Problemes 2

2.1. **(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: $T = \sum_{i=1}^{n} e_i$.

2.2. **(Subgraf induït)** Tenim un graf no dirigit $G = (V, E)$. Donat un subconjunt $V' \subseteq V$ el *subgraph 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 $G[V']$.

2.3. **(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.

2.4. **(Recobriment minimal)** Sigui $G = (V, E)$ un graf no dirigit. Un subconjunt $C \subseteq V$ s'anomena *recobriment de vèrtexs* de G si

$$\forall \{u, v\} \in E : \{u, v\} \cap C \neq \emptyset.$$

Donat $G = (V, E)$, un recobriment de vèrtexs C és diu que és *minimal* si per qualsevol $C' \subseteq V$ a G, tal que $C' \subset C$ hem de tenir que C' no és un recobriment de vèrtexs.

Un conjunt $C \subset V$ és un *recobriment de vèrtexs mínim* si C és un recobriment de vèrtexs a G amb mínima cardinalitat. (Quan G és un arbre, hi ha un algorisme polinòmic per a trobar un recobriment de vèrtexs mínim a G, però per a G generals el problema és NP-hard).

En aquest problema demostrareu que, a diferència del problema de trobar un recobriment de vèrtexs mínim, el problema de trobar un recobriment de vèrtexs minimal pot ser resolt en temps polinòmic.

(a) Demostreu que un recobriment de vèrtexs minimal no necessàriament ha de ser un recobriment de vèrtexs mínim.

(b) Demostreu que tot recobriment de vèrtexs mínim també és minimal.

(c) Doneu un algorisme polinòmic per trobar un recobriment de vèrtexs minimal a G.

2.5. **(Ratolins i caus)** A un experiment sociològic es disposen n ratolins a les posicions $R = \{r_1, \ldots, r_n\}$ determinades sobre una línia recta. A la mateixa línia hi ha n caus localitzats a les posicions $C = \{c_1, \ldots, c_n\}$. 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. **(SOS Rural)** Considerad un camino rural en el Pirineo con casas muy dispersas a lo largo de él. Por motivos de seguridad se quieren colocar estaciones SOS en algunos puntos de la carretera. Los expertos indican que, teniendo en cuenta las condiciones climáticas de la zona, se debería garantizar que cada casa se encuentre como máximo a una distancia de 15km, siguiendo la carretera, de una de las estaciones SOS.

Proporcionad un algoritmo eficiente que consiga este objetivo utilizando el mínimo número de estaciones SOS posibles. Justificad la corrección y el coste de vuestro algoritmo e indicad la complejidad en tiempo de la solución propuesta.

2.7. **(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

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

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

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

2.9. **(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 esta 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)

2.10. **(Bottleneck ST)** Un *bottleneck spanning tree* T d'un graf no dirigit i ponderat $G = (V, E, w)$, on $w : E \to \mathbb{R}^+$, 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 $\leq b$.

2.11. **(MST únic?)** Demostreu que un graf G té un únic MST si, per a tot tall C de G, existeix una única aresta $e \in C$ amb valor mínim. Demostreu que el el reciproc no és cert, i.e. pot ser el cas de que per un o més talls C tinguem més d'una aresta mínim pes, però que el MST sigui únic

2.12. **(MST amb colors)** Tenim un graf no dirigit i connex $G = (V, E)$ i una coloració de les arestes amb dos colors, roig i blau ($c : E \to \{R, B\}$). Doneu un algorisme per a obtenir un arbre d'expansió amb el mínim nombre d'arestes blaves.

2.13. **(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.

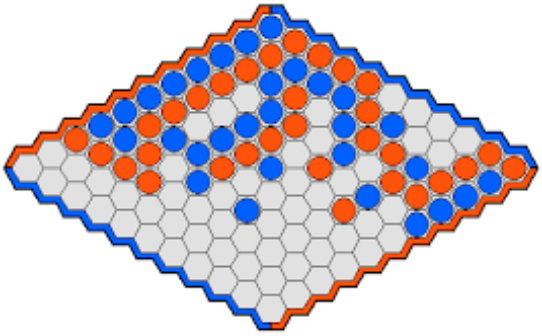
(a) És possible obtenir una puntuació millor deixant algun dels nodes sense marcar? Justifiqueu la vostra resposta.

