infty$  para todo  $Y < 0, i = 1, \dots, n$

Algorítmica - José María Gómez Hidalgo

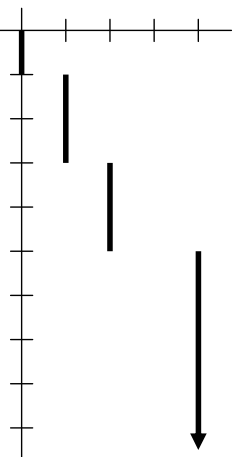
## Problema de la mochila 0/1

- Enfoque de cómputo
  - En vez de calcular  $f_n(M)$ , se calcula la función  $f_n(X)$  usando
    - $f_i(X) = \max(f_{i-1}(X), f_{i-1}(X - w_i) + p_i)$
    - Llamamos  $g_{i-1}(X) = f_{i-1}(X - w_i) + p_i$
    - $f_{i-1}(X)$  se calcula a partir de  $f_{i-1}(X)$  realizando una translación de la curva  $w_i$  a la derecha, y  $p_i$  hacia arriba
  - Ejemplo:  $n = 3, M = 6, W = \{2,3,4\}, P = \{1,2,5\}$

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Computacionalmente
  - Las funciones  $f_i(X)$  y  $g_i(X)$  son escalonadas y se pueden representar como listas de puntos  
e.g.  $f(X) = \{(0,0), (1,1), (3,2), (5,4)\}$  corresponde a la función



Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Definimos dos funciones
  - desplazar( $f,(x,y)$ ) que suma  $(x,y)$  a todos los pares de la lista que representa la función  $f$
  - máximo( $f,g$ ) que obtiene el máximo de las dos funciones  $f$  y  $g$ , por medio de una mezcla ordenada de los pares de ambas
    - se eliminan los pares  $(x,y)$  t.q.  
(a) existe  $(x',y')$  con  $x' \leq x$  é  $y' \geq y$   
(b)  $y > M$

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Recuperación de la solución
    - Queremos obtener  $x_i$  a partir de  $f_i$   
Inicio: Buscar  $(M, f(M))$  en  $f_{n-1}$   
Si  $(M, f(M)) \in f_{n-1}$ , entonces  $x_n = 0$   
y buscar  $(M, f(M))$  en  $f_{n-2}$   
Si  $(M, f(M)) \notin f_{n-1}$ , entonces  $x_n = 1$   
y buscar  $(M - w_n, f_n(M) - p_n)$  en  $f_{n-2}$
- Repetir: ...

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Complejidad de un algoritmo de vuelta atrás
  - Construir todas las secuencias de  $n$  valores 0 ó 1  
 $\Rightarrow O(2^n)$
- Complejidad del algoritmo de PD
  - La lista para la función  $f_i(X)$  contiene a los sumo  $2^i$  pares, y las funciones son lineales en la longitud de las listas  $\Rightarrow$  caso peor  $O(\sum 2^i) = O(2^n)$   
!!! Tan ineficiente como la vuelta atrás!!!

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- **Caso especial**
    - Los elementos de P y de W son números enteros
    - Podemos realizar una implementación substancialmente más eficiente aprovechando una estructura de matriz lineal para representar una función  $f_i(X)$
- $f_i(\cdot) \equiv F(i) = \text{matriz } [0..M] \text{ de entero}$   
de manera que  $f_i(X) = F(i)(X) = F(i, X)$
- Ejemplo:  $n = 3$ ,  $M = 6$ ,  $W = \{2,3,4\}$ ,  $P = \{1,2,5\}$

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

```
Función mochilaPD(P,W: matriz [1..n] de entero, M: entero
    S: matriz [1..n] de entero): entero
    para i = 1 hasta n hacer
        para j = 0 hasta M hacer
            F(i,j) = max(F(i-1,j),F(i-1,j-W(i))+P(i))
    i = n; w = M
    mientras i ≥ 1 hacer
        si F(i,w) = F(i-1,w)
            entonces S(i) = 0
        sino S(i) = 1; w = w - W(i)
        i = i - 1
    devolver F(n,M)
```

