

Problemes 2

- 2.1. (**Planifica unitats**) Tenim un conjunt de n tasques de durada 1. Cada tasca i té associada una data prevista de finalització d_i i una penalització p_i que s'hauria de pagar si la tasca s'acaba després de la seva data prevista de finalització.

Volem trobar una planificació de les n tasques a un processador que només pot processar una tasca per a cada instant de temps. Volem que la planificació minimitzi la penalització total pagada. Considereu el següent algorisme:

```
function PLANIFICA( $d_1, \dots, d_n, p_1, \dots, p_n$ )
  Ordena les tasques per  $d_i$  creixentment
   $t \leftarrow 1$ ;  $S \leftarrow \emptyset$ 
  for  $i = 1, \dots, n$  do
    if  $d_i \geq t$  then
       $S \leftarrow S \cup \{i\}$ 
       $t \leftarrow t + 1$ 
    else
      if existeix  $k \in S$  amb  $p_k < p_i$  then
         $S \leftarrow S - \{k\}$ , on  $k \in S$  té l'índex més gran d'entre les de  $p_k$  mínima a  $S$ 
         $S \leftarrow S \cup \{i\}$ 
  return (Una planificació en què es processa de forma continuada, i on primer
    s'executen les tasques de  $S$  en ordre creixent de  $d_i$ 's i, seguidament,
    les tasques que no són a  $S$  en qualsevol ordre.)
```

- Demostreu que PLANIFICA resol correctament el problema plantejat.
- Proporcioneu una implementació eficient de PLANIFICA i analitzeu-ne el cost.
- Es podria modificar PLANIFICA per tal de continuar tenint un algorisme correcte però en què es millori el cost de la primera ordenació? Raoneu la vostra resposta.

- 2.2. **(Revisió)** Un professor rep n sol·licituds de revisions d'examen. Abans de començar, el professor mira la llista dels n estudiants que han sol·licitat revisió i pot calcular, per a cada estudiant i , el temps t_i que utilitzarà per atendre l' i -èsim estudiant. Per a estudiant i , el temps d'espera e_i és el temps que el professor triga a revisar els exàmens dels estudiants que fan la revisió abans que i .

Dissenyeu un algorisme per a computar l'ordre en que s'han de revisar els exàmens dels n estudiants de manera que es minimitzi el temps total d'espera: $T = \sum_{i=1}^n e_i$.

- 2.3. **(Subgraf induït)** Tenim un graf no dirigit $G = (V, E)$. Donat un subconjunt $V' \subseteq V$ el *subgraf induït* per V' és el graf $G[V'] = (V', E')$ on $E' = E \cap (V' \times V')$, és a dir, conté totes les arestes que tenen els dos extrems a V' . El *grau* d'un vèrtex a un graf és el nombre d'arestes incidents al vèrtex. Doneu un algorisme eficient per al següent problema: donat G i un enter positiu k , trobar el subconjunt (si hi ha algun) més gran V' de V , tal que cada vèrtex a V' té grau $\geq k$ a $G[V']$.

- 2.4. **(Partició interval)** El problema de la *partició interval* (*Interval Partitioning Problem*) és similar al problema de la selecció d'activitats vist a classe però, en lloc de tenir un únic recurs, tenim molts recursos (és a dir, diverses còpies del mateix recurs). Doneu un algorisme que permeti programar totes les activitats fent servir el menor número possible de recursos.

- 2.5. **(Ratolins i caus)** A un experiment sociològic es disposen n ratolins a les posicions $R = \{r_1, \dots, r_n\}$ determinades sobre una línia recta. A la mateixa línia hi ha n caus localitzats a les posicions $C = \{c_1, \dots, c_n\}$. A cada cau només es podrà allotjar un ratolí.

Un ratolí pot romandre a la seva posició, moure's un pas cap a la dreta de la línia (és a dir, de la posició x a la posició $x + 1$), o moure's un pas cap a l'esquerra de la línia (és a dir, de la posició x a la posició $x - 1$). Qualsevol d'aquests moviments consumeix un minut de temps.

Dissenyeu un algorisme, el més eficient possible, per a assignar els ratolins als caus de manera que es minimitzi el temps en què l'últim ratolí entra dins d'un cau. Raoneu el cost de l'algorisme proposat.