(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 proporcioni la màxima puntuació possible.

2.14. **(HEX)** El joc d'HEX té com un dels seus inventors, al matemàtic John Nash. En aquest joc, dos jugadors, un amb color negre i l'altre amb color e blanc, fan torns on a cada torn el jugador que li toca col·loca una pedra del seu colors a una posició encara buida, a una xarxa $n \times n$ de cel.les hexagonals. Un cop col·locada una pedra, no es pot moure. L'objectiu de cada jugador és connectar els costats del mateix color a la graella, amb un camí continuo fet amb les seves pedres. Dues cel·les es consideren connectades si comparteixen una vora les dues tenen la pedres amb el mateix color.



Red won

Descriviu un algorisme eficient per al seguiment d'una partida. L'algorisme, després de cada jugada, ha d'indicar si el jugador que acaba de jugar ha guanyat el joc de HEX.

2.15. **(Moviments tauler)** Tenim un tauler de dimensions $n \times n$, amb n fitxes col·locades a certes posicions $(x_1, y_1), \ldots, (x_n, y_n)$ i una fila i. Volem determinar el mínim nombre de moviments necessaris per a posar les n fitxes a la fila i (una a cada casella). Els moviments permesos són: cap a la dreta, esquerra, amunt i avall. Durant aquests moviments es poden apilar tantes fitxes a la mateixa posició com calgui. Pero en finalitzar ha de quedar una fitxa per casella.

Pista: El nombre de moviments verticals (amunt/avall) necessaris es pot calcular fàcilment.

2.16. **(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 proporcioni màxim benefici. Dissenyeu un algorisme, el més eficient que podeu, per a resoldre aquest problema.

2.17. **(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:

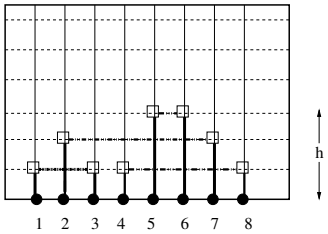
- en un instant de temps només podeu realitzar una única tasca,
- una vegada comenceu a fer una tasca, heu de continuar-la fins a finalitzar-la, i
- s'han de completar totes les tasques.

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.

[1] Observeu 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 $t_i - d_i$.

2.18. **(VLSI)** Als circuits VLSI, s'utilitza un encaminament Manhattan sobre la placa aïllant on va muntat el circuit. Les connexions horitzontals van per la cara de sota i les connexions verticals per la cara de sobre. Quan es necessiten connectar les connexions horitzontals amb les verticals, es perfora la placa amb el que s'anomena una *via*. Les connexions del circuit amb l'exterior es realitzen amb *pins* (veure la figura, on les connexions de sobre estan dibuixades en sòlid i les que van per sota amb línia discontínua). Tant les connexions horitzontals com verticals segueixen unes pistes dibuixades sobre la placa en forma d'una graella, i els pins estat alineats a un extrem de la placa. Sigui h el nombre de pistes horitzontals utilitzades. Si $L = \{(p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n)\}$ són una seqüència de parells de pins a connectar (dos a dos), volem dissenyar un algorisme que connecti els parells de pins utilitzant el mínim nombre de pistes horitzontals h. Per exemple, considereu la següent figura:



Tenim com a entrada $L = \{(1, 3), (2, 7), (4, 8), (5, 6)\}$, el nombre de h és 3, i no es pot fer amb $h = 2$.

En particular: dissenyeu un algorisme eficient, que donats n parells de pins, resolgui el problema de l'encaminament, de manera que es minimitzi h. Doneu-ne la complexitat i demostreu-ne la correctesa.