Algorítmica - José María Gómez Hidalgo

## Problema de la mochila 0/1

- Complejidad
  - Recorremos la matriz completa  $\Rightarrow \theta(n.M)$ 
    - si  $n.M \ll 2^n$  la implementación es muy eficiente
    - aunque, para el ejemplo,  $n.M = 18$  y  $2^n = 8$

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

- Muy similar en su solución al problema de la mochila 0/1
- Planteamiento
  - Dado  $D$  un sistema monetario con unos valores de monedas  $D_1, \dots, D_n$ , devolver una cantidad  $C$  *minimizando el número de monedas*
  - Se suponen  $\infty$  monedas de cada valor  $D_i$
- Observación: el avance rápido no siempre funciona ( $D = (1, 4, 6)$ ,  $C = 8$ )

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

- ¿Cumple las condiciones?
  - ☑ Es de optimización
  - ☑ La solución (óptima) se puede construir en base a una serie de decisiones sucesivas e.g.
    - dec<sub>i</sub> ≡ número de monedas de valor d<sub>i</sub> que se toman => n posibles decisiones
  - ☑ El PO

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

- Para facilitar implementación
  - la decisión dec<sub>i</sub> no será el número de monedas de valor d<sub>i</sub> que tomamos, sino
  - la decisión dec<sub>k</sub> es si tomamos una moneda más de valor d<sub>i</sub> o no
    - Es posible que haya más de n decisiones

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

- Definimos  $f_i(Y)$  como el número mínimo de monedas para devolver  $Y$  con los valores de monedas  $d_1, \dots, d_i$ , es decir, el valor de la solución óptima del problema CAMBIO(1,  $i, Y$ )
- Queremos calcular  $f_n(C)$
- Aplicamos P.O. con enfoque hacia detrás  
 $f_n(C) = \min(f_{n-1}(C), f_n(C-d_i)+1)$

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

- Generalización  
 $f_i(Y) = \min(f_{i-1}(Y), f_i(Y-d_i)+1)$   
 $f_i(Y) = -\infty$  si  $Y < 0$   
 $f_1(Y) = Y \text{ div } d_1$  si  $(Y \text{ mod } d_1) = 0$  (es posible devolver cambio)  
 $f_1(Y) = -\infty$  si  $(Y \text{ mod } d_1) \neq 0$  (no es posible devolver cambio)
- Ejemplo:  $D = (1, 4, 6)$ ,  $C=8$

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

- Como en el problema de la mochila entera
  - En lugar de calcular  $f_n(C)$ , calculamos la función  $f_n(X)$  completa
  - Como  $d_i$  y  $C$  enteros, usamos como representación de  $f_i(X)$  la estructura  $F = \text{matriz } [1..n, 0..C]$  de entero
  - Para recuperar la solución, examinamos  $F(n,C)$  y vamos reconstruyendo las decisiones tomadas

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

```
Función cambioPD(D: matriz [1..n] de entero, C: entero
    S: matriz [1..n] de entero): entero
    para i = 1 hasta n hacer
        para j = 0 hasta C hacer
            F(i,j) = min(F(i-1,j),F(i,j-D(i))+1)
    i = n; K = C
    mientras i > 1 hacer
        si F(i,K) ≠ F(i-1,K)
            entonces S(i) = S(i) + 1; K = K - D(i)
        sino i = i - 1
    S(1) = F(1,K)
    devolver F(n,C)
```

Algorítmica - José María Gómez Hidalgo

## Problema de la devolución del cambio

- Complejidad
  - Construcción de la matriz  $\in \theta(n.C)$
  - Reconstrucción de la solución  $\in \theta(n+C)$
  - Global  $\in \theta(n.C)$

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

- Dado un grafo  $G = (V,A)$ , encontrar el camino de coste mínimo entre cada par de vértices  $i, j$  de  $V$  tal que  $i \neq j$
- Disponemos de un enfoque de avance rápido (algoritmo de Dijkstra) que, adaptado, funciona en  $\theta(n^3)$  (restricción  $|a| \geq 0$  para toda  $a \in A$ )
- Sólo imponemos que  $G$  no tenga ciclos de longitud negativa