2.6. **(Graf centre)** Sigui $T = (V, E)$ un arbre no dirigit amb n vèrtexs. Per a $u, v \in V$, sigui $d(u, v)$ el nombre d'arestes del camí de u a v en T . Definim el *centre* de T com el vèrtex

$$c = \operatorname{argmin}_{v \in V} \{ \max_{u \in V} d(u, v) \}.$$

Describiu un algorisme de cost $O(n)$ per a obtenir un centre de T .

- 2.7. (**Huffmann màxim**) Si utilitzem l'algorisme de Huffmann per a comprimir un text format per n , símbols que apareixen amb freqüències $f_1, f_2, f_3, \dots, f_n$ quina és la màxima longitud de compressió d'un símbol que podem obtenir? Doneu un exemple de les freqüències on es compleix aquesta condició.

- 2.8. **(Googol)** La companyia Googol vol detectar a la web en quin idioma està escrita cada pàgina a la web. Per això, ha dissenyat un programa que quan descarrega una pagina, pot demanar si la pàgina està escrita en idioma L_i ($1 \leq i \leq 8$) i el programa respon SI o NO. Googol sap que de totes les pàgines a internet, el 40% estan escrites en L_1 , el 17% en L_2 , el 15% en L_3 , el 11% en L_4 , el 9% L_5 , el 5% L_6 , el 2% L_7 , i el 1% L_8 .
- (a) Googol, ha dissenyat un programa al que, quan descarrega una pagina, pot demanar si la pàgina està escrita en idioma L_i ($1 \leq i \leq 8$) i el programa respon SI o NO. En quin ordre s'han a de fer les preguntes per que el nombre mitjà de preguntes necessaries per a identificar l'idioma sigui mínim?
 - (b) Un dels informàtics a Googol pensa que potser seria millor fer servir un programa que, donats dos conjunts disjunts d'idiomes, diu si la pàgina està escrita amb un idioma del primer conjunt o amb un idioma del segon. Dissenya un algorisme, que fent servir aquest tipus de qüestions, trobi l'idioma en el que està escrit la pàgina. Quin serà el nombre esperat de qüestions utilitzant el teu algorisme? (Ajut: Penseu en el codis de Huffman)

2.9. **(Bottleneck ST)** Un *bottleneck spanning tree* T d'un graf no dirigit i ponderat $G = (V, E, w)$, on $w : E \rightarrow \mathbb{R}^+$, és un arbre d'expansió de G on el pes més gran és mínim sobre tots els arbres d'expansió de G . Diem que el valor d'un bottleneck spanning tree és el pes de la aresta de pes màxim a T .

(a) Demostreu la correctesa o trobeu un contraexemple pels enunciats següents:

- *Un bottleneck spanning tree és també un arbre d'expansió mínim.*
- *Un arbre d'expansió mínim és també un bottleneck spanning tree.*

(b) Doneu un algorisme amb cost $O(|V| + |E|)$ que donat un graf G i un enter b , determini si el valor d'un bottleneck spanning tree és $\leq b$.

- 2.10. **(MST?)** Aquí teniu el pseudocodi de quatre algorismes diferents. Tots tenen com a entrada un graf connex ponderat $G = (V, E, w)$, amb pesos no negatius a les arestes, i retornen un subconjunt d'arestes F .

Per cadascun dels algorismes, heu de determinar si (V, F) és un arbre d'expansió amb cost mínim o no, i justificar la vostra resposta amb la demostració adient. A més, heu de donar la implementació més eficient de cadascun dels algorismes, independentment de si retornen o no un MST.

(a) MAYBE-MST-A(G, w)

```
sort the edges into nonincreasing order of edge weights w
F = E
for each edge e, taken in nonincreasing order by weight
    if F - {e} is a connected graph
        F = F - {e}
return F
```

(b) MAYBE-MST-B(G, w)

```
F = ∅
for each edge e, taken in arbitrary order
    if F + {e} has no cycles
        F = F + {e}
return F
```

(c) MAYBE-MST-C(G, w)

```
F = ∅
for each edge e, taken in arbitrary order
    F = F + {e}
    if F has a cycle c
        let e' be a maximum-weight edge on c
        F = F - {e'}
return F
```

