Portofoliu Algoritmi și complexitate - Probleme rezonabile

Anul I, semestrul 2

Numele tău aici

Contribuitori:
- Sabina
- Carla

- Dacă vreți probleme rezolvate mai grele intrati aici;
- Dacă nu înțelegeți ceva întrebați-mă;
- Vă invit sa mai scrieți și problemele voastre (da, știu);
- Nu garantez că problemele puse de alții sunt corecte (să citiți și cerințele);
- Dacă găsiți ceva greșit spuneți-mi;
- O parte din comentarii sunt pentru amuzamentul meu, iar în restul incerc să explic ce face (și de ce face) programul;
- Voi mai actualiza acest pdf cand mai primim fișe de laboratoare sau cand mai primesc exercitii rezolvate de la cineva. Deci intrați din nou pe acelasi link (aici).

z sercem, Pąblo.

Laborator 1

8. Aproximați cu precizia ε limita la $+\infty$ a șirului $(s_n)_{n\in\mathbb{N}^*}$, definit prin

$$x_n = \left(1 + \frac{1}{n}\right)^n, n \geq 1.$$

```cpp
#include <iostream>
//includem cmath pentru ca avem nevoie de 'pow'
#include <cmath>

// 'using namespace std;' nu e cea mai bună practică,
// în special pentru proiecte mai mari.
// De exemplu, daca definim o functia asta:
//double riemann_zeta(double n) {
//   std::cout << "majie";
//   return 0;
//}
// și în main avem asta:
//cout << riemann_zeta(-1) << end;
// Când rulăm programul, pe ecran se afiseaza "majie0", are sens nu?
// Asta se intampla daca compilăm cu versiuni mai vechi de c++17
// (o data cam la 3 ani mai apare o versiune de c++).
// Dar daca compilăm cu c++17 (sau mai recent), pe ecran apare
// "-0.0833333". De ce?
// Păi din c++17 în "cmath" mai este definã și funcția "riemann_zeta",
// care sigur nu e definită ca funcția noastră (deși tot 'majie' e și
// acolo).
// Iar daca nu știați asta și încercați ceva de genul ăsta, rămaneți
// fară păr în cap (sau fără tastatură) până aflați de ce nu va merge
// programul cum trebuie.
// Dar pentru ce facem noi la info, să zicem că e ok să scriem asta:
using namespace std;

// E (foarte) posibil ca aceste comentarii sa fie foarte evidente dar
// nu știu altceva ce sa scriu.
int main() {
    // x0 ar fi x_{n-1} si x1, x_n
    double eps, x0 = 0, x1 = 0;
    // delta e diferența intre termenii consecutivi
    // o initializam cu un numar mare care sigur e mai mare decat eps
    double delta = 10000;
    cin << eps;

    int n = 1;
    // daca ne simțeam aventuroși puteam scrie
    // for (int n = 1; delta > eps; n++) {
    while (delta > eps) {
        // am incrementat n, deci x_{n-1} devine x_n
        x0 = x1;
        // implementam definitia care ni se da
```

```
        x1 = pow(1 + 1.0 / n, n);
        n++;
        // actualiazm delta
        delta = x1 - x0;
    }
    cout << x1 << endl;

    return 0;
}
```

1. (Carla) BMI

```
#include <iostream>
using namespace std;

// Pąblo: good luck
int main() {
    unsigned kg, m;
    float ma = 0;
    cin >> kg >> m;
    ma=kg/m^2;
    if (ma<18.5)
        cout << "risc minim pentru sanatate" << endl;
    else if (ma>=18.5 && ma<25)
        cout << "risc minim/scazut pentru sanatate" << endl;
    else if (ma>=25 && ma<30)
        cout << "risc scazut/moderat pentru sanatate" << endl;
    else if (ma>=30 && ma<35)
        cout << "risc moderat/ridicat pentru sanatate" << endl;
    else cout << "risc ridicat pentru sanatate" << endl;
    return 0;
}
```

2. (Carla) functia

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    float x, f=0;
    cin >> x;
    if (x>=-5 && x<=5)
        f=sqrt(abs(x))+6;
    else f=5*x*x+7;
    cout << f << endl;
    return 0;
}
```

3. (Carla) alta functie

```
int functie(int x, int y, int a, int b) {
    int c=0;
    if (abs(x)<=8 && abs(y)<=8)
        c=a*b*sqrt(x*x+y*y)*sin(x*x+y*y);
    return c;
}
```

4. (Carla) cifra destinului (uau)

