- Algorismes voraços
  - Arbres d'expansió mínims
  - Camins mínims
  - Compressió de dades
- Problemes

- La tècnica voraç consisteix a prendre sempre una decisió localment òptima (la tria d'un valor candidat que dóna la millor extensió possible de la solució parcial actual) i tot seguit resoldre l'únic subproblema resultat de la decisió presa.
- La tècnica voraç es pot aplicar a aquells problemes que mostren tant el pricipi d'optimalitat local, segons el qual es pot arribar a una solució globalment òptima mitjançant una decisió localment òptima, com el principi d'estructura òptima, que estableix que tota solució òptima d'un problema conté solucions òptimes de subproblemes.
- El pricipi d'optimalitat local permet prendre la decisió que sembla millor per al problema actual, sense considerar decisions futures o solucions de subproblemes més petits.
- El principi d'estructura òptima garanteix que es pot obtenir una solució òptima d'un problema donat mitjançant una decisió voraç en cada pas.

Exemple (Màquina expenedora, cont.)

- Un algorisme voraç per al problema de la màquina expenedora consisteix a donar sempre una moneda de la denominació més gran possible i repetir el procés sobre la quantitat restant.
- Siguin $D_1 > D_2 > \cdots > D_k = 1$ les denominacions disponibles.

- ```
  procedure monedes (D, n)
    i := 1
    while n > 0 do
      if D[i] ⩾ n then
        print "moneda de denominació" D[i]
        n := n − D[i]
      else
        i := i + 1
  ```
- El cost temporal d'aquest algorisme és $\Theta(n)$.
  ```

Exemple (Màquina expenedora, cont.)

- Un algorisme voraç per al problema de la màquina expenedora consisteix a donar sempre una moneda de la denominació més gran possible i repetir el procés sobre la quantitat restant.
- Siguin $D_1 > D_2 > \cdots > D_k = 1$ les denominacions disponibles.

- **procedure** monedes (D, n)
  $i := 1$
  while $n > 0$ **do**
    $c := n$ **div** $D[i]$
    print c "monedes de denominació" $D[i]$
    $n := n - c * D[i]$
    $i := i + 1$
- El cost temporal d'aquest algorisme és $\Theta(k)$.

Lema (Principi d'optimalitat local)

Siguin $25 > 10 > 5 > 1$ *les denominacions disponibles. Sempre existeix una solució òptima al problema de la màquina expenedora per a* n *cèntims que inclou una moneda de denominació* d*, on* d *és la denominació més gran disponible tal que* $d \leqslant n$*.*

Demostració.

Considerem una solució òptima al problema de la màquina expenedora per a n cèntims que no inclou cap moneda de denominació d. Si $1 \leqslant n < 5$, aleshores $d = 1$ i la solució òptima consisteix només de monedes de 1 cèntim.

Si $5 \leqslant n < 10$, aleshores $d = 5$ i com que aquesta solució òptima no conté cap moneda de 5 cèntims, només conté monedes de 1 cèntim i canviant-ne cinc per una moneda de 5 cèntims obtenim una solució amb menys monedes.

Demostració.

Si $10 \leqslant n < 25$, aleshores $d = 10$ i com que aquesta solució òptima no conté cap moneda de 10 cèntims, només conté monedes de 5 i 1 cèntim. En particular, conté monedes de 5 i 1 cèntim que sumen 10 cèntims, i canviant-les per una moneda de 10 cèntims obtenim una solució amb menys monedes.

Finalment, si $25 \leqslant n$, aleshores $d = 25$ i com que aquesta solució òptima no conté cap moneda de 25 cèntims, només conté monedes de 10, 5 i 1 cèntim. Si conté tres monedes de 10 cèntims, les podem canviar per una moneda de 25 cèntims més una de 5 cèntims, obtenint una solució amb una moneda menys. Si conté com a molt dues monedes de 10 cèntims, aleshores conté monedes de 10, 5 i 1 cèntim que sumen 25 cèntims, i canviant-les per una moneda de 25 cèntims obtenim una solució amb menys monedes. □

Lema (Principi d'estructura òptima)

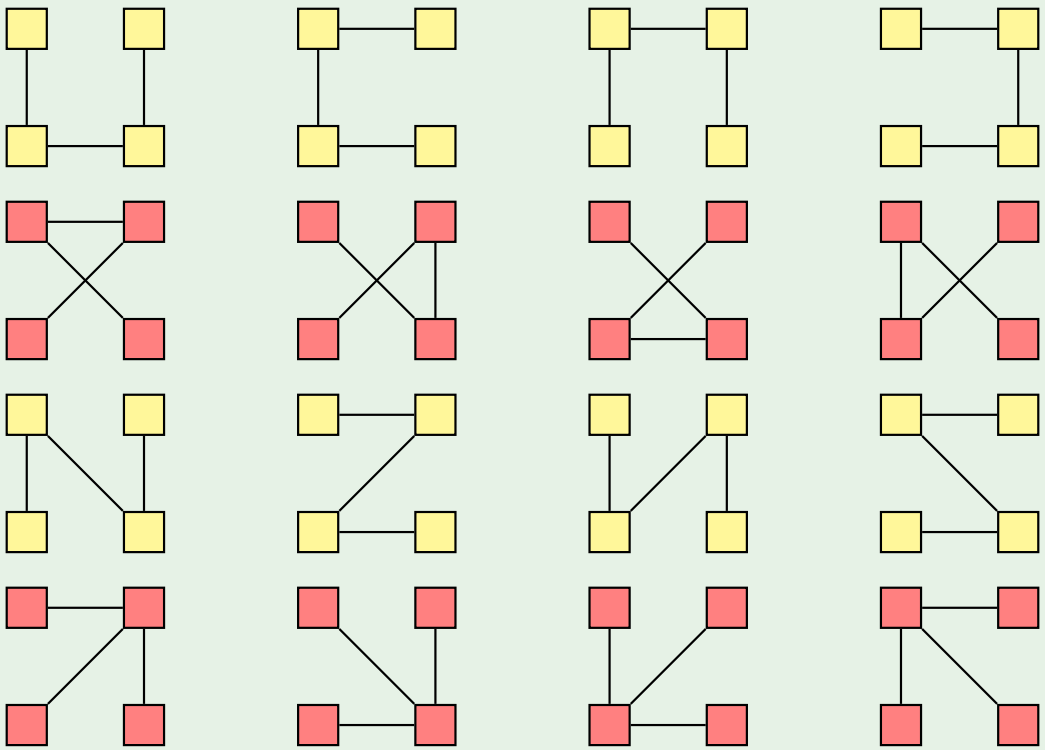
Sigui una solució òptima al problema de la màquina expenedora per a n cèntims, sigui d una denominació inclosa en aquesta solució òptima, i sigui k el nombre de monedes d'aquesta solució òptima. Aleshores, aquesta solució òptima de la instància n cèntims conté una solució òptima de la instància $n - d$ cèntims.

Demostració.

La solució de la instància $n - d$ cèntims inclosa dins la solució òptima de la instància n cèntims, conté $k - 1$ monedes. Suposem que existeix una solució de la instància $n - d$ cèntims amb menys de $k - 1$ monedes. Aleshores, podem estendre-la en una solució de la instància n cèntims amb menys de k monedes, la qual cosa contradiu la hipòtesi que la solució de la instància n cèntims és òptima. □

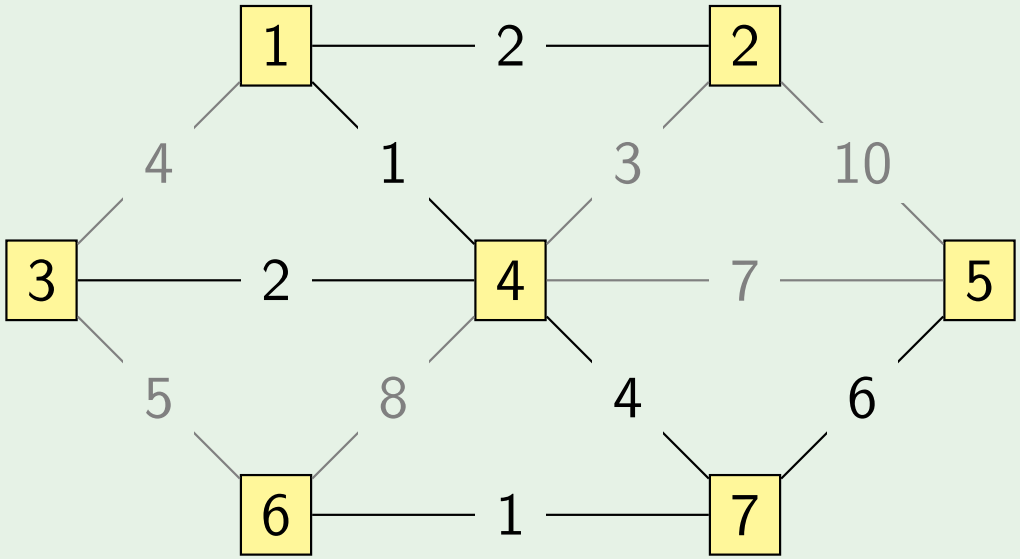
- Un arbre d'expansió d'un graf és un subgraf que conté tots els vèrtexs i que és un arbre.
- Un graf pot tenir molts d'arbres d'expansió.
- Tot graf connex té un arbre d'expansió.

Exemple (El graf complet de 4 vèrtexs, K_4, té 16 arbres d'expansió.)



- Un arbre d'expansió mínim d'un graf és un subgraf que conté tots els vèrtexs i que és un arbre de cost mínim.

Exemple



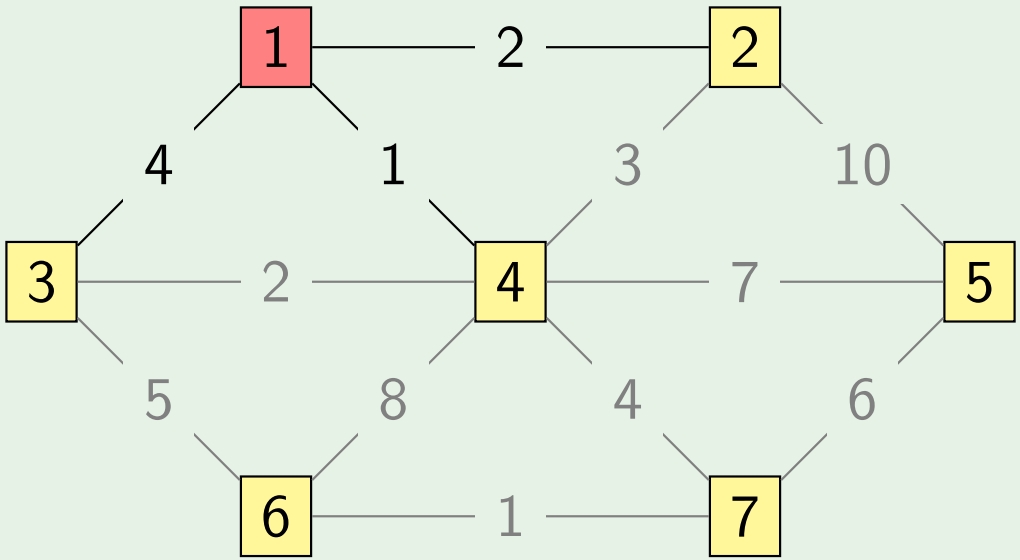
- Es pot obtenir un arbre d'expansió mínim d'un graf no dirigit G amb pesos amb l'anomenat algorisme de Prim, que consisteix a estendre un arbre inicial T amb una aresta de cost mínim entre totes aquelles arestes que uneixen un vèrtex de T amb un vèrtex de $G \setminus T$.

- **procedure** prim (G: graf)
  for all vèrtex v de G **do**
  $vist[v] := false$
  sigui v un vèrtex qualsevol de G
  for all vèrtex w adjacent amb el vèrtex v **do**
  $inserir(Q, \{v, w\}, pes[v, w])$
  $vist[v] := true$
  while not $buit(Q)$ **do**
  $\{u, v\} := elim\_min(Q)$
  if not $vist[v]$ **then**
  $push\_back(L, \{u, v\})$
  for all vèrtex w adjacent amb el vèrtex v **do**
  if not $vist[w]$ **then**
  $inserir(Q, \{v, w\}, pes[v, w])$
  $vist[v] := true$
- L'algorisme de Prim es pot implementar amb cost temporal $\Theta(m \log n)$.