(d) MAYBE-MST-D(G, w)

```
Compute A, B a partition of V
FA = MAYBE-MST-D( $G[A], w$ )
FB = MAYBE-MST-D( $G[B], w$ )
let e be a minimum-weight edge connecting a vertex in A with a vertex in B
F = F + {e}
return F
```

2.11. **(Mark and collect)** Tenim un graf connex no dirigit $G = (V, E)$ on cada node $v \in V$ té associat un valor $\ell(v) \geq 0$; considerem el següent joc unipersonal:

- (a) Els nodes inicialment no estan marcats i la puntuació del jugador és 0.
- (b) El jugador selecciona un node $u \in V$ no marcat. Sigui $M(u)$ el conjunt de veïns de u a G que ja han sigut marcats. Aleshores, s'afegeix a la puntuació del jugador el valor $\sum_{v \in M(u)} \ell(v)$ i marquem u .
- (c) El joc es repeteix fins que tots els nodes siguin marcats o el jugador decideixi finalitzar la partida, deixant possiblement alguns nodes sense marcar.

Per exemple, suposeu que el graf té tres nodes A, B, C on A està connectat a B i B amb C , amb $\ell(A) = 3$, $\ell(B) = 2$, $\ell(C) = 3$. En aquest cas, una estratègia òptima seria marcar primer A , després C i finalment B . Aquest ordre dona al jugador una puntuació total de 6.

- (a) És possible obtenir una puntuació millor deixant algun dels nodes sense marcar? Justifiqueu la vostra resposta.
- (b) Dissenyeu un algorisme voraç per tal d'obtenir la millor puntuació possible. Justifiqueu la seva correctesa i doneu-ne el cost.
- (c) Suposeu ara que $\ell(v)$ pugui ser negatiu. Continua el vostre algorisme proporcionant la puntuació màxima possible?
- (d) Considereu la següent modificació del joc per aquest cas en què els nodes poguessin tenir valors negatius: eliminem primerament de G tots els nodes $v \in V$ amb $\ell(v) < 0$, per tal de seguidament executar el vostre algorisme sobre el graf resultant d'aquesta eliminació. Doneu un exemple on aquesta variació no proporcionï la màxima puntuació possible.

- 2.12. **(Gotes)** Considereu una canonada d'alçada L . La canonada té n gotes d'aigua a n alçades diferents dins d'ella. Les posicions on es troben les gotes es representen per nombres enters dins del rang $0 \dots L$. La posició 0 és la base de la canonada (el final) i la posició L n'és la part més alta.

Cada gota d'aigua es mou cap al final de la canonada a velocitat uniforme. Les gotes es mouen a diferents velocitats. La velocitat d'una gota és un nombre racional. Quan una gota d'aigua atrapa a una altra gota es barregen, i la nova gota assumeix la velocitat de la gota d'aigua atrapada (és a dir, la velocitat de la gota que estava més propera a la base de la canonada). Doneu un algorisme, el més eficient que pugueu, per determinar el nombre de gotes que surten de l'extrem final de la canonada. Justifica la seva correctesa, i dona el seu cost en funció d' n .

- 2.13. **(CinemaVis)** L'empresa CinemaVis ha de programar l'aparició d'un seguit d'anuncis a una pantalla gegant a la Plaça del Mig la diada de Sant Jordi. La tirada d'anuncis es pot iniciar a temps 0 (l'inici programat) però mai abans. A més, CinemaVis disposa d'un conjunt de n anuncis per fer la selecció. L'anunci i té una durada de 1 minut i té associat dos valors reals no negatius t_i i b_i . L'anunciat pagarà b_i euros a CinemaVis si l'anunci i s'emet a l'interval $[0, t_i]$ i 0 euros si s'emet després. Cap dels n anuncis es pot mostrar més d'una vegada. CinemaVis vol projectar la selecció d'anuncis que li proporcionin màxim benefici. Dissenyeu un algorisme, el més eficient que podeu, per a resoldre aquest problema.

2.14. (**Agenda**) A la vostra agenda teniu una llista L de totes les tasques que heu de completar en el dia de avui. Per a cada tasca $i \in L$ s'especifica la durada $d_i \in \mathbb{N}$ que indica el temps necessari per a completar-la i un factor de penalització $p_i \in \mathbb{Z}^+$ que n'agreuja el retard. Heu de determinar en quin ordre realitzar totes les tasques per obtenir el resultat que menys penalització total acumuli.

