ESTRUCTURES DE DADES I ALGORISMES
COL·LECCIÓ DE PROBLEMES

Albert Atserias Amalia Duch Jordi Petit
Enric Rodríguez-Carbonell
(Editors)

27 de juliol de 2021

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

Taula de continguts

1

Anàlisi d'algorismes

1. **Sumatoris fins a la sopa**. Demostreu les igualtats següents:
   (a) $\sum_{i=1}^{n} i = n(n+1)/2$.
   (b) $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$.
   (c) $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

2. **Temps exacte**. Suposem que les assignacions, els increments i les comparacions d'enters triguen 15 μs, 21 μs i 34 μs, respectivament, i que la resta d'instruccions (increments del *program counter*, salts, etc.) triguen un temps negligible. Doneu la fórmula que expressa, en funció del valor de l'enter d'entrada n, el temps d'execució $T(n)$ de l'algorisme següent:

```
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
        ++s;
}   }
```

3. **Triga la tira**. Per a cada funció $f(n)$ i temps t de la taula següent, determineu la talla més gran que es pot resoldre en temps t d'un problema que necessita un temps de $f(n)$ microsegons (μs) per executar-se sobre una entrada de talla n.

	1 segon	1 minut	1 hora	1 dia	1 mes	1 any	1 segle
$\log n$							
$\sqrt{n}$							
n							
n^2							
n^3							
2^n							

4. **Ordenació per selecció en casos pitjor i millor**. Suposem que les assignacions, els increments d'una unitat, les comparacions i els accessos a vectors d'enters triguen

15 μs, 21 μs, 34 μs i 12 μs, respectivament, i que la resta d'instruccions (increments del *program counter*, salts, etc.) triguen un temps negligible. Per al programa següent, doneu la fórmula que expressa, en funció de la mida n del vector d'enters d'entrada $A[0,\ldots,n-1]$, el temps d'execució $T_W(n)$ en el cas pitjor segons el contingut de A (és a dir, un contingut de A que el fa trigar el màxim de temps). Repetiu per al temps d'execució $T_B(n)$ en el cas millor segons el contingut de A (és a dir, un contingut de A que el fa trigar el mínim de temps).

```
for (int i = 0; i < n; ++i) {
    int k = i;
    for (int j = i; j < n; ++j) {
        if (A[j] < A[k]) k = j;
    }
    int aux = A[i]; A[i] = A[k]; A[k] = aux;
}
```

5. **Polinomis vs. Exponencials**. L'objectiu d'aquest problema és demostrar que si $a > 0$ i $b > 1$ són dos reals fixats, llavors $n^a = O(b^n)$ però $n^a \neq \Omega(b^n)$. Per dos reals $r > 0$ i $s > 1$ fixats, considereu la funció real $f(n) = \frac{n^r}{s^n}$ en l'interval $(0, +\infty)$.
   (a) Calculeu la derivada $f'(n)$ de $f(n)$.
   (b) Resoleu l'equació $f'(n) = 0$ en n i demostreu que té una única solució (que d'ara en endavant anomenarem $n_{r,s}$).
   (c) Analitzeu el signe de $f'(n)$ per concloure que $f(n_{r,s})$ és el màxim de $f(n)$.
   (d) Concloeu-ne que $n^r = O(s^n)$. Quines constants c i n_0 heu escollit per l'O?
   (e) Fent ús d'això proveu que, de fet, $\lim_{n\to+\infty} \frac{n^a}{b^n} = 0$ per a tots reals $a > 0$ i $b > 1$.
   (f) Concloeu-ne que $n^a = O(b^n)$ però $n^a \neq \Omega(b^n)$ per a tots reals $a > 0$ i $b > 1$.

6. **Logaritmes vs. Arrels**. Utilitzeu les conclusions del problema "Polinomis vs. Exponencials" per veure que per a qualsevol parell de constants reals $a, b \geq 1$ tenim que $(\ln n)^a = O(n^{1/b})$ però $(\ln n)^a \neq \Omega(n^{1/b})$.

7. **Polinomis i logaritmes**. Demostreu les propietats asimptòtiques següents:
   (a) Tot polinomi $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 n^0$ amb $a_k > 0$ és $\Theta(n^k)$.
   (b) Per a qualssevol bases $a, b > 1$, es té que $\log_a n$ és $\Theta(\log_b n)$.

8. **Ordres de creixement I**. Considereu les funcions següents: $5n \ln n$, $4n\sqrt{n}$, $\log_\pi(n^n)$, $n/\log_2 n$, $n/\ln n$, $3\ln(7^n)$. Ordeneu-les, de menor a major, segons el seu creixement asimptòtic. Si dues o més funcions tenen el mateix grau de creixement, indiqueu-ho.

9. **Ordres de creixement II**. Agrupeu les funcions següents de manera que $f(n)$ i $g(n)$ pertanyin al mateix grup si i només si $f(n) \in \Theta(g(n))$, i enumereu els grups segons el seu ordre de creixement: n, $\sqrt{n}$, $n^{1.5}$, n^2, $n \log n$, $n \log\log n$, $(\log\log n)^2$, $n \log^2 n$, $n^3/(n-1)$, $n^{\log n}$, $n \log(n^2)$, $21n$, 2^n, $2^{\sqrt{n}}$, $2^{\sqrt{\log n}}$, $2^{n/2}$, 37, 2^{2^n}, $2/n$ i $n^2 \log n$.

10. **Aparella la dotzena**. Aparella cadascun dels 12 ordres de creixement següents amb la seva fórmula: 1) logarítmic, 2) polilogarítmic, 3) radical (o sublineal), 4) lineal,

5) quasilineal, 6) quadràtic, 7) polinòmic, 8) exponencial polilogarítmic (o quasipolinòmic), 9) exponencial radical (o exponencial sublineal, o subexponencial), 10) exponencial lineal (o exponencial), 11) factorial, 12) exponencial polinòmic (o quasiexponencial).

a) $\Theta(n^2)$,

b) $\Theta(2^{cn})$ per una constant $c > 0$,

c) $\Theta(\log n)$,

d) $\Theta(2^{(\log n)^c})$ per una constant $c \geq 1$,

e) $\Theta(n \log n)$,

f) $\Theta(n^c)$ per una constant $c \geq 1$,

g) $\Theta(2^{n^c})$ per una constant $c \geq 1$,

h) $\Theta(n)$,

i) $\Theta(n!)$,

j) $\Theta(n^c)$ per una constant $0 < c < 1$,

k) $\Theta((\log n)^c)$ per una constant $c \geq 1$,

l) $\Theta(2^{n^c})$ per una constant $0 < c < 1$.

11. **Simplificant expressions**. Per a cadascuna de les funcions següents, doneu el seu ordre de magnitud de la forma més senzilla possible utilitzant la notació asimptòtica Θ. Per exemple, $3n^2 = \Theta(n^2)$.

(a) $23n^2 + 3n - 4 = \cdots$
(b) $42 = \cdots$
(c) $2n = \cdots$
(d) $3n = \cdots$
(e) $2^n = \cdots$
(f) $3^n = \cdots$
(g) $0.001n \log n + 1000n = \cdots$
(h) $n^2 + n - \sqrt{n} = \cdots$
(i) $n/\log n + 1/n = \cdots$
(j) $n! + 2^n = \cdots$
(k) $n^n + n! = \cdots$

12. **Fors, i fors dins de fors**. De cadascun dels fragments de codi següents, analitzeu el seu cost.

```
// Fragment 1
int s = 0;
for (int i = 0; i < n; ++i) {
  ++s;
}

// Fragment 2
int s = 0;
for (int i = 0; i < n; i += 2) {
```

```
    ++s;
}

// Fragment 3
int s = 0;
for (int i = 0; i < n; ++i) {
    ++s;
}
for (int j = 0; j < n; ++j) {
    ++s;
}

// Fragment 4
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
        ++s;
} }

// Fragment 5
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i; ++j) {
        ++s;
} }

// Fragment 6
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = i; j < n; ++j) {
        ++s;
} }

// Fragment 7
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
        for (int k = 0; k < n; ++k) {
            ++s;
} } }

// Fragment 8
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i; ++j) {
        for (int k = 0; k < j; ++k) {
            ++s;
} } }
```

```
// Fragment 9
int s = 0;
for (int i = 1; i ≤ n; i *= 2) {
    ++s;
}

// Fragment 10
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i * i; ++j) {
        for (int k = 0; k < n; ++k) {
            ++s;
}   }   }

// Fragment 11
int s = 0;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i * i; ++j) {
        if (j % i == 0) {
            for (int k = 0; k < n; ++k) {
                ++s;
}   }   }   }
```

13. **Pas de paràmetres**. Digueu quin és el cost de les instruccions (1), (2) i (3) del programa següent.

```
int funcio1 (vector<int>& v) { return v[0]; }
int funcio2 (vector<int> v) { return v[0]; }

int f (int n) {
    vector<int> v(n, 33);          // (1)
    int a = funcio1(v);            // (2)
    int b = funcio2(v);            // (3)
    return a + b;
}
```

14. **Os, Omegues, o Thetes**. Les frases següents es refereixen a l'algorisme d'ordenació per inserció i les X amaguen una ocurrència de O, Ω o Θ. Per a cadascuna de les frases, digueu quina o quines opcions per $X \in \{O, \Omega, \Theta\}$ fan que l'afirmació sigui certa, quina o quines opcions fan que l'afirmació sigui falsa, i justifiqueu les vostres respostes.

    (a) El cas pitjor és $X(n^2)$.

    (b) El cas pitjor és $X(n)$.

    (c) El cas millor és $X(n^2)$.

    (d) El cas millor és $X(n)$.

    (e) Per a tota distribució de probabilitat, el cas mitjà és $X(n^2)$.

    (f) Per a tota distribució de probabilitat, el cas mitjà és $X(n)$.

    (g) Per a alguna distribució de probabilitat, el cas mitjà és $X(n \log n)$.

15. **Algorismes d'ordenació elementals**. Digueu quan triguen els algorismes d'ordenació elementals (selecció, inserció i bombolla) quan l'entrada es troba...

    (a) permutada a l'atzar,

    (b) ordenada,

    (c) ordenada del revés,

    (d) amb tots els elements iguals.

16. **L'Omer calcula medianes adormit**. Malgrat anar tot el dia adormit, l'Omer ha escrit un algorisme *correcte* que retorna la mediana d'una taula de n elements. Vosaltres no heu vist l'algorisme de l'Omer. Tanmateix, podeu assegurar que només una de les afirmacions següents és certa. Digueu raonadament quina.