Exemple

- Les arestes a la cua de prioritats fins al moment són les següents.

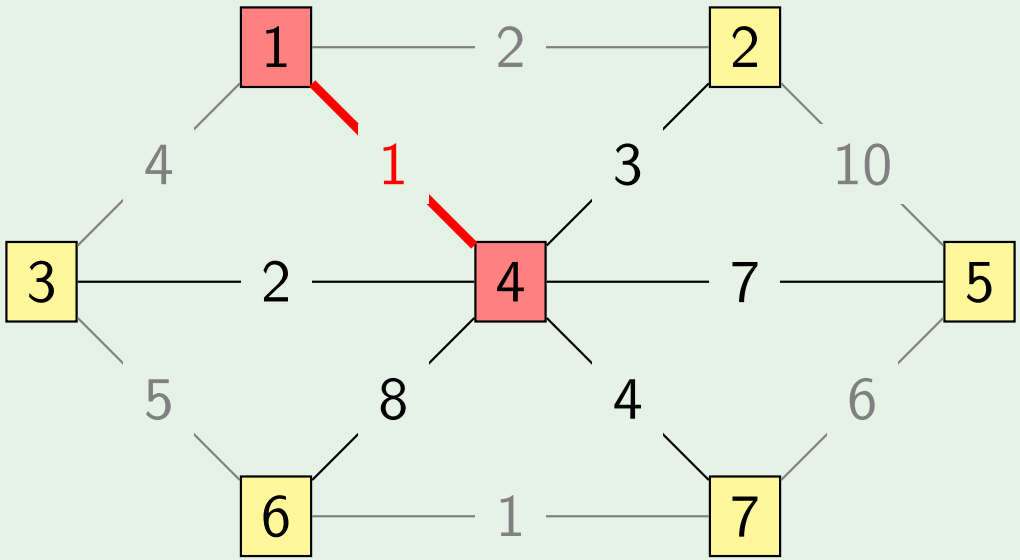
aresta	1–4	1–2	1–3
pes	1	2	4



Exemple

- Les arestes a la cua de prioritats fins al moment són les següents.

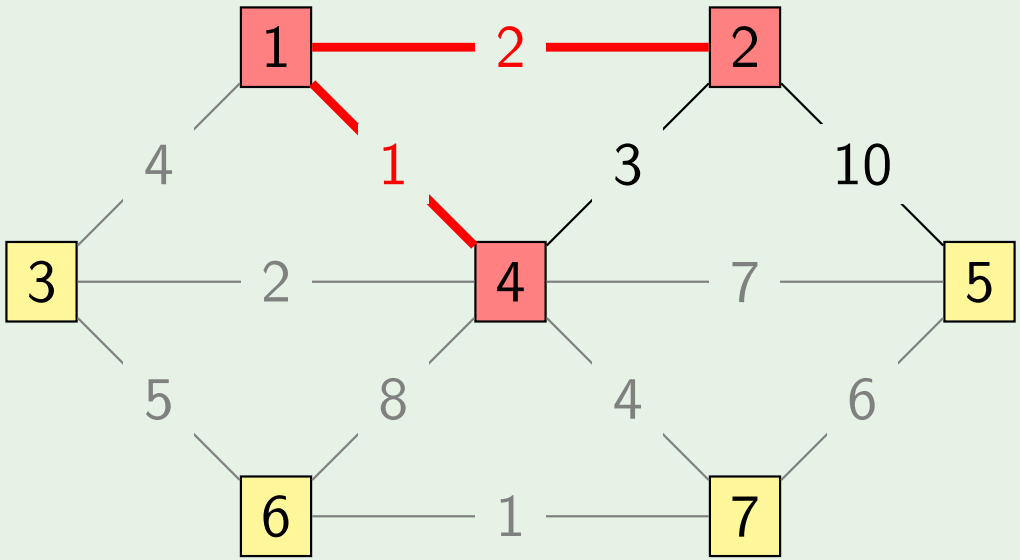
aresta	1–2	3–4	2–4	1–3	4–7	4–5	4–6
pes	2	2	3	4	4	7	8



Exemple

- Les arestes a la cua de prioritats fins al moment són les següents.

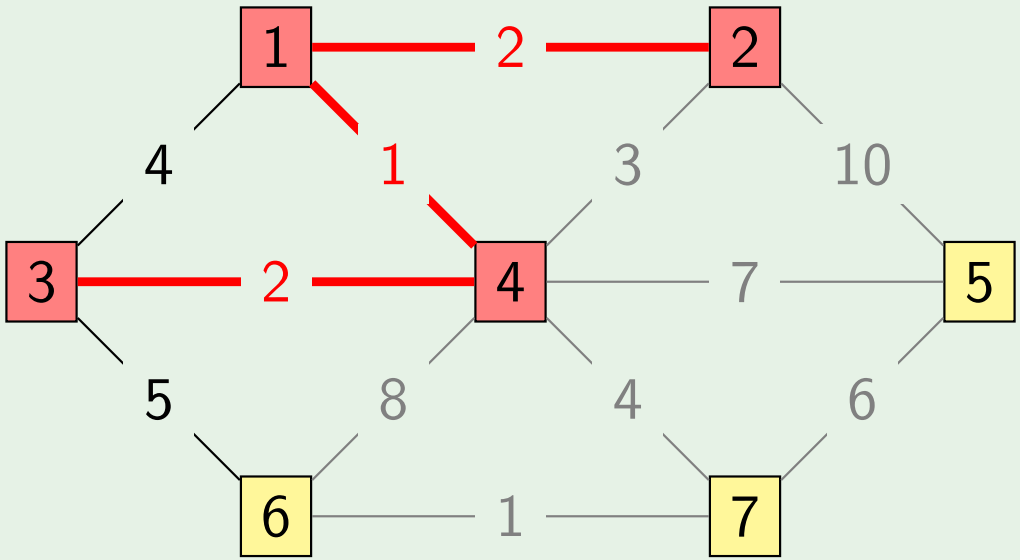
aresta	3–4	2–4	1–3	4–7	4–5	4–6	2–5
pes	2	3	4	4	7	8	10



Exemple

- Les arestes a la cua de prioritats fins al moment són les següents.

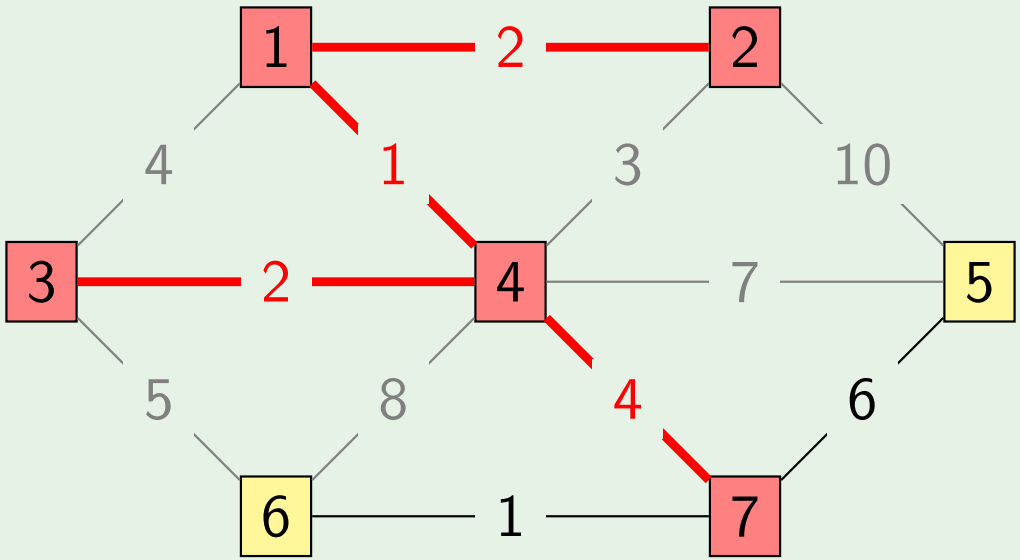
aresta	2–4	1–3	4–7	3–6	4–5	4–6	2–5
pes	3	4	4	5	7	8	10



Exemple

- Les arestes a la cua de prioritats fins al moment són les següents.

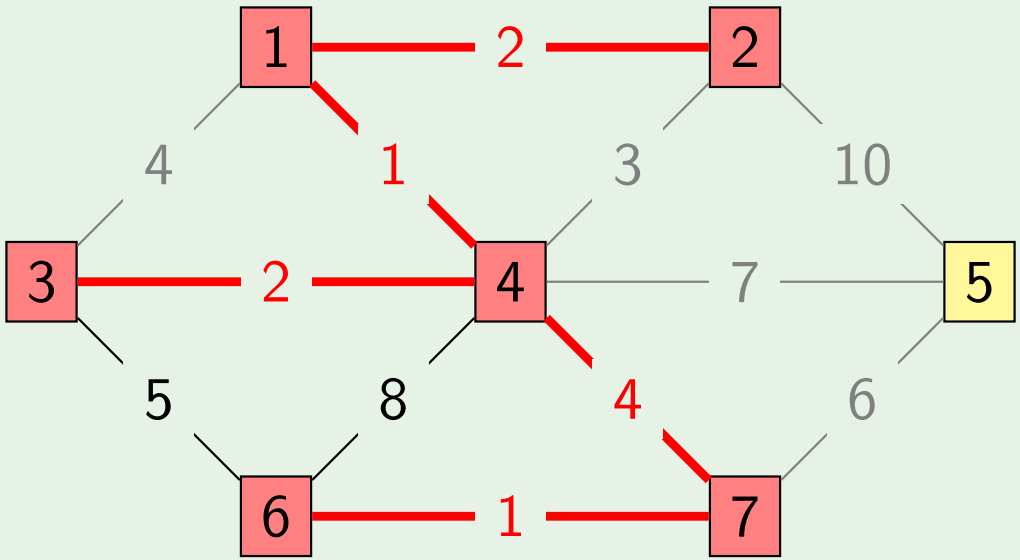
aresta	6–7	3–6	5–7	4–5	4–6	2–5
pes	1	5	6	7	8	10



Exemple

- Les arestes a la cua de prioritats fins al moment són les següents.

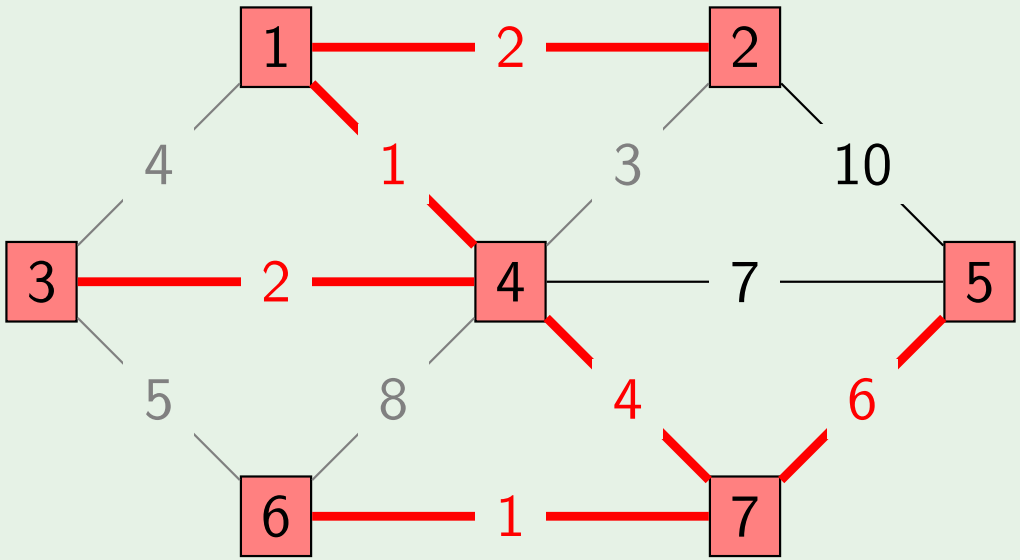
aresta	3–6	5–7	4–5	4–6	2–5
pes	5	6	7	8	10



Exemple

- Les arestes a la cua de prioritats fins al moment són les següents.

