18. El Professor JD ha corregit els exàmens finals del curs, de cara a tenir una distribució maca de les notes finals decideix formar $k$ grups, cada grup amb el mateix nombre d'alumnes, i donar la mateixa nota a tots els alumnes que són al mateix grup. La condició més important és que qualsevol dels alumnes al grup $i$ han de tenir nota d'examen superior a qualsevol alumne d'un grup inferior (grups de 1 fins a $i-1$). L'ordre dintre de cadascun dels grups es irrellevant. Dissenyeu un algorisme que donada una taula $A$ no ordenada, que a cada registre conté la identificació d'un estudiant amb la seva notes d'examen, divideix $A$ en els $k$ grups, amb les propietat descrita a dalt. El vostre algorisme ha de funcionar en temps $O(n \lg k)$. Al vostre anàlisis podeu suposar que $n$ és múltiple de $k$ i $k$ és una potencia de 2.

    **Una solució:** Sigui AGRUPAR l'algorisme recursiu que, té com a entrada una taula de alumnes-notes $N$ i dos enters $\ell$ i $t$, i fa el següent:

    - Mentre $\ell \neq 1$ troba la mediana de $A$ i fa una partició al seu voltant en temps $O(|N|)$.
    - Considerem la sub-taula $N_e$ esquerra i la sub-taula dreta $N_d$.
    - Cridem recursivament
      AGRUPAR$(N_e, \ell/2, 2t)$ i AGRUPAR$(N_d, \ell/2, 2t+1)$.
    - Quan $\ell = 1$, la taula constitueix la partició $t$.

    La crida inicial la farem amb $N$, $\ell = k$ i $t = 0$. La correctesa ve de com particionem els elements. Sempre tenim dos meitats i els elements a $N_e$ són més petits que la mediana i els elements a $N_d$ són més grans o iguals que la mediana. Aconseguirem $\ell = 1$ després de $\lg k$ iteracions, en aquell moment la taula té $n/k$ elements. La variable $t$ comptabilitza l'ordre de las crides. Al primer nivell tenim només una taula i $t = 0$. Al segon tindrem dos taules, la de l'esquerra etiquetada amb 0 i la de la dreta amb 1. Al següent nivell, tindrem 0,1,2,3 (e-e,e-d,d-e,d-d). Llavors $t$ comptabilitza l'ordre correcte de les particions per garantir la propietat requerida.

    El cost de l'algorisme és $T(n,k) = 2T(n/2, k/2) + \Theta(n)$ amb $T(n,1) = \Theta(1)$, per a tot $n$. Desplegant la recursió tenim

$$\begin{aligned} T(n,k) &= 2T(n/2,k/2) + cn = 4T(n/4,k/4) + 2c(n/2) + cn \\ &= 4T(n/4,k/4) + 2cn = k + cn\lg k. \end{aligned}$$

    llavors, $T(n) = \Theta(n \lg k)$.

19. Resoleu les següents recurrències

(a) $T(n) = 16T(n/2) + \binom{n}{3} \lg^4 n$

(b) $T(n) = 5T(n/2) + \sqrt{n}$

(c) $T(n) = 2T(n/4) + 6.046\sqrt{n}$

(d) $T(n) = 2T(n/2) + \frac{n}{\lg n}$

(e) $T(n) = T(n-10) + n$

20. Considereu el següent algorisme de dividir-i-vèncer per al problema de *trobar un clique* en un graf no dirigit $G = (V, A)$ (recordeu un clique és un subgraf $C$ de $G$ on tots els vèrtexs estan connectats entre ells).

