# Problema 2.3: temps d'espera de revisions d'examen

A.22 Grup 2

## Enunciat

Un professor rep $n$ sol·licituds de revisións d'examen. Abans de començar, el professor mira la llista dels $n$ estudiants que han sol·licitat revisió i pot calcular, per a cada estudiant $i$, el temps $t_i$ que utilitzarà per atendre l'$i$-èssim estudiant. Per a estudiant $i$, el temps d'espera $e_i$ és el temps que el professor triga a revisar els exàmens dels estudiants que fan la revisió abans que $i$.

Dissenyeu un algorisme per a computar l'ordre en que s'han de revisar els exàmens dels $n$ estudiants de manera que es minimitzi el temps total d'espera: $T = \sum_{i=1}^{n} e_1$.

## Solució

### Codi en C++

```
#include<vector>
#include<algorithm>
using namespace std;

// post: es retorna l'ordre en que s'ha d'atendre els n estudiants
vector<int>& calcular_ordre(const vector<double>& t_atencio) {
    vector<pair<double,int>> v(t_atencio.size());
    vector<int> sol(t_atencio.size());

    for (int i = 0; i < t_atencio.size(); ++i) v[i] = make_pair(t_atencio[i],i);

    sort(v.begin(), v.end());

    for (int i = 0; i < t_atencio.size(); ++i) sol[i] = v[i].second;

    return sol;
}
```

### Idea general

La solució és molt senzilla i es tracta d'una aproximació golafre, basada només en ordenar els estudiants per $t_i$ creixent i antenent-los en aquest ordre, prioritzant aquells que tenen un temps d'atenció menor. En el següent apartat es veurà la correcció d'aquest procediment.

### Correcció

Sigui $\Pi(t)$ una permutació dels $n$ temps d'atenció dels estudiants, i $\Pi(t)_i$ denoti a l'estudiant $i$-èssim en aquesta permutació i el seu temps d'atenció corresponent. Anem a veure perquè aquesta permutació ha de ser la seqüència ordenada creixentment si volem tenir la solució òptima al problema.

En realitat el que es vol minimitzar és el total dels temps d'espera $T$ definit a l'enunciat. Aplicant la definició de temps d'espera, aquest sumatori és el següent:

$$T = \sum_{i=1}^{n} \sum_{i=1}^{i-1} \Pi(t)_i = (n-1) \cdot \Pi(t)_1 + (n-2) \cdot \Pi(t)_2 + \cdots \Pi(t)_{n-1}$$

En l'anterior expressió apareixen els diferents temps d'atenció dels estudiants, "multiplicats per un cert pes". D'aquesta manera, $\Pi(t)_1$ és el que té el major pes, i va decreixent fent que $\Pi(t)_n$ ni tan sols contribueixi a la suma.

És ben sabut que per a minimitzar una suma de productes així, cal assignar al major pes el valor més petit, al segon pes més gran el penúltim valor més petit, ... i el pes més petit s'emparella amb el valor més gran. Per tant, si $\Pi(t)$ és una solució òptima al problema, inevitablement es tracta de la seqüència de temps d'espera ordenada creixentment.

## Cost de l'algorisme

Cada línia del codi en la secció del codi en *C++* té cost lineal excepte

- **return**: és constant al retornar una referència i no una còpia
- **sort**: assumirem que *C++* implementa òptimament per a la tasca d'ordenar un vector, amb cost $O(n \log n)$

Cost: $O(n \log n)$