# Problemes 1

1.1. Un vector $A[n]$ conté tots els enters entre 0 i $n$ excepte un.

(a) Dissenyeu un algorisme que, utilitzant un vector auxiliar $B[n+1]$, detecti l'enter que no és a $A$, i ho faci en $O(n)$ passos.

(b) Suposem ara que $n = 2^k - 1$ per a $k \in \mathbb{N}$ i que els enters a $A$ venen donats per la seva representació binària. En aquest cas, l'accés a cada enter no és constant, i llegir qualsevol enter té un cost $\lg n$. L'única operació que podem fer en temps constant es "recuperar" el $j$-èsim bit de l'enter a $A[i]$. Dissenyeu un algorisme que, utilitzant la representació binària per a cada enter, trobi l'enter que no és a $A$ en $O(n)$ passos.

1.2. El coeficient de Gini és una mesura de la desigualtat ideada per l'estadístic italià Corrado Gini. Normalment s'utilitza per mesurar la desigualtat en els ingressos, dins d'un país, però pot utilitzar-se per mesurar qualsevol forma de distribució desigual. El coeficient de Gini és un nombre entre 0 i 1 , on 0 es correspon amb la perfecta igualtat (tots tenen els mateixos ingressos) i on el valor 1 es correspon amb la perfecta desigualtat (una persona té tots els ingressos i els altres cap).

Formalment, si $r = (r_1, \ldots, r_n)$, amb $n > 1$, és un vector de valors no negatius, el *coeficient de Gini* es defineix com:

$$G(r) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \mid r_i - r_j \mid}{2(n-1) \sum_{i=1}^{n} r_i}.$$

Proporcioneu un algorisme eficient per calcular el coeficient de Gini donat el vector $r$.

1.3. Per a cadascú dels algorismes, digueu quin és el temps en cas pitjor, quan l'entrada és un enter positiu $n > 0$.

(a)
```
for i = 1 to n do
    j = i
    while j < n do
        j = 2 * j
```

(b)
```
for i = 1 to n do
    j = n
    while i * i < j do
        j = j - 1
```

(c)
```
for i = 1 to n do
    j = 2
    while j < i do
        j = j * j
```

(d)
```
i = 2
while (i * i < n) i (n  mod i ≠ 0)  do
    i = i + 1
```

1.4. El problema 2SAT té com a entrada un conjunt de clàusules, on cada clàusula és la disjunció (OR) de dos literals (un literal és una variable booleana o la negació d'una variable booleana). Volem trobar una manera d'assignar un valor cert o fals a cadascuna de les variables perquè totes les clàusules es

satisfaguin - és a dir, hi hagi al menys almenys un literal cert a cada clàusula. Per exemple, aquí teniu una instància de 2SAT:

$$(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_1 \vee x_2) \wedge (\neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_4).$$

Aquesta instància és satisfactible: fent $x_1, x_2, x_3, x_4$ cert, fals, fals i cert, respectivament.

El propòsit d'aquest problema és conduir-vos a una manera de resoldre 2SAT de manera eficient reduint-ho al problema de trobar les components connexes fortes d'un graf dirigit. Donada una instància $F$ de 2SAT amb $n$ variables i $m$ clàusules, construïm un graf dirigit $G_F = (V, E)$ de la següent manera.

- $G_F$ té $2n$ nodes, un per a cada variable i un per a la seva negació.
- $G_F$ té $2m$ arcs: per a cada clàusula $(\alpha \vee \beta)$ de $F$ (on $\alpha, \beta$ són literals), $G_F$ té un arc des de la negació d'$\alpha$ a $\beta$, i un de la negació de $\beta$ a $\alpha$.

Tingueu en compte que la clàusula $(\alpha \vee \beta)$ és equivalent a qualsevol de les implicacions $\neg\alpha \Rightarrow \beta$ o $\neg\beta \Rightarrow \alpha$. En aquest sentit, $G_F$ representa totes les implicacions directes en $F$.

(a) Realitzeu aquesta construcció per a la instància de 2SAT indicada amunt.

(b) Demostreu que si $G_F$ té una component connexa forta que conté $x$ i $\neg x$ per a alguna variable $x$, llavors no és satisfactible.

(c) Ara demostreu la inversa: és a dir, que si no hi ha cap component connexa forta que contingui tant un literal com la seva negació, llavors la instància ha de ser satisfactible.

(d) A la vista del resultat previ, hi ha un algorisme de temps lineal per resoldre 2SAT?

