

**Cognoms**

**Nom**

**DNI**

**Examen Final EDA**

**Duració: 3h**

**08/01/2024**

- 
- *L'enunciat té 5 fulls, 9 cares, i 4 problemes.*
  - *Poseu el vostre nom complet i número de DNI a cada full.*
  - *Contesteu tots els problemes en el propi full de l'enunciat a l'espai reservat.*
  - *Llevat que es digui el contrari, sempre que parlem de cost ens referim a cost asimptòtic en temps.*
  - *Llevat que es digui el contrari, cal justificar les respostes.*
- 

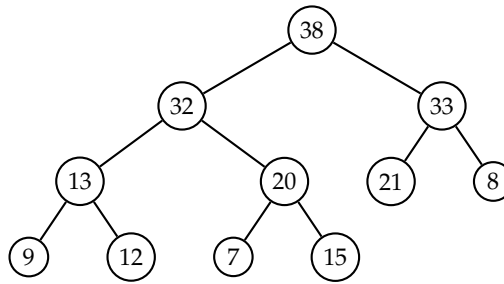
**Problema 1**

**(2 pts.)**

Responen les preguntes següents:

- (a) (1 pt.) Sabem que el cost d'un cert algorisme ve donat per la recurrència  $T(n) = aT(n/4) + \Theta(n^2)$ . Analitzeu el cost de l'algorisme en funció del paràmetre natural  $a \geq 1$ .

- (b) (1 pt.) El max-heap de la figura següent és el resultat d'una seqüència d'operacions d'inserció i esborrat-del-màxim. La darrera operació va ser una inserció.



Expliqueu per què el 38 **no** pot ser l'últim element afegit.

Escriviu la llista dels elements que **sí** poden haver estat l'últim en ser afegit; aquesta llista no cal justificar-la.

**Cognoms**

**Nom**

**DNI**

**Problema 2**

**(3 pts.)**

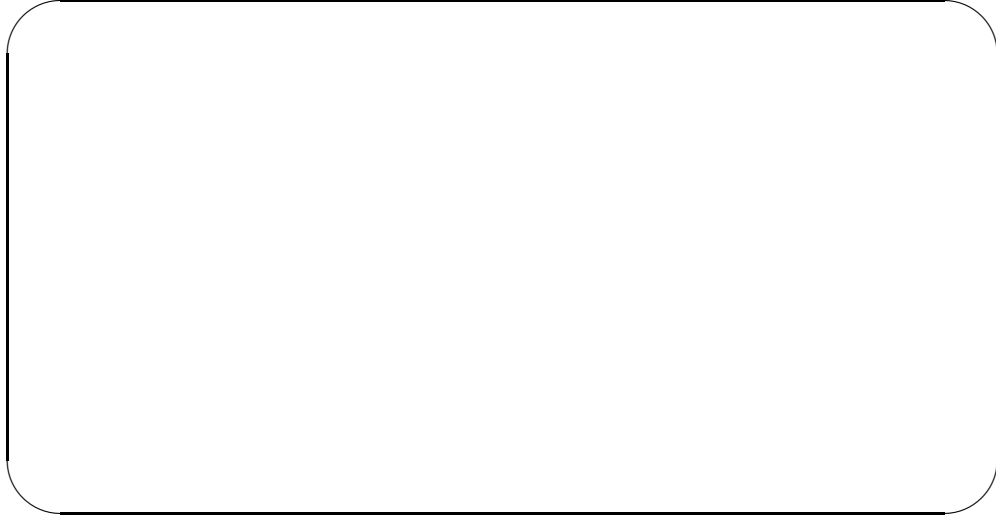
Donat un vector  $v$  d' $n$  nombres naturals, volem calcular un vector que contingui totes les parelles  $\langle z, t \rangle$  tal que el nombre  $z$  apareix a  $v$  exactament  $t$  vegades, amb  $t > 0$ . L'ordre de les parelles en el vector no ens importa. Per exemple, donat el vector  $(4, 1, 5, 1, 3, 4, 5, 1)$  un resultat correcte seria  $(\langle 3, 1 \rangle, \langle 5, 2 \rangle, \langle 1, 3 \rangle, \langle 4, 2 \rangle)$ .