    (a) El cost de l'algorisme de l'Omer és per força $\Theta(n)$.

    (b) El cost de l'algorisme de l'Omer és per força $\Omega(n)$.

    (c) El cost de l'algorisme de l'Omer en el cas mitjà és per força $O(\log n)$.

    (d) El cost de l'algorisme de l'Omer en el cas pitjor és per força $\Theta(n \log n)$.

17. **Sumant matrius**. Escriviu l'algorisme clàssic per sumar dues matrius $n \times n$. Expliqueu perquè aquest algorisme de cost $O(n^2)$ és lineal.

    Demostreu que qualsevol algorisme per sumar dues matrius $n \times n$ requereix $\Omega(n^2)$ passos.

18. **Un misteri**. Digueu què fa l'algorisme següent i quin és el seu cost en funció de n.

```
int misteri(int n) {
    int f = 1;
    for (int i = 2; i ≤ n; ++i) f *= i;
    return f;
}
```

19. **Primalitat**. Els algorismes següents pretenen determinar si el natural n és un nombre primer o no. Digueu quins són correctes, quins no i quin és el seu cost en funció de n.

```
bool es_primer_v1(int n) {
    if (n ≤ 1) return false;
    for (int i = 2; i < n; ++i) if (n%i == 0) return false;
    return true;
}

bool es_primer_v2(int n) {
    if (n ≤ 1) return false;
    for (int i = 2; i*i < n; ++i) if (n%i == 0) return false;
    return true;
}

bool es_primer_v3(int n) {
    if (n ≤ 1) return false;
```

```
    for (int i = 2; i*i ≤ n; ++i) if (n%i == 0) return false;
    return true;
}

bool es_primer_v4(int n) {
    if (n ≤ 1) return false;
    if (n == 2) return true;
    if (n%2 == 0) return false;
    for (int i = 3; i*i ≤ n; i += 2) if (n%i == 0) return false;
    return true;
}
```

20. **Garbell d'Eratóstenes**. El programa següent implementa el garbell d'Eratóstenes per determinar, per a cada nombre entre 0 i n (inclosos), si aquest nombre és primer o no. Analitzeu el seu cost.

```
vector<bool> primers(int n) {
    vector<bool> p(n + 1, true);
    p[0] = p[1] = false;
    for (int i = 2; i*i ≤ n; ++i) {
        if (p[i]) {
            for (int j = 2*i; j ≤ n; j += i) p[j] = false;
    }   }
    return p;
}
```

21. **Subtractores**. Resoleu les recurrències subtractores següents:

    (a) $T(n) = T(n-1) + \Theta(1)$,

    (b) $T(n) = T(n-2) + \Theta(1)$,

    (c) $T(n) = T(n-1) + \Theta(n)$,

    (d) $T(n) = 2T(n-1) + \Theta(1)$.

22. **Divisores**. Resoleu les recurrències divisores següents:

    (a) $T(n) = 2T(n/2) + \Theta(1)$,

    (b) $T(n) = 2T(n/2) + \Theta(n)$,

    (c) $T(n) = 2T(n/2) + \Theta(n^2)$,

    (d) $T(n) = 2T(n/2) + O(1)$,

    (e) $T(n) = 2T(n/2) + O(n)$,

    (f) $T(n) = 2T(n/2) + O(n^2)$,

    (g) $T(n) = 4T(n/2) + n$,

    (h) $T(n) = 4T(n/2) + n^2$,

    (i) $T(n) = 4T(n/2) + n^3$,

    (j) $T(n) = 9T(n/3) + 3n + 2$,

(k) $T(n) = T(9n/10) + \Theta(n)$,

(l) $T(n) = 2T(n/4) + \sqrt{n}$,

23. **Exponenciació lenta i ràpida**. Analitzeu el cost de les funcions recursives següents per calcular x^n per a $n \geq 0$.

```
double potencia_1 (double x, int n) {
    if (n==0) {
        return 1;
    } else {
        return x*potencia_1(x,n-1);
}   }

double potencia_2 (double x, int n) {
    if (n==0) {
        return 1;
    } else if (n%2==0) {
        double y = potencia_2(x,n/2);
        return y*y;
    } else {
        double y = potencia_2(x,n/2);
        return y*y*x;
}   }

double potencia_3 (double x, int n) {
    if (n==0) {
        return 1;
    } else if (n%2==0) {
        return potencia_3(x,n/2) * potencia_3(x,n/2);
    } else {
        return potencia_3(x,n/2) * potencia_3(x,n/2) * x;
}   }
```

24. **Trobant l'x**. Un algorisme A té un cost donat per la recurrència $T_A(n) = 7T_A(n/2) + n^2$. Un algorisme competidor B té cost $T_B(n) = xT_B(n/4) + n^2$. Quin és l'enter x més gran per al qual B és asimptòticament millor que A?

25. **Trencant-ho a troços**. Calculeu el cost de l'algorisme següent:

```
double F (vector<double>& v, int i, int j) {
    int aux = (j-i+1)/4;
    if (aux>0) {
        int m1 = i-1+aux;
        int m2 = m1+aux;
        int m3 = m2+aux;
        return F(v,i,m1) + F(v,m1+1,m2) + F(v,m2+1,m3) + F(v,m3+1,j);
    } else {
        return i+j-1;
}   }
```

26. **Trencant-ho a troços II**. Considereu les dues funcions següents:

```
double A (vector<double>& v, int i, int j) {
    if (i < j) {
        int x = f(v, i, j);
        int m = (i + j) / 2;
        return A(v, i, m-1) + A(v, m, j) + A(v, i+1, m) + x;
    } else {
        return v[i];
}   }

double B (vector<double>& v, int i, int j) {
    if (i<j) {
        int x = g(v, i, j);
        int m1 = i + (j - i + 1) / 3;
        int m2 = i + (j - i + 1) * 2 / 3;
        return B(v, i, m1-1) + B(v, m1, m2-1) + B(v, m2, j) + x;
    } else {
        return v[i];
}   }
```

Suposant que el cost de la funció $f()$ és $\Theta(1)$ i que el cost de $g()$ és proporcional a la grandària del vector a processar (és a dir, $\Theta(j-i+1)$), quin és el cost asimptòtic de $A()$ i de $B()$ en funció de n?

Quina de les dues funcions escollirieu si fan el mateix?

27. **Recurrència genèrica**. El cost $T(n)$ d'un algorisme recursiu satisfà $T(n) = xT(n/4) + \Theta(\sqrt{n})$, amb $x > 0$. Digueu quin és el cost $T(n)$ tenint en compte que hi ha tres situacions possibles, segons l'interval en el qual cau la x.

28. **Màxim i mínim amb truc**. Considereu una taula T amb n elements.

    (a) Escriviu un algorisme que usi exactament $n-1$ comparacions per trobar l'element mínim de T.

    (b) Escriviu un algorisme que usi exactament $n-1$ comparacions per trobar l'element màxim de T.

    (c) Escriviu un algorisme que trobi els elements mínim i màxim de T amb només unes $\frac{3}{2}n$ comparacions.

29. **Canviant la variable**. Resoleu la recurrència $T(n) = T(\sqrt{n}) + 1$ tot utilitzant un canvi de variables. Assumiu que n és de la forma 2^{2^i} per un enter $i \geq 0$.

30. **Taula bidimensional biordenada**. Considereu una taula bidimensional de talla $n \times n$ on cada columna té els elements ordenats en ordre estrictament creixent de dalt a baix i cada fila té els elements ordenats en ordre estrictament creixent d'esquerra a dreta.

    (a) Doneu un algorisme de temps $\Theta(n)$ que, donat un element x, determini si x és o no dins de la matriu.

    (b) Doneu un algorisme de temps $\Theta(n)$ que, donat un element x, determini quants elements de la matriu són estrictament menors que x.

2

Dividir i vèncer

1. **Més rigor, si us plau**. Considereu el problema de, donada una taula ordenada T amb n elements i un element x, retorna -1 si x no és a T o un índex i tal que $T[i] = x$ altrament.

   Digueu què penseu dels tres fragments de codi següents (el fragment 1 es troba en una web de programadors, el fragment 2 és una adaptació del codi del llibre *Modern software development using Java* de Tymann i Schneider, i el fragment 3 és una adaptació del codi del llibre *Developing Java software* de Winder i Roberts):

```
// Fragment 1
int lookup1 (vector<int>& T, int x) {
    int l = 0;
    int r = T.size () - 1;
    while (l <= r) {
        int m = (l+r) / 2;
        if (x < T[m]) r = m;
        else if (x > T[m]) l = m + 1;
        else return m;
    }
    return -1;
}

// Fragment 2
int lookup2 (vector<int>& array, int target ) {
    int start = 0;
    int end = array . size ();
    int position = -1;
    while ( start <=end and position==-1) {
        int middle = ( start +end)/2;
        if ( target <array[middle]) end = middle - 1;
        else if ( target >array[middle]) start = middle + 1;
```

```
        else position = middle;
    }
    return position ;
}

// Fragment 3
int lookup2 (vector<int>& v, int o) {
    int hi = v.size ();
    int lo = 0;
    while (true) {
        int centre = (hi+lo)/2;
        if (centre==lo) {
            if (v[centre]==o) return centre ;
            else if (v[centre+1]==o) return centre+1;
            else return -1;
        }
        if (v[centre]<o) lo = centre ;
        else if (o<v[centre]) hi = centre ;
        else return centre ;
} }
```

2. **Rang**. Escriviu un algorisme de cost $\Theta(\log n)$ que, donada una taula ordenada T amb n elements diferents i dos elements x i y amb $x \leq y$, retorni el nombre d'elements en T que es troben entre x i y (x i y inclosos).

3. **Fusió amunt, fusió avall**. Ordeneu la taula $\langle 3, 8, 15, 7, 12, 6, 5, 4, 3, 7, 1\rangle$ amb l'algorisme d'ordenació per fusió recursiu i amb l'algorisme d'ordenació per fusió d'avall cap amunt.

4. **Ordenació per fusió**. Digueu quant triga l'ordenació per fusió quan l'entrada es troba...
   (a) permutada a l'atzar,
   (b) ordenada,
   (c) ordenada del revés,
   (d) amb tots els elements iguals.

5. **Alçada de pila**. Quantes crides recursives cal guardar com a màxim en un instant donat a la pila per a ordenar per fusió una taula de n elements?

6. **Quicksort a mà**. Ordeneu la taula $\langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2\rangle$ amb l'algorisme d'ordenació ràpida i la partició de Hoare.