2.19. **(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.

2.20. **(BCN-Paris)** Doneu un algorisme que resolgui el següent problema i doneu la seva complexitat en funció de n. Justifiqueu-ne la correctesa:

Volem anar de Barcelona a Paris seguint l'autopista a través de Lyon, i tenim n benzineres, $B_1 \ldots, B_n$, al llarg d'aquesta ruta. Amb el diposit ple, el vostre cotxe pot funcionar K km. La benzinera B_1 és a Barcelona, i cada B_i, $2 \leq i \leq n$ és a $d_i < K$ km. de la benzinera B_{i-1}. La benzinera B_n és a Paris. Quina és l'estratègia per aturar-se el mínim nombre de cops al llarg del viatge?

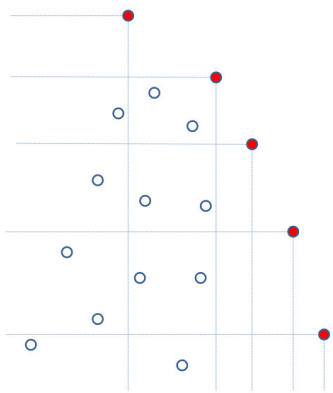
2.21. **(Fusionant seqüències)** Ja sabeu que fer la fusió ordenada de dues seqüències ordenades d'm i n elements, respectivament, comporta fer $m+n$ moviments de dades (penseu, per exemple, en un *merge* durant l'ordenació amb *mergesort* d'un vector). Però si hem de fer la fusió d'N seqüències, dos 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.

Dissenyeu 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.

2.22. **(Òptims de Pareto)**

Considereu un conjunt $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ d'n punts en el pla Euclidià (2-dimensional). Diem que un punt (x_i, y_i) *domina* un altre punt (x_j, y_j) si és més gran en totes dues coordenades, és a dir, si $x_i > x_j$ i $y_i > y_j$. Diem que un punt és *maximal* si no és dominat per cap altre.

Per exemple, al conjunt de punts mostrat en aquesta figura, els punts sòlids són els punts maximals, mentre que els punts buits són dominats. Les línies discontínues emmarquen quins punts són dominats per cadascun dels punts maximals.



Donat un conjunt P d'n punts, dissenyeu un algorisme eficient per a trobar tots els punts maximals de P (els anomenats *Òptims de Pareto*).

2.23. **(Streaming)** Tenim n paquets de vídeo que s'han d'enviar seqüencialment per una única línia de comunicació (això s'anomena *streaming*). El paquet i-èsim té una grandària de b_i bits i triga t_i segons a travessar, on t_i i b_i són enters positius (no es poden enviar dos paquets al mateix temps). Podeu assumir que la velocitat de transmissió d'un paquet es uniforme.

Hem de decidir una planificació de l'ordre en què hem d'enviar els paquets de manera que, un cop tenim un ordre, no pot haver-hi retard entre la fi d'un paquet i el començament del següent. Assumirem que comencem a l'instant 0 i finalitzem al $\sum_{i=1}^{n} t_i$. A més, el proveïdor de la connexió vol utilitzar una amplada de banda no massa gran, per tant s'afegeix la restricció següent: Per a cada enter $t > 0$, el nombre total de bits que enviem de temps 0 a t ha de ser $\leq rt$, per a una $r > 0$ fixada (noteu que aquesta restricció no diu res per a períodes de temps que no comencen a l'instant 0). Direm que una planificació és *vàlida* si satisfà la restricció prèvia.

El problema a resoldre és el següent: donat un conjunt de n paquets, cadascun especificat per la seva grandària b_i i la durada de la seva transmissió t_i, i donat el valor del paràmetre r, hem de determinar si existeix una planificació vàlida dels n paquets. Per exemple, si $n = 3$ amb $(b_1, t_1) = (2000, 1)$, $(b_2, t_2) = (6000, 2)$ i $(b_3, t_3) = (2000, 1)$ amb $r = 5000$, aleshores la planificació $1, 2, 3$ és vàlida, donat que compleix la restricció.

