Nume: Fusneica Florentin-Cristian
Grupa: 241

Tema 1. Algoritmi Avansati

1) *Fie un șir de numere naturale $S=\{s_1, s_2, \ldots, s_n\}$ și un număr natural K, cu $K \geq s_i$ pentru orice i între 1 și n.*
    a) *Să se scrie un algoritm pseudo-polinomial care găsește suma maximă, dar care să fie ≤K, ce poate fi formată din elemente din S (numere întregi, pozitive, luate cel mult o singură dată).*

```
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

ifstream f("date.in");

unsigned int suma_maxima(int K, const vector<unsigned int> valori)
{
    unsigned int maximul = 0;
    vector <unsigned int> sume;
    sume.push_back(0);
    for(unsigned int valoare: valori)
        for(unsigned int suma: sume)
            if(suma + valoare <= K){
                maximul = max(total, suma + valoare);
                sume.push_back(suma + valoare);
            }
    return maximul;
}

int main()
{
    int K, val;
    vector <unsigned int> S;

    f >> K;
    while(f >> val)
        S.push_back(val);

    cout << suma_maxima(K, S);
}
```

K si S sunt valorile date in cerinta (valoarea maxima la care poate ajunge suma, si numerele cu care lucram).

Subprogramul suma_maxima() va returna suma maxima pe care o poti forma din elementele din S (folosind numerel o singura data). In acel for dublu se vor forma toate sumele dintre numere <= K, iar maximul va tine minte maximul dintre sume. vectorul “sume” va fi “refolosit”, adaugand la fiecare termen din vectorul “valori” (care ia locul multimii de numere S din cerinta) noile posibilitati de sume.

Exemplu pe datele de intrare: 16 1 4 9 1 8, raspunsul va fi 14.

b) ***Să se găsească un algoritm aproximativ care calculează o sumă cel puțin pe jumătate de mare ca cea optimă dar rulează în timp O(n) și complexitate spațiu O(1). Mai exact: aveți voie să parcurgeți fiecare element din S cel mult o singură dată, respectiv aveți memorie alocată doar pentru 3 variabile de tip int (dintre care una este K) + varabile de tip stream***

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream f("date.in");
    int K, s, maximum = 0;

    f >> K;

    while(f >> s)
        if(maximum + s <= K)
            maximum += s;
        else if(maximum < s)
            maximum = s;

    cout << maximum;
}
```

Vom primi elementele din fisier: in cazul in care suma elementelor pana la un anumit punct, maximul va primi acea valoare; daca cumva valoarea urmatoare este <= K, adunata cu maximul gasit pana in punctul acela ar depasi valoarea K si totodata este mai mare decat maximul, atunci maximul va lua acea valoare.
Prin al doilea “if” ne asiguram ca fie maximul, fie acel numar care urmeaza ar fi mai mare decat ½ * K: daca nu, asta ar insemna ca numarul care este verificat fie nu ar fi fost adaugat

inainte la maximum, fie este mai mic ca maximum (ceea ce nu ne intereseaza), dar nu o sa fie cazul sa nu fie adaugat maximum in cazul in care inca i se permite.