7. **Quicksort**. Digueu quant triga l'algorisme d'ordenació ràpida amb la partició de Hoare quan l'entrada es troba...
   (a) permutada a l'atzar,
   (b) ordenada,
   (c) ordenada del revés,

(d) amb tots els elements iguals.

8. **Ordenant per dos camps**. Tenim un vector de vectors $V[0\,..\,n-1][2]$ amb informació sobre n persones. Cada posició $V[i]$ guarda els cognoms de la i-èsima persona: $V[i][0]$ guarda el primer cognom, $V[i][1]$ guarda el segon cognom.

   Volem ordenar el vector amb l'ordre habitual: Primer, les persones amb primer cognom més petit. En cas d'empat, van abans les persones amb segon cognom més petit. Suposeu que no hi ha dues persones amb els mateixos dos cognoms.

   Per exemple, si el contingut inicial de V fos

	0	1	2	3
0	Garcia	Roig	Garcia	Grau
1	Pi	Negre	Cases	Negre

   el resultat final hauria de ser

	0	1	2	3
0	Garcia	Garcia	Grau	Roig
1	Cases	Pi	Negre	Negre

   De les combinacions següents, només una resol aquest problema en general. Digueu raonadament quina és i quin cost té en temps i en espai:

   (a) Primer ordenem V amb *quicksort* usant el primer cognom, després amb *mergesort* usant el segon cognom.

   (b) Primer ordenem V amb *mergesort* usant el primer cognom, després amb *quicksort* usant el segon cognom.

   (c) Primer ordenem V amb *quicksort* usant el segon cognom, després amb *mergesort* usant el primer cognom.

   (d) Primer ordenem V amb *mergesort* usant el segon cognom, després amb *quicksort* usant el primer cognom.

9. **Unimodal**. Es diu que una seqüència $A = (a_1, \ldots, a_n)$ de longitud $n \geq 3$ és *unimodal* si existeix un índex p, $1 \leq p \leq n$, tal que $a_1 < a_2 < \cdots < a_p$ i $a_p > a_{p+1} > \cdots > a_n$, és a dir, creix fins a a_p i després decreix. A l'element a_p se l'anomena *pic*. Escriu un algorisme de dividir i vèncer que donat un vector que conté una seqüència unimodal trobi el pic de la seqüència, amb cost $O(\log n)$. Justifica que el cost de l'algorisme proposat és $O(\log n)$ i la correctesa de l'algorisme.

10. **Ordenació boja d'en Quim**. Sigui $T[i..j]$ una taula amb $n = j - i + 1$ elements. Considereu l'algorisme d'ordenació anomenat l'*ordenació boja d'en Quim*:

    (a) Si $n \leq 2$, la taula s'ordena trivialment.

    (b) Si $n \geq 3$, "dividim" la taula en tres intervals $T[i..k-1]$, $T[k..\ell]$ i $T[\ell+1..j]$, on $k = i + \lfloor n/3 \rfloor$, $\ell = j - \lfloor n/3 \rfloor$. L'algorisme ordena recursivament $T[i..\ell]$, després ordena $T[k..j]$, i finalment ordena de nou $T[i..\ell]$.

    Demostreu que aquest algorisme ordena correctament. Doneu, raonadament, una recurrència per al temps que triga aquest algorisme i resoleu-la.

11. **Comptant inversions**. Recordeu que una *inversió* en una taula $A[1 \ldots n]$ és un parell de posicions de la taula en desordre, és a dir, un parell (i, j) tal que $1 \leq i < j \leq n$ i $A[i] > A[j]$. Escriviu un algorisme que compti el nombre d'inversions d'una taula amb n elements en temps $O(n \log n)$ en el cas pitjor.

12. **Una inversió**. Responeu les següents qüestions:

    (a) Demostreu que, si en una taula hi ha una única inversió, llavors els dos elements d'aquesta apareixen consecutivament, és a dir, si (i, j) és l'única inversió de la taula, llavors $i + 1 = j$.

    (b) Descriviu un algorisme de cost $\Theta(\log n)$ en el cas pitjor que, donada una taula d'enters $T[1 \ldots n]$ que només té una inversió i un element x, digui si x és a T.

13. **Subtaula quasi nula**. Dissenyeu un algorisme de cost $O(n \log n)$ tal que, donada una taula amb n enters, calculi la subtaula consecutiva no buida tal que la seva suma és el més propera possible a 0.

14. **Subtaula quasi nula II**. Trobeu un algorisme de cost $O(n)$ per al problema anterior suposant aquest cop que la taula està ordenada.

15. **Cercant en dues taules**. Dissenyeu un algorisme de cost $\Theta(\log n)$ que, donades dues taules ordenades creixentment de n elements cadascuna i un cert k, trobi el k-èsim element més petit global.

16. **Trobar un punt fix**. Sigui V un vector de n enters diferents en ordre creixent. Dissenyeu un algorisme $O(\log n)$ que retorni un índex i tal que $V[i] = i$ si aquest existeix, -1 altrament.

17. **Multiplicació de Karatsuba**. Aquest problema estudia un algorisme de dividir i vèncer per multiplicar dos números de $n = 2^k$ bits. Recordeu que l'algorisme escolar utilitza $\Theta(n^2)$ passos.

    a) Siguin x i y dos números de n bits cadascun de forma que $x = a2^{n/2} + b$ i que $y = c2^{n/2} + d$. Comproveu que $xy = ac\ 2^n + (ad + bc)2^{n/2} + bd$.

    Utilitzant aquesta idea, dissenyeu un algorisme de dividir i vèncer per reduir la multiplicació de dos nombres de n bits a quatre multiplicacions de nombres de $n/2$ bits que es combinen mitjançant addicions i desplaçaments.

    Analitzeu el cost de l'algorisme resultant.

    b) L'any 1962, Karatsuba i Ofman van descobrir una forma molt astuta per reduir les anteriors quatre multiplicacions de nombres de $n/2$ bits a només tres: Comproveu que $xy = ((a + b)(c + d) - ac - bd)2^{n/2} + ac2^n + bd$.

    Utilitzant aquesta idea, dissenyeu un nou algorisme de dividir i vèncer i analitzeu-ne el cost.

    c) Com es generalitzen els algorismes anteriors quan n no és una potència de 2?

18. **Multiplicació de matrius de Strassen**. Aquest problema estudia un algorisme de dividir i vèncer per multiplicar dues matrius $n \times n$. Recordeu que l'algorisme clàssic utilitza $\Theta(n^3)$ passos.

Utilitzant la vella dita del "fila per columna", el producte de dues matrius 2×2 s'obté amb 8 productes de reals:

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} * \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} = \begin{pmatrix} a_{00} * b_{00} + a_{01} * b_{10} & a_{00} * b_{01} + a_{01} * b_{11} \\ a_{10} * b_{00} + a_{11} * b_{10} & a_{10} * b_{01} + a_{11} * b_{11} \end{pmatrix}.$$

Cap a l'any 1967, Strassen va descobrir que aquest producte de matrius es podia fer utilitzant només 7 productes de reals. Per això, va definir

$$\begin{aligned} m_1 &= (a_{00} + a_{11}) * (b_{00} + b_{11}) \\ m_2 &= (a_{10} + a_{11}) * (b_{00}) \\ m_3 &= (a_{00}) * (b_{01} - b_{11}) \\ m_4 &= (a_{11}) * (b_{10} - b_{00}) \\ m_5 &= (a_{00} + a_{01}) * (b_{11}) \\ m_6 &= (a_{10} - a_{00}) * (b_{00} + b_{01}) \\ m_7 &= (a_{01} - a_{11}) * (b_{10} + b_{11}) \end{aligned}$$

i va comprovar que

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} * \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} = \begin{pmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{pmatrix}.$$

L'avantatge d'aquest fet és que, aplicant la idea recursivament, podem multiplicar dues matrius grans amb un temps inferior al $\Theta(n^3)$ de l'algorisme clàssic: Suposeu que tenim dues matrius A i B de talla $n \times n$ cadascuna, essent n una potència de 2. Llavors podem descompondre el producte de les dues matrius en quatre blocs de la mateixa talla de la forma següent:

$$\left(\begin{array}{c|c} C_{00} & C_{01} \\ \hline C_{10} & C_{11} \end{array}\right) = \left(\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right) * \left(\begin{array}{c|c} B_{00} & B_{01} \\ \hline B_{10} & B_{11} \end{array}\right).$$

No és difícil veure que hom pot tractar els blocs d'aquestes matrius com si fossin nombres. Per exemple, la matriu C_{00} de talla $\frac{n}{2} \times \frac{n}{2}$ correspon al producte de matrius $M_1 + M_4 - M_5 + M_7$ on els M_i vénen donats per les fórmules de Strassen, substituint números per matrius.

a) Suposant que n és una potència de 2, quants passos requereix l'algorisme descrit per multiplicar dues matrius $n \times n$?

b) Quina senzilla modificació aplicaríeu a l'algorisme per multiplicar matrius $n \times n$ quan n no és una potència de 2? Quin cost tindria l'algorisme llavors?

c) Doneu una fita inferior per al problema de la multiplicació de dues matrius $n \times n$.

19. **Subseqüència màxima**. El problema de la Subseqüència Màxima consisteix en, donada una taula A amb n enters, determinar la suma màxima que es pot trobar en qualsevol subtaula consecutiva de l'entrada. Per exemple, si la taula t conté els 10 elements

$$31, -41, 59, 26, -53, 58, 97, -93, -23, 84$$

llavors la Subseqüència Màxima és 187, que correspon a la suma dels elements $A[2\,..\,6]$. Formalment, es vol calcular

$$\max\left\{\sum_{k=i}^{j} A[k] \;:\; 0 \le i < n,\ 0 \le j < n\right\}.$$

Fixeu-vos que es pot triar una subtaula buida, que té suma 0.

(a) Escriviu un algorisme $\Theta(n \log n)$ per resoldre el problema de la Subseqüència Màxima utilitzant dividir i vèncer.

(b) Escriviu un algorisme $\Theta(n)$ per resoldre el mateix problema utilitzant dividir i vèncer.

Nota: existeixen algorismes lineals per aquest problema que no estan basats en dividir i vèncer.

20. **Jornades d'un campionat**. Cal organitzar l'horari d'un campionat entre n jugadors, cadascun dels quals ha de jugar exactament una vegada contra cada adversari. A més, cada jugador ha de jugar exactament un partit diari. Suposant que n és potència de 2, dissenyeu i implementeu un algorisme per construir l'horari que permeti acabar el campionat en $n-1$ dies. Analitzeu el cost de l'algorisme.

