

# 3. Algoritmos de avance rápido

## 1. Planteamiento general

## 2. Ejemplos de aplicación

2.1. Problema de la mochila

2.2. Problemas sobre grafos

2.3. Otros problemas

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Problemas de optimización
  - Dados un conjunto de datos de entrada y una función objetivo (a maximizar o minimizar), encontrar un subconjunto de los datos que
    - (a) optimice la función objetivo y
    - (b) satisfaga un conjunto de restricciones adicionales

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- **Solución factible**
  - Cualquier subconjunto del conjunto de entrada que satisface las restricciones
- **Solución óptima**
  - Solución factible para la que la función objetivo alcanza un valor óptimo

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- **Idea de un algoritmo de avance rápido**
  - Avanzar paso a paso, seleccionando un elemento por vez
- **Esquema**
  - Empezar con la solución =  $\emptyset$
  - Repetir
    - Seleccionar el elemento más prometedor de la entrada
    - Si al agregarlo a la solución, permanece factible, hacerlo

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Ejemplo - problema de devolución del cambio
  - 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  $D_i$
  - En España con monedas de valores  $D = \{1, 5, 10, 25, 100\}$ , devolver  $C = 289$ 
    - Solución factible = 289 monedas de valor 1
    - Solución óptima = 2 de 100, 3 de 25, 1 de 10, 4 de 1

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

```
función DevolverCambio(D: matriz[1..n] de entero, C:
entero): multiconjunto de entero
sol =  $\emptyset$ 
cant = 0
mientras (cant  $\neq$  C)
  hacer x = seleccionar(C, cant, D)
  si x = 0 /* no hay valor adecuado de moneda */
    entonces devolver  $\emptyset$  /* error */
  sino   sol = sol U {x}
        cant = cant + x
devolver sol
```

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- **Observaciones**
    - Este algoritmo se basa en la función seleccionar seleccionar( $C$ , cant,  $D$ ) obtiene el mayor valor de moneda de  $D$  que sea menor o igual que  $C$  - cant
    - El algoritmo devuelve un error codificado como  $\emptyset$  cuando no es posible efectuar la devolución
- p. ej.  $D = [2, 4, 6]$  y  $C = 5$

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- **Observaciones**
    - En determinados sistemas monetarios, no funciona correctamente
- p. ej. Sistema Isabelino (Inglaterra, s. XIX)
- $D = [1, 3, 6, 12, 24, 30]$ ,  $C = 48$
- DevolverCambio  $\Rightarrow$   $\{30, 12, 6\}$
- Solución óptima  $\Rightarrow$   $\{24, 24\}$
- En la actualidad, los sistemas monetarios están diseñados de modo que funciona correctamente

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- Características del enfoque
  - Se selecciona un sólo elemento por vez, aceptándolo o rechazándolo
  - No cambia de opinión => la decisión es definitiva
  - La selección se realiza de acuerdo con la función objetivo => el elemento más prometedor

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- Esquema

función AvanceRápido(C: conjunto): (multi)conjunto

S =  $\emptyset$

mientras (C  $\neq \emptyset$ ) y no(solucion(S))

hacer x = seleccionar(C)

C = C - {x}

si (factible(S U {x}))

entonces S = S U {x}

si solucion(S) entonces devolver S

sino devolver error

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- Complejidad
  - Dependiente de la función de selección
  - Dependiente del orden inicial de los elementos de entrada
  - En general, cuando es aplicable, más rápido que los demás enfoques (excepto, tal vez, divide y vencerás)

Algorítmica - José María Gómez Hidalgo

## 2.1. Problema de la mochila

- 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
  - Se pueden tomar partes de los objetos
    - La solución es un vector  $X$  de tal que  $0 \leq x_i \leq 1$