(a) (1 pt.) Considereu el codi següent, que resol el problema plantejat:

```
vector<pair<int,int>> map_count (const vector<int>& v){  
    map<int,int> m; // Podeu assumir que m és un AVL  
    for (int x : v) ++m[x];  
    vector<pair<int,int>> res;  
    for (pair<int,int> p : m) res.push_back({p.first ,p.second});  
    return res;  
}
```

Quin és el cost en cas pitjor de la funció anterior en funció d' $n$ ? *Nota:* en tot aquest problema assumiu que el cost d'un *push\_back* és  $\Theta(1)$  i que recórrer els elements d'un *map* té cost lineal respecte el seu nombre d'elements. Us pot ser útil saber que  $\log 1 + \log 2 + \dots + \log n = \Theta(n \log n)$ .

- (b) (1 pt.) Assumim (**només en aquest apartat**) que tots els nombres de  $v$  són menors estrictes que 100 (que és un nombre fixat que no depèn d' $n$ ). Com aconseguiríeu resoldre el problema en temps  $O(n)$  en cas pitjor? No cal que doneu codi, amb una explicació d'alt nivell n'hi ha prou. Tampoc cal que justifiqueu el cost.

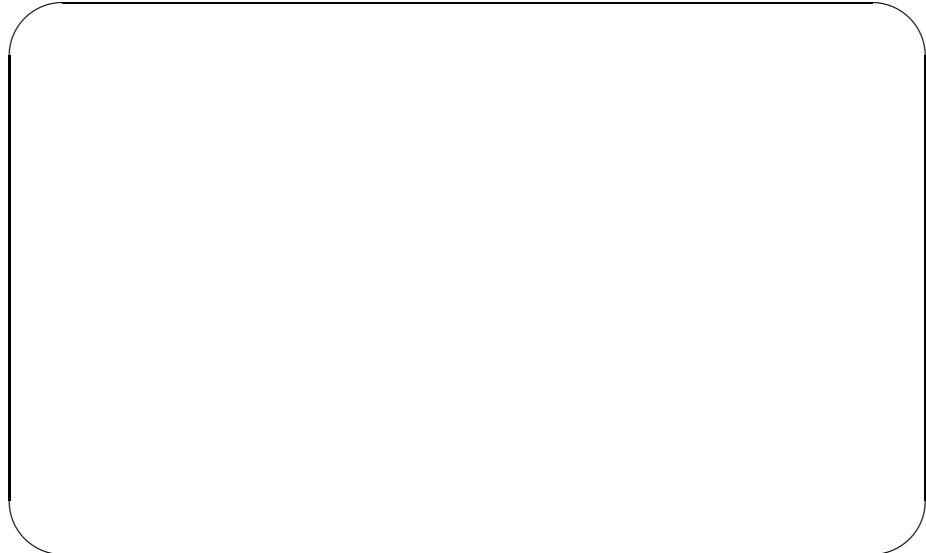


- (c) (1 pt.) Ompliu el codi següent per tal de resoldre el problema que tenim entre mans.

```
vector<pair<int,int>> priority (const vector<int>& v) {  
    priority_queue<int> q;  
    for (int x : v) q.push(x);  
    vector<pair<int,int>> res;
```



```
while (not q.empty()) {
```



```
}  
return res; }
```

**Cognoms**

**Nom**

**DNI**

**Problema 3**

**(2 pts.)**

Disposem de dues gerres  $A$  i  $B$  amb capacitat  $cap\_A > 0$  i  $cap\_B > 0$  litres, respectivament. Tenim també una font per poder omplir les gerres. L'objectiu és aconseguir tenir exactament  $k$  litres en una de les gerres amb les següents operacions:

- Omplir una de les gerres fins a dalt.
- Buidar completament una de les gerres.
- Buidar el contingut d'una gerra origen cap a una gerra destí fins que, o bé la gerra origen quedi buida, o bé la gerra destí quedi plena.

Ens demanen que calculem el mínim nombre d'operacions per aconseguir  $k$  litres en una de les gerres si comencem amb les gerres buides, o que indiquem que no és possible obtenir  $k$  litres. Per exemple, si  $cap_A = 10$ ,  $cap_B = 7$  i  $k = 4$ , ho podem fer amb 4 operacions:

1. Omplim la gerra  $B$  fins a dalt ( $A$  tindrà 0 litres, i  $B$  en tindrà 7).
2. Buidem la gerra  $B$  cap a  $A$  ( $A$  tindrà 7 litres i  $B$  estarà buida).
3. Omplim la gerra  $B$  fins a dalt (tant  $A$  com  $B$  tenen 7 litres).
4. Buidem la gerra  $B$  cap a  $A$  ( $A$  tindrà 10 litres, i  $B$  en tindrà 4).