21. **Quicksort en el cas mitjà**. L'objectiu d'aquest exercici és veure que el nombre de comparacions de quicksort, en el cas mitjà i quan l'entrada $A[1,\ldots,n]$ és una permutació del conjunt $\{1,\ldots,n\}$ escollida a l'atzar, és $O(n \log n)$. Considerem la versió[1] de l'algorisme que, amb l'entrada $A[l,\ldots,r]$ amb $|r-l+1| \ge 3$, agafa el primer element $q := A[l]$ com a pivot i fa crides recursives amb $A[l,\ldots,q-1]$ i $A[q+1,\ldots,r]$ després d'haver executat un algorisme de partició que fa el següent: compara $A[l]$ amb cadascun dels altres elements de $A[l,\ldots,r]$ i posa els elements de $A[l,\ldots,r]$ que són més petits que q a l'esquerra de $A[q]$ (però en el mateix ordre que estaven entre ells), posa els elements de $A[l,\ldots,r]$ que són més grans que q a la dreta de $A[q]$ (però en el mateix ordre que estaven entre ells), i posa q a $A[q]$.

(a) Argumenteu que, abans de qualsevol crida, la distribució de probabilitat de l'entrada $A[l,\ldots,r]$ és uniforme entre les permutacions de $\{l,\ldots,r\}$.

(b) Demostreu que si $T(n)$ és el nombre de comparacions esperat de l'algorisme amb un subvector de longitud $n \ge 3$, llavors $T(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} T(i)$.

(c) Proveu que $\sum_{i=1}^{n-1} i\ln(i) \le \int_1^n x\ln(x)dx = \frac{1}{2}n^2\ln(n) - \frac{1}{4}n^2 + \frac{1}{4}$ per a tot $n \ge 1$.

(d) Aprofiteu els dos punts anteriors per veure que $T(n) \le 2n\ln(n)$ per inducció.

[1] Aquesta versió fa servir un algorisme de partició que no es pot implementar *in-place* com la partició de Hoare vista a classe; podríem haver plantejat el mateix problema amb la partició de Hoare però llavors la primera part d'aquest problema seria bastant més difícil.

3

Diccionaris

1. **Taula de dispersió**. Considereu una taula de dispersió d'encadenament separat amb $M = 10$ posicions per a claus enteres i funció de dispersió $h(x) = x \mod M$.

   (a) Començant amb una taula buida, mostreu com queda després d'inserir les claus 3441, 3412, 498, 1983, 4893, 3874, 3722, 3313, 4830, 2001, 3202, 365, 128181, 7812, 1299, 999 i 18267.

   (b) Mostreu com queda la taula de dispersió anterior quan s'aplica una redispersió per enmagatzemar 20 posicions.

2. **Taula de dispersió II**. Construïu la taula de dispersió d'encadenament separat per a les claus 30, 20, 56, 75, 31 i 19, amb 11 posicions i funció de dispersió $h(x) = x \mod 11$.

   (a) Calculeu el nombre màxim de comparacions en una cerca amb èxit en aquesta taula.

   (b) Calculeu el nombre esperat de comparacions en una cerca amb èxit en aquesta taula.

3. **El renum del BASIC**. Un programa en BASIC consisteix en una sèrie d'instruccions numerades en ordre creixent. El control de flux es gestiona a través de les instruccions *GOTO x* i *GOSUB x*, on *x* és un número d'instrucció. Per exemple, el programa següent calcula el factorial d'un número:

```
50 INPUT N
60 LET F = 1
61 LET I = 1
73 IF I=N THEN GOTO 99
76 LET F = F*I
80 LET I = I+1
81 GOTO 73
99 PRINT F
```

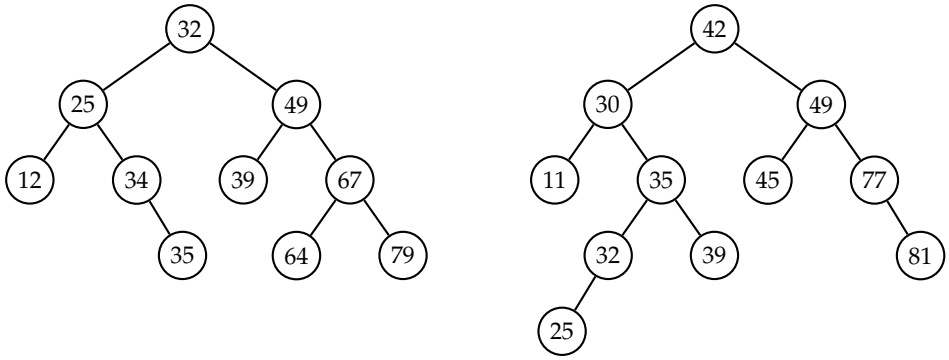
El procediment *RENUM* renumera les instruccions del programa de forma que les línies vagin de 10 en 10. Per exemple, després de renumerar, el programa anterior queda:

```
10 INPUT N
20 LET F = 1
30 LET I = 1
40 IF I=N THEN GOTO 80
50 LET F = F*I
60 LET I = I+1
70 GOTO 40
80 PRINT F
```

Descriviu com implementar el procediment *RENUM* en temps lineal en el cas mitjà.

4. **Tots diferents**. Dissenyeu i analitzeu un algorisme que utilitzi taules de dispersió per comprovar que tots els elements d'una llista són diferents.

5. **L'ABC dels ABCs**. Digueu si els arbres següents són arbres binaris de cerca o no i per què.

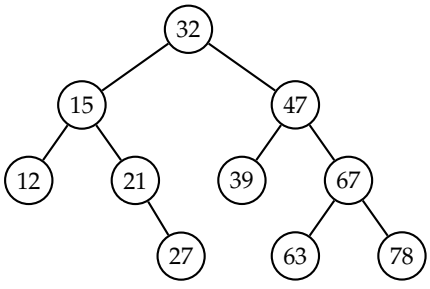


6. **Inserir en un ABC**. Partint d'un arbre binari de cerca buit, inseriu amb l'algorisme clàssic, l'una rera l'altra, la seqüència de claus $32, 15, 47, 67, 78, 39, 63, 21, 12, 27$.

7. **Inserir en un ABC II**. Partint d'un arbre binari de cerca buit, inseriu amb l'algorisme clàssic, l'una rera l'altra, la seqüència de claus $12, 15, 21, 27, 32, 39, 47, 63, 67, 78$.

8. **Sobre el màxim d'un ABC**. Expliqueu si és cert o no que en un arbre binari de cerca no buit, l'element màxim pot tenir fill esquerre però no pot tenir fill dret.

9. **Esborrar en un ABC**. Partint de l'arbre binari de cerca següent, elimineu les claus $63, 21, 15, 32$, l'una rera l'altra. Expliqueu quin algorisme heu usat per eliminar les claus.



10. **Inordre d'un ABC**. Demostreu que el recorregut en inordre d'un arbre binari de cerca visita els elements en ordre creixent.

11. **L'ABC d'un preordre donat**. Dibuixeu l'arbre binari de cerca el recorregut en preordre del qual és 8, 5, 2, 1, 4, 3, 6, 7, 11, 9, 13, 12.

12. **ABCs implementats**. Per aquest problema i els següents, utilitzeu aquesta definició de tipus per als arbres binaris de cerca:

```
struct node {
    Elem x;              // Informació en el node
    node* fe;            // Punter al fill esquerre
    node* fd;            // Punter al fill dret

    node(Elem _x, node* _fe, node* _fdr)
      : x(_x), fe(_fe), fd(_fd) {
    }

    ~node () {
        delete fe; delete fd;
    }
};

typedef node* abc;       // Un ABC es denota per un punter a la seva arrel
                         // (null si és buit)
```

    Implementeu una operació *abc crear* () que retorni un arbre binari de cerca buit. Quin és el seu cost?

13. **Cerca en un ABC**. Implementeu una operació **bool** *hi_es* (*abc a*, *Elem x*) que indiqui si l'element *x* és a l'arbre binari de cerca *a*. Quin és el seu cost?

14. **Inserir en un ABC**. Implementeu una operació **void** *afegir* (*abc*& *a*, *Elem x*) que afegeixi l'element *x* a l'arbre binari de cerca *a*. Quin és el seu cost?

15. **Esborrar en un ABC**. Implementeu una operació **void** *treure* (*abc*& *a*, *Elem x*) que tregui l'element *x* de l'arbre binari de cerca *a*. Quin és el seu cost?

16. **Mínim en un ABC**. Implementeu una operació *Elem minim* (*abc a*) que retorni l'element més petit de l'arbre binari de cerca no buit *a*. Quin és el seu cost?

17. **Màxim en un ABC**. Implementeu una operació *Elem maxim* (*abc a*) que retorni l'element més gran de l'arbre binari de cerca no buit *a*. Quin és el seu cost?

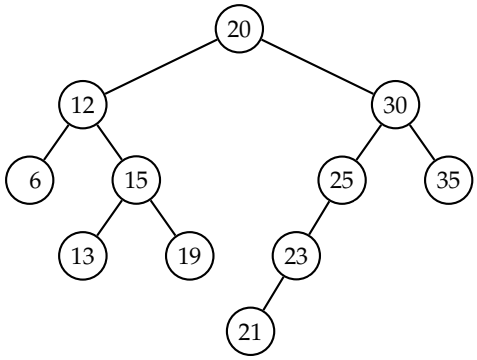
18. **Fusió de dos ABCs**. La *fusió* de dos arbres binaris de cerca *a1* i *a2* és un arbre binari de cerca que conté tots els elements de *a1* i *a2*. Implementeu una funció

    abc fusio (*abc*& *a1*, *abc*& *a2*)

    que retorni la fusió dels arbres binaris de cerca *a1* i *a2* (els dos arbres donats han de quedar buits després de la crida). Quin és el seu cost?

19. **Segment d'un ABC**. Dissenyeu una funció *list <Elem> entre(abc a, Elem x1, Elem x2)* que, donat un arbre binari de cerca *a* i dos elements *x1* i *x2* amb $x1 \leq x2$, retorni una llista amb tots els elements de *a* que es trobin entre *x1* i *x2*, en ordre decreixent.

    Per exemple, donat l'arbre

    i els valors 18 i 26, cal retornar la llista $\langle 25, 23, 21, 20, 19 \rangle$.

