Crețu Laurențiu Vasile
CTI, Grupa 161

# Tema 2 - Greedy

## Problema 12

## Enunț:

Un bijutier are la dispoziţie N bucăţi de metal preţios. Pentru fiecare bucată i de metal el cunoaşte greutatea gi (in grame) şi preţul pi (in $). Din aceste metale el trebuie să confecţioneze un colier împletit, având greutatea G impusă de către cumpărător. Știind că bijuteria trebuie realizată cu un cost minim, să se precizeze cum foloseşte bijutierul bucăţile de metal preţios. Obs. Bucăţile de metal preţios pot fi folosite în întregime sau numai parțial.

**Date de intrare:** fișierul date.in conține:

- pe prima linie două numere naturale N și G, cu semnificația din enunț; - pe următoarele N linii triplete de numere naturale de forma id g p, unde unde id reprezintă identificatorul unui metal, g greutatea acestuia, iar p prețul sau integral.

**Date de ieșire:** fișierul date.out conține pe fiecare linie identificatorul unui metal selectat și procentul în care au fost inclus în bijuterie. Pe ultima linie se scrie valoarea totală a acesteia.

**Exemplu:**

| date.in | date.out |
|---|---|
| 4 10<br>1 2 3<br>2 6 10<br>3 3 15<br>4 8 13 | 1 100%<br>4 100%<br>16.26 |

## Metoda de rezolvare:

Metoda de rezolvare folosită este una de tip Greedy, având o complexitate $\mathcal{O}(n \cdot \log_2 n)$ egală cu complexitatea algoritmului de sortare, ce este unul de tip Merge Sort. Ideea principală a algoritmului implementat este bazată pe sortarea metalelor după prețul lor per gram și alegerea primelor k metale a căror sumă a greutăților este egală cu greutatea G impusă de cumpărător. Un metal este definit drept o structură, având ca și câmpuri: indexul, greutatea totală, prețul și greutatea unui gram.

### *Definirea structurii:*

```
typedef struct{
    int index;
    float greutate;
    float pret;
    float pret_gram;
}metal;
```

## *Algoritmul de sortare:*

```
void interclasare(metal *v,int st,int mij,int dr){
    int i,j,k;
    metal *aux = (metal*)malloc((dr-st+1)*sizeof(metal));
    i = st; j = mij+1; k = 0;
    while (i<=mij && j<=dr)
        if (v[i].pret_gram<v[j].pret_gram)
            aux[k++] = v[i++];
        else
            aux[k++] = v[j++];
    while (i<=mij)
        aux[k++] = v[i++];
    while (j<=dr)
        aux[k++] = v[j++];
    k = 0;
    for (i = st; i<=dr; i++)
        v[i] = aux[k++];
    free(aux);
}

void mergeSort(metal *v,int st,int dr){
    if (st<dr){
        int mij = (st+dr)/2;
        mergeSort(v,st,mij);
        mergeSort(v,mij+1,dr);
        interclasare(v,st,mij,dr);
    }
}
```

## *Algoritmul Greedy:*

```
void greedy(metal *v,int n,float g){
    FILE *fout = fopen("date.out","w");
    float pretTotal = 0;
    float cantitateTotala = 0;
    for (int i=0; i<n; i++){
        if (cantitateTotala+v[i].greutate<=g){
            cantitateTotala += v[i].greutate;
            pretTotal += v[i].pret;
            fprintf(fout,"%d 100%%\n",v[i].index);
        }
        else{
            float cantitateFractionara = 100*(g-
cantitateTotala)/v[i].greutate;
```

```
            pretTotal += (g-cantitateTotala)*v[i].pret_gram;
            cantitateTotala = g;
            fprintf(fout,"%d %f%%\n",v[i].index,cantitateFractionara);
        }
        if (cantitateTotala==g)
            break;
    }
    fprintf(fout,"%.2f",pretTotal);
    fclose(fout);
}
```