Ompliu el codi següent per tal de resoldre aquest problema. *Pista:* Fixeu-vos que hi ha  $(cap_A + 1) \times (cap_B + 1)$  possibles estats. No esperem una solució per *backtracking*.

```
int cap_A, cap_B; // variables globals
int operacions(const pair<int,int>& ini, int k);
int main ( ) {
    int k;
    cin >> cap_A >> cap_B >> k;
    pair<int,int> ini = {0,0}; // un parell es (litres_dins_A, litres_dins_B)
    int res = operacions (ini, k);
    if (res == -1) cout << "No es poden aconseguir " << k << " litres." << endl;
    else cout << "Necessitem " << res << " operacions." << endl; }

// Omplir A
pair<int,int> mov_1 (const pair<int,int>& p) {return {cap_A, p.second};}
// Omplir B
pair<int,int> mov_2 (const pair<int,int>& p) {return {p.first , cap_B};}

// Buidar A
pair<int,int> mov_3 (const pair<int,int>& p) {return {0, p.second};}
// Buidar B
pair<int,int> mov_4 (const pair<int,int>& p) {return {p.first , 0};}
```

```
pair<int,int> mov_5 (const pair<int,int>& p) { // A cap a B
```

```
}
```

```
pair<int,int> mov_6 (const pair<int,int>& p) { // B cap a A
```

```
}
```

```
vector<pair<int,int>> un_pas (const pair<int,int>& p) {  
    return {mov_1(p), mov_2(p), mov_3(p), mov_4(p), mov_5(p), mov_6(p)}; }
```

```
// retorna el mínim nombre d'operacions o -1 si no hi ha solució
```

```
int operacions (const pair<int,int>& ini, int k) {
```

```
}
```

**Cognoms**

**Nom**

**DNI**

**Problema 4**

**(3 pts.)**

Donat un graf  $G = (V, E)$  no dirigit, el problema **3-COL** consisteix en determinar si existeix una funció  $c : V \rightarrow \{1, 2, 3\}$  de manera que per a tota aresta  $\{u, v\} \in E$  es compleixi  $c(u) \neq c(v)$ .

Donat un graf  $G = (V, E)$  no dirigit amb  $|V| = 3n$ , el problema **3-COL-EQUIL** consisteix en determinar si existeix una funció  $c : V \rightarrow \{1, 2, 3\}$  de manera que per a tota aresta  $\{u, v\} \in E$  es compleixi  $c(u) \neq c(v)$  i tal que la cardinalitat dels conjunts  $C_i = \{v \in V \mid c(v) = i\}$  sigui  $n$  per a tota  $1 \leq i \leq 3$ .

- (a) (0.5 pts.) Construïu una instància positiva de **3-COL** amb 6 vèrtexs que sigui una instància negativa de **3-COL-EQUIL**. Justifiqueu per què aquesta instància compleix el que es demana.

- (b) (1.25 pts.) Demostreu que **3-COL-EQUIL** pertany a la classe NP.

(c) (1.25 pts.) Donat  $G = (V, E)$  una instància de **3-COL** amb  $V = \{v_1, v_2, \dots, v_n\}$  podem generar una instància  $G' = (V', E')$  de **3-COL-EQUIL** de la manera següent:

- $V' = \{v_1^1, v_1^2, v_1^3, v_2^1, v_2^2, v_2^3, \dots, v_n^1, v_n^2, v_n^3\}$
- $E' = \{\{v_i^k, v_j^k\} \mid \{v_i, v_j\} \in E \text{ i } 1 \leq k \leq 3\}$

és a dir, creem 3 còpies de  $G$ .

Demostreu que la funció anterior és un reducció polinòmica de **3-COL** cap a **3-COL-EQUIL**.



**Cognoms**

**Nom**

**DNI**

*Aquesta cara estaria en blanc intencionadament si no fos per aquesta nota.*