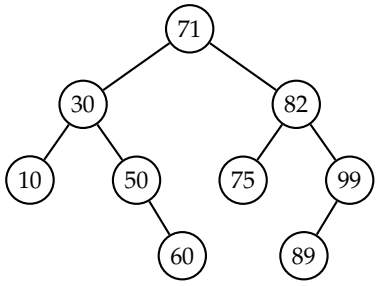
    Calculeu el cost del vostre algorisme en el cas pitjor.

20. **Menors en un ABC**. Implementeu una funció **int** *menors(abc a, Elem x)* que retorni el nombre d'elements de l'arbre binari de cerca *a* que són estrictament inferiors a *x*. Quin és el seu cost?

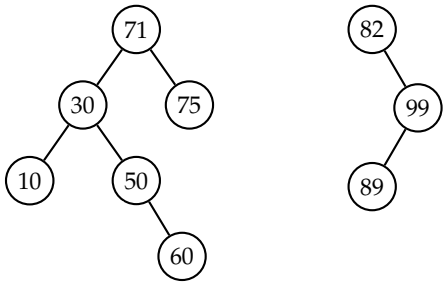
21. **Construir l'ABC d'un preodre donat**. Implementeu la funció *abc refer* (**vector**<*Elem*> *pre*) que reconstrueix un arbre binari de cerca a partir del seu recorregut en preordre emmagatzemat en un vector *pre*. Quin és el seu cost?

22. **Separar un ABC**. Implementeu una funció **void** *separa*(*abc*& *a*, *Elem x*, *abc*& *le*, *abc*& *gt*) que donat una arbre binari de cerca *a*, retorni dos arbres binaris de cerca *le* i *gt* on *le* conté tots els elements de *a* que són més petits o iguals que *x* i *gt* conté tots els elements de *a* que són més grans que *x*. L'arbre original *a* ha de quedar destruït.

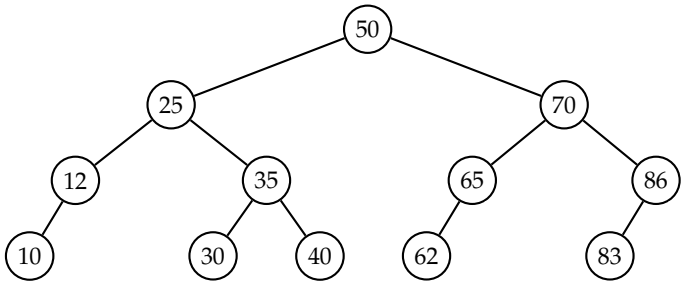
    Per exemple, donat l'arbre binari de cerca següent

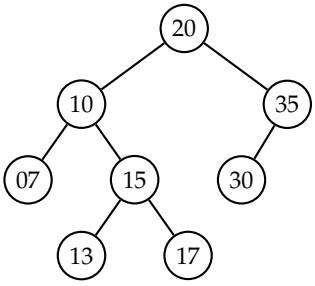
    i l'element $x = 75$, els arbres *le* i *gt* són els següents:



23. **Inserir en un AVL**. Doneu els tres arbres AVL resultants d'afegir les claus 31, 32 i 33 l'una després de l'altra en l'arbre AVL següent:



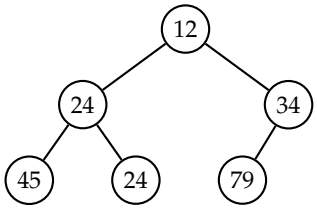
24. **Inserir i esborrar en un AVL**. Doneu els quatre arbres AVL resultants d'afegir les claus 18 i 12 i d'esborrar les claus 7 i 30 a l'arbre AVL següent (apliqueu cada operació a l'arbre obtingut anteriorment):

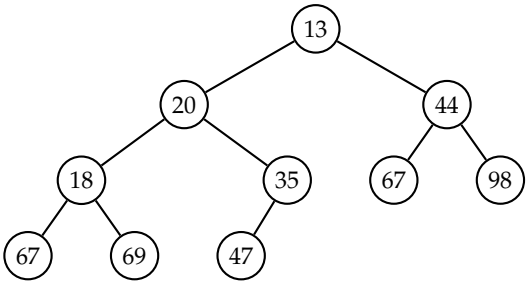


4

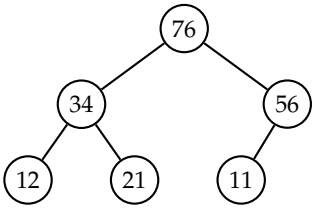
Cues amb prioritats

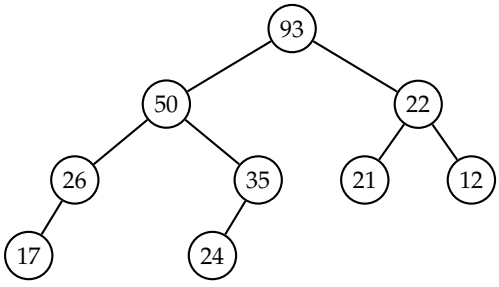
1. **Són min-heaps?**. Digueu si els arbres binaris següents són min-heaps o no i per què.





2. **Són max-heaps?**. Digueu si els arbres binaris següents són max-heaps o no i per què.

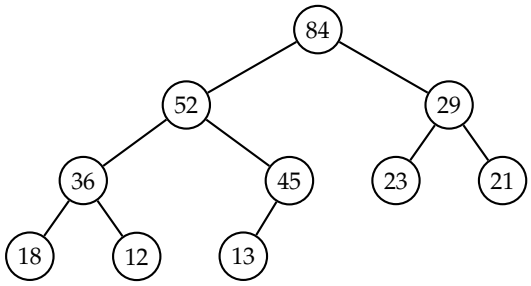




3. **Inserir en un heap**. Partint d'un min-heap buit, inseriu successivament els nombres 45, 67, 23, 46, 89, 65, 12, 34, 98, 76.

4. **Inserir en un heap II**. Partint d'un max-heap buit, inseriu successivament els nombres 45, 67, 23, 46, 89, 65, 12, 34, 98, 76.

5. **Esborrant en un heap**. Partint del max-heap següent, elimineu successivament l'element màxim fins que quedi buit.



6. **Reconstruïnt el heap**. Donat el min-heap següent implementat en una taula,

7	11	9	23	41	27	12	29

dibuixeu l'arbre representat per la taula i, a continuació, dibuixeu l'evolució dels continguts de la taula i de l'arbre representat, en aplicar successivament les operacions d'inserir l'element 3 i eliminar el mínim.

7. **Reconstruïnt el heap II**. Convertiu la taula següent en un min-heap tot aplicant l'algorisme de construcció de heaps de dalt cap a baix.

45	53	27	21	11	97	34	78

8. **Reconstruïnt el heap III**. Convertiu la taula següent en un min-heap tot aplicant l'algorisme de construcció de heaps de baix cap a dalt.

45	53	27	21	11	97	34	78

9. **Test**. Valideu o refuteu les afirmacions següents:
   (a) "Una taula amb els elements ordenats de menor a major és un min-heap."
   (b) "Inserir en un max-heap amb n elements té cost $\Theta(\log n)$ en el cas pitjor."
   (c) "Trobar l'element màxim en un min-heap té cost $O(\sqrt{n})$."
   (d) "Eliminar el màxim d'un max-heap de n elements diferents té cost $\Theta(1)$ en el cas millor."

10. **Partial sort**. L'algorisme `partial_sort` de la STL rep un vector amb n elements i un valor m, $1 \leq m \leq n$, i reordena el vector de manera que els seus m primers elements són els m elements més petits del vector original, en ordre creixent.
    (a) Expliqueu com es pot implementar aquesta funció de manera que el seu cost en cas pitjor sigui $\mathcal{O}(n + m \log n)$.
    (b) I com es podria implementar per a que el seu cost sigui $\mathcal{O}(n \log m)$?

11. **El k-èssim d'n taules ordenades**. Dissenyeu un algorisme de cost $\Theta(k \log n + n)$ que, donades n taules ordenades creixentment amb n elements cadascuna i un cert k, trobi el k-èsim element més petit global.

5

Grafs

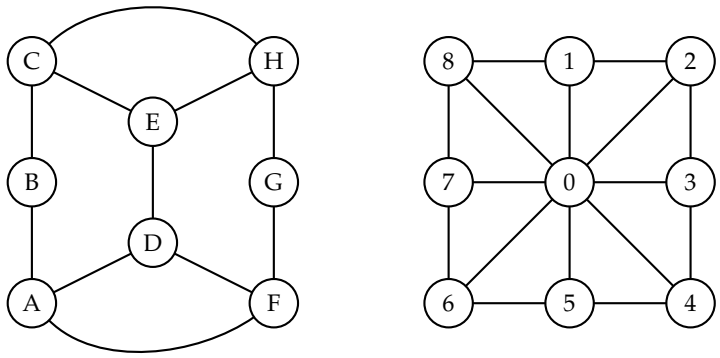
1. **Dibuixeu el graf**. Dibuixeu el graf $G = (V, E)$ donat per:
   (a) $V = \{1, 2, 3, 4, 5, 6\}$
   (b) $E = \{\{1, 2\}, \{1, 4\}, \{3, 2\}, \{4, 5\}, \{5, 1\}, \{5, 2\}\}$

   És connex? Si no ho és, quants components connexos té?

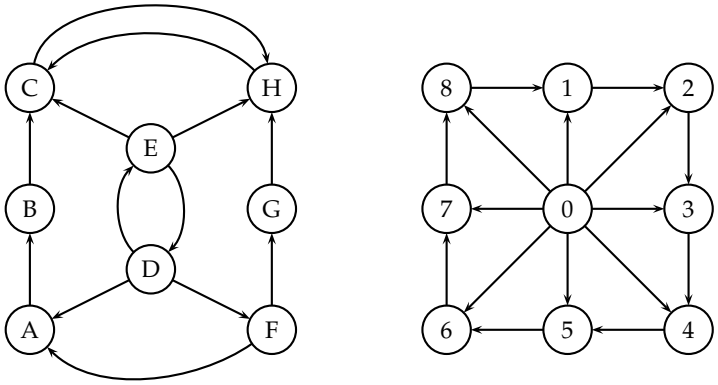
2. **Dibuixeu el graf II**. Dibuixeu el graf dirigit $G = (V, E)$ donat per:
   (a) $V = \{1, 2, 3, 4, 5\}$
   (b) $E = \{(1, 2), (1, 4), (3, 2), (4, 5), (5, 1), (5, 2)\}$

   És fortament connex? És feblement connex?

3. **Representeu els grafs**. Mostreu com es representen els dos grafs següents utilitzant una matriu d'adjacència i llistes d'adjacència. Compteu el grau de cada vèrtex. Calculeu el diàmetre de cada graf.