Algorítmica - José María Gómez Hidalgo

## 2.1. Problema de la mochila

- Formalmente

$$\begin{aligned} &\text{maximizar} && f(X) = \sum p_i \cdot X_i \\ &\text{sujeto a} && \sum w_i \cdot X_i \leq M \\ &\text{con} && 0 \leq X_i \leq 1, p_i, w_i > 0 \text{ para } 1 \leq i \leq n \end{aligned}$$

Algorítmica - José María Gómez Hidalgo

## 2.1. Problema de la mochila

- Condiciones
  - Sólo interesa el caso  $W \cdot 1 > M$
  - Toda solución óptima en ese caso cumple que  $W \cdot X = M$

Algorítmica - José María Gómez Hidalgo

## 2.1. Problema de la mochila

- Ejemplo

$M = 20$ ,  $P = (25, 24, 15)$ ,  $W = (18, 15, 10)$

Algunas soluciones factibles

$X1 = (1/2, 1/3, 1/4)$ ,  $f(X) = 24.25$

$X2 = (1, 2/15, 0)$ ,  $f(X) = 28.2$

$X3 = (0, 2/3, 1)$ ,  $f(X) = 31$

$X4 = (0, 1, 1/2)$ ,  $f(X) = 31.5$  (óptima)

Algorítmica - José María Gómez Hidalgo

## 2.1. Problema de la mochila

- Enfoques de avance rápido
  - Tomar los elementos por orden de beneficio
    - Estrategia no óptima
  - Tomar los elementos por orden inverso de peso
    - Estrategia no óptima
  - Tomar los elementos por orden de  $p_i/w_i$ 
    - Estrategia óptima

Algorítmica - José María Gómez Hidalgo

## 2.1. Problema de la mochila

```
función mochila(P, W: matriz [1..n] de real, M: real): matriz
[1..n] de real
/* Suponemos P, W ordenados por P[i] / W[i] */
para i = 1 hasta n hacer S[i] = 0
capacidad = M; i = 1
mientras ((i ≤ n) y (W[i] ≤ capacidad)) /* (1) */
hacer capacidad = capacidad - W[i]
    S[i] = 1; i = i + 1
si (i ≤ n)
entonces S[i] = capacidad / W[i]
devolver S
```

Algorítmica - José María Gómez Hidalgo

## 2.1. Problema de la mochila

- Complejidad
  - Caso peor
    - Tomando la condición (1) como barómetro
    - $t(n) \in \theta(n)$
  - Caso mejor
    - $t(n) \in \theta(1)$
  - Si se tiene en cuenta la ordenación previa
    - $t(n) = t_{\text{ord. rápida}}(n)$

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

- Grafo  $G = (V, A)$ , con  $V$  conjunto de etiquetas (vértices) y  $A \subseteq V \times V$  conjunto de aristas
- Grafo dirigido versus grafo no dirigido
- Camino y ciclo
- Grafo conexo y fuertemente conexo
- Representación como matriz de adyacencia vs. listas de adyacencia
  - Acceso  $\in \theta(1)$ , espacio  $\in \theta(n^2)$  (matriz)
  - Acceso  $\in O(n)$ , espacio  $\in O(n^2)$  (lista)

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos Árbol generador mínimo

- Problema
  - Dado un grafo conexo  $G = (V, A)$  con aristas etiquetadas con números positivos (por ejemplo, longitudes), encontrar  $A' \subseteq A$  tal que  $G' = (V, A')$  sea conexo y la suma de las etiquetas de las aristas de  $A'$  sea mínima
  - $G'$  = árbol generador mínimo de  $G$

Algorítmica - José María Gómez Hidalgo

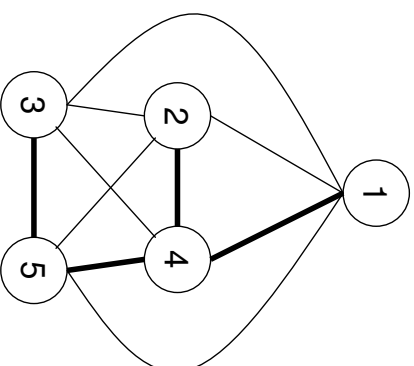
## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Ejemplo
  - $V = (1, \dots, 5)$
  - $A = ($ 