Tingueu en compte que:

- en un instant de temps només podeu realitzar una única tasca,
- una vegada comenceu a fer una tasca, heu de continuar-la fins a finalitzar-la, i
- s'han de completar totes les tasques.

El criteri d'optimització és la penalització total que s'acumula. La penalització real associada a una tasca $i \in L$ és el temps de finalització t_i de la seva realització, multiplicat per la seva penalització p_i . El temps de finalització t_i es correspon al temps transcorregut des de l'inici de la jornada laboral (és a dir, des de l'instant de temps 0) fins al moment en que s'ha finalitzat la tasca.¹

Considereu l'algorisme voraç que programa les tasques en ordre decreixent de factor de penalització p_i . Determineu si aquest algoritme resol el problema. En cas que no ho faci, proporcioneu un algorisme (tant eficient com pogueu) per resoldre'l.

¹Observeu que, segons les restriccions del problema, la realització d'una tasca $i \in L$ amb temps de finalització t_i haurà començat a l'instant $t_i - d_i$.

- 2.15. **(WebSocial)** A l'empresa *WebSocial* li han donat un premi molt important per la seva contribució tecnològica a l'estudi de les relacions interpersonals a xarxes socials. Per celebrar-ho han decidit organitzar una gran festa amb els seus clients en què volen fer gala dels bons resultats que obtenen.

Els CEOs de WebSocial han determinat un conjunt d' n tasques que s'han de portar a terme per a la correcta organització de la festa, i disposen d'un conjunt S d' n treballadors per encarregar-se'n. Totes les tasques requereixen un encarregat i totes menys una requereixen que una altra persona del grup de treballadors seleccionats actuï com a supervisor. Un treballador pot supervisar cap, una o més d'una tasca.

Fent ús de la seva pròpia tecnologia l'equip directiu de WebSocial ha mesurat, per a cada parell de treballadors (a, b) , un coeficient que indica la insatisfacció derivada del fet que un d'ells hagi de supervisar una tasca encarregada a l'altre. També ha determinat un subconjunt de treballadors $I \subseteq S$ que no poden ser supervisors de cap tasca (tot i ells sí que poden ser supervisats), i també un conjunt de treballadors $J \subseteq S$, $I \cap J = \emptyset$, que podrien fer-se càrrec de tasques sense necessitat de cap supervisió.

També han fixat un llindar màxim d'incompatibilitat u .

- (a) Doneu un algorisme amb cost polinòmic per determinar si és possible trobar una estructura jeràrquica compatible amb les restriccions de WebSocial de manera que, entre tot parell de supervisor i supervisat, el nivell d'insatisfacció sigui com a màxim u .
- (b) Assumint que hi ha una estructura jeràrquica compatible amb les restriccions de l'apartat (a), proporcioneu un algorisme eficient per obtenir una estructura jeràrquica en la qual es minimitzi la suma dels coeficients d'insatisfacció entre supervisors i supervisats.

- 2.16. **(Fusionant seqüències)** Ja sabeu que fer la fusió ordenada de dues seqüències ordenades d' m i n elements, respectivament, comporta fer $m + n$ moviments de dades (penseu, per exemple, en un *merge* durant l'ordenació amb *mergesort* d'un vector). Però si hem de fer la fusió d' N seqüències, dos a dos, l'ordre en què es facin les fusions és rellevant. Imagineu que tenim tres seqüències A , B i C amb 30, 50 i 10 elements, respectivament. Si fusionem A amb B i després el resultat el fusionem amb C , farem $30 + 50 = 80$ moviments per a la primera fusió i $80 + 10 = 90$ per a la segona, amb un total de 170 moviments. En canvi, si fusionem primer A i C i el resultat el fusionem amb B farem un total de 130 moviments.

Dissenyau un algorisme golafre (*greedy*) per fer les fusions i obtenir la seqüència final ordenada amb mínim nombre total de moviments. Justifiqueu la seva correctesa i calculeu-ne el cost temporal del vostre algorisme en funció del nombre de seqüències N .

2.17. (Shell decomposition)

Amb l'aparició de les xarxes socials modernes han sorgit nous problemes i elements a estudiar que són de gran interès algorítmic. Un d'ells, per exemple, és l'anàlisi de la importància dels actors (usuaris) de la xarxa en relació a un procés de transmissió d'influència que estigui passant en un moment donat. Per a estudiar això s'utilitza una descomposició particular de la xarxa que s'anomena *shell decomposition* (descomposició de closca); per tal d'entendre en què consisteix aquesta descomposició, necessitem conèixer els conceptes de *k-core* i *k-shell*.