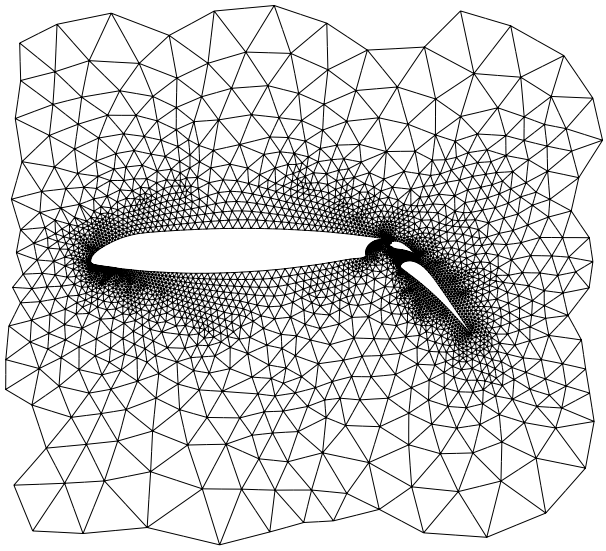
4. **Representeu els grafs II**. Mostreu com es representen els grafs dirigits següents utilitzant una matriu d'adjacència i llistes d'adjacència. Compteu el grau d'entrada i de sortida de cada vèrtex. Digueu si els grafs són fortament o feblement connexos.



5. **Suma dels graus**. Demostreu que en un graf no dirigit $G = (V, E)$ es té $\sum_{u \in V} \text{grau}(u) = 2|E|$.

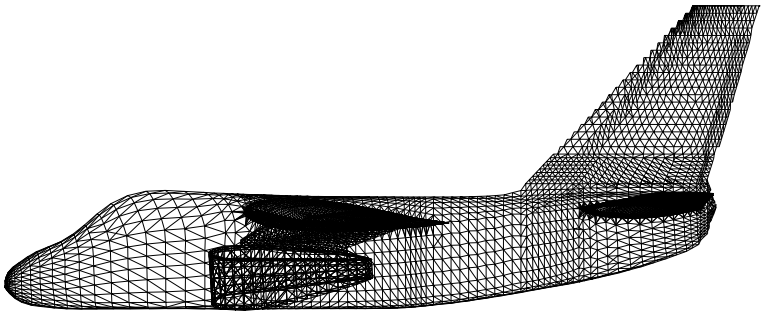
6. **Existeix un graf així?**. Considereu un graf no dirigit amb cinc vèrtexs, tots de grau tres. Si existeix tal graf, feu-ne un dibuix. Si no n'existeix cap, doneu-ne una explicació raonada.

7. **L'espai que ocupa**. El graf de la figura següent té $n = 4253$ vèrtexs i $m = 12289$ arestes.

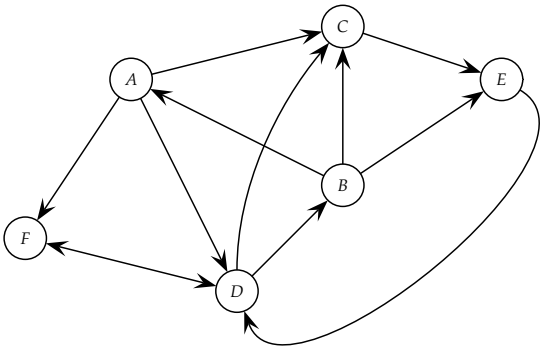


Suposeu que al vostre ordinador cada booleà ocupi un byte, cada enter quatre i cada apuntador quatre. Digueu quanta memòria cal per emmagatzemar aquest graf utilitzant una representació amb matriu d'adjacència i utilitzant una representació amb llistes d'adjacència.

8. **L'espai que ocupa II**. Repetiu el problema anterior per al graf següent, que té $n = 156317$ vèrtexs i $m = 1059331$ arestes.



9. **Recorregut en profunditat**. Llisteu totes les possibles seqüències de visita dels vèrtexs d'aquest graf dirigit, tot aplicant un recorregut en profunditat que comenci en el vèrtex E.



10. **Recorregut en amplada**. Repetiu el problema anterior tot aplicant un recorregut en amplada que comenci en el vèrtex B.

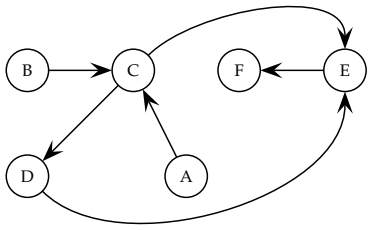
11. **Misteris gràfics**. Considereu la funció següent sobre un graf no dirigit $G = (V, E)$ amb $n = |V|$ vèrtexs i $m = |E|$ arestes, implementat amb llistes d'adjacència (on els vèrtexs s'identifiquen amb els enters 0, ... $n - 1$):

```
int misteri(const vector<vector<int>>& G) {
    int x = 0;
    for (int u = 0; u < G.size(); ++u)
        for (int w : G[u])
            ++x;
    return x;
}
```

    (a) Expliqueu breument què retorna la funció.

    (b) Quin cost té la funció?

12. **Ordenacions topològiques**. Doneu, en ordre lexicogràfic, totes les possibles ordenacions topològiques del següent graf dirigit acíclic (DAG, de *Directed Acyclic Graph*):



13. **I si afegim un vèrtex aïllat?**. Si al graf anterior li afegíssim un vèrtex aïllat, quantes ordenacions topològiques tindria? (No les llisteu, només digueu quantes i expliqueu per què.)

14. **Ordenació topològica inversa**. Una *ordenació topològica inversa* d'un DAG és una seqüència formada per tots els vèrtexs del graf tal que si (v, w) és una aresta del graf, llavors w apareix abans que v a la seqüència. A partir de l'algorisme del recorregut en profunditat, dissenyeu i analitzeu un algorisme que obtingui una ordenació topològica inversa d'un DAG.

15. **Arrels i fulles en un DAG**. Escriu un algorisme que donat un DAG $G = \langle V, E \rangle$ calculi quantes *arrels* i quants *fulles* conté el DAG G. Un vèrtex v és una *arrel* del DAG si el seu grau d'entrada és 0, i és un *fulla* si el seu grau de sortida és 0. El graf està implementat amb llistes d'adjacència.

    Calcula el cost de l'algorisme desenvolupat en funció de $n = |V|$ i $m = |E|$, i justifica la seva correctesa.

16. **Transposant**. La *transposició* d'un graf dirigit $G = (V, E)$ és un nou graf dirigit $G^T = (V, E^T)$ on $E^T = \{(u, v \mid (v, u) \in E\}$. Dissenyeu i analitzeu un algorisme que transposi un graf dirigit representat amb una matriu d'adjacència.

    Feu el mateix amb un graf dirigit representat amb llistes d'adjacència.

17. **Elevant al quadrat**. El *quadrat* d'un graf $G = (V, E)$ és un altre graf $G^Q = (V, E^Q)$ on $E^Q = \{\{u, v\} \mid \exists w \in V \mid \{u, w\} \in E \wedge \{w, v\} \in E\}$. 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.

18. **Elevant al quadrat II**. Es pot calcular el quadrat d'un graf amb n vèrtexs representat amb matrius d'adjacència en temps inferior a $\Theta(n^3)$?

    Pista: Strassen.

19. **Buscar quadres**. En un graf no dirigit, un *quadre* és un cicle de llargada 4. Dissenyeu un algorisme que, donat un graf no dirigit, digui si aquest conté algun quadre. Analitzeu la seva eficiència en diferents representacions.

20. **Buscar quadres II**. Aprofitant la solució del problema **Elevant al quadrat II**, dissenyeu un algorisme per al problema anterior de temps inferior a $\Theta(n^3)$, on n és el nombre de vèrtexs del graf.

21. **La celebritat**. 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.

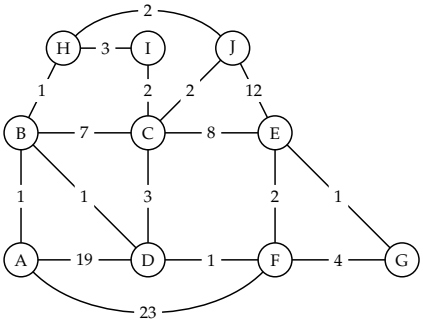
    Doneu un algorisme que, donat un graf dirigit representat amb una matriu d'adjacència, indica si hi ha o no cap celebritat. En el cas que hi sigui, cal dir qui és. El vostre algorisme ha de funcionar en temps $O(n)$, on n és el nombre de vèrtexs.

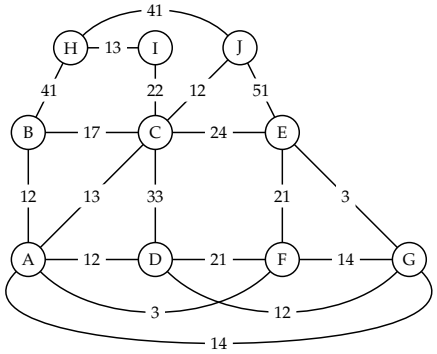
22. **Comptant components connexos**. Dissenyeu i analitzeu un algorisme que calculi el nombre de components connexos d'un graf no dirigit.

23. **És acíclic?**. Dissenyeu i analitzeu un algorisme que en temps $O(|V|)$ determini si un graf no dirigit conté cicles o no.

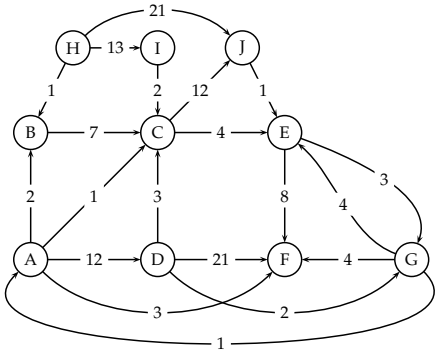
24. **És acíclic? II**. Dissenyeu i analitzeu un algorisme que determini si un graf dirigit conté cicles o no.

25. **Camí mínim**. Trobeu el camí mínim per anar del vèrtex A al vèrtex G en els dos grafs següents:





26. **Un de Dijkstra**. Mostreu com l'algorisme de Dijsktra calcula els camins mínims per anar del vèrtex C a tots els altres vèrtexs en el graf dirigit següent:



27. **Amb cicles negatius?**. Quin sentit té calcular camins mínims quan hi ha cicles de cost negatiu en un graf?

    Mostreu un graf amb pesos negatius per al qual l'algorisme de Dijsktra no funciona, malgrat no haver-hi cicles de cost negatiu.

