1. Suposem que tenim un vector A amb n nombres enters diferents, amb la propietat: existeix un únic índex p tal que els valors $A[1 \cdots p]$ estan en ordre creixent i els valors $A[p \cdots n]$ estan en ordre decreixent. Per exemple, en la següent vector tenim $n = 10$ i $p = 4$:

$$A = (2, 5, 12, 17, 15, 10, 9, 4, 3, 1)$$

Dissenyeu un algorisme eficient per trobar p donada una matriu A amb la propietat anterior.

Una solució: L'algorisme recursiu es descriu en l'Algorisme FindPeak. Donada la matriu A, la resposta s'obté amb una crida amb $i = 1$ i $j = n$. L'algorisme és una cerca binària, en cada pas, comparem els dos elements intermedis i veurem si estem en la part creixent o decreixent. El cas base ressòl el problema d'obtenir la posició del màxim, però per a una entrada amb mida constant.

```
function FindPeak(A, i, j)
    n = j − i + 1
    if n ≤ 5 then
        return PosMax(A, i, j)
    end if
    k = (i + j)/2
    if A[k] < A[k + 1] then
        return FindPeak(A, k + 1, j)
    else
        return FindPeak(A, i, k)
    end if
end function
```

Correctesa: Volem trobar l'índex p. Si $A[k] < A[k+1]$, sabem que $A[i] < \cdots < A[k]$ per $i < k$ i podem prescindir de forma segura els elements $A[i \cdots k]$. De la mateixa manera, si $A[k] > A[k+1]$, sabem que $A[k+1] > \cdots > A[j]$ per $j > k+1$ i podem descartar amb seguretat els elements $A[k+1 \cdots j]$. La posició de p coincideix amb la del valor màxim al vector, per tant el cas base és correcte.

Cost temporal: El cas base té cost constant. A cada pas, es redueix la mida del problema a la meitat i, a més, el cost de les operacions és constant. Així, tenim la recurrència $T(n) = T(n/2) + c$ per a alguna constant c. Sabem que, com a la cerca binària, $T(n) = O(\log n)$.

2. 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.

3. 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.