## *Generarea datelor de test:*

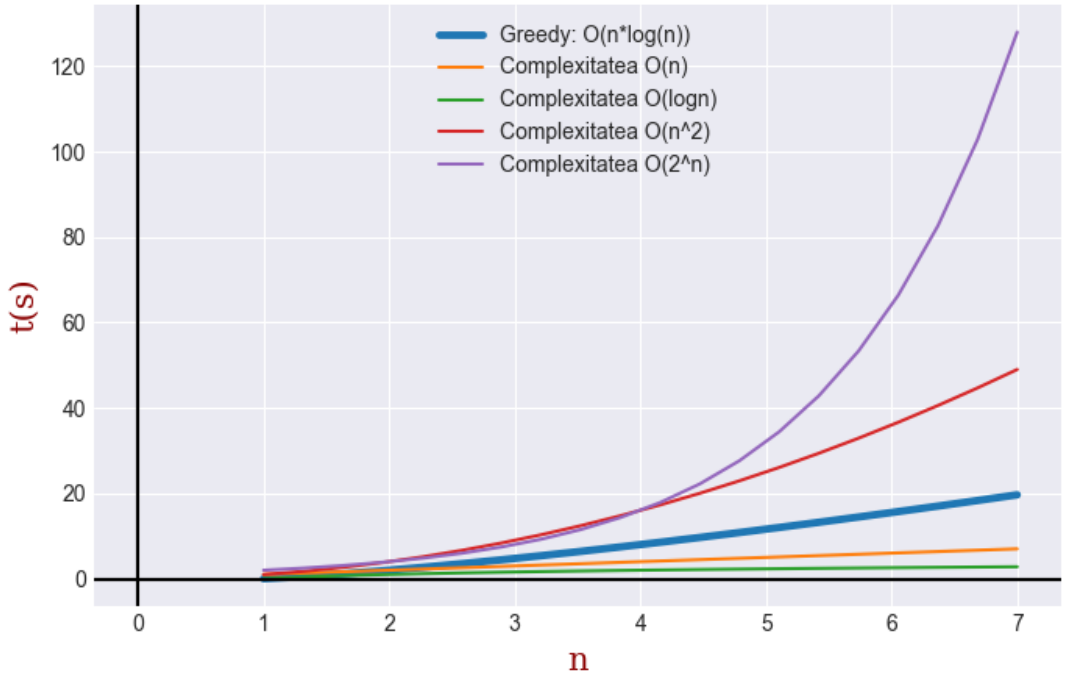
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define LOWER 1
#define UPPER 10000

typedef struct{
    int index;
    float greutate;
    float pret;
}metal_pretios;

void generare(){
    FILE *fin = fopen("date.in","w");
    srand(time(0)); rand();
    int n = rand()/(RAND_MAX/UPPER)+LOWER;
    metal_pretios *v = (metal_pretios*)malloc(n*sizeof(metal_pretios));
    float cantitate = 0;
    for (int i=0; i<n; i++){
        v[i].index = i+1;
        v[i].greutate = (float)rand()/(float)(RAND_MAX/(UPPER+1));
        v[i].pret = (float)rand()/(float)(RAND_MAX/(UPPER+1));
        cantitate += v[i].greutate;
    }
    float g = (float)rand()/(float)(RAND_MAX/(cantitate+1));
    fprintf(fin,"%d %.2f\n",n,g);
    for (int i=0; i<n; i++)
        fprintf(fin,"%d %.2f %.2f\n",v[i].index,v[i].greutate,v[i].pret);
    fclose(fin);
    free(v);
}
```

# Analiza timpului de executare:

| Nr. Crt. | Numarul de pietre N | Timpul de executare T |
|---|---|---|
| 1 | 6 | 0.000287s |
| 2 | 53 | 0.000717s |
| 3 | 86 | 0.000379s |
| 4 | 658 | 0.001296s |
| 5 | 921 | 0.001630s |
| 6 | 6839 | 0.008572s |
| 7 | 7305 | 0.010533s |
| 8 | 23196 | 0.031472s |
| 9 | 126064 | 0.162528s |
| 10 | 843095 | 1.202296s |

* Date calculate pe Linux 5.10.0-kali3-amd64, compilator gcc 10.2.1 20210110

# Complexitatea în timp a algoritmului raportată la alte complexități:




* Grafic creat în Python 3.8.5 folosind librăriile Numpy 1.20.2 și Matplotlib 3.4.1