```
#include <iostream>
#include <cmath>
using namespace std;

int nastere(int zi, int luna, int an) {
    int x = 0;
    while (zi) {
        x=x+zi%10;
        zi=zi/10;
    }
    while (luna) {
        x=x+luna%10;
        luna=luna/10;
    }
    while (an) {
        x=x+an%10;
        an=an/10;
    }
    return x;
}

int destin(int x) {
    unsigned s=0;
    if (x<10) return x;
    else while (x) {
        s=s+x%10;
        x=x/10;
    }
    return s;
}
int main() {
    int a,b,c;
    cin >> a >> b >> c;
    int x=nastere(a,b,c);
    cout << destin(x) << endl;
    return 0;
}
```

5. (Carla) ăla cu multe subpuncte

a)

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main () {
    float a, b, c, d, x1, x2;
    cin >> a >> b >> c; // coeficientii
    d=b*b-4*a*c; //delta
    x1=(-b-sqrt(d))/2*a; //prima solutie
    x2=(-b+sqrt(d))/2*a; // a doua solutie
    cout << x1 << ' ' << x2 << endl;
    return 0;
}
```

b)

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    float x, n, max=-100000, min=100000;
    cin >> n; //numarul de elemente
    while (n) {
        cin >> x;// element cu element
        if (max<x)
            max=x;
        if (min>x)
            min=x;
        n--;
    }
    cout << max << ' ' << min << endl;
    return 0;
}
```

k)

```cpp
#include <iostream>
using namespace std;

int main() {
    unsigned n,d;
    cin>>n;
    for(d=2;d<=n/2;d++)
        if(n%d==0)
            cout<<d<<" ";
```

```
    cout << endl;
    return 0;
}
```

6. (Carla) nr perfect

```
#include <iostream>

using namespace std;


// Pąblo: poate fi bool, avem și d-ăștea in c++
int perfect(int n) {
    int d;
    int s = 1;
    if (n<=2) return 0;
    else {
        for (d=2; d*d<=n; d++)
            if (n%d==0) {
                s=s+d;
                if (n/d!=d)
                    s=s+(n/d);
            }
        if (n==s)
            return 1;
    }
    return 0;
}
int main() {
    int n;
    cin >> n;
    if (perfect(n))
        cout << n << " este perfect" << endl;
    else cout << n << " nu este perfect" << endl;

    return 0;
}
```

10. (Carla) Operatii cu multimi

Apartenenta

```
#include <iostream>

using namespace std;

int main() {
    int n, ok=0, c;
    cin >> n;
    int v[n+1];
```

```
    for (int i=1;i<=n;i++)
        cin >> v[i];
    cin >> c;
    for (int i=1;i<=n;i++)
        if (v[i]==c)
            ok=1;
    if (ok==1)
        cout << "apartine" << endl;
    else cout << "nu apartine" << endl;

    return 0;
}
```

Reuninunea

```
#include<iostream>
using namespace std;

int main() {
    int a[20], b[20], c[50], i, j, m, n, k, gasit;
    cin>>n;
    for (i=0; i< n; i++) {
        cin>>a[i];
    }
    cin>>m;

    for(i=0; i<=m-1; i++) {
        cin>>b[i];
    }

    k=0;
    for(i=0; i<=n-1; i++) {
        gasit=0;
        for(j=0; j<=m-1 && !gasit; j++)
            if (a[i]==b[j])
                gasit=1;
        if(!gasit)
            c[k++]=a[i];
    }

    for (i=0; i<=m-1; i++)
        cout << b[i] << " ";
    for (i=0; i<k; i++)
        cout<<c[i]<<" ";

    return 0;
}
```

Laborator 2

2. (Inspirat de Sabina) Suma cifrelor - corectitudinea

```cpp
#include <iostream>
using namespace std;

int main() {
    // sper ca intelegeți ce face codul asta,
    // daca nu incercati sa dati un exemplu pe o foaie
    // și efectuati pas cu pas etapele algoritmului
    // (puteti citi și ce scrie mai jos, deși nu cred că ajută)
    int n, s = 0;
    cin << n;
    int i = 0;
    while (n > 0) {
        // se putea și cu 's += n%10; n /= 10;'
        s = s + n % 10;
        n = n / 10;
        i++;
    }
    cout << "suma = " << n << endl;
    return 0;
}
```

I. Parțial corectitudinea

Considerăm aserțiunile de intrare și ieșire:

$P_{in} = \left\{ n = \sum_{j=0}^{k} c_j 10^j;\ c_j \in \overline{0,9},\ \forall j \in \overline{0,k};\ c_k \neq 0 \right\},$

$P_{out} = \left\{ s = \sum_{j=0}^{k} c_j \right\}.$

Alegem proprietatea:

$I = \left\{ n = \sum_{j=0}^{k-i} c_{i+j} 10^j; s = \sum_{j=0}^