4. 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
    end while
end for
```

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

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

$j = j * j$
end while
end for

(d)
$i = 2$
while $(i * i < n)$ i $(n \mod i \neq 0)$ **do**
$i = i + 1$
end while

5. 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.

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

6. Digueu si són certes o no ho són (i per què) les afirmacions següents :

(a) $n^{1/2} = \Theta(n^{2/3})$
(b) $n! = o(n^n)$.
(c) $n! = \omega(2^n)$
(d) $\sum_{i=1}^{\sqrt{n}} i$ és $\Omega(n)$
(e) $\lceil \log n \rceil !$ és $O(n)$
(f) Per a tot $f(n)$ positiu, tenim $f(n) + o(f(n)) = \Theta(f(n))$
(g) Sigui $f(n) = \sqrt{n}$ i $g(n) = n \cdot (n \mod 100)$, aleshores $f(n) = O(g(n))$

7. Quantes vegades, com a funció de n, escriu 'A' el programa següent

Funció $f(n)$
si $n > 1$
escriure ('A')
$f(n/2)$
$f(n/2)$

Podeu assumir que n és un enter no negatiu.

8. Donat el següent algorisme per conduir un robot, que té com a entrada un enter no negatiu n,

Funció CAMINAR-(n)
si $n \leq 1$ retornar i aturar-se
caminar nord n metres
caminar est n metres
caminar sud n metres
caminar oest $n - 1$ metres
caminar nord 1 metre
CAMINAR $(n - 2)$

Sigui $C(n)$ el nombre de metres que el robot camina quan executem l'algorisme amb paràmetre n. Doneu una estimació asimptòtica del valor de $C(n)$.

9. Sigui A una matriu real $m \times n$. Diem que A té la propietat de Monge si per a tots els enters i, j, k i l tals que $1 \leq i < k \leq m$ i $1 \leq j < l \leq n$ tenim

$$A[i, j] + A[k, l] \leq A[i, l] + A[k, j].$$

(És a dir, si escollim 2 files i 2 columnes de A, aquestes intersecten en 4 elements. La propietat Monge requereix que la suma dels elements més a dalt-esquerra i l'element més a baix-dreta sigui menor o igual a la suma dels elements més a baix-esquerra i l'element més a dalt-dreta.) Per exemple, la següent matriu té la propietat Monge:

$$\begin{bmatrix} 10 & 17 & 13 & 28 & 23 \\ 17 & 22 & 16 & 29 & 23 \\ 24 & 28 & 22 & 34 & 24 \\ 11 & 13 & 6 & 17 & 7 \\ 45 & 44 & 32 & 37 & 23 \\ 36 & 33 & 19 & 21 & 6 \\ 75 & 66 & 51 & 53 & 34 \end{bmatrix}$$

(a) Demostreu que una matriu és Monge si, i únicament si, totes les files i columnes adjacents tenen la propietat Monge. Formalment: demostreu que per a tot $i = 1, 2, \ldots, m-1$ i $j = 1, 2, \ldots, n-1$ teniu

$$A[i, j] + A[i+1, j+1] \leq A[i, j+1] + A[i+1, j].$$

(b) Per a cada fila i, $1 \leq i \leq m$, sigui $f_A(i)$ l'índex de la columna que conté l'element mínim a la fila i, en cas d'empat escolliu el de mes a l'esquerra. Demostreu que, per a qualsevol matriu $m \times n$, A que té la propietat Monge, es compleix: $f_A(1) \leq f_A(2) \leq \cdots \leq f_A(m)$.

(c) Donada una matriu A amb la propietat Monge, considereu el següent algorisme de dividir i vèncer per a computar $f_A(i)$:

i. Construïu la submatriu A' de A, que consisteix de les files amb índex senar (les files d'A estan enumerades d'1 fins a m).

ii. Recursivament determinar $f_{A'}(i)$ per a cada fila i

iii. Computar $f_A(i)$ per a les files d'A, utilitzant els valors obtinguts al pas previ.

Explicar com dissenyaríeu el darrer pas de l'algorisme de DiV. Quina és la seva complexitat?

(d) Escriure una recurrència per l'algorisme de dividir i vèncer. Quina és la complexitat de tot l'algorisme? (Podeu assumir que al primer pas de l'algorisme, podeu construir A' a partir de A, en $O(m)$ passos)

10. El quadrat d'un graf $G = (V, E)$ és un altre graf $G' = (V, E')$ on $E' = \{(u, v) | \ \exists w \in V, (u, w) \in E \wedge (w, v) \in E\}$. Dissenyeu i analitzeu un algorisme que, donat un graf representat amb una matriu d'adjacència, calculi el seu quadrat. Feu el mateix amb un graf representat amb llistes d'adjacència.

11. 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.

12. Es diu que un vèrtex d'un graf connex és un *punt d'articulació* del mateix si, en suprimir aquest vèrtex i totes les arestes que hi incideixen, el graf resultant deixa de ser connex. Per exemple, un graf en forma d'anell no té cap punt d'articulació mentre que tot node que no sigui fulla d'un arbre és punt d'articulació. Dissenyeu un algorisme que, donat un graf connex no dirigit $G = (V, E)$, indiqui quins vèrtexs del graf són punts d'articulació. Calculeu el seu cost. Indiqueu quina implementació del graf és la què proporciona un algorisme més eficient.