28. **Nombre de camins mínims**. Modifiqueu l'algorisme de Dijsktra per tal que compti el nombre de camins mínims entre dos vèrtexs donats. Què passaria si el graf tingués cicles de cost zero? I si tingués arcs de cost zero però no cicles de cost zero?

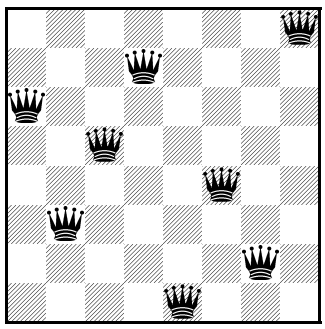
29. **Camins mínims més curts**. Modifiqueu l'algorisme de Dijsktra per tal que si hi ha més d'un camí mínim entre dos vèrtexs donats, en retorni un de qualsevol amb el nombre mínim d'arcs.

6

Generació i cerca exhaustiva

1. **Vuit reines**. Dissenyeu un algorisme que escrigui totes les possibles maneres de col·locar n reines en un tauler amb $n \times n$ escacs de forma que cap reina n'amenaci cap altra.

   Per exemple, aquesta és una configuració de 8 reines:

2. **Vuit reines II**. Dissenyeu un algorisme que compti de quantes maneres es poden col·locar n reines en un tauler amb $n \times n$ escacs de forma que cap n'amenaci cap altra.

3. **Vuit reines III**. Dissenyeu un algorisme que trobi una possible manera de col·locar n reines en un tauler amb $n \times n$ escacs de forma que cap n'amenaci cap altra o indiqui que no hi ha tal col·locació.

4. **Salts de cavall**. En un tauler amb $n \times n$ escacs es col·loca un cavall en una casella donada. Dissenyeu un algorisme per trobar si existeix alguna forma d'aplicar $n^2 - 1$ moviments de cavall de forma que el cavall visiti totes les caselles del tauler.

   Per exemple, aquesta és una possible solució per a un tauler 5×5 començant des del centre:

22	5	16	11	24
15	10	23	6	1
4	21	12	17	12
9	14	19	2	7
20	3	8	13	18

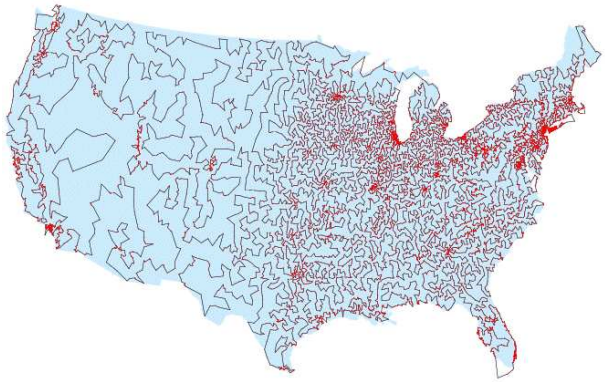
5. **Quadrats llatins**. Un quadrat llatí d'ordre n és una taula $n \times n$ on cada casella està pintada amb un color escollit d'entre n possibles sense que cap fila ni cap columna contingui colors repetits. Per exemple, aquest és un quadrat llatí d'ordre 5:

0	2	3	1	4
1	3	4	2	0
2	1	0	4	3
3	4	1	0	2
4	0	2	3	1

   Dissenyeu un algorisme que escrigui tots els quadrats llatins d'ordre n.

6. **Graf Hamiltonià**. Dissenyeu i analitzeu un algorisme per determinar si un graf connex no dirigit és hamiltonià. Modifiqueu-lo per obtenir-ne un cicle hamiltonià si aquest existeix.

7. **El viatjant**. Un viatjant ha de visitar n clients que viuen en n ciutats diferents. La distància per anar de la ciutat i a la ciutat j ve donada per $D[i][j]$. El viatjant vol, sortint de la seva ciutat, passar exactament un cop per cada ciutat on viuen els seus clients i retornar al punt de partida. A més, vol que la distància total recorreguda sigui mínima. Per exemple, aquest és el pla de viatge d'un viatjant de comerç pels EEUU:



   Dissenyeu i analitzeu un algorisme per resoldre el problema del viatjant. Feu-ho amb l'esquema de tornada enrera (*backtracking*) i amb l'esquema de ramificació i poda (*branch and bound*).

8. **Clique**. Donat un graf no dirigit $G = (V, E)$, una *clique* és un subconjunt de vèrtexs $S \subseteq V$ tal que per a qualsevol parell de vèrtexs u i v en S hi ha una aresta entre u i v en el graf.

   Dissenyeu i analitzeu un algorisme que donat un graf G i un natural k, determini si G conté alguna clique de talla k.

9. **Conjunt independent**. Dissenyeu i analitzeu un algorisme que, donat un graf $G = (V, E)$ i un nombre natural k, determini si G té un conjunt independent de k vèrtexs (és a dir, si existeix un subconjunt $A \subseteq V$ de k vèrtexs tal que $\{u, v\} \notin E$ per a tot $u, v \in A$).

10. **Recobriment**. Dissenyeu i analitzeu un algorisme que, donat un graf $G = (V, E)$ i un nombre natural k, determini si G té un recobriment de les arestes amb k vèrtexs. Recordeu que això vol dir determinar si existeix un subconjunt $A \subseteq V$ de k vèrtexs tal que per a cada $\{u, v\} \in E$, al menys un dels extrems u o v pertany A.

11. **Suma de subconjunt**. Dissenyeu un algorisme que, donada un col·leció C de n nombres enters positius (amb possibles repeticions) i un nombre enter positiu S, determini si hi ha un subconjunt de C que sumi exactament S.

12. **Coloració de grafs**. Es disposa de $c \geq 3$ colors diferents i d'un mapa de n països, juntament amb una taula de booleans $v[i][j]$ que indica si dos països i i j són veïns. Escriviu un algorisme de cerca exhaustiva que trobi totes les coloracions possibles del mapa amb la restricció que dos països veïns han de ser pintats amb colors diferents.

    Com solucionareu el problema per a $c = 2$ (bicoloració)?

13. **Empaquetament**. Tenim una col.lecció de n objectes que cal empaquetar en envasos de capacitat C. L'objecte i té volum $v[i]$. Dissenyeu un algorisme que calculi quin és l'empaquetament òptim, és a dir, aquell que minimitza el nombre d'envasos. Els objectes no es poden fraccionar però, per simplificar el problema, podeu considerar que un objecte cap dins d'un envàs si el volum de l'objecte és igual o inferior a l'espai lliure que queda a l'envàs (independentment de la forma que l'objecte pugui tenir.)

7

Intractabilitat

1. **Tragicomèdia en tres actes**. Considereu els problemes **NP**-complets següents:

   (a) BIN-PACKING: Donades n peces de mides $d_1, \ldots, d_n$ i k contenidors de capacitat m, decidir si es poden col·locar totes les peces en els contenidors sense sobrepassar la seva capacitat.

   (b) TRIPARTITE-MATCHING: Donats tres conjunts C_1, C_2, C_3 disjunts amb n elements cadascun d'ells, i un subconjunt $S \subseteq C_1 \times C_2 \times C_3$, decidir si es poden escollir n elements de S de manera que tot element de C_1, C_2 i C_3 aparegui una única vegada entre aquestes n ternes.

   (c) 3-COLORABILITAT: Donat un graf, decidir si es poden assignar colors als vèrtexs, a escollir d'entre tres colors possibles, de manera que cap aresta tingui els seus extrems del mateix color.

   (d) GRAF-HAMILTONIÀ: Donat un graf, decidir si hi ha un cicle que passa per tots els seus vèrtexs un sol cop.

   (e) CONJUNT-DOMINADOR: Donat un graf i un natural k, decidir si hi ha un subconjunt de k vèrtexs tals que tot altre vèrtex és adjacent a un d'ells.

   Cadascuna de les escenes següents presenta un problema. Mostreu que aquest problema és difícil tot veient que la seva versió decisional es pot identificar amb algun del problemes de la llista anterior.

   Escena 1. En Roy, a petició d'en Johnny, és l'encarregat d'organitzar un sopar per al grupet de col·legues de sempre. Ha demanat taula en un restaurant; és una taula rodona molt gran i hi caben tots. Quan arriben al lloc, però, apareixen les primeres dificultats.

   JOHNNY: Escolta Roy, es veu que algunes persones del grup estan enfadades entre elles. Seria convenient que dues persones enfadades no seguessin l'una al costat de l'altra.

ROY: Ja comencem... Mira, anem al gra. Vull que recopilis tota la informació al respecte; és a dir, per cada parell de persones, si estan enfadades o no. Jo miraré com les podem fer seure sense que ningú tingui un veí de taula amb el qual està enfadat.

Escena 2. Tenint totes les dades a la mà, i després d'una bona estona, en Roy no ha trobat encara cap solució. A més, en Johnny torna a aparèixer amb cara de circumstàncies.

JOHNNY: Ep, es veu que els que estan enfadats mútuament no volen ni tan sols seure a la mateixa taula. Però tranquil, he preguntat al restaurant i m'han dit que ens deixen tres taules grans.

ROY: Vejam, intentaré distribuir la gent en tres grups de manera que en un mateix grup no hi hagi dues persones mútuament enfadades.

Escena 3. Mentre en Roy pensa com distribuir la gent al voltant de les tres taules, en Johnny apareix de nou informant que el restaurant és a punt de tancar, i que ja poden anar buscant un altre lloc per sopar.

ROY: Mira, abans de buscar un altre lloc, m'agradaria aclarir quatre coses a la gent: Això de "ai, ai, no vull seure amb en tal o amb en qual" és una criaturada! Vull parlar amb ells seriosament. El problema és que estic una mica afònic i si parlo amb tothom em quedaré sense veu. Vull que escullis k representants de manera que, per a qualsevol dels del grup, hi hagi un dels k representants amb qui no estigui enfadat. D'aquesta manera els k representants faran arribar a tot el grup el missatge.

Escena 4. Com que en Johnny no se'n surt en triar els k representants, finalment agafa un megàfon i comunica a tothom que si continuen amb tanta conya, hauran d'anar cap a caseta.

JOHNNY: Escolta, tinc un mapa de la ciutat on he marcat les places on hi ha bons restaurants, i els carrers que les uneixen. Podríem anar passejant per totes i veure si tenen taules lliures.

ROY: Ok, deixa'm una estona el mapa per escollir el trajecte, que fa fred i no tinc ganes de passar dues vegades pel mateix lloc.