CLIQUE($G$)

1 Enumereu els vèrtexs $V$ com $1, 2, \ldots, n$, on $n = |V|$

2 Si $n = 1$ tornar $V$

3 Dividir $V$ en $V_1 = \{1, 2, \ldots, \lfloor n/2 \rfloor\}$ i $V_2 = \{\lfloor n/2 \rfloor + 1, \ldots, n\}$

4 Sigui $G_1 = G[V_1]$ el subgraf induït per $V_1$ i sigui $G_2$ el subgraf induït per $V_2$ (les arestes de $G_1$ són les arestes que connecten vèrtexs en $V_1$ i similar amb $G_2$)

5 Recursivament trobeu cliques $C_1$=CLIQUE($G_1$) i $C_2$=CLIQUE($G_2$)

6 Combineu aquests dos cliques de la manera següent:

6.1 Inicialitzar $C_1^+$ com a $C_1$ i $C_2^+$ com a $C_2$

6.2 Per a cada vèrtex $v \in C_2$, si $v$ està connectat a tots els vèrtexs a $C_1^+$, aleshores afegir $v$ a $C_1^+$

6.3 Per a cada vèrtex $u \in C_1$, si $u$ està connectat a tots els vèrtexs a $C_2^+$, aleshores afegir $u$ a $C_2^+$

6.4 Retorneu el més gran d'entre $C_1^+$ i $C_2^+$

Contesteu les següents preguntes:

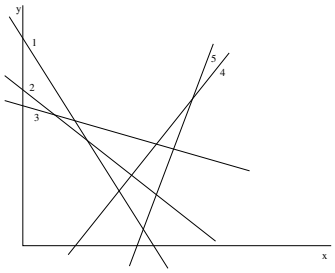
(a) Demostreu que l'algorisme CLIQUE sempre retorna un subgraf de $G$ que és un clique.

(b) Doneu una expressió asimptòtica del nombre de passos de l'algorisme CLIQUE.

(c) Doneu un exemple d'un graf $G$ on l'algorisme CLIQUE retorna un clique que no és de grandària màxima.

(d) Creieu que és fàcil modificar CLIQUE de manera que sempre done el clique màxim, sense incrementar el temps pitjor de l'algorisme? Expliqueu la vostra resposta.

21. El problema de l'eliminació de superfícies ocultes és un problema important en informàtica gràfica. Si des de la teva perspectiva, en Pepet està davant d'en Ramonet, podràs veure en Pepet però no en Ramonet. Considereu el següent problema, restringit al pla. Us donen $n$ rectes no verticals al pla, $L_1, \ldots, L_n$, on la recta $L_i$ ve especificada per l'equació $y = a_i x + b_i$. Assumim, que no hi han tres rectes que es creuen exactament al mateix punt. Direm que $L_i$ és *maximal* en $x_0$ de la coordenada $x$, si per qualsevol $1 \leq j \leq n$ amb $j \neq i$ tenim que $a_i x_0 + b_i > a_j x_0 + b_j$. Direm que $L_i$ és *visible* si té algun punt maximal.




Donat com a entrada un conjunt de $n$ rectes $\mathcal{L} = \{L_1, \ldots, L_n\}$, doneu un algorisme que, en temps $O(n \lg n)$, torne las rectes no visibles. A la figura de sobre teniu un exemple amb $\mathcal{L} = \{1, 2, 3, 4, 5\}$. Totes les rectes excepte la 2 són visibles (considerem rectes infinites).

22. Suposeu que sou consultors per a un banc que està molt amoïnat amb el tema de la detecció de fraus. El banc ha confiscat $n$ targetes de crèdit que se sospita han estat utilitzades en negocis fraudulents. Cada targeta conté una banda magnètica amb dades encriptades, entre elles el número del compte bancari on es carrega la tageta. Cada targeta es carrega a un únic compte bancari, però un mateix compte pot tenir moltes targetes. Direm que dues targetes són *equivalents* si corresponen al mateix compte.

    És molt difícil de llegir directament el número de compte d'una targeta intel·ligent, però el banc té una tecnologia que donades dues targetes permet determinar si són equivalents.

    La qüestió que el banc vol resoldre és la següent: donades les $n$ targetes, volen conèixer si hi ha un conjunt on més de $n/2$ targetes són totes equivalents entre si. Suposem que les úniques operacions possibles que pot fer amb les targetes és connectar-les de dues en dues, al sistema que comprova si són equivalents.

    Doneu un algorisme que resolgui el problema utilitzant només $O(n \lg n)$ comprovacions d'equivalència entre targetes. Sabríeu com fer-ho en temps lineal?