1.5. En una festa, un convidat es diu que és una celebritat si tothom el coneix, però ell no coneix a ningú (tret d'ell mateix). Les relacions de coneixença donen lloc a un graf dirigit: cada convidat és un vèrtex, i hi ha un arc entre $u$ i $v$ si $u$ coneix a $v$. Doneu un algorisme que, donat un graf dirigit representat amb una matriu d'adjacència, indica si hi ha o no cap celebritat. En el cas que hi sigui, cal dir qui és. El vostre algorisme ha de funcionar en temps $O(n)$, on $n$ és el nombre de vèrtexs.

1.6. Llisteu les següents funcions en ordre *creixent*, és a dir, si l'ordre és $f_1; f_2; \ldots$, aleshores $f_2 = \Omega(f_1)$; $f_3 = \Omega(f_2)$; etc.

$$(\log n)^{100},\ n \log n,\ 3^n,\ \tfrac{n^2}{\log n},\ n2^n,\ 0.99^n,\ n^3,\ \sqrt{n}.$$

1.7. Digueu si cadascuna de les afirmacions següents són certes o falses (i per què).

(a) Asimptòticament $(1 + o(1))^{\omega(1)} = 1$

(b) Si $f(n) = (n + 2)n/2$ aleshores $f(n) \in \Theta(n^2)$.

(c) Si $f(n) = (n + 2)n/2$ aleshores $f(n) \in \Theta(n^3)$.

(d) $n^{1.1} \in O(n(\lg n)^2)$

(e) $n^{0.01} \in \omega((\lg n)^2)$

1.8. Digueu si la següent demostració de

$$\sum_{k=1}^{n} k = O(n)$$

és certa o no (i justifiqueu la vostra resposta).
**Demostració:** Per a $k = 1$, tenim $\sum_{k=1}^{1} k = 1 = O(1)$. Per hipòtesi inductiva, assumim $\sum_{k=1}^{n} k = O(n)$, per a una certa $n > 1$. Llavors, per a $n + 1$ tenim

$$\sum_{k=1}^{n+1} k = n + 1 + \sum_{k=1}^{n} k = n + 1 + O(n) = O(n).$$

1.9. Un graf dirigit és *fortament connex* quan, per cada parell de vèrtexs $u, v$, hi ha un camí de $u$ a $v$. Doneu un algorisme per determinar si un graf dirigit és fortament connex.

1.10. Un graf dirigit $G = (V, E)$ és *semiconnex* si, per qualsevol parell de vèrtexs $u, v \in V$, tenim un camí dirigit de $u$ a $v$ o de $v$ a $u$. Doneu un algorisme eficient per determinar si un graf dirigit $G$ és semiconnex. Demostreu la correctesa del vostre algorisme i analitzeu-ne el cost. Dissenyeu el vostre algorisme fent us d'un algorisme que us proporcioni les components connexes fortes del graf en temps $O(n + m)$.

1.11. Definim els *k-mers* com les subcadenes de DNA, amb grandària $k$. Per tant, per a un valor donat $k$ podem assumir que tenim una base de dades amb tots els $4^k$ $k$-mers. Una manera utilitzada en l'experimentació clínica per a identificar noves seqüències de DNA, consisteix a agafar mostres aleatòries de una cadena i determinar quins $k$-mers conté, on els $k$-mers es poden solapar. A partir d'aquest procés, podem reconstruir tota la seqüència de DNA.
Formalment, donada una cadena $w \in \{A, C, T, G\}^*$, i un enter $k$, sigui $S(w)$ el multi-conjunt de tots els $k$-mers que apareixen a $w$. Notem que $|S(w)| = |w| - k + 1$. El problema consisteix en, donat un multi-conjunt $C$ de $k$-mers, trobar la cadena de DNA $w$ tal que $S(w) = C$.

a Demostreu que hi ha una reducció polinòmica d'aquest problema al problema del camí Hamiltonià, que és NPC (utilitzeu els $k$-mers com a vèrtexs i el solapament entre $k$-mers com condició d'existència d'arestes).

b Demostreu que hi ha una reducció d'aquest problema al problema del camí Eulerià, que és a P (aquest cop, utilitzeu $k$-mers com a arestes dirigides).

c Vol dir això que aquest problema és a P i a NPC, i per tant P=NP?