Escena 5. Malgrat tenir totes les dades a mà, i després d'una bona estona, en Roy no se'n surt. En Johnny el vol ajudar:

JOHNNY: Escolta Roy, jo tinc un munt de col·legues a la ciutat i a cadascuna d'aquestes places hi tinc un amic que hi viu. Dona'm pasta que els truco i els demano que mirin si hi ha lloc als restaurants de la seva plaça.

ROY: Mira tio, estic escurat i no tinc diners per tantes trucades. Te'n dono prou per fer m trucades. Si de cas, demana als teus amics que mirin no només a la plaça on viuen, sinó també a les places que distin un carrer d'on ells són. Escull als amics adequats perquè puguin mirar totes les places.

Escena 6. En no sortir-se'n, en Johnny truca al primer de la llista, i resulta que a prop seu hi ha un restaurant amb taules lliures. Quan hi arriben, resulta que només hi ha una taula amb k cadires i que tanquen el restaurant d'aquí poc.

JOHNNY: Ja està, tinc la solució: Anirem sopant primer uns i després els altres, quan algú hagi acabat, sortirà i n'entrarà un altre. Mira, fa un temps vaig dedicar-me a recopilar informació sobre quan trigava cada persona en sopar. Ja saps que tinc hobbies una mica estranys...

ROY: Mare meva...

JOHNNY: Amb aquestes dades, podrem fer una planificació distribuint les persones a les cadires, de manera que tothom pugui sopar abans que tanquin.

ROY: D'acord, dona'm aquesta llista del temps que triga cadascú a sopar i veuré què hi puc fer. Algun dia m'hauràs d'explicar quines altres dades reculls sobre la gent...

Escena 7. En l'entretant, en Johnny rep la trucada d'un amic dient que a prop d'allà hi ha un restaurant molt guai. Se n'hi van tots de dret. Quan hi arriben, resulta que només hi ha un munt de tauletes petites, i a sobre, totes de colors diferents.

JOHNNY: Escolta Roy. Es veu que la colla estan una mica avorrits i volen aprofitar la situació per a asseure's adequadament per parelles. Resulta que som tants nois com noies, i cadascú té les seves preferències. Però les taules de colors també juguen el seu paper. Per exemple, tal persona accepta seure amb tal altra només si la taula és de color rosa, i amb no sé quina altra només si la taula és de color blau. Haurem de trobar una distribució que satisfaci les restriccions de tothom. Es veu que amb tanta estona sense menjar, la penya està una mica anada.

ROY: A la penya els clavaré jo una bona pinya. Mare de déu, no estic ara per gaires tonteries, eh! Però vaja, ho intentaré.

Escena 8. No se'n surten i s'ha fet molt tard. En Roy i en Johnny només troben una tasca on hi ha una taula amb una cadira. Això sí, obren un munt d'hores.

JOHNNY: Resulta que la gent estan una mica ratllats. Alguns se'n van a fer un volt i tornaran més tard. Aquí tinc apuntat, per cada persona, a quina hora tornarà i a quina hora se n'haurà d'anar a casa. I encara tinc la llista del temps que necessita cada persona per sopar. Crec que, amb aquestes dades, hauria de ser fàcil veure si podem distribuir a tothom segons els seus horaris perquè puguin anar seient a la cadira i sopant.

ROY: Escolta Johnny, n'estic fart. Vull que sàpigues que aquesta és la darrera cosa que faig per aquesta gent. Va, vinga, dona'm aquestes dades i veuré què hi puc fer.

[En aquesta escena no n'hi ha prou amb identificar un problema de la llista; cal fer una reducció.]

Escena 9. Finalment, en Roy i en Johnny decideixen sopar tots dos junts en aquell lloc, només els ha calgut demanar una altra cadira a la cambrera. Als altres, els han donat una adreça d'un lloc on només hi ha un descampat, a la zona amb major índex de delinqüència i criminalitat de la ciutat.

JOHNNY: Vinga home, no t'ho prenguis així! Això d'avui simplement ha estat mala sort.

ROY: Mira, de mala sort, res. No és el primer cop que em passa això. És que sóc un passarell i sembla que no n'aprendré mai. Ara, vull que escoltis amb atenció.

Sé que t'ho he dit altres cops, però aquesta vegada va de debò: No tornaré a organitzar res per aquesta penya mai més. I quan dic mai és mai. De fet, ni tan sols tornaré a quedar amb ells.

JOHNNY: D'acord, d'acord! Llàstima, però. La setmana vinent havíem pensat anar tots a una casa de colònies. Ja saps, natura, aire pur, tranquil·litat, diversió, carn a la brasa, banys al riu tots despullats...

ROY: Vaja! Sona bé això! Però no sé, després les coses sempre acaben sortint malament.

JOHNNY: A més, aquest cop vindrà l'Steffy. Te'n recordes?

ROY: L'STEFFY !?!?!?

JOHNNY: Sí, i diu que té moltes ganes de veure't. Crec que s'endurà una desil·lusió molt gran si no véns.

ROY: Hmmm... bé... potser n'he fet un gra massa. Si en el fons són bons paios... A vegades estan inaguantables, però són bons paios en definitiva, i això és el que compta. Al cap i a la fi, l'amistat està plena de mals moments, però sobretot de moments meravellosos plens de joia. Va vinga! Què punyetes? M'hi apunto! I si vols t'ajudo a organitzar les coses.

JOHNNY: Perfecte! Ets el millor! Un crack, sí senyor! Va, vinga, posem-nos-hi ara mateix. A veure, l'únic problema que podria haver-hi és que només hi ha k cotxes i...

2. **Entendre les definicions**. Tenim un problema X i hem pogut establir que SAT es redueix a X (en símbols, $\text{SAT} \leq_m X$). Quina de les següents afirmacions podem deduir:

   (a) X pertany a la classe **NP**, però no sabem si és o no **NP**-complet.

   (b) X és un problema **NP**-complet.

   (c) Cap de les anteriors.

   Addicionalment, un company ha trobat ara una reducció en sentit invers, $X \leq_m \text{SAT}$. Què podem deduir ara?

   (i) X pertany a la classe **NP**, però no sabem si és o no **NP**-complet.

   (ii) X és un problema **NP**-complet.

   (iii) Cap de les anteriors.

3. **Reduccions en quin sentit?**. Els problemes de la satisfactibilitat de fórmules booleanes (SAT) i de determinar si un graf donat és hamiltonià (HAM) són dos problemes NP-complets ben coneguts. El problema de determinar si un graf donat conté un circuit eulerià (EUL) té una solució en temps $\Theta(n + m)$, sent n el nombre de vèrtexs del graf i m el nombre d'arestes.

   Quina de les següents frases podem afirmar que és certa?

   (a) SAT es redueix a HAM i HAM es redueix a EUL.

   (b) EUL es redueix a SAT i SAT es redueix a HAM.

   (c) SAT i HAM es redueixen l'un a l'altre; EUL només es redueix a HAM.

(d) SAT es redueix a EUL i EUL es redueix a HAM.

4. **PARTICIÓ**. Reduïu el problema SUBSET-SUM al problema de la PARTICIÓ: Donat un conjunt d'enters S (possiblement amb repeticions), trobar si aquest es pot *particionar* en dos subconjunts S_1 i S_2 (és a dir, cada element de S pertany a S_1 o a S_2, però no als dos alhora) de forma que $\sum_{x \in S_1} x = \sum_{x \in S_2} x$. Demostreu, a més, que PARTICIÓ pertany a **NP**.

5. **SUBGRAF-ESBORRAT**. Reduïu el problema de GRAF-HAMILTONIÀ a SUBGRAF-ESBORRAT: Donats dos grafs G_1 i G_2, determinar si un graf isomorf a G_1 s'obté a partir de G_2 tot esborrant arestes. Demostreu, a més, que SUBGRAF-ESBORRAT pertany a **NP**.

6. **CLIQUE**. Reduïu el problema de CONJUNT-INDEPENDENT a CLIQUE: Donat un graf i un natural k, decidir si hi ha un subconjunt de k vèrtexs tal que entre tot parell de vèrtexs d'aquest subconjunt hi ha una aresta. Demostreu, a més, que CLIQUE pertany a **NP**.

7. **SUBGRAF-INDUÏT**. Reduïu el problema de CLIQUE al de SUBGRAF-INDUÏT: Donats dos grafs G_1 i G_2, determinar si G_1 és un subgraf induït de G_2. Demostreu, a més, que SUBGRAF-INDUÏT pertany a **NP**.

8. **Un de l'Steffy**. Possiblement recordareu que en Johnny i en Roy havien quedat per anar amb la seva nombrosa colla (i l'Steffy!) a passar un cap de setmana a una casa de colònies. Avui han decidit anar a fer ràfting, però només queda una barca. A més, la barca només funciona si les persones que hi pugen pesen exactament T kilograms en total (perquè si pesen més, la barca s'enfonsa i, si pesen menys, el corrent no és prou fort per portar-los fins al final).

   Llegiu el diàleg següent, assumint que tots els personatges diuen la veritat.

   JOHNNY: Roy, demana a tothom el seu pes, per veure si val la pena que lloguem la barca.

   ROY: Aquí ho tens, ja saps que m'agrada fer llistes de la gent.

   STEFFY: D'això, nois... val més que no us hi poseu. El problema que voleu resoldre (donat un enter T i donats n enters estrictament positius $p_1, \ldots, p_n$, determinar si n'hi ha uns quants que sumen T) és **NP**-complet.

   JOHNNY: Ostres, així ja l'hem fotuda!

   ROY: Espera! Casualment, he portat una llàntia màgica que vaig comprar al mercat d'Istambul, en una botiga on parlaven català. Em van dir que si la fregues, surt un geni que resol eficientment qualsevol problema **NP**.

   JOHNNY: Va, vinga, passa-me-la!

   [En Johnny frega la llàntia i en surt un geni envoltat de fum.]

   GENI: Salutacions, noi. Si em dónes m enters $b_1, \ldots, b_m$, determinaré al moment si hi ha algun subconjunt no buit que sumi zero.

   JOHNNY: Ostres, Roy! Aquest no és el nostre problema!

   ROY: Ummm... potser aquell venedor d'Istambul em va enganyar.

STEFFY: No patiu, nois. El venedor no va mentir... Tinc la solució!

(a) Expliqueu quina solució ha trobat l'Steffy per saber si alguns d'ells poden fer ràfting (és a dir, doneu una reducció del problema d'en Johnny al problema del geni).

(b) Acabeu de demostrar que el problema del geni és **NP**-complet.