23. Donada una taula $A$ amb $n$ registres, on cada registre conté un enter de valor entre 0 i $2^n$, i els continguts de la taula estan desordenats, dissenyeu un algorisme lineal per a obtenir una llista ordenada dels elements a $A$ que tenen valor més gran que els $\log n$ elements més petits a $A$, i al mateix temps, tenen valor més petit que els $n - 3\log n$ elements més grans a $A$.

24. Tenim un taula $T$ amb $n$ claus (no necessàriament numèriques) que pertanyen a un conjunt totalment ordenat. Doneu un algorisme $O(n + klogk)$ per a ordenar els $k$ elements a $T$ que són els més petits d'entre els més grans que la mediana de les claus a $T$.

25. 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 $G[V']$.

26. 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.

27. 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,

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

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 $X$.

28. **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 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 $n$.

29. 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 $T$.

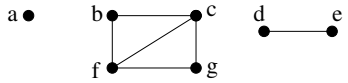
(a) Demostreu la correctesa o trobeu un contraexemple pels enunciats següents:

- *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 $\leq b$.

30. 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

31. Donat un graf no dirigit $G = (V, E)$ amb $n$ vèrtexs i $m$ arestes, dissenyeu un algorisme que utilitzant UNION-FIND ( tal com s'ha explicat a classe) ens digui si $G$ és o no connex i en cas de que no ho sigui, puguem averigus a quina component connexa pertany un vèrtex $v$ donat. Quina és la complexitat del vostre algorisme?. Apliqueu el vostre algorisme al següent exemple




El problema de trobar la connectivitat dún graf no dirigit té complexitat $O(n + m)$ utilitzant DFS o BFS, en quins casos UNION-FIND és més ràpid?

32. 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 $G$.

33. 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.

34. **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.

35. 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.

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

00110001011111011001110111011000001001110100101 01

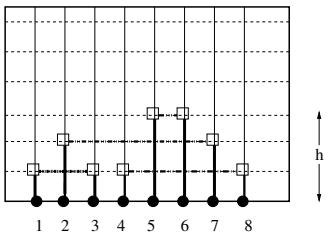
Donades les freqüències de les lletres que es donen a la taula de sota, obtingueu els codis de Huffmann i feu-los servir per decodificar el títol. Tingueu en compte que, quan hi ha múltiples eleccions, s'han d'agafar les dues primeres (d'esquerra a dreta, a l'ordre donat a la taula). Heu d'assignar a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

37. Tenim un alfabet $\Sigma$ 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.)

38. Una tribu de matemàtics ha decidit utilitzar un alfabet molt concís de 3 símbols (més el blanc $\flat$) per a comunicar-se entre ells. Utilitzant cadenes dels símbols $\{+, -, \star\}$ els matemàtics codifiquen totes els paraules que necessiten per a comunicar-se. Donada la seqüència $+ - - + \star\flat \star + \star - \star \flat\star$, trobeu la freqüència de cada un dels símbols $\{+, -, \star, \flat\}$. Dibuixeu l'arbre de Huffman. Quina és la compressió màxima que podeu obtenir utilitzant Huffman? Quin és el guany de compressió respecte a utilitzar una compressió de longitud fixada? (i.e. utilitzant $\{0, 1\}^2$). Tingueu en compte que els matemàtics es comuniquen entre si utilitzant Internet, per tant, abans d'enviar el missatges, l'ordinador converteix els símbols del seu alfabet en cadenes binàries via ASCII.

39. 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.

40. 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

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

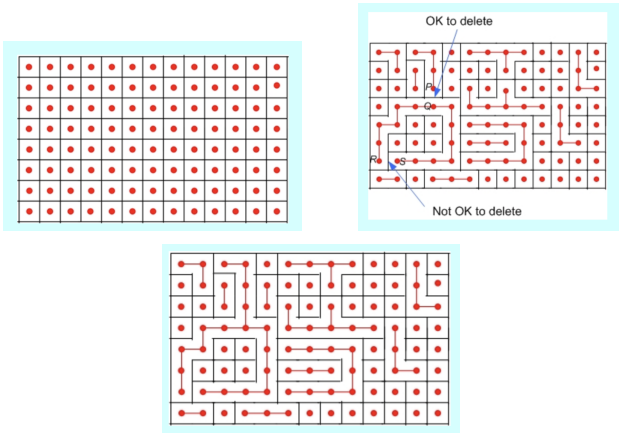
Describiu un algorisme de cost $O(n)$ per a obtenir un centre de $T$.