0	7.5	10	7	9.5
7.5	0	3.5	1.5	4
10	3.5	0	4	2.5
7	1.5	4	0	3
9.5	4	2.5	3	0

 $)$
  - Solución =  $((1,4), (2,4), (3,5), (4,5))$  (suma = 14)



Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Características de la solución
  - Si el grafo es conexo, existe solución
  - La solución debe tener al menos  $n-1$  aristas
    - Para que el árbol generador mínimo sea conexo
  - La solución debe tener como mucho  $n-1$  aristas
    - Para que el árbol generador mínimo no tenga ciclos, en cuyo caso sobran aristas

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Enfoques de avance rápido
  - (Kruskal) Seleccionar en cada paso la arista de peso mínimo de las que quedan y decidir si se agrega o no a la solución
    - No debe formar un ciclo
  - (Prim) Partir de un vértice cualquiera y en cada paso agregar una arista de peso mínimo para alcanzar un vértice no visitado

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

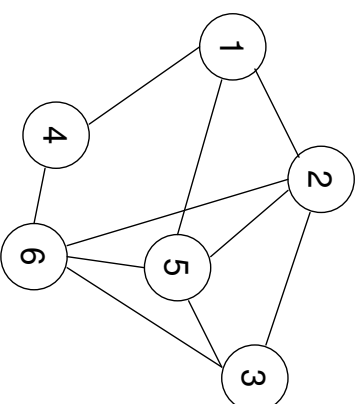
- Algoritmo de Kruskal
  - Partir de Sol = vacía
  - Mientras  $|Sol| < n-1$  y quedan aristas sin marcar
    - Seleccionar la arista A más pequeña no marcada
    - Marcarla
    - Si A no crea un ciclo al agregarla a Sol, hacerlo
    - Si no, descartar A

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Ejemplo

$$- A = \begin{pmatrix} 0 & 10 & \infty & 30 & 45 & \infty \\ 10 & 0 & 50 & \infty & 40 & 25 \\ \infty & 50 & 0 & \infty & 35 & 15 \\ 30 & \infty & \infty & 0 & \infty & 20 \\ 45 & 40 & 35 & \infty & 0 & 55 \\ \infty & 25 & 15 & 20 & 55 & 0 \end{pmatrix}$$


Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Solución del ejemplo

ARISTA	COSTE	COMPONENTES CONEXAS
(1, 2)	10	(1), (2), (3), (4), (5), (6)
(3, 6)	15	(1, 2), (3), (4), (5), (6)
(4, 6)	20	(1, 2), (3, 6), (4), (5)
(2, 6)	25	(1, 2), (3, 4, 6), (5)
(1, 4)	30	rechazada
(3, 5)	35	(1, 2, 3, 4, 5, 6)

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

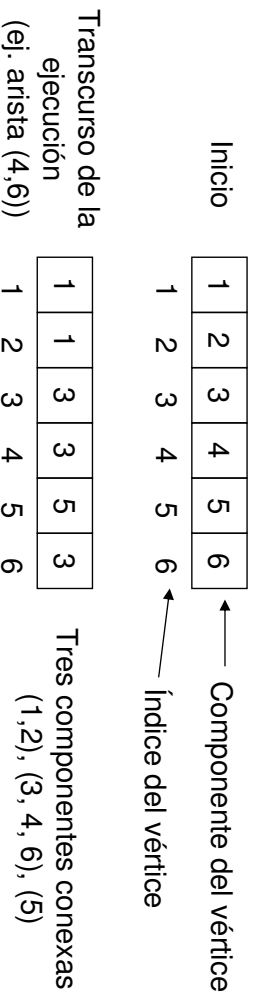
función kruskal(G: grafo): lista de arista  
sol =  $\emptyset$   
resto = G.A /\* conjunto de aristas \*/  
mientras ( $(|\text{sol}| < (n-1) \text{ y } (\text{resto} \neq \emptyset))$ )  
hacer a = seleccionar(resto)  
    resto = resto - {a}  
    si (no(ciclo(sol,a)))  
        entonces sol = sol U {a}  
devolver sol