Si tenim una xarxa social G (graf no dirigit) amb n actors (vèrtexs), el ***k-core*** és el subgraf induït de mida màxima en què tots els vèrtexs tenen com a mínim grau k , amb $0 \leq k \leq n$. La ***k-shell*** és el conjunt d'actors que pertanyen al k -core però no pertanyen al $(k + 1)$ -core.² Aleshores, la ***shell decomposition*** de G és la partició de vèrtexs definida per les k -shells de G que no són buides.

Donada una xarxa G , dissenyeu un algoritme amb cost lineal que proporcioni la shell decomposition de G . El vostre algoritme ha d'indicar també, per a cada shell a la descomposició, el corresponent valor de k .

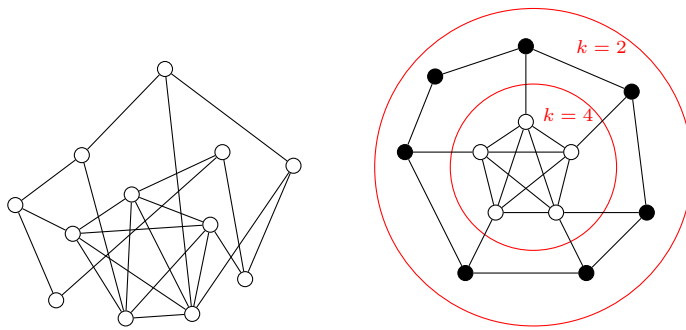


Figura 1: Un graf i una il·lustració de la seva *shell decomposition*

Exemple: Per al graf de la Figura 1 (esquerra), el 4-core és el subgraf induït per els vèrtexs dins del cercle interior que es destaquen a la figura dreta (nodes buits ○), i el 2-core és tot el graf. Per a la resta de k -cores: noteu que el 3-core és igual al 4-core, que el 0-core i l'1-core són iguals al 2-core, i que el k -core és buit $k \geq 5$.

Per tant, la *shell decomposition* té dues shells no buides: la 4-shell, composta pels vèrtexs del 4-core (nodes buits ○), i la 2-shell, composta pels vèrtexs del 2-core que no són al 3-core (nodes plens ●).

²En alguns grafs pot passar que el k -core o la k -shell siguin buits per a algunes k .

2.18. (Reduint CO_2)

Els nostres polítics han decidit contribuir a la reducció d'emissions de CO_2 compartint el seus vehicles oficials. En assistir a la reunió anual d'alcaldes, un vehicle oficial pot donar servei entre les ciutats A i B, portant a la ciutat B tots els alcaldes que es trobin a A (inclòs l'alcalde d'A). Per a cada possible trajecte (A, B) entre dues ciutats de la província s'ha estimat un valor $c(A, B)$ corresponent a les tones de CO_2 que emet el vehicle oficial en fer aquest trajecte.

Dissenyeu un algorisme que, atesa una descripció dels possibles trajectes entre N ciutats, les corresponents estimacions d'emissions de CO_2 i la ciutat on es durà a terme la propera reunió, determini els trajectes que han de ser operats de manera que es minimitzi el total de CO_2 emès i tots els alcaldes puguin arribar a la reunió.

Justifiqueu-ne la correctesa i l'anàlisi del cost en funció del nombre N de ciutats.

Observacions:

- Totes les ciutats tenen alcalde i tots han d'anar a la reunió.
- Inicialment cada alcalde es troba a la seva ciutat.
- El valor $c(A, B)$ és independent del nombre de persones que viatgen al vehicle i del sentit en què es faci el trajecte (és a dir, $c(A, B) = c(B, A), \forall A, B$).
- Els vehicles oficials són molt grans; podeu suposar que hi caben tantes persones com necessiteu.

- 2.19. **(SenseAll)** La companyia SENSEALL ha rebut l'encàrrec de dotar el parc de Coltxerola d'un sistema de prevenció d'incendis. D'acord amb la Conselleria de Turisme, han fixat les ubicacions dels sensors que s'utilitzaran al sistema. Hi ha dos tipus de sensors, **RevSen** i **RevSpr**. Tots dos són capaços de detectar un senyal de perill directament de l'entorn o procedent d'un altre sensor. A més, un sensor **RevSen** pot també transmetre senyals de perill a altres sensors, i un **RevSpr** pot activar els aspersors del seu entorn.