13. 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.

14. 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.

15. Recordeu que un graf no dirigit $G = (V, E)$ és **bipartit** si existeix un partició de $V = V_1 \cup V_2$ ($V_1 \neq \emptyset \neq V_2$) de manera totes les arestes a E són entre V_1 i V_2. Donat un graf $G = (V, E)$ doneu un algorisme per a determinar si el graf és bipartit. Quina és la seva complexitat?

16. 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.

    Volem resoldre un problema més senzill. 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 $|S(w)| = |w| - k + 1$. *Donat un multi-conjunt C de k-mers, trobar, si n'hi ha, la cadena de DNA w tal que $S(w) = C$.*

Trobeu un algorisme per resoldre aquest problema i analitzeu la seva complexitat.

(Ajut: A partir de qualsevol entrada al problema de la seqüenciació, hem de construir en temps polinòmic un graf dirigit G que sigui entrada al problema del camí Eulerià, i tal que existeixi una solució al problema de la seqüenciació sii G té un camí Eulerià. Podeu considerar com a vèrtexs els $k-1$-mers que apareixen en el multiconjunt. Heu de decidir com definir les arestes $(u, \vec{v})$ i demostrar la correctesa de la vostra construcció.)

17. Tenim un conjunt de robots que es mouen en un edifici, cadascun d'ells és equipat amb un transmissor de ràdio. El robot pot utilitzar el transmissor per comunicar-se amb una estació base. No obstant això, si els robots són massa a prop un de l'altre hi ha problemes amb la interferència entre els transmissors. Volem trobar un pla de moviment dels robots, de manera que puguin procedir al seu destí final, sense perdre mai el contacte amb l'estació base.

    Podem modelar aquest problema de la següent manera. Se'ns dóna un graf $G = (V, E)$ que representa el plànol d'un edifici, hi ha dos robots que inicialment es troben en els nodes a i b. El robot en el node a voli viatjar a la posició c, i el robot en el node b vol viatjar a la posició d. Això s'aconsegueix per mitjà d'una planificació: a cada pas de temps, el programa especifica que un dels robots es mou travessant una aresta. Al final de la planificació, els dos robots han d'estar en les seves destinacions finals.

    Una planificació és *lliure* d'interferència si no hi ha un punt de temps en el qual els dos robots ocupen nodes que es troben a distància menor de r, per a un valor determinat del paràmetre r.

    Doneu un algorisme de temps polinomial que decideixi si hi ha una planificació lliure donats, el graf, les posicions inicials i finals dels robots i el valor de r.

    Una solució. Per resoldre el problema considerarem l'espai de configuracions on els robots es poden moure. És a dir, el conjunt de parells de posicions que estan a distància més gran o igual que r:

    $$C = \{(u, v) \mid u, v \in V \text{ i } d(u, v) \geq r\}$$

    Podem considerar la relació entre configuracions definida pels moviments permesos. Així tenim

    $$M = \{((u, v), (u', v')) \mid (u, v), (u', v') \in C \text{ i } ((u = u' \text{ i } (v, v') \in E) \text{ o } (v = v' \text{ i } (u, u') \in E))\}.$$

    A l'espai de configuracions podem considerar el graf $\mathcal{G} = (C, M)$ on dos configuracions son veïnes si i només si un del robots pot canviar de posició sense interferir amb la posició de l'altre.

Els robots són inicialment a la configuració (a, b) i s'han de desplaçar amb moviments vàlids fins a la configuració (c, d). Això serà possible únicament si hi ha un camí de (a, b) a (c, d). D'acord amb el raonament anterior tenim que comprovar hi ha un camí entre dos vèrtexs a $\mathcal{G}$. Podem detectar-ho amb un BFS en tems $O(|C| + |M|)$.

Per calcular el cost hem de tenir en compte la mida de l'entrada. Si $G = (V, E)$ i $n = |V|$ i $m = |E|$, tenim $|C| \leq n^2$ i $|M| \leq 2m$. Suposant que ens donen G mitjançant llistes d'adjacència la mida de l'entrada és $n + m$. Construir una descripció de $\mathcal{G}$ mitjançant llistes d'adjacència té cost $O(n^2 + m)$. Fer un BSF sobre $\mathcal{G}$ té cost $O(n^2 + m)$. El cost total es $O(n^2 + m)$ però $m \leq n^2$. Llavors l'algorisme proposat té cost $O(n^2)$.