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Implementación - función de verificar ciclos
  - Mantener información de las componentes conexas



Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Complejidad
  - Existen  $m$  aristas a recorrer (peor caso)
  - Por cada arista, se busca si los vértices están en la misma componente (tiempo constante) y se unen las componentes (peor caso  $n$ )
  - En el peor caso
    - $t(m,n) \in O(m.n)$  (cerca de  $O(n^2)$  si el grafo es poco poblado,  $O(n^3)$  si es completo)
  - Si las componentes conexas se implementan como montículos,  $t(m,n) \in O(n.\log n)$

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Algoritmo de Prim
  - Comenzar con dos componentes conexas  $B$  y  $N$ , la primera con un vértice y la segunda con los demás
  - En cada paso, seleccionar la arista de menor peso que relaciona un vértice de una componente con un vértice de la otra componente
  - Mantiene la conexión del subgrafo construido progresivamente

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Solución del ejemplo

ARISTA	COSTE	B	N
(1, 2)	10	(1)	(2, 3, 4, 5, 6)
(2, 6)	25	(1, 2)	(3, 4, 5, 6)
(3, 6)	15	(1, 2, 6)	(3, 4, 5)
(4, 6)	20	(1, 2, 3, 6)	(4, 5)
(3, 5)	35	(1, 2, 3, 4, 6)	(5)
		(1, 2, 3, 4, 5, 6)	()

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

función prim(G: grafo): lista de arista

sol =  $\emptyset$

B = {1}

N = G.V - {1}

para i = 1 hasta n-1

hacer a = seleccionar(B, N)

/\* origen en B, destino en N \*/

sol = sol U {a}

B = B U {a[2]} /\* segunda componente de a \*/

N = N - {a[2]}

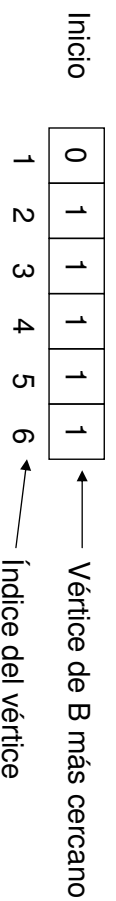
devolver sol

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Implementación - representación de B y N
  - Representamos ambos en un vector P
  - Cada componente indica el vértice más cercano en B al vértice de N
  - Un 0 indica que el vertice pertenece a B



Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Implementación - inserción en B, eliminación de N
  - Para insertar un vértice v en B, poner su componente a 0
  - Es preciso actualizar el vector
    - Se recorre el vector, y para cada componente no nula se verifica si v está más cerca que su anterior vértice cercano de B
    - Si  $P(i) \neq 0$  y  $\text{coste}(i, P(i)) > \text{coste}(i, v)$ , poner  $P(i) = v$

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Ejemplo

Inicio	0	1	1	1	1	1	0	0	0	6	3	0	3
	-	10	$\infty$	30	45	$\infty$	-	-	-	20	35	-	
1	0	0	2	1	2	2	0	0	0	0	3	0	4
	-	-	50	30	40	25	-	-	-	-	35	-	
2	0	0	6	6	2	0	0	0	0	0	0	0	Fin
	-	-	15	20	40	-	-	-	-	-	-	-	

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Árbol generador mínimo

- Complejidad
  - La selección de la arista se hace en tiempo  $n$
  - La inserción den B, eliminación de N y actualización de P se hace en tiempo de  $n$
  - Los procesos anteriores se hacen  $n-1$  veces
  - $t(m,n) \in \theta(n^2)$