aresta	4–5	4–6	2–5
pes	7	8	10



Lema (Principi d'optimalitat local)

Sigui $G = (V, E)$ un graf no dirigit amb pesos, sigui T un subarbre de G, i sigui $\{v, w\} \in E$ una aresta de pes mínim que uneix un vèrtex v de T amb un vèrtex w de $G \setminus T$. Si existeix un arbre d'expansió mínim de G que conté T, aleshores existeix un arbre d'expansió mínim de G que conté T i $\{v, w\}$.

Demostració.

Sigui U un arbre d'expansió mínim de G que conté T, i suposem que no existeix cap arbre d'expansió mínim de G que contingui T i $\{v, w\}$. Com que U conté T, existeix una aresta $\{x, y\}$ de U que uneix un vèrtex x de T amb un vèrtex y de $U \setminus T$. Sigui ara V l'arbre d'expansió de G resultat de canviar $\{x, y\}$ per $\{v, w\}$ en U. Però, V no pot esser mínim, perquè conté T i $\{v, w\}$, i aleshores U tampoc no és mínim, perquè $\{v, w\}$ és una aresta de pes mínim que uneix T amb $G \setminus T$ i $pes[v, w] \leqslant pes[x, y]$ implica $pes[V] \leqslant pes[U]$, contradicció. □

Lema (Principi d'estructura òptima)

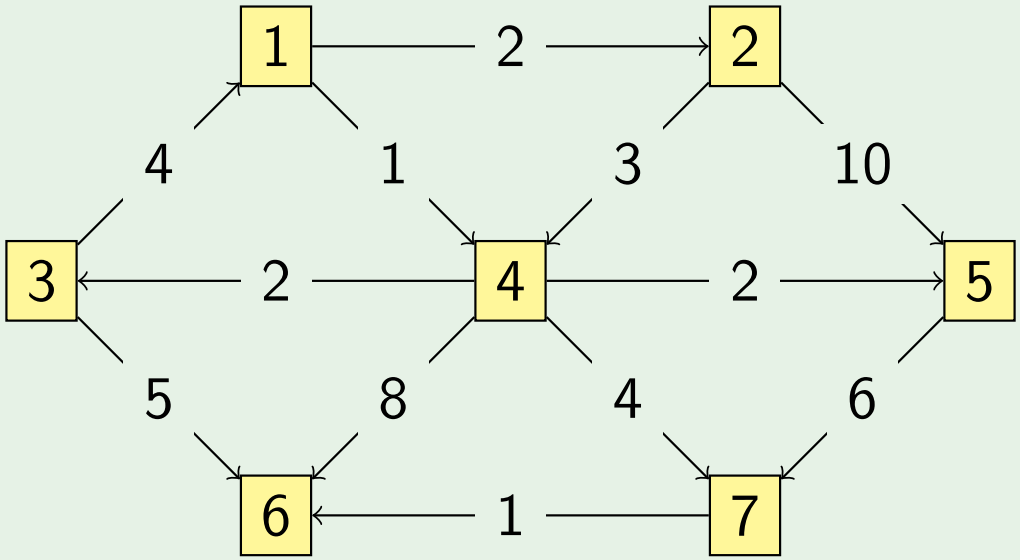
Sigui $G = (V, E)$ un graf no dirigit amb pesos, sigui T un arbre d'expansió mínim de G, i sigui $\{v, w\}$ una aresta de T. Siguin també $T_1 = (V_1, E_1)$ i $T_2 = (V_2, E_2)$ els subarbres de $T \setminus \{v, w\}$, i siguin G_1 i G_2 els subgrafs de G induïts per V_1 i V_2. Aleshores, T_1 i T_2 són arbres d'expansió mínims de G_1 i G_2.

Demostració.

Suposem que T_1 no és un arbre d'expansió mínim de G_1. Aleshores, existeix un arbre d'expansió mínim U_1 de G_1 tal que $pes[U_1] < pes[T_1]$. Però, això vol dir que T no és un arbre d'expansió mínim de G, perquè $pes[T] = pes[T_1] + pes[v, w] + pes[T_2] > pes[U_1] + pes[v, w] + pes[T_2]$, contradicció. Per tant, T_1 és un arbre d'expansió mínim de G_1. De la mateixa manera, T_2 és un arbre d'expansió mínim de G_2. □

- A partir d'un vèrtex inicial s en un graf G, trobar el camí més curt des de s fins a cada altre vèrtex de G.

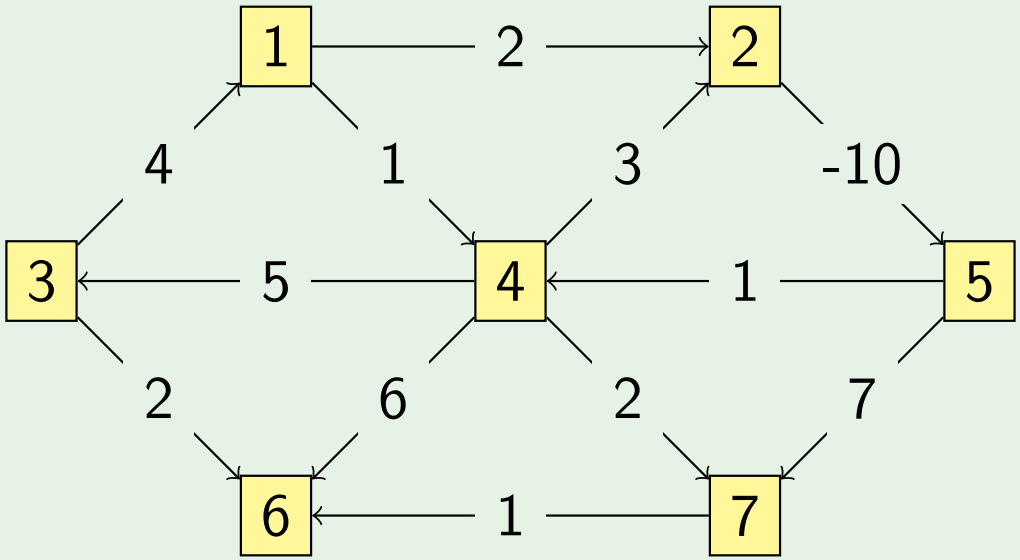
Exemple



- $1 \longrightarrow 4 \longrightarrow 7 \longrightarrow 6$ amb cost 6 (amb pesos)
- $1 \longrightarrow 4 \longrightarrow 6$ de longitud 2 (sense pesos)

- Els camins mínims poden no estar definits si el graf conté cicles amb pes negatiu.

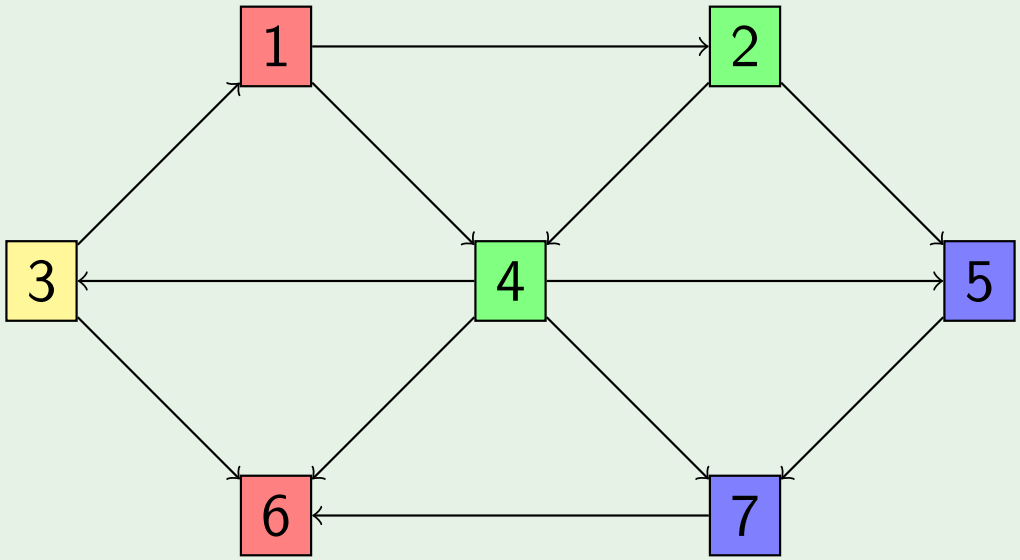
Exemple



- $5 \longrightarrow 4$ té cost 1
- $5 \longrightarrow 4 \longrightarrow 2 \longrightarrow 5 \longrightarrow 4$ té cost -5
- $5 \longrightarrow 4 \longrightarrow 2 \longrightarrow 5 \longrightarrow 4 \longrightarrow 2 \longrightarrow 5 \longrightarrow 4$ té cost -11
- $\ldots$

- En un graf sense pesos, els camins més curts des d'un vèrtex inicial fins a cada altre vèrtex del graf venen donats per un recorregut en amplada del graf.

Exemple



- $3 \longrightarrow 1$, $3 \longrightarrow 6$ (longitud 1)
- $3 \longrightarrow 1 \longrightarrow 2$, $3 \longrightarrow 1 \longrightarrow 4$ (longitud 2)
- $3 \longrightarrow 1 \longrightarrow 2 \longrightarrow 5$, $3 \longrightarrow 1 \longrightarrow 4 \longrightarrow 7$ (longitud 3)

- En un graf $G = (V, E)$ sense pesos, els camins més curts des d'un vèrtex inicial fins a cada altre vèrtex del graf venen donats per un recorregut en amplada del graf.
- **procedure** camins (G: graf; s: vèrtex)
  for all vèrtex v de G **do**
  dist$[v] := \infty$
  encuar(Q, s)
  dist$[s] := 0$
  repeat
  $v :=$ front(Q)
  desencuar(Q)
  for all vèrtex w adjacent amb el vèrtex v **do**
  if dist$[w] = \infty$ **then**
  dist$[w] :=$ dist$[v] + 1$
  encuar(Q, w)
  until buit(Q)

- En un graf $G = (V, E)$ amb pesos, els camins més curts des d'un vèrtex inicial fins a cada altre vèrtex del graf es poden obtenir mitjançant l'anomenat algorisme de Dijkstra.
- Aquest algorisme consisteix a estendre un conjunt S de vèrtexs (la distància als quals des del vèrtex inicial ha estat calculada) amb un vèrtex de $V \setminus S$ de distància mínima.

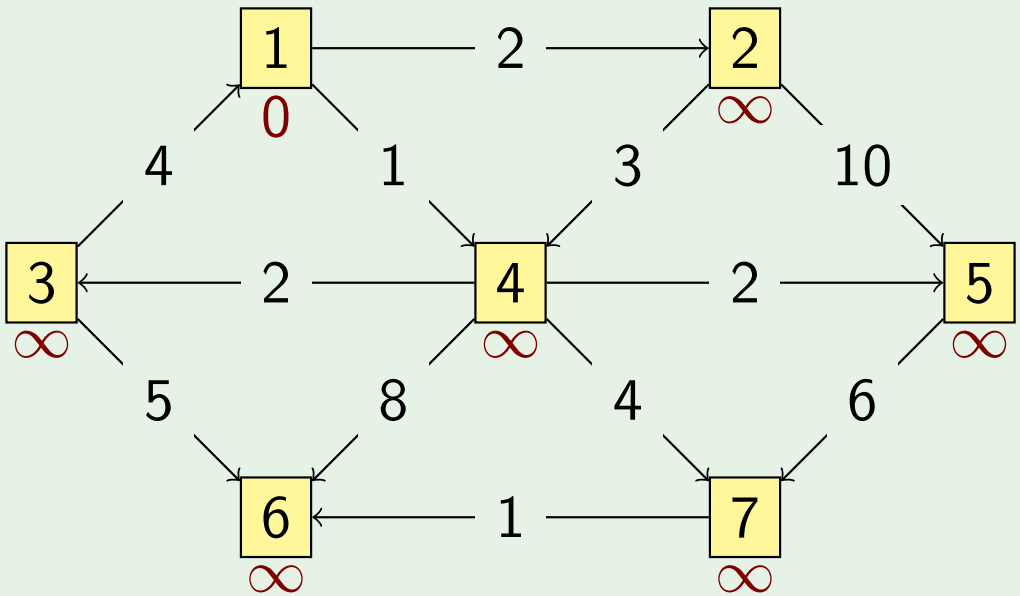
- **procedure** dijkstra (G: graf; s: vèrtex)
  for all vèrtex v de G **do**
    $vist[v] := false$
    $d[v] := \infty$
  $d[s] := 0$
  $inserir(Q, \{s, s\}, d[s])$
  while not $buit(Q)$ **do**
    $\{u, v\} := elim\_min(Q)$
    if not $vist[v]$ **then**
      for all vèrtex w adjacent amb el vèrtex v **do**
        if not $vist[w]$ **then**
          if $d[w] > d[v] + pes[v, w]$ **then**
            $d[w] := d[v] + pes[v, w]$
            $inserir(Q, \{v, w\}, d[w])$
      $vist[v] := true$

- L'algorisme de Dijkstra es pot implementar amb cost temporal $\Theta(m \log n)$.

Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

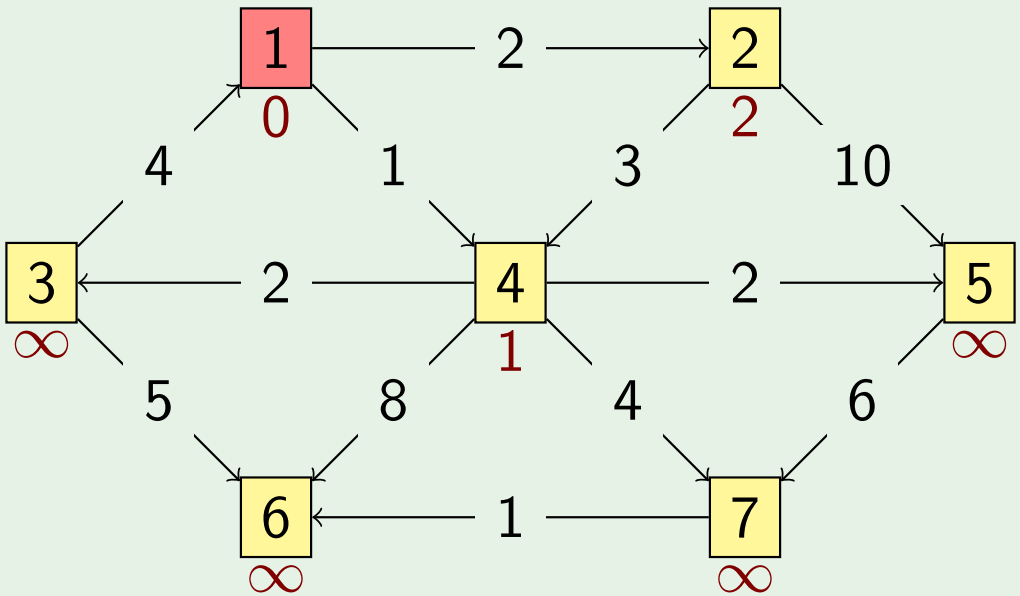
w	1	2	3	4	5	6	7
$d[w]$	0	∞	∞	∞	∞	∞	∞



Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

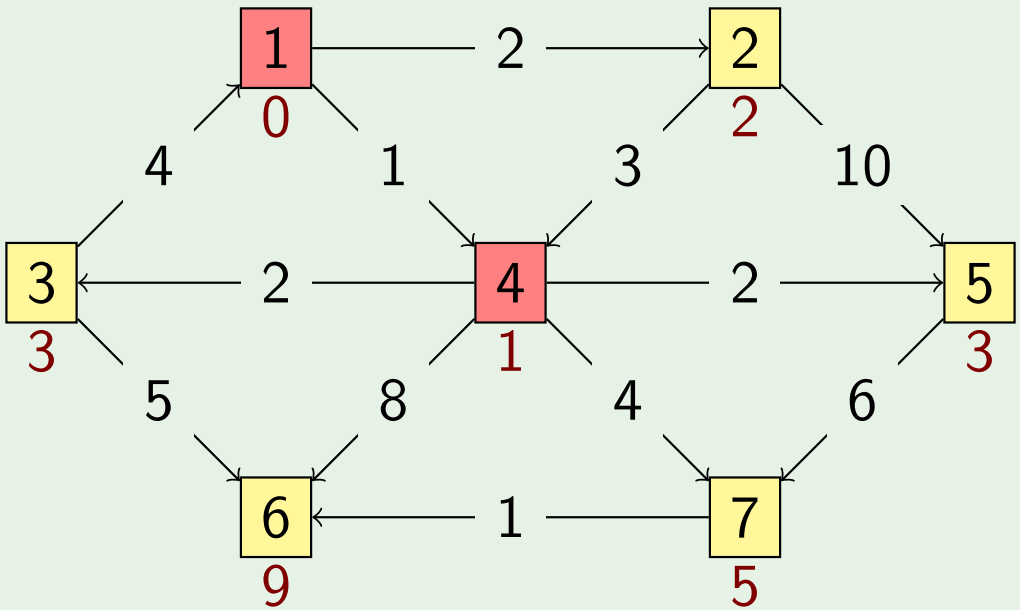
w	1	2	3	4	5	6	7
$d[w]$	0	2	∞	1	∞	∞	∞



Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

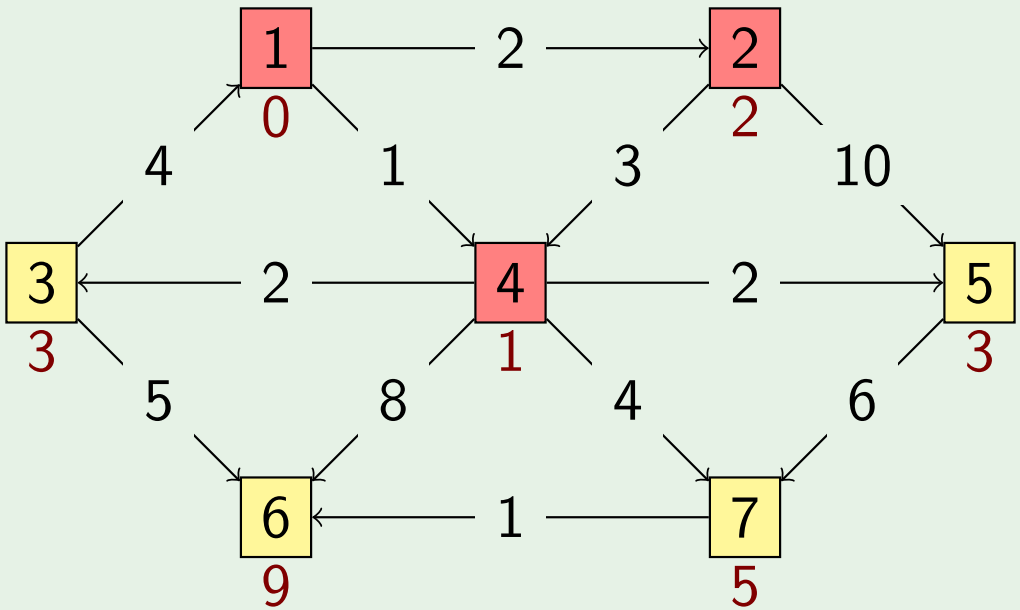
w	1	2	3	4	5	6	7
$d[w]$	0	2	3	1	3	9	5



Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

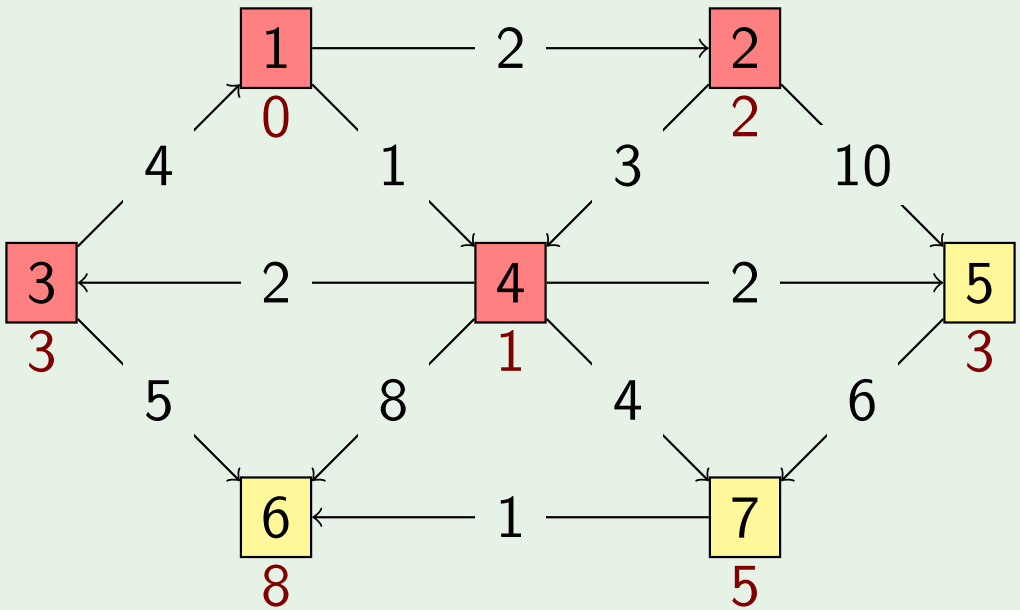
w	1	2	3	4	5	6	7
$d[w]$	0	2	3	1	3	9	5



Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

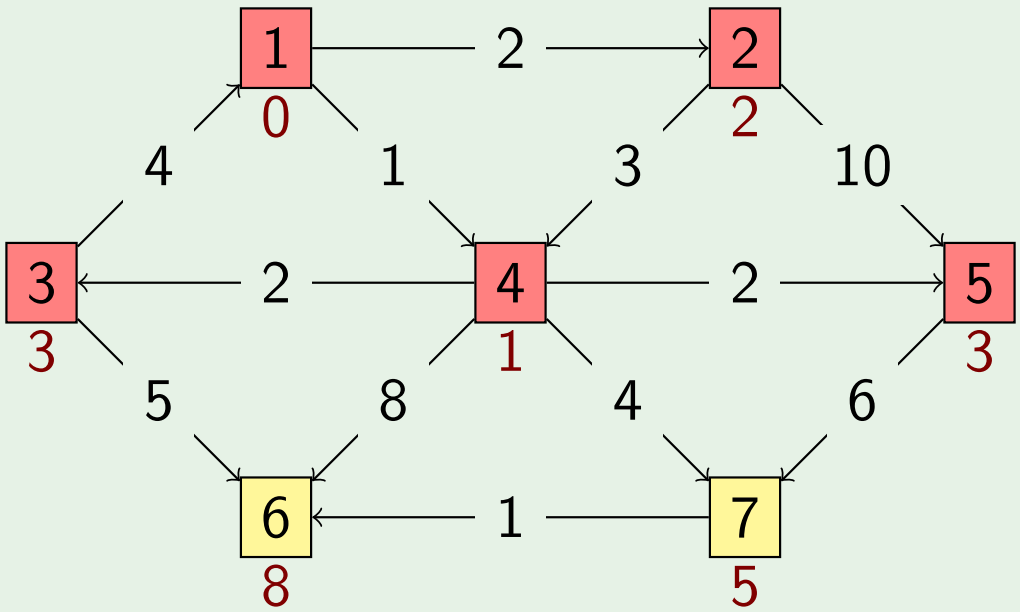
w	1	2	3	4	5	6	7
$d[w]$	0	2	3	1	3	8	5



Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

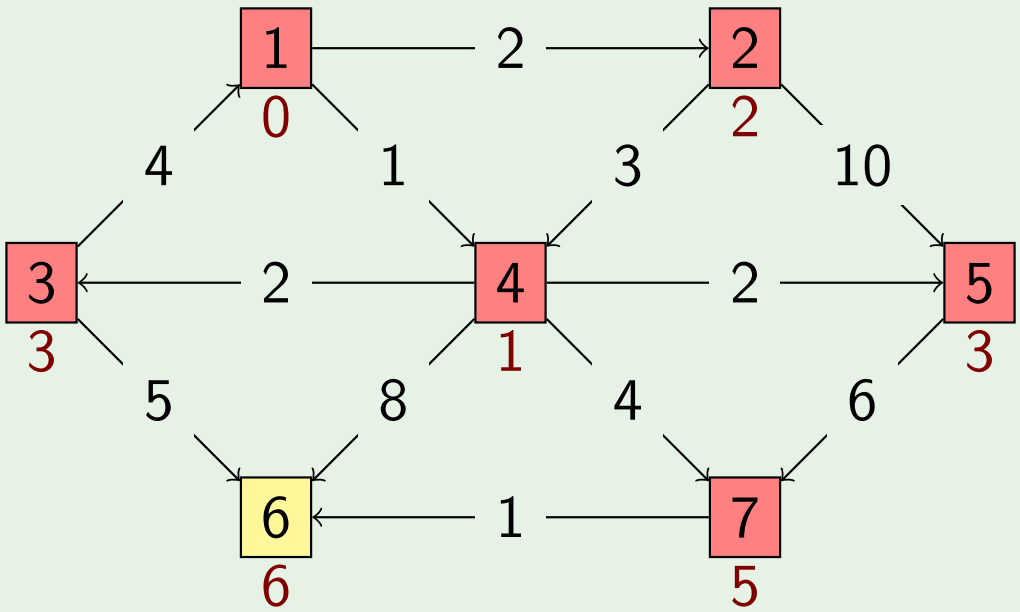
w	1	2	3	4	5	6	7
$d[w]$	0	2	3	1	3	8	5



Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

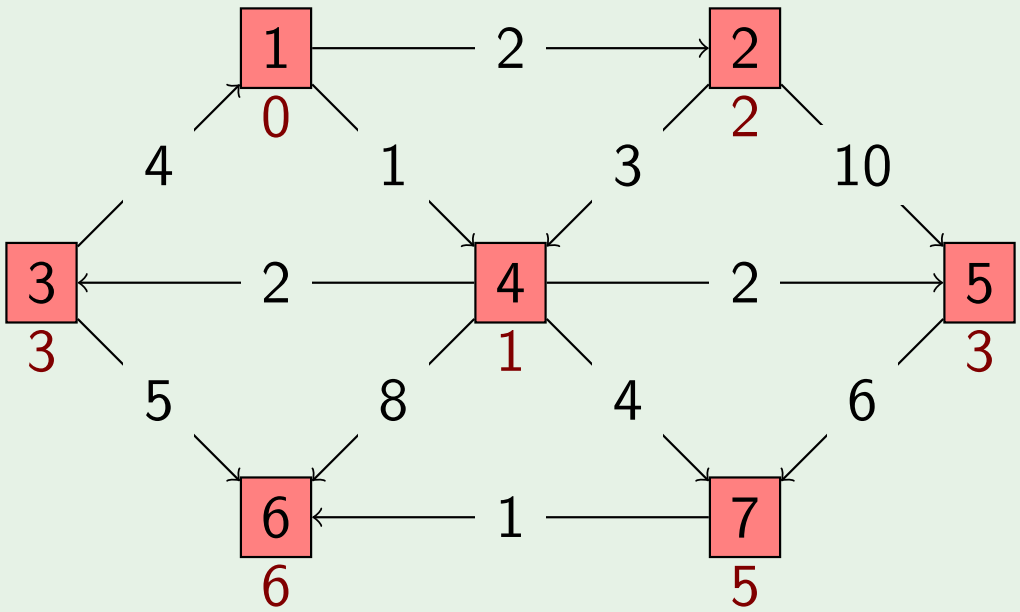
w	1	2	3	4	5	6	7
$d[w]$	0	2	3	1	3	6	5



Exemple

- Els camins més curts trobats fins al moment des del vèrtex $s = 1$ fins a cada altre vèrtex w del graf, tenen els pesos següents.