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

- ¿Cumple las condiciones?
    - ☑ Es de optimización
    - ☑ La solución (óptima) se puede construir en base a una serie de decisiones sucesivas e.g.
      - dec  $\equiv$  a que vértice vamos desde aquél en el que estamos
    - ☑ El PO
- Sea  $CAM(i,j)$  el camino de coste mínimo entre  $i$  y  $j$ ,  $k$  en  $CAM(i,j)$ ; entonces, los subcaminos  $CAM(i,k)$  y  $CAM(k,j)$  también son óptimos

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

- Primero construimos una función de optimización
  - $C(i,j)$  es la longitud del camino de coste mínimo entre  $i$  y  $j$
- La decisión en un momento dado es
  - dec = ¿cuál es el vértice  $k$  de índice mayor que está en el camino  $CAM(i,j)$ ?

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

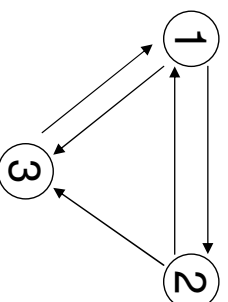
- Definimos
    - $A_k(i,j)$  = longitud del camino de coste mínimo entre  $i$  y  $j$  pasando a lo sumo por los vértices  $1$  a  $k$
    - Buscamos  $A_n(i,j)$  para todos  $i,j \in V, i \neq j$
  - Por el principio del optimo
    - $A_n(i,j)$  dependerá de si el vértice  $n$  aparece en el camino de coste mínimo entre  $i$  y  $j$
- $$A_n(i,j) = \min(A_{n-1}(i,j), A_{n-1}(i,n) + A_{n-1}(n,j))$$

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

- Generalizando
  - $A_k(i,j) = \min(A_{k-1}(i,j), A_{k-1}(i,k) + A_{k-1}(k,j))$
  - $A_0(i,j) = G.A(i,j)$
- Ejemplo

$$G.A = \begin{pmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{pmatrix}$$



Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

```
Función caminosPD1(G: grafo): matriz [1..n, 1..n] de real  
var C: matriz [1..n, 1..n] de real  
para i = 1 hasta n hacer  
  para j = 1 hasta n hacer  
    C(i,j) = G.A(i,j)  
  para i = 1 hasta n hacer  
    para j = 1 hasta n hacer  
      para k = 1 hasta n hacer  
        C(i,j) = min(C(i,j), C(i,k) + C(k,j))  
    devolver C
```

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

- **Complejidad**
  - $\Theta(n^3)$  equivalente al algoritmo de Dijkstra pero con menor constante oculta
- **Falta reconstruir los caminos y no hemos almacenado las decisiones**
  - Seguiremos una estrategia similar a la del algoritmo de Dijkstra

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

- En la matriz  $P$  se almacena el valor de la decisión ( $k$  está en el camino entre  $i$  y  $j$  o no) a medida que se toma  
 $P =$  matriz  $[1..n, 1..n]$  de entero
- Si no se toma  $k$ , no se modifica  $P(i,j)$
- Si se toma  $k$ , se pone  $P(i,j) = k$   
 $\Rightarrow P(i,j)$  almacena siempre el vértice de mayor índice entre  $i$  y  $j$

Algorítmica - José María Gómez Hidalgo

## Problema de los caminos mínimos

- Se puede diseñar un algoritmo que, a partir de  $P$ , devuelva el camino más corto entre  $i$  y  $j$  (similar a Dijkstra)
- Funciona en  $O(n)$  para dos vértices, y en  $O(n^3)$  para los caminos para todos los pares de vértices

Algorítmica - José María Gómez Hidalgo

# Otros problemas

- Otros problemas clásicos
  - Problema del viajante
  - Problema de los árboles de decisión mínimos
  - Problema de la subcadena común más larga