1.12. 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) $T(n) = T(n - 10) + n$

1.13. Donat un graf no dirigit $G = (V, E)$ i un subconjunt de vèrtex $V_1$, el subgraf induït per $V_1$, $G[V_1]$ té com a vèrtex $V_1$ i con a arestes totes les arestes a $E$ que connecten vèrtexs en $V_1$. Un clique és un subgraf indiut per un conjunt $C$ on tots els vèrtexs estan connectats entre ells.

Considereu el següent algorisme de dividir-i-vèncer per al problema de *trobar un clique* en un graf no dirigit $G = (V, A)$.

```
CliqueDV(G)
1: Enumereu els vèrtexs V com 1, 2, ..., n, on n = |V|
2: Si n = 1 tornar V
3: Dividir V en V_1 = {1, 2, ..., ⌊n/2⌋} i V_2 = {⌊n/2⌋ + 1, ..., n}
4: Sigui G_1 = G[V_1] i G_2 = G[V_2]
5: C_1= CliqueDV(G_1) i C_2= CliqueDV(G_2)
6: C_1^+ = C_1 i C_2^+ = C_2
7: for u ∈ C_1 do
8:     if u està connectat a tots els vèrtexs a C_2^+ then
9:         C_2^+ = C_2^+ ∪ {u}
10: for u ∈ C_2 do
11:     if u està connectat a tots els vèrtexs a C_1^+ then
12:         C_1^+ = C_1^+ ∪ {u}
```

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

Contesteu les següents preguntes:

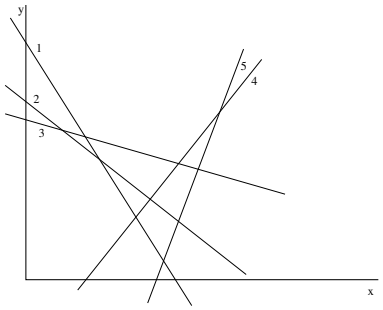
(a) Demostreu que l'algorisme CliqueDV sempre retorna un subgraf de $G$ que és un clique.

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

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

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

1.14. 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).

1.15. 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 $\lceil n/2 \rceil$ 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?

1.16. 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 $A$.

1.17. Tenim un taula $T$ amb $n$ claus (no necessàriament numèriques) que pertanyen a un conjunt totalment ordenat. Doneu un algorisme $O(n + k \log k)$ 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 $T$.

1.18. Com ordenar eficientment elements de longitud variable:

(a) Donada una taula d'enters, on els enters emmagatzemats poden tenir diferent nombre de dígits. Però sabem que el nombre total de dígits sobre tots els enters de la matriu és $n$. Mostreu com ordenar la matriu en $O(n)$ passos.

(b) Se us proporciona una sèrie de cadenes de caràcters, on les diferents cadenes poden tenir diferent nombre de caràcters. Com en al cas previ, el nombre total de caràcters sobre totes les cadenes és $n$. Mostreu com ordenar les cadenes en ordre alfabètic fent servir $O(n)$ passos. (Tingueu en compte que l'ordre desitjat és l'ordre alfabètic estàndard, per exemple, $a < ab < b$.)

1.19. Hi ha un concurs de TV amb $n$ participants on cada participant escull un enter entre 0 i 1000000. El premi és per als dos concursants que escullen els enters més propers. Dissenyeu un algorisme que, en temps lineal, li digui al presentador quins són els dos concursants guanyadors o l'indiqui que hi ha més d'un parell de concursants candidats a rebre el premi.

1.20. Tenim un conjunt de $2n$ valors tots diferents. Una meitat dels valors estan emmagatzemats a una taula $A$ i l'altra meitat a una taula $B$. Cadascuna de les dues taules està ordenada en ordre creixent i es troba a un ordinador diferent. No hi ha cap relació d'ordre entre els valors a $A$ i els valors a $B$. Volen trobar la mediana del total dels $2n$ valors. Doneu un algorisme amb cost $O(\lg n)$ que permeti obtenir la mediana sota la hipòtesis que només podeu fer crides de la forma $\mathsf{Element}(i, A)$ o $\mathsf{Element}(i, B)$, per $1 \leq i \leq n$, que retornen l'element $i$-ésim a $A$ o a $B$, respectivament (amb cost $O(1)$).

1.21. Tenim un vector $A[1, \ldots, n]$ no ordenat amb claus no necessàriament numèriques, però que pertanyen a un conjunt totalment ordenat de claus. Sigui $x_i$ el $2^i$-èsim element més petit en $A$. Doneu un algorisme per calcular la suma dels valors $x_i$, per $1 \leq 2^i \leq n$, en $\Theta(n)$ passos.