41. 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.

42. Considereu el següent algoritme de generació de laberint aleatori basat en una graella en el que ens donen dos cel.las $x$ e $y$ a les cantonades.

Comenceu amb cada vèrtex a la xarxa $n \times n$ formant el seu propi conjunt
  identificats amb números en $[0, n^2 - 1]$
Crea totes les parets interiors i exteriors
**while** $x$ i $y$ no estan al mateix conjunt
  Trieu una paret interior a l'atzar
  **if** els vèrtex adjacents a aquesta paret estan en diferents conjunts
    traieu la paret i feu una UNIO
  **endif**
**endwhile**

A la figura següènt teniu un exemple de l'estat inicial. Seguit d'una il.lustració de casos en els que el if intermedi és cert/fals i del resultat després de la UNIO en el cas en que es pot..




Expliqueu el següent:

(a) Què representa un conjunt a mida que l'algorisme progressa?

(b) Per què no deixem de realitzar UNION una vegada que la cèl·la inicial i la cel·la final estan en la mateixa classe d'equivalència?

(c) Com garanteix l'algorisme que només hi haurà una ruta des de la cel·la inicial fins a la cel·la final?

(d) Exactament quants UNION es duen a terme? Expliqueu la vostra resposta.

43. Donat com a entrada un graf no dirigit $G = (V, E)$, definim el problema de GST (graf sense triangles) com el problema de seleccionar el màxim nombre d'arestes $E$, de manera que el graf $G' = (V, E')$ no contingui cap triangle (i.e. no hi han 3 vèrtexs $x, y, z$ tal que $(x, y), (x, z)$ i $(y, z)$ siguin arestes a $G'$). Aquest problema és NP-hard. Dissenyeu un algorisme que done una 2-aproximació al problema. Quina és la complexitat del vostre algorisme?.

44. El problema del Tall Maxim (MAX-CUT). Donat un graf no dirigit $G = (V, E)$, el problema del maximum cut (MAX-CUT) es trobar la partició $S \cup \bar{S}$ de $V$ tal que maximitze el nombre d'arestes entre $S$ i $\bar{S}$ ($C(S, \bar{S})$ ,on $S \subseteq V$ i $\bar{S} = V - S$. La maximització entre totes les possibles particiones de $V$. El problema del MAX-CUT és NP-hard en general. Donat $G = (V, E)$ considereu el següent

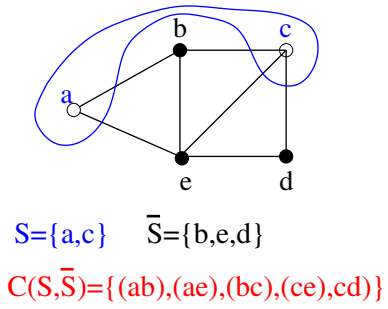



Figura 1: Exemple de MAX-CUT, amb cost òptim 5

algorisme golafre (greedy) pel problema del MAX-CUT:
Ordeneu els vèrtexs en ordre decreixent respecte al seu grau (nombre veïns). Considereu el resultat de l'ordenació s'escriu com a $(v_1, v_2, \ldots, v_n)$.
Inicialment tenim $S = \emptyset = \bar{S}$. Al primer pas $v_1 \in S$. Al pas $i$-èsim col.loquem $v_i$ a $S$ o $\bar{S}$ de manera que maximitze $|C(S, \bar{S})|$.

(a) Demostreu la correctessa i doneu la complexitat de l'algorisme golafre?

(b) Demostreu que aquest algorisme golafre és una 2-aproximació al MAX-CUT

45. 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 $O(n)$.