Per a resoldre aquest problema, heu de resoldre els apartats següents:

(a) Demostreu o doneu un contraexemple a l'enunciat de: és veritat que existeix una planificació vàlida si, i únicament si, per a cada paquet i tenim $b_i \leq rt_i$.

(b) Doneu un algorisme que per a una entrada de n paquets (cadascun especificat per (b_i, t_i)) i el paràmetre r, determini si existeix una planificació vàlida. El vostre algorisme hauria de tenir complexitat polinòmica en n.

2.24. **(CatCar)** Un grup de persones que necessiten esporàdicament un cotxe han creat CatCar, una cooperativa per gestionar l'ús compartit dels seus cotxes. La companya té una web, quan una membre de CatCar necessita un cotxe, la nit prèvia, escriu a la pàgina web de la CatCar, l'hora i lloc on vol recollir el cotxe i l'hora i lloc on el deixarà. CatCar ha de preveure tenir suficients cotxes disponibles a cada lloc per a satisfer les demandes del dia, però, per altra banda, també és important tenir un excés de cotxes immobilitzats a un lloc. Per tant, CatCar vol un algorisme eficient que minimitzi el nombre total de cotxes que la companya ha de deixar a cada lloc.

Formalment, el problema és el següent: Com a entrada tenim m llocs (1 fins a m) i n tuples (l, t, l', t'), a on l, t és el lloc i temps de recollida i l', t' és el lloc i tems de retornar el cotxe. Quan una persona s'emporta un cotxe d'un lloc determinat, el nombre de cotxes al lloc disminueix en 1, i quan el torna, incrementa en 1. Doneu un algorisme que calculi el nombre de cotxes que CatCar ha de deixar a cada un dels m llocs, amb les restriccions que cada lloc sempre ha de tenir un nombre no negatiu de cotxes i que el nombre de cotxes immobilitzats ha de ser mínim.

2.25. **(Punts)** Donat un conjunt $\{x_1, x_2, \ldots, x_n\}$ de punts de la recta real, doneu un algorisme per a determinar el conjunt més petit d'intervals tancats amb longitud unitat, que cobreixen tots els punts.

2.26. **(MST?)** Argumenteu si el següent algorisme per a trobar el MST d'un graf donat $G = (V, E)$, que és connex, no dirigit i amb pesos $w : E \to \mathbb{N}^+$, és o no és correcte, i si és correcte doneu la seva complexitat:

(a) Particioneu V en dos subconjunts S_1 i S_2 (i.e. $V = S_1 \cup S_2$ i $S_1 \cap S_2 = \emptyset$ tal que cadascun dels subgrafs resultants G_{S_1} i G_{S_2} son connexes.

(b) Recursivament trobeu un MST T_{S_1} per a G_{S_1} i un MST T_{S_2} per a G_{S_2}.

(c) Com G és connex hi hauran arestes entre els vèrtexs de T_{S_1} i de T_{S_2}, escolliu l'atesta amb menys pes per a formar un MST de G.

2.27. **(MST a grafs petits)** Tenim un graf no dirigit i connex $G = (V, E)$ amb pesos $w : E \to \mathbb{N}^+$, i tal que $|V| = n$ i $|E| = n$. Doneu un algorisme **linear** per a trobar el MST de G.

2.28. **(About Huffman)** Per cadascuna de les afirmacions següents indique-ho si és o no certa.

(a) En un codi de Huffman, si tots els caràcters tenen freqüència $< 1/3$, aleshores **NO** existeix cap caràcter que es pugui codificar amb longitud 1.

(b) En un codi de Huffman, si tots els caràcters tenen freqüència $< 2/5$, aleshores no hi han caràcters que es pugui codificar amb longitud 1.

(c) Si a un codi Huffman tenim una fulla a profunditat 1 i una altra fulla a profunditat 3, això implica que ha d'haver una fulla a profunditat 2.