42. Suposem que tenim dos conjunts A i B cadascun amb n enters positius (desordenats). Podeu optar per canviar l'ordre de cada conjunt com vulgueu. Després de reordenar els conjunts, sigui a_i l'element i-èsim d'A i sigui b_i l'element i-èsim de B (després de la vostra re-ordenació). Aleshores rebeu un pagament de $\prod_{i=1}^{n} a_i^{b_i}$ bitcoins. Doneu un algorisme que maximitzi el vostre guany. Demostreu la correctesa i digueu quina és la complexitat del vostre algorisme.

    Una solució:

    Idea: per maximitzar els beneficis cal reordenar els dos conjunts d'entrada. Farem ús de la tècnica dels algorismes voraços per seleccionar l'ordenació que proporciona l'òptim valor.

    Estructura de suboptimalitat: Si a_i és el valor més gran a A, i b_j és el valor més gran a B, si $i \neq j$, podem comparar els termes $a_i^{b_i} a_j^{b_j}$ i $a_i^{b_j} a_j^{b_i}$. Tenim

    $$\frac{a_i^{b_j} a_j^{b_i}}{a_i^{b_i} a_j^{b_j}} = (\frac{a_j}{a_i})^{b_j - b_i} \leq 1.$$

    Independentment de l'ordre, si la solució és optima els dos valors màxims han d'aparèixer junts i la resta d'elements s'han d"ordenar de forma òptima. Això ens porta a la regla golafre:

    > *Mentre quedin elements,*
    > *seleccioneu els elements amb valor màxim d'A i d'B.*

    D'acord amb aquesta regla, l'ordenació que proporciona l'òptim pagament és l'ordre decreixent.

    La complexitat d'ordenar és $O(n \lg n)$, però si n té valor suficientment petit, la complexitat pot ser $\Theta(n)$ fent ús de l'algorisme Radix sort.

43. 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 $E' = E \cap (V' \times V')$. 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 $G[V']$.

44. Sigui $G = (V, E)$ un graf no dirigit. Un subconjunt $I \subseteq V$ s'anomena *subconjunt independent* si, per a tot parell de vèrtexs $u, v \in I$ tenim que $(u, v) \notin E$. Donat $G = (V, E)$, un conjunt independent I es diu que és *maximal* si, per qualsevol altre conjunt independent $V' \subseteq V$ a G tal que $I \subset V'$, forçosament hem de tenir que V' no és independent. Un conjunt $M \subset V$ és un *conjunt independent màxim* si M és un conjunt independent

a G amb màxima cardinalitat.[1] En aquest problema demostrareu que, a diferència de trobar el conjunt independent màxim, el problema de trobar el conjunt independent maximal por ser resolt en temps polinòmic, per a qualsevol G.

(a) Demostreu que un conjunt independent maximal no necessàriament és un conjunt independent màxim.

(b) Demostreu que tot conjunt independent màxim també és maximal.

(c) Doneu un algorisme polinòmic que trobi un conjunt maximal a G. Quina és la seva complexitat?

45. 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.

46. 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.

47. Sigui $X = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ un conjunt de n intervals. Diem que un conjunt de punts $P = \{p_1, \ldots, p_k\}$ *intercepta* un conjunt d'intervals X si tot interval a X conté com a mínim un punt de P, és a dir,

$$\forall_{1 \leq i \leq n} \exists_{1 \leq j \leq k} \;\; a_i \leq p_j \leq b_i.$$

Describiu i analitzeu un algorisme eficient per, donats el conjunt d'intervals X i el conjunt de punts P, calculi el subconjunt més petit de punts que intercepta X.

48. **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). 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

[1] Quan G és un arbre, hi ha un algorisme polinòmic per a trobar un conjunt independent màxim a G, però per a G generals, el problema és NP-hard.

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.

49. Tenim un alfabet Σ on per a cada símbol $a \in \Sigma$, p_a es la probabilitat que aparegui el caràcter a. Demostreu que, per a qualsevol símbol $a \in \Sigma$, la seva profunditat en un arbre prefix que produeix un codi de Huffman òptim és $O(\lg \frac{1}{p_a})$. (Ajuts: en un arbre prefix que s'utilitzi per a dissenyar el codi Huffman, la probabilitat d'un nus és la suma de les probabilitats dels fills. La probabilitat de l'arrel és, doncs, 1.)

50. 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 $\leq b$.

51. Un graf és *unicíclic* si conté exactament un cicle. Un *subgraf d'expansió unicíclic* d'un graf $G = (V, E)$ és un subgraf connex $H = (V, F)$, amb $F \subseteq E$ que, a més, és unicíclic.

(a) Quants arbres d'expansió té un graf unicíclic?

(b) Doneu un algorisme eficient per trobar un subgraf unicíclic d'expansió amb pes mínim.

52. 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.