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Camino de coste mínimo

- Problema
  - Dado un grafo  $G$  y un vértice cualquiera (por ejemplo, el primero), encontrar el camino más corto desde el mismo a todos los demás vértices del grafo

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Camino de coste mínimo

- Enfoque de avance rápido
  - Seleccionar cada vez un vértice, y actualizar las distancias a los vértices que faltan por alcanzar
  - Criterio de selección: el vértice no alcanzado más cercano

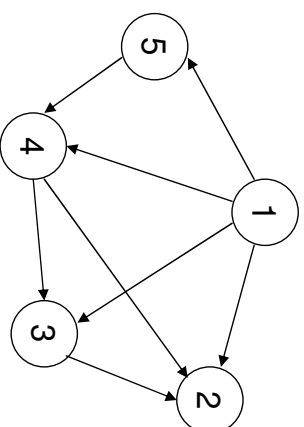
Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Camino de coste mínimo

- Ejemplo

$$- A = \begin{pmatrix} 0 & 50 & 30 & 100 & 10 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & \infty \\ \infty & 20 & 30 & 0 & \infty \\ \infty & \infty & \infty & 10 & 0 \end{pmatrix}$$



Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Camino de coste mínimo

- Ejemplo

Paso	Vértice	Resto	Distancia
Inicio	1	(2, 3, 4, 5)	(50, 30, 100, 10)
1	5	(2, 3, 4)	(50, 30, 20, 10)
2	4	(2, 3)	(40, 30, 20, 10)
3	3	(2)	(35, 30, 20, 10)

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Caminos de coste mínimo

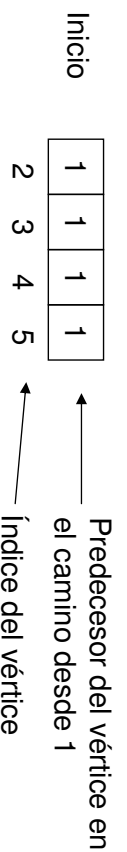
función dijkstra(G: grafo) : matriz [2..n] de real  
resto = G.V - {1}  
para i = 2 hasta n  
hacer distancia[i] = G.A(1, i)  
para i = 1 hasta n-1  
hacer v = seleccionar(resto, distancia)  
resto = resto - {v}  
para cada w ∈ resto  
hacer distancia[w] = min(distancia[w],  
distancia[v] + G.A(v,w))  
devolver distancia

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Caminos de coste mínimo

- **Observaciones**
  - El algoritmo produce las distancias mínimas, no los caminos
  - Para obtener los caminos, precisamos una matriz adicional



Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Caminos de coste mínimo

- Ejemplo

Paso	Vértice	Resto	Distancia	Predecesor
Inicio	1	(2, 3, 4, 5)	(50, 30, 100, 10)	(1, 1, 1, 1)
1	5	(2, 3, 4)	(50, 30, 20, 10)	(1, 1, 5, 1)
2	4	(2, 3)	(40, 30, 20, 10)	(4, 1, 5, 1)
3	3	(2)	(35, 30, 20, 10)	(3, 1, 5, 1)

Algorítmica - José María Gómez Hidalgo

## 2.2. Problemas sobre grafos

### Caminos de coste mínimo

- Observaciones
  - A partir de la matriz predecesor, es posible reconstruir el camino mínimo desde 1 a cualquier otro vértice
  - Complejidad
    - Inicialización  $\theta(n)$
    - Cada vuelta  $\theta(n)$
    - Además,  $n$  vueltas
      - $t(m,n) \in \theta(n^2)$

Algorítmica - José María Gómez Hidalgo

## 2.3. Otros problemas

- Existen otros problemas en los cuales los métodos de avance rápido son efectivos
- Problemas de almacenamiento
  - Almacenamiento de archivos secuenciales
  - Mezcla de archivos
- Problemas de planificación
  - Planificación de tareas con plazo fijo