SENSEALL ha de determinar com connectar els sensors. L'acord estableix que tots els sensors han de formar una xarxa connectada segura, ja que, en cas d'emergència, s'haurien d'activar tots els aspersors del parc després que un **RevSen** detecti un possible incendi. D'altra banda, SENSEALL vol que el cost de construcció de la xarxa de comunicació sigui el mínim possible. Per això, per a cada parell de sensors que es poden connectar, disposa d'una estimació del cost de construir una línia segura de connexió entre ells. El cost correspon a l'establiment de comunicació bidireccional entre dos sensors **RevSen**, i unidireccional entre **RevSen** i **RevSpr**.

Dissenyeu un algorisme que, atès el nombre de sensors de cada tipus i els costos d'establiment de comunicació segura entre ells, determini si és possible construir una xarxa que compleixi els requeriments i, en cas de ser-ho, proporcioneu el detall de les connexions a construir.

- 2.20. (**Agenda II**) A la vostra agenda teniu una llista L de totes les tasques que heu de completar en el dia d'avui. Per a cada tasca $i \in L$ s'especifica la durada $d_i \in \mathbb{N}$ que indica el temps necessari per a completar-la i un factor de penalització $p_i \in \mathbb{Z}^+$ que n'agreuja el retard. Heu de determinar en quin ordre realitzar totes les tasques per obtenir el resultat que **menys** penalització total acumuli.

Tingueu en compte que:

- en un instant de temps només podeu realitzar una única tasca,
- una vegada comenceu a fer una tasca, heu de continuar-la fins a finalitzar-la, i
- s'han de completar totes les tasques.

El criteri d'optimització és la penalització total que s'acumula. La penalització real associada a una tasca $i \in L$ és l'ordre que ocupa la tasca dins de la planificació final, multiplicat per la seva penalització p_i ; per exemple, si la tasca i es fa en 5a posició, la seva penalització serà de $5 \cdot p_i$.

Considereu l'algorisme voraç que programa les tasques en ordre decreixent segons el ratio p_i/d_i . Determineu si aquest algoritme resol el problema. En cas que ho faci, justifiqueu-ne el perquè. En cas que no ho faci, proporcioneu un algorisme (tant eficient com pugueu) per resoldre'l.

2.21. (Xarxa de fibra)

El govern d'Atlàntida t'ha encarregat la instal·lació d'una xarxa de fibra òptica que connecti totes les seves n ciutats. Hi ha dues tecnologies de fibra òptica que es poden utilitzar:

- FIBRA 1.0: És una tecnologia fiable i relativament econòmica. Es disposa d'una llista de parelles de ciutats entre les quals és possible instal·lar un enllaç directe de FIBRA 1.0. A més, per a cada parella hi ha associat un cost d'establiment, que és un enter positiu.
- FIBRA 2.0⁺: És una tecnologia emergent d'alt rendiment que pot connectar directament qualsevol parella de ciutats. Tanmateix, el seu cost és tan elevat que el govern no pot permetre's ni un sol enllaç de FIBRA 2.0⁺.

Cal tenir en compte que tots els enllaços directes són bidireccionals. La xarxa instal·lada ha de connectar totes les ciutats d'Atlàntida: entre qualsevol parella de ciutats ha d'existir un camí format per enllaços directes que les connecti.

Un filantrop s'ha ofert a finançar exactament $k < n$ enllaços directes de FIBRA 2.0⁺, que es poden utilitzar per connectar k parelles qualssevol de ciutats. L'objectiu és minimitzar el cost que ha d'assumir el govern pels enllaços de FIBRA 1.0 necessaris per construir una xarxa connexa.

Descriviu, justifiqueu i analitzeu un algorisme que trobi la xarxa amb el cost mínim per al govern. Observeu que és possible construir una xarxa que connecti totes les ciutats utilitzant únicament enllaços de FIBRA 1.0, però es vol aprofitar la donació dels k enllaços de FIBRA 2.0⁺ per tal de reduir al màxim el cost assumit pel govern.