

Cognoms**Nom****DNI****Examen Parcial EDA****Duració: 2h****02/11/2023**

-
- L'enunciat té 3 fulls, 6 cares, i 2 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**(3.5 pts.)**

Respon a les següents preguntes:

- (a) (1 pt.) Cada fila de la següent taula representa una certa etapa de l'execució d'un algorisme d'ordenació conegut (selecció, inserció, quicksort i mergesort) sobre el vector

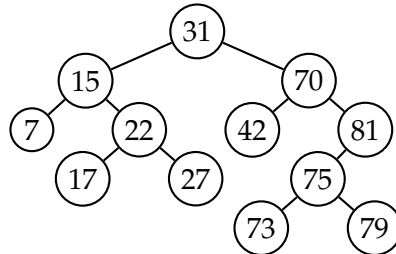
10	2	5	3	7	13	1	6
----	---	---	---	---	----	---	---

. Escriviu al costat de cada fila a quin d'aquests algorismes correspon. Cada algorisme ha d'aparèixer exactament una vegada. 1 o cap resposta correcta puntuarà 0 punts i 2 respostes correctes, 0.5 punts. Si repetiu algun algorisme, us atorgarem 0 punts. No cal que justifiqueu la vostra resposta.

2	3	5	10	1	6	7	13	
2	3	5	10	7	13	1	6	
6	2	5	3	7	1	13	10	
1	2	3	5	6	13	10	7	

- (b) (1 pt.) Ordena les funcions $(\log_2 \log_2 n)^2$ i $\log_2 n$ segons el seu creixement asimptòtic. Recorda que cal justificar la resposta.

- (c) (1.5 pts.) Sigui T un arbre binari de cerca que implementa un diccionari, on les claus són nombres enters. Donades dues claus diferents x, y , que apareixen a T , definim el *menor ancestre comú* de x i y com l'ancestre comú de x i y que té major distància respecte l'arrel. Per exemple, en el següent arbre el menor ancestre comú de 7 i 17 és el 15, el de 42 i 73 és el 70, i el de 15 i 17 és 15.



Descriviu a alt nivell (no cal que doneu codi en C++) un algorisme per calcular el menor ancestre comú de dos claus x i y que compleixen $x < y$. Es valorarà l'eficiència (no només asimptòtica) de l'algorisme.

Cognoms**Nom****DNI****Problema 2****(6.5 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 següent codi, que resol el problema plantejat:


```
vector<pair<int,int>> sort_and_process (vector<int>& v) {  
    vector<pair<int,int>> res;  
    sort(v.begin(), v.end());  
    int i = 0;  
    while (i < v.size()) {  
        int elem = v[i];  
        int times = 0;  
        while (i < v.size() and v[i] == elem) {++times; ++i;}  
        res.push_back({elem, times});  
    }  
    return res;  
}
```

Quin és el cost en cas pitjor del programa anterior en funció d' n si la funció *sort* implementa una ordenació per inserció? I si implementa un mergesort? *Nota:* en tot aquest problema assumiu que el cost d'un *push_back* és $\Theta(1)$.

(b) (2 pts.) Considerem una segona solució al mateix problema:

```
vector<pair<int,int>> mark_and_process (vector<int>& v) {  
    vector<pair<int,int>> res;  
    for (int i = 0; i < v.size (); ++i){  
        if (v[i] != -1) {  
            int times = 1, elem = v[i], j = i+1;  
            while (j < v.size ()) {  
                if (v[j] == elem) {++times; v[j] = -1;}  
                ++j;  
            }  
            res.push_back({elem,times});  
        }  
    }  
    return res;  
}
```

Quin és el cost en cas pitjor d'aquest programa en funció d' n ? I el seu cost en cas millor? Doneu dos vectors pels quals s'apliqui el cas pitjor i millor, respectivament.