w	1	2	3	4	5	6	7
$d[w]$	0	2	3	1	3	6	5



Lema (Principi d'optimalitat local)

Sigui $G = (V, E)$ un graf dirigit amb pesos no negatius, sigui $s \in S \subseteq V$, i sigui $v \in S$ un vèrtex tal que $d[v] \leqslant d[w]$ per a tot vèrtex $w \in S$. Aleshores, $d[v] = dist(s, v)$.

Demostració.

Suposem que existeix un camí $s \rightsquigarrow x \longrightarrow y \rightsquigarrow v$ del vèrtex s al vèrtex v amb $pes[s \rightsquigarrow x \longrightarrow y \rightsquigarrow v] < d[v]$, on $x \in S$ i $y \in V \setminus S$. Aleshores, $pes[s \rightsquigarrow x \longrightarrow y \rightsquigarrow v] = pes[s \rightsquigarrow x \longrightarrow y] + pes[y \rightsquigarrow v] = d[x] + pes[x, y] + pes[y \rightsquigarrow v] \geqslant d[x]$, perquè $x \in S$ i $y \notin S$ i els pesos són no negatius. Com que $x \in S$, però, $d[x] \geqslant d[v]$ i per tant, $pes[s \rightsquigarrow x \longrightarrow y \rightsquigarrow v] \geqslant d[x] \geqslant d[v] > pes[s \rightsquigarrow x \longrightarrow y \rightsquigarrow v]$, contradicció. □

Lema (Principi d'estructura òptima)

Sigui $G = (V, E)$ un graf dirigit amb pesos no negatius, siguin $u, v, w \in V$, i sigui $u \rightsquigarrow v \rightsquigarrow w$ un camí de pes mínim del vèrtex u al vèrtex w. Aleshores, $dist(u, w) = dist(u, v) + dist(v, w)$.

Demostració.

Sabem que $pes[u \rightsquigarrow v] \geqslant dist(u, v)$, $pes[v \rightsquigarrow w] \geqslant dist(v, w)$, i $pes[u \rightsquigarrow v] + pes[v \rightsquigarrow w] = pes[u \rightsquigarrow v \rightsquigarrow w]$. Si $pes[u \rightsquigarrow v] > dist(u, v)$, sigui $u \dashrightarrow v$ un camí de pes mínim del vèrtex u al vèrtex v. Així, el camí $u \dashrightarrow v \rightsquigarrow w$ té pes $pes[u \dashrightarrow v] + pes[v \rightsquigarrow w] < pes[u \rightsquigarrow v] + pes[v \rightsquigarrow w] = pes[u \rightsquigarrow v \rightsquigarrow w]$, contradicció. Aleshores, $pes[u \rightsquigarrow v] = dist(u, v)$. De la mateixa manera, es demostra que $pes[v \rightsquigarrow w] = dist(v, w)$ i per tant, $dist(u, v) + dist(v, w) = pes[u \rightsquigarrow v] + pes[v \rightsquigarrow w] = pes[u \rightsquigarrow v \rightsquigarrow w] = dist(u, w)$. □

- La codificació ASCII usa 8 bits per representar $2^8 = 256$ caràcters. Per comprimir dades que contenen menys caràcters diferents, es pot usar una codificació alternativa amb un nombre menor de bits.

Exemple

- La codificació dels 11 caràcters ABRACADABRA en ASCII usa 88 bits, 8 bits per caràcter.
- Considerem una codificació alternativa de 5 bits, que permet representar $2^5 = 32$ caràcters. Per exemple, A = 00000, B = 00001, C = 00010, etc. Amb aquesta codificació, la representació dels 11 caràcters ABRACADABRA,

  00000000011000010000000010000000011000000000110001000000

  usa només 55 bits, és a dir, comporta una compressió del 37,5%.

- Tota codificació de longitud fixa és no ambigüa.

- Es pot aconseguir un factor de compressió més gran amb una codificació de longitud variable, assignant representacions més curtes als caràcters més freqüents.

Exemple (cont.)

- Considerem una codificació de longitud variable. Per exemple, A = 0, B = 1, C = 01, D = 10, R = 00. Amb aquesta codificació, la representació dels 11 caràcters ABRACADABRA,

  $$01000010100100 0$$

  usa només 15 bits, és a dir, comporta una compressió del 82,95%.
- Aquesta codificació, però, és ambigüa. El primer bit podria esser tant la representació d'una A com l'inici de la representació d'una R.

- Una codificació és lliure de prefixos si la representació de cap caràcter no és un prefix de la representació de cap altre caràcter.
- Tota codificació de longitud variable lliure de prefixos és no ambigüa.

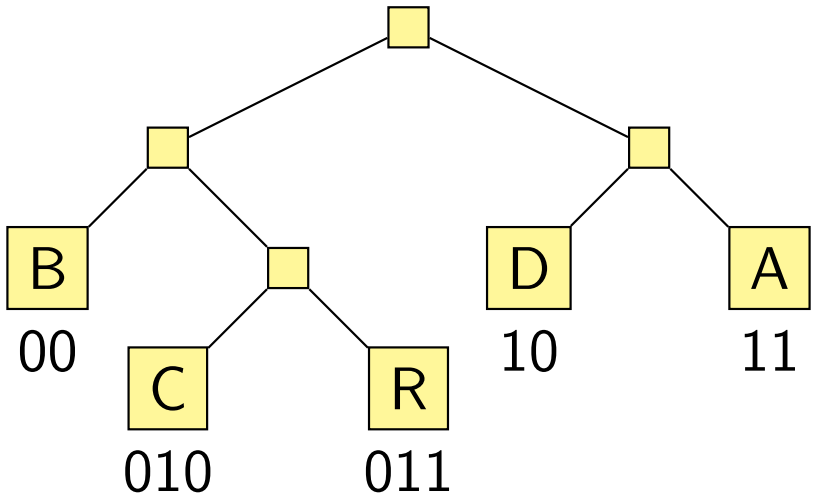
Exemple (cont.)

- Considerem una codificació alternativa, de longitud variable, lliure de prefixos. Per exemple, A = 11, B = 00, C = 010, D = 10, R = 011. Amb aquesta codificació, la representació dels 11 caràcters ABRACADABRA,

  $$11000111101011011000111$$

  usa només 25 bits, és a dir, comporta una compressió del 71,59%.

- Un arbre de codificació permet descodificar fàcilment el text original.

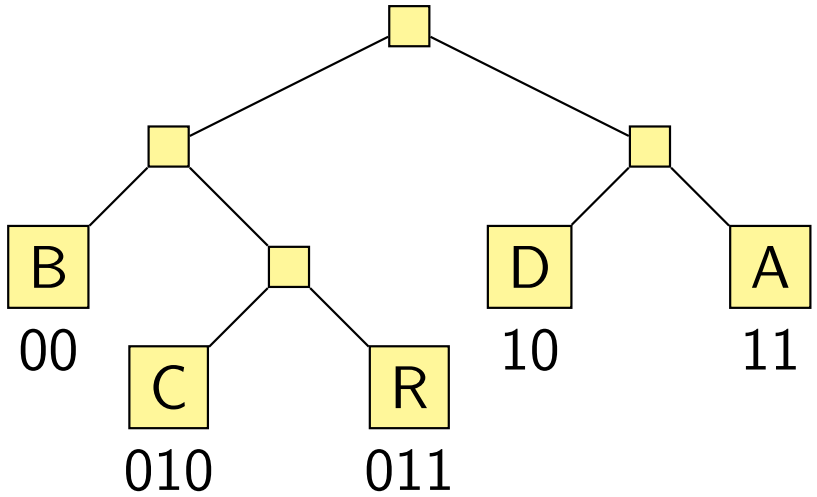


- Donat un text (seqüència de caràcters) sobre un alfabet X,
  - $f(x)$ denota la freqüència del caràcter $x \in X$
  - $d_T(x)$ denota la profunditat (nombre de bits de la representació) del caràcter $x \in X$ en l'arbre de codificació T
- El cost (nombre de bits) d'un arbre de codificació T per representar un text $S \in X^*$ amb freqüències $\{f(x) : x \in X\}$, ve donat per

$$B(T) = \sum_{x \in X} f(x) d_T(x)$$

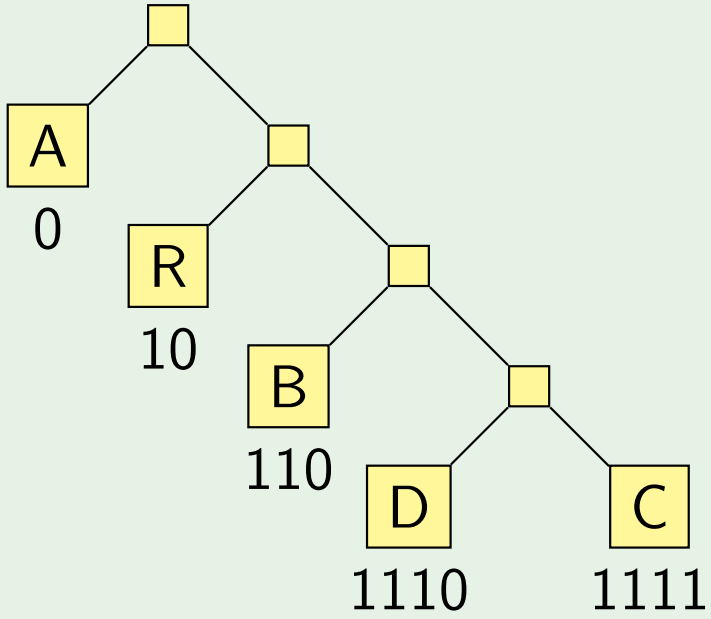
- Per exemple, $B(T) = 5 \cdot 2 + 2 \cdot 2 + 1 \cdot 3 + 1 \cdot 2 + 2 \cdot 3 = 25$

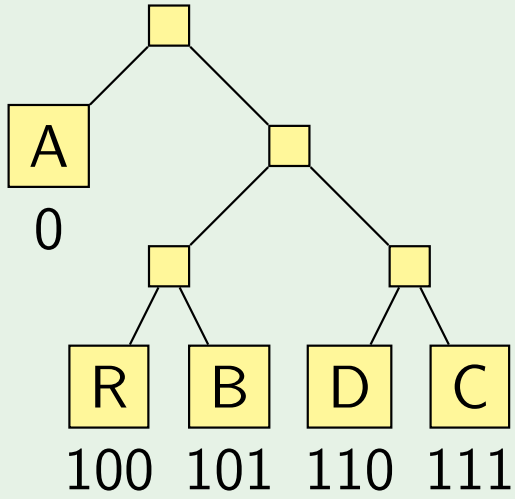
x	$f(x)$
A	5
B	2
C	1
D	1
R	2



Exemple (cont.)