53. CoopC és una companyia de lloguer de cotxes que vol diversificar el seu negoci per tal de competir en els desplaçaments curts. La companyia té oficines distribuïdes a Barcelona i un dipòsit central de cotxes al Prat. CoopC vol un mètode que li permeti processar les peticions de trajectes curts rebuts pel dia següent. Per això, vol determinar el nombre de cotxes que ha de deixar a cada oficina a l'inici del dia dedicats a trajectes curts. Aquesta assignació ha de permetre tenir suficients cotxes disponibles al llarg del dia (en cadascuna de les oficines) per tal de poder cobrir tots els trajectes curts del dia. A més, CoopC vol minimitzar el nombre de cotxes destinats durant el dia a desplaçaments curts.

La informació de la qual disposa CoopC per a cada sol·licitud de trajecte curt per al proper dia és la següent: una tupla (l, t, l', t') on el parell (l, t) indica l'oficina i l'hora de recollida del vehicle i el parell (l', t') indica l'oficina i l'hora de devolució.

A efecte d'aquesta aproximació CoopC assumeix que tots els cotxes són idèntics i que disposa d'una flota prou gran per a poder cobrir qualsevol nivell de demanda de trajectes curts. A més, assumeix també que els cotxes seran recollits i retornats puntualment a les hores i locals estipulats.

Dissenyeu un algorisme voraç que permeti obtenir el nombre de cotxes que s'han d'assignar a cada oficina, de manera que, al llarg del dia, sempre hi hagi almenys un cotxe disponible en una oficina a l'hora en què algun client l'ha de recollir d'allà. Tenint en compte que mantenir immobilitzat

un cotxe té un cost alt, l'assignació obtinguda ha de garantir el requisit de disponibilitat i ha d'assignar el menor nombre possible de cotxes a cada oficina.

54. **Clustering.** Una versió senzilla del *problema de l'agrupació (clustering)* és la següent. Com a entrada al problema tenim: un conjunt de n objectes $X = \{x_1, x_2, \ldots, x_n\}$, una distància $d(x_i, x_j) > 0$ per a cada parell i, j amb $i \neq j$, una $k \in \mathbb{N}$. El problema consisteix a particionar X en k subconjunts no buits $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ (anomenats clústers) de manera que es maximitzi la separació entre clústers. La separació entre clústers a $\mathcal{C}$ és la distància mínima entre tots els possibles parells de punts a diferents clústers de $\mathcal{C}$. Doneu un algorisme polinòmic per solucionar aquest problema.

55. 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. 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.

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

56. La cadena de sota és el títol d'una cançó codificat amb codis de Huffman.

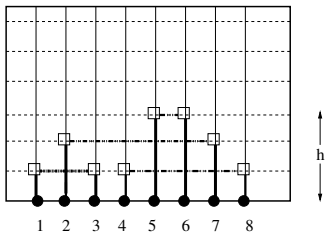
    00110001011111011001110111011000001001110100101 01

    Donades les freqüències de les lletres que es donen a la taula de sota, obtingueu els codis de Huffmann i feu-los servir per decodificar el títol. Tingueu en compte que, quan hi ha múltiples eleccions, s'han d'agafar les dues primeres (d'esquerra a dreta, a l'ordre donat a la taula). Heu d'assignar a la branca esquerra el símbol 0 i a la dreta el símbol 1.

lletra	a	h	v	w	' '	e	t	l	o
freqüència	1	1	1	1	2	2	2	3	3

57. 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.

58. 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?

Una solució: L'estratègia que farem servir és esperar el màxim per a carregar benzina, però sense quedar-nos amb el dipòsit buit. Aquest criteri es pot implementar com el següent algorisme voraç:

```
function GREEDY(d_1, ... d_n, K)
    d = 0; R = ∅; j = 1
    for i = 1 ... n − 1 do
        if d + d_{i+1} > K then
            j := j + 1;
            R = R ∪ {B_j};
            d = 0;
        end if
        d := d + d_{i+1};
    end for
    return (j);
end function
```

Per demostrar la correctesa de l'algorisme compararem la solució obtinguda pel nostre algorisme amb una posible solució òptima amb menys

aturades. Sigui $R = \{b_1, \ldots, b_j\}$ les benzineres seleccionades per GREEDY. Suposem que la solució òptima requereix $m < j$ aturades, sigui $S = \{c_1, \ldots, c_m\}$ el conjunt de les benzineres a una solució òptima.

Primer demostrarem que, per $j = 1, \ldots, m$, $d(B_1, b_i) \geq d(B_1, c_i)$. L'enunciat és cert per $j = 1$, si no ens quedaríem sense benzina abans d'arribar a c_1. Suposem que l'enunciat és cert per $i < j$. Llavors tenim

$$d(B_1, c_j) - d(B_1, c_{j-1}) \leq K,$$

ja que S és una solució, i

$$d(B_1, c_j) - d(B_1, b_{j-1}) \leq d(B_1, c_j) - d(B_1, c_{j-1}),$$

per hipòtesi d'inducció. Combinant les dues desigualtats tenim

$$d(B_1, c_j) - d(B_1, b_{j-1}) \leq K.$$

Llavors, c_j ha de ser abans de b_j.

Com a conseqüència del resultat tenim que $d(b_m, B_n) \geq d(c_m, B_n) \leq K$, llavors l'algorisme voraç no s'aturaria a cap benzinera després de b_m i tenim $j = m$.

Per tant GREEDY és òptim i la seva complexitat és $O(n)$.