Cognoms**Nom****DNI**

--	--	--

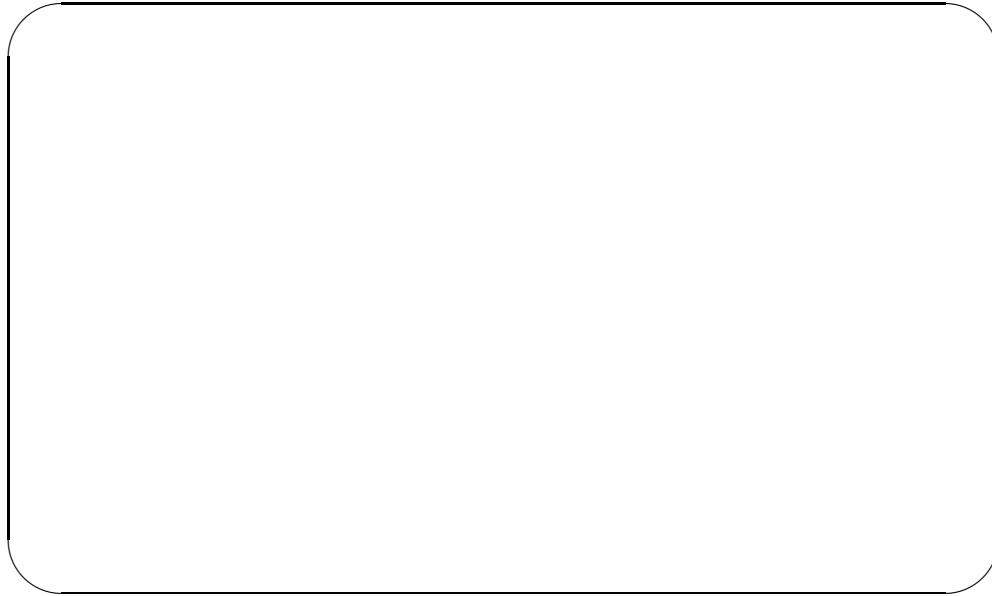
- (c) (1.5 pts.) Ens proporcionen ara una solució al problema basada en dividir i vèncer:

```
vector<pair<int,int>> combine (const vector<pair<int,int>>& v1,
                               const vector<pair<int,int>>& v2) {
    vector<pair<int,int>> res;
    vector<bool> present(v2.size (), false);
    for (int i = 0; i < v1.size (); ++i) {
        int elem = v1[i].first;
        int times = v1[i].second;
        for (int j = 0; j < v2.size (); ++j) {
            if (v2[j].first == elem) {
                times += v2[j].second;
                present[j] = true;
            }
        }
        res.push_back({elem, times});
    }
    for (int j = 0; j < v2.size (); ++j)
        if (not present[j]) res.push_back(v2[j]);
    return res;
}

vector<pair<int,int>> merge_count (const vector<int>& v, int l, int r) {
    if (r == l) return {{v[l],1}};
    else {
        int m = (l+r)/2;
        vector<pair<int,int>> r1 = merge_count(v,l,m);
        vector<pair<int,int>> r2 = merge_count(v,m+1,r);
        return combine(r1,r2);
    }
}

vector<pair<int,int>> merge_count (const vector<int>& v) {
    return merge_count(v,0,v.size ()-1);
}
```


Quin és el cost en cas pitjor d'una crida a *merge_count(v)* en funció d'*n*?



- (d) (2 pts.) Finalment, ens demanen modificar la funció *combine* per tal que una crida a *merge_count(v)* tingui cost $\Theta(n \log n)$ en cas pitjor. Ompliu el codi següent per tal que proporcioni una solució correcte i tingui el cost desitjat.

Pista: intenteu que *merge_count* retorni el vector de parelles **ordenat**.

```
vector<pair<int,int>> combine (const vector<pair<int,int>>& v1,  
                             const vector<pair<int,int>>& v2) {  
    vector<pair<int,int>> res;  
    int i = 0, j = 0;  
    while (i < v1.size () and j < v2.size ()) {
```



```
    }  
    while (i < v1.size ()) { res.push_back(v1[i]); ++i;}  
    while (j < v2.size ()) { res.push_back(v2[j]); ++j;}  
    return res; }
```