x	$f(x)$
A	5
B	2
C	1
D	1
R	2





- Amb aquestes codificacions, la representació dels 11 caràcters ABRACADABRA usa només 23 bits, és a dir, comporta una compressió del 73,86%.

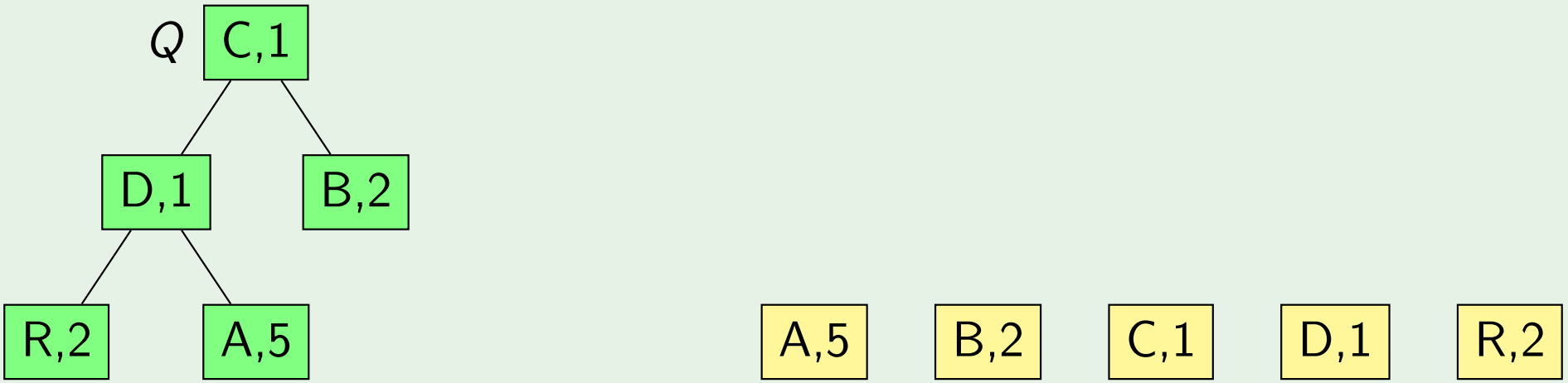
– $B(T_1) = 5 \cdot 1 + 2 \cdot 3 + 1 \cdot 4 + 1 \cdot 4 + 2 \cdot 2 = 23$

– $B(T_2) = 5 \cdot 1 + 2 \cdot 3 + 1 \cdot 3 + 1 \cdot 3 + 2 \cdot 3 = 23$

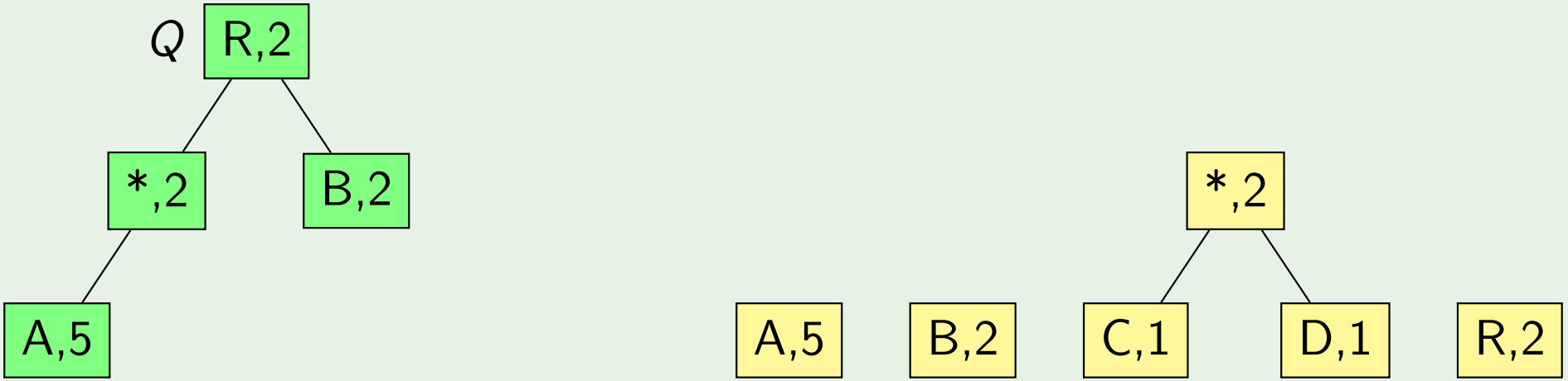
- Es pot aconseguir una compressió de dades òptima, és a dir, un arbre de codificació T amb el mínim cost $B(T)$ possible, amb l'anomenat algorisme de Huffman.
- **function** codificació (X: alfabet; f: freqüències)
  for all $x \in X$ **do**
    pointer to node z := **new**(node)
    $z \rightarrow$ info := $f(x)$
    $z \rightarrow$ left := **null**
    $z \rightarrow$ right := **null**
    inserir($Q, z, z \rightarrow$ info)
  for $i := 1$ to $|X| - 1$ **do**
    pointer to node z := **new**(node)
    $z \rightarrow$ left := elim_min(Q)
    $z \rightarrow$ right := elim_min(Q)
    $z \rightarrow$ info := $z \rightarrow$ left $\rightarrow$ info + $z \rightarrow$ right $\rightarrow$ info
    inserir($Q, z, z \rightarrow$ info)
  return min(Q)

- L'algorisme de Huffman es pot implementar amb cost temporal $\Theta(n \log n)$.

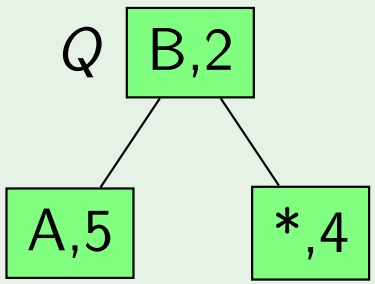
Exemple (Algorisme de Huffman)

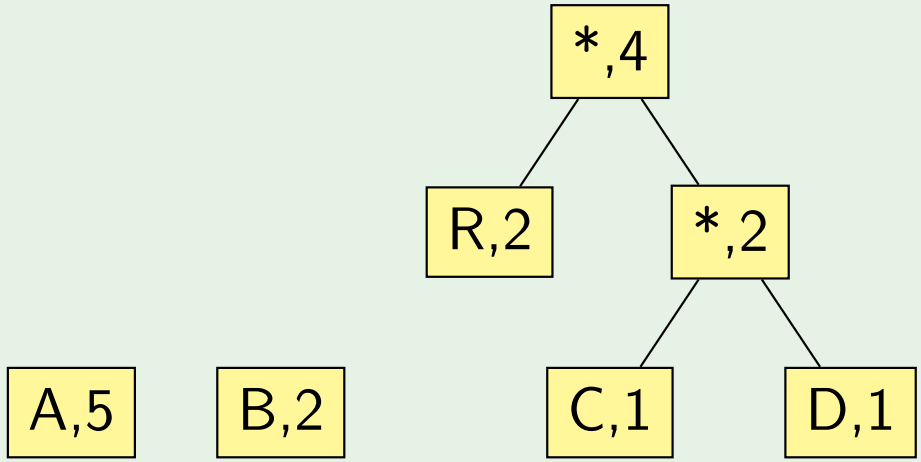


Exemple (Algorisme de Huffman)

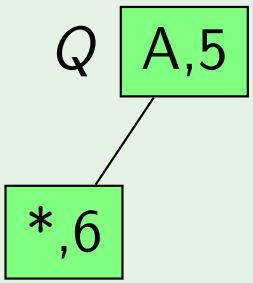


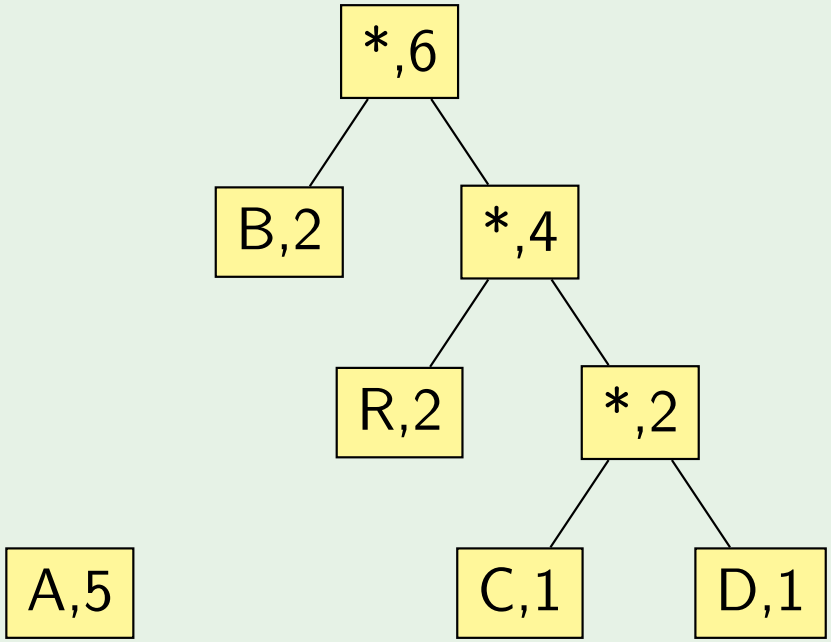
Exemple (Algorisme de Huffman)





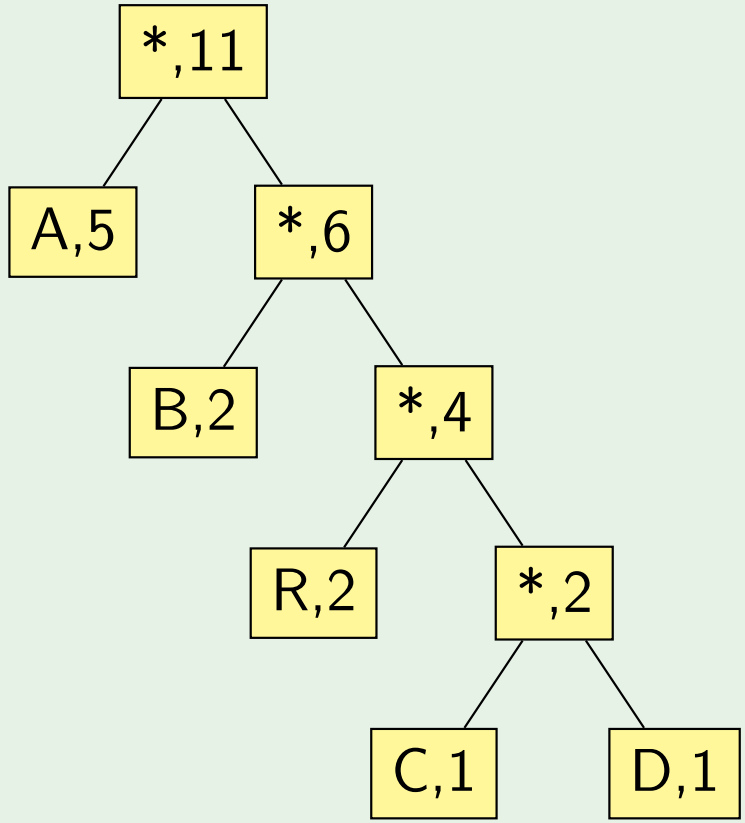
Exemple (Algorisme de Huffman)





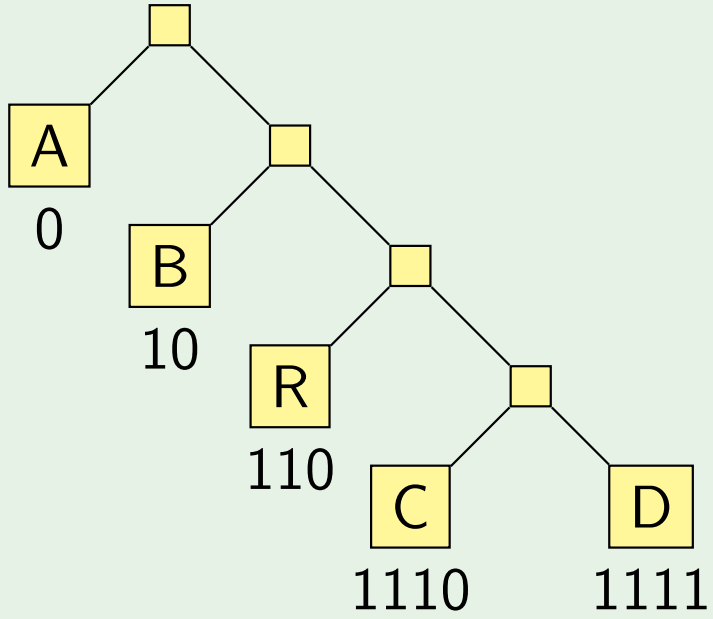
Exemple (Algorisme de Huffman)

Q *,11



Exemple (Algorisme de Huffman)

x	$f(x)$
A	5
B	2
C	1
D	1
R	2



- $B(T) = 5 \cdot 1 + 2 \cdot 2 + 1 \cdot 4 + 1 \cdot 4 + 2 \cdot 3 = 23$

Lema (Principi d'optimalitat local)

Sigui un text sobre un alfabet X, on cada caràcter $x \in X$ hi apareix amb freqüència $f(x)$, i siguin $x, y \in X$ dos caràcters amb freqüència mínima. Aleshores, existeix una codificació lliure de prefixos de cost mínim per a X en què les representacions de x i y són de la mateixa llargària i només difereixen en el darrer bit.

Demostració.

Sigui T un arbre que representa una codificació lliure de prefixos de cost mínim per a X, siguin $x, y \in X$ dos caràcters amb freqüència mínima, i siguin a, b dues fulles adjacents de T de profunditat màxima. Suposem que $f(a) \leqslant f(b)$ i $f(x) \leqslant f(y)$. Aleshores, $f(x) \leqslant f(a)$ i $f(y) \leqslant f(b)$. Sigui T' l'arbre T amb les fulles a i x intercanviades, i sigui T'' l'arbre T' amb les fulles b i y intercanviades. Com que $B(T) - B(T') \geqslant 0$ i $B(T') - B(T'') \geqslant 0$, $B(T'') \leqslant B(T)$ i aleshores $B(T'') = B(T)$, perquè T representa una codificació lliure de prefixos de cost mínim per a X.

Demostració.

De fet,

$$\begin{aligned} B(T) - B(T') &= f(x)d_T(x) + f(a)d_T(a) - f(x)d_{T'}(x) + f(a)d_{T'}(a) \\ &= f(x)d_T(x) + f(a)d_T(a) - f(x)d_T(a) + f(a)d_T(x) \\ &= (f(a) - f(x))(d_T(a) - d_T(x)) \\ &\geqslant 0 \end{aligned}$$

on $f(a) - f(x) \geqslant 0$ perquè $x \in X$ és un caràcter amb freqüència mínima i $d_T(a) - d_T(x) \geqslant 0$ perquè a és una fulla de T de profunditat màxima. La desigualtat $B(T') - B(T'') \geqslant 0$ es demostra de manera similar. □

Lema (Principi d'estructura òptima)

Sigui un text sobre un alfabet X, on cada caràcter $x \in X$ hi apareix amb freqüència $f(x)$, siguin $x, y \in X$ dos caràcters amb freqüència mínima, i sigui T un arbre que representa una codificació lliure de prefixos de cost mínim per a X. Aleshores, l'arbre $T' = T \setminus \{x, y\}$ representa una codificació lliure de prefixos de cost mínim per a l'alfabet $X' = X \setminus \{x, y\} \cup \{z\}$, on z és un nou caràcter amb freqüència $f(z) = f(x) + f(y)$.

Demostració.

$B(T') = B(T) - f(x)d_T(x) - f(y)d_T(y) + f(z)d_{T'}(z) = B(T) - (f(x) + f(y))(d_{T'}(z) + 1) + f(z)d_{T'}(z) = B(T) - f(x) - f(y)$. Suposem que existeix un arbre de codificació T'' per a X' amb $B(T'') < B(T')$. Siguin x, y dues fulles adjacents de T'' i sigui T''' l'arbre T'' amb el pare de x, y reemplaçat per una fulla z amb $f(z) = f(x) + f(y)$. Aleshores, $B(T''') = B(T'') - f(x) - f(y) < B(T') - f(x) - f(y) = B(T)$, contradicció. □

Els problemes marcats amb una estrelleta ($\star$) són més difícils. Si no es diu el contrari i us cal, considereu que tots els logaritmes són en base 2.

- 6.1
- 6.2
- 6.3
- 6.4
- 6.5
- 6.6
- 6.7
- 6.8
- 6.9
- 6.10
- 6.12
- 6.13
- 6.14
- 6.15
- 6.16
- 6.17
- 6.18
- 6.19
- 6.20
- 6.21

Problema (6.1)

Cal programar una màquina expenedora per tal que torni canvi usant el mínim nombre de monedes possibles. Dissenyeu un algorisme voraç que resolgui aquest problema quan el canvi s'ha de donar en monedes de cèntims d'euros de curs legal (monedes de 50, 20, 10, 5, 2 i 1 cèntims).

Mostreu que el vostre algorisme troba la solució òptima.

Trobeu una combinació de valors de les monedes per a la qual el vostre algorisme no trobi necessàriament el nombre mínim de monedes per fer el canvi.

Problema (6.2)

Un automobilista ha de fer un viatge seguint una ruta predeterminada. Amb el dipòsit ple pot fer x km. Sabem d'antuvi les ubicacions de totes les benzineres del trajecte, i que no n'hi ha cap tram de més de x km sense benzinera. Dissenyeu i analitzeu un algorisme voraç que permeti realitzar el trajecte fent un mínim d'aturades per posar benzina.

Problema (6.3)

Donat un conjunt de reals $X = \{x_1, x_2, \ldots, x_n\}$, *es vol trobar el conjunt més petit possible d'intervals unitaris que recobreixin tots els punts. És a dir, trobar un conjunt d'intervals*
$I = \{[i_1, i_1 + 1], [i_2, i_2 + 1], \ldots, [i_m, i_m + 1]\}$ *tal que*
$\forall j \mid 1 \leqslant j \leqslant n \mid \exists k \mid 1 \leqslant k \leqslant m \mid x_j \in [i_k, i_k + 1]$ *amb el mínim valor possible de* m. *Dissenyeu i analitzeu un algorisme voraç per resoldre aquest problema.*

Problema (6.4)

Es vol emmagatzemar en un dispositiu de capacitat L un conjunt de n arxius. La talla de l'arxiu i-èsim ve donada per $\ell[i]$, i se sap que $\sum_{i=1}^{n} \ell[i] > L$. Dissenyeu un algorisme que seleccioni un subconjunt dels arxius tot maximitzant el nombre d'arxius que es poden emmagatzemar al dispositiu.

Problema (6.5)

Dissenyeu un algorisme per calcular com emmagatzemar n arxius de talla $\ell[i]$ en una cinta magnètica, de forma que el temps mitjà de lectura d'un fitxer sigui mínim. Se suposa que tots els fitxers es llegeixen amb la mateixa freqüència, i que la cinta es rebobina després de llegir cada fitxer. Així doncs, el temps mitjà de lectura serà $\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{i}\ell[p[j]]$ on p és la permutació escollida.

Problema (6.6)

S'han de planificar n tasques en un processador que triga un temps $\ell[i]$ en completar la i-èsima tasca. A més, cada tasca ha d'acabar, com a molt tard, a l'instant $t[i]$. Una seqüència factible és una ordenació de les tasques tal que la seva execució en aquell ordre fa que cada tasca i acabi a l'instant $t[i]$. Dissenyeu un algorisme voraç per calcular una seqüència factible o indicar que no n'hi ha cap.

Problema (6.7)

S'han de planificar n tasques en un processador que triga una unitat de temps en completar cadascuna. Per a cada tasca i hi ha un instant d'acabament $t[i]$ tal que si completem la tasca abans d'aquell instant rebem un benefici $b[i] \geqslant 0$. Dissenyeu un algorisme voraç que calculi una permutació p que representi l'ordre en el qual executar les tasques tot maximitzant el benefici net $\sum_{p[i] \leqslant t[i]} b[i]$.

Problema (6.8)

Donat un graf no dirigit $G = (V, E)$, es diu que un subconjunt dels vèrtexs $U \subseteq V$ és un recobriment de G si cada aresta en E incideix en almenys un vèrtex de U. Un recobriment U és mínim si no hi ha cap altre recobriment amb menys vèrtexs.

El Professor Gran proposa l'algorisme voraç següent: Començem amb $E' := E$, $U := \emptyset$ i $S := V$. Mentre $E' \neq \emptyset$, agafem un vèrtex u de S; si el nombre de veïns de u en E' és zero, esborrem u de S; sinó incloem u a U, l'esborrem de S i esborrem de E' totes les seves arestes adjacents.

Mostreu que aquest algorisme proporciona un recobriment però que no és necessàriament òptim.

El Professor Alzina proposa modificar l'algorisme anterior triant sempre el vèrtex de S que tingui més veïns en E'.

Mostreu que aquest algorisme tampoc no és correcte.

Problema (6.9)

Repetiu el problema anterior restringit a arbres.

Proposeu un algorisme que resolgui el problema del recobriment mínim restringit a arbres en temps lineal.

Problema (6.10)

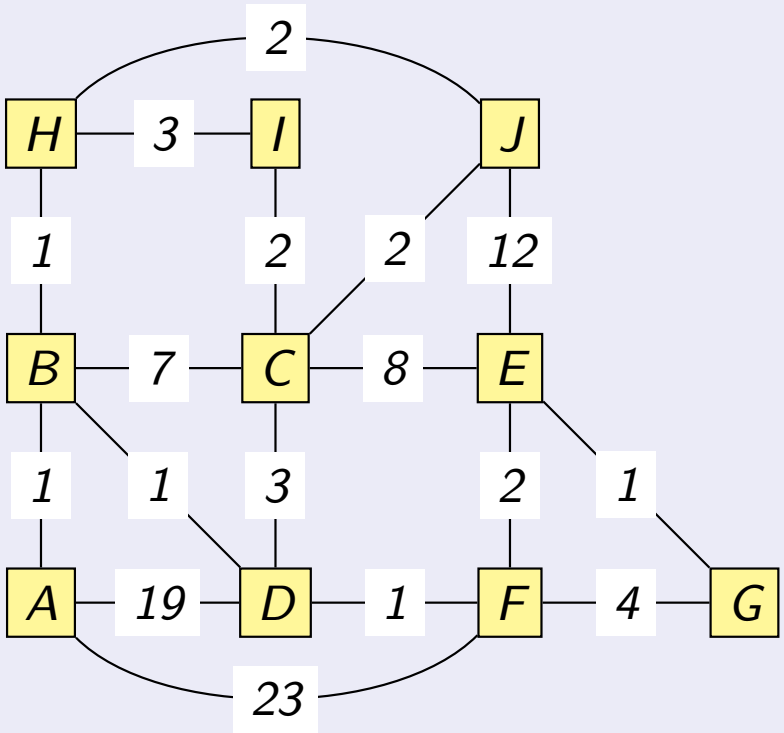
Tenim informació emmagatzemada en n cintes, on la i-èsima cinta conté m[i] registres ordenats en forma creixent. Es vol obtenir una única cinta amb tota la informació ordenada, efectuant $n-1$ operacions de fusió entre dues cintes (una fusió llegeix dues cintes i en grava una de nova amb la reunió ordenada de la informació continguda a les dues cintes d'entrada). L'ordre en el qual es facin les fusions afecta l'eficiència del procés. Dissenyeu un algorisme voraç que minimitzi el nombre de còpies de registres.

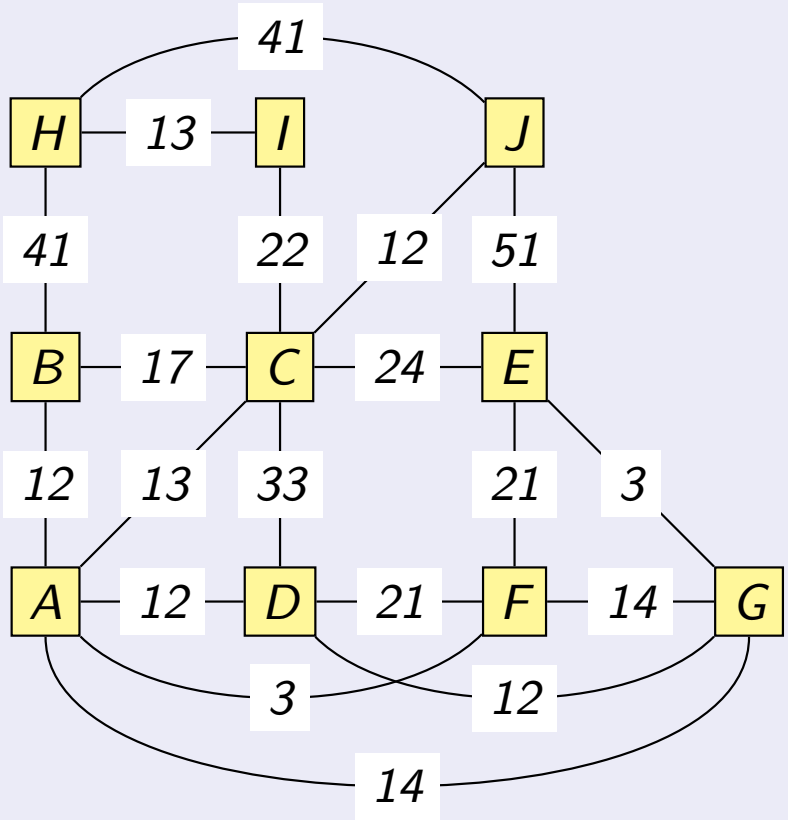
Exemple: Si $n = 3$, i $m_A = 30$, $m_B = 20$ i $m_C = 10$, tenim:

opció	*fusió*	*còpies*	*fusió*	*còpies*	*total*
$((A,B),C)$	$X = (A,B)$	*50*	$Y = (X,C)$	*60*	*110*
$((C,B),A)$	$X = (C,B)$	*30*	$Y = (X,A)$	*60*	*90*
$((A,C),B)$	$X = (A,C)$	*40*	$Y = (X,B)$	*60*	*100*

Problema (6.12)

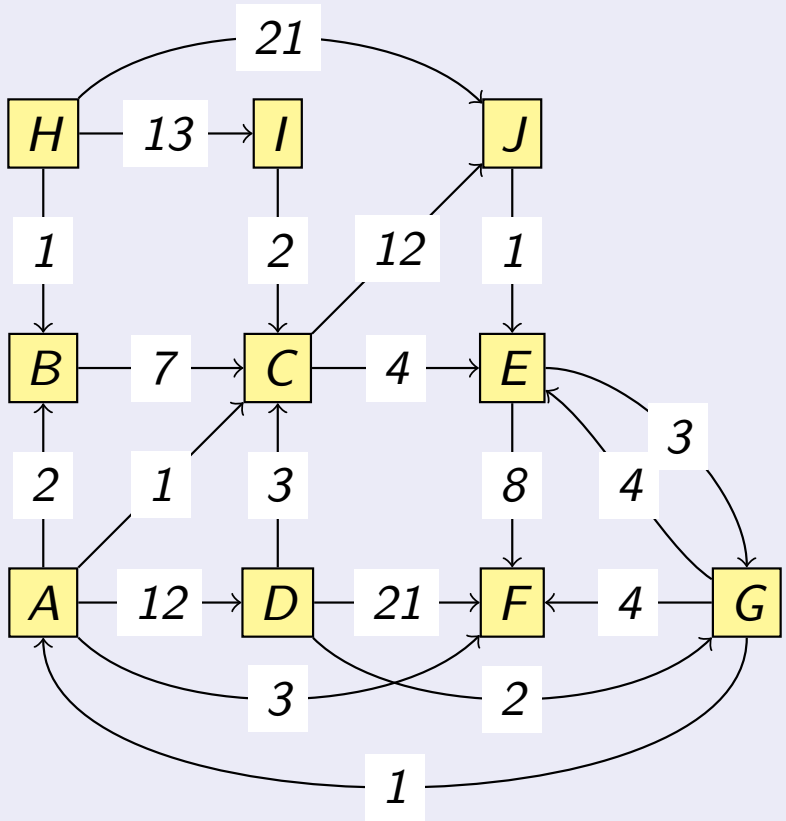
Trobeu el camí mínim per anar del vèrtex A al vèrtex G en els dos grafs següents:





Problema (6.13)

Mostreu com l'algorisme de Dijsktra calcula els camins mínims per anar del vèrtex C a tots els altres vèrtexs en el graf dirigit següent:



Problema (6.14)

Quin sentit té calcular camins mínims quan hi ha cicles de cost negatiu en un graf?

Mostreu un graf per al qual l'algorisme de Dijsktra no funciona quan hi ha pesos negatius, malgrat no haver-hi cicles de cost negatiu.

Problema (6.15)

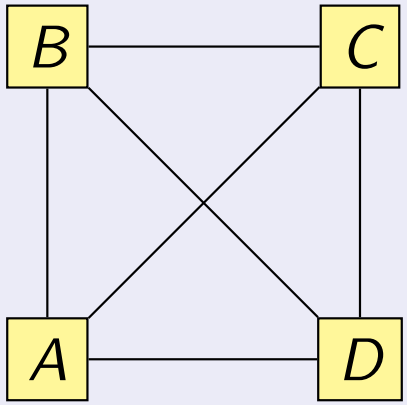
Modifiqueu l'algorisme de Dijsktra per tal que compti el nombre de camins mínims entre dos vèrtexs donats. Què passaria si el graf tingués cicles de cost zero? I si tingués arcs de cost zero però no cicles de cost zero?

Problema (6.16)

Modifiqueu l'algorisme de Dijsktra per tal que si hi ha més d'un camí mínim entre dos vèrtexs donats, en retorni un de qualsevol amb el nombre mínim d'arcs.

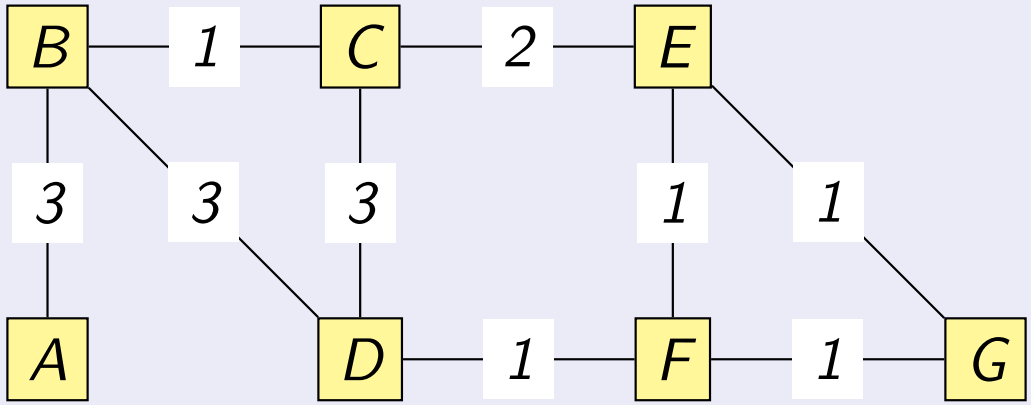
Problema (6.17)

Doneu tots els possibles arbres d'expansió del graf següent:



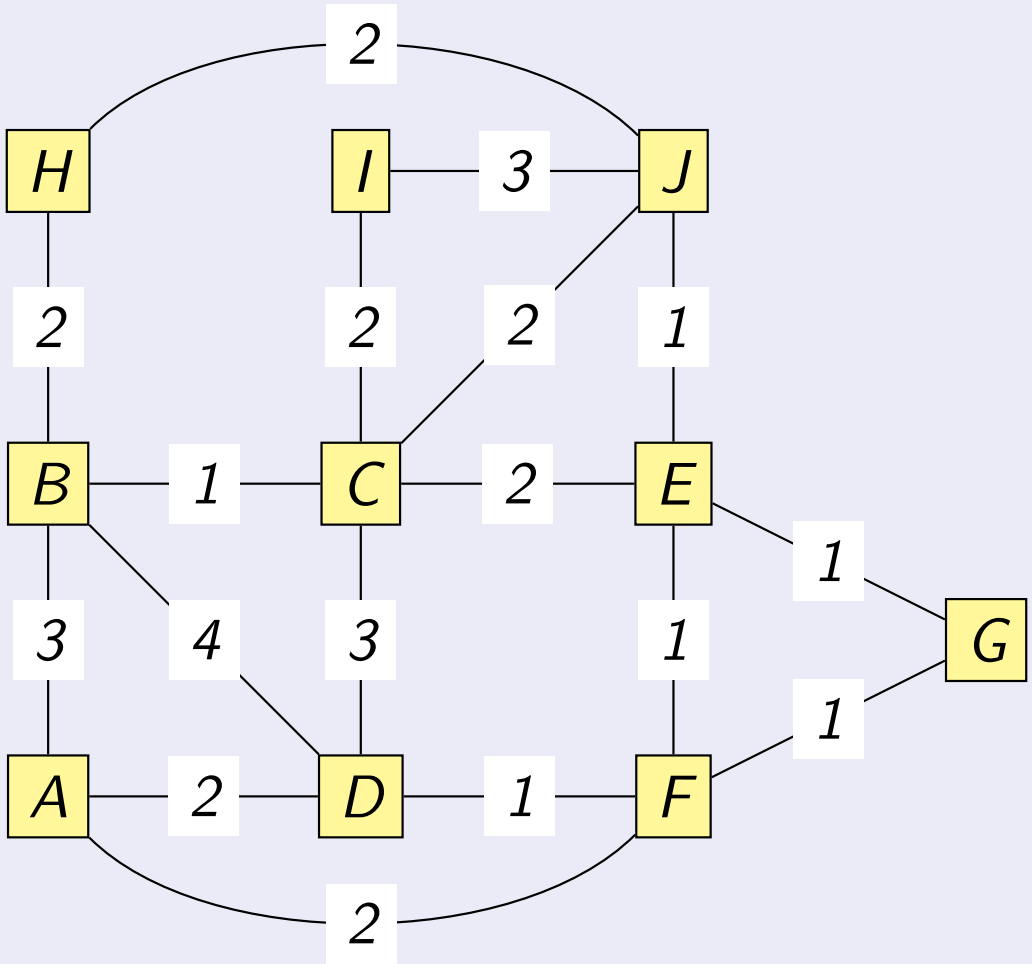
Problema (6.18)

Doneu tots els possibles arbres d'expansió mínims del graf següent:



Problema (6.19)

Mostreu quin arbre retorna l'algorisme de Prim sobre el graf següent. En cas d'haver-hi més d'una possibilitat, trenqueu-la segons l'ordre alfabètic.



Problema (6.20)

Digueu si l'algorisme de Prim funciona amb grafs amb pesos negatius.

Problema (6.21)

Considereu un fitxer que només conté els caràcters `A`, `B`, `C`, `D` i `E`. El caràcter `A` apareix 35000 cops, el caràcter `B` apareix 1000 cops, els caràcters `C` i `D` apareixen 2000 cops, i el caràcter `E` apareix 1500 cops. Dibuixeu un arbre de Huffman que minimitzi la llargada d'aquest fitxer. Quina serà la llargada del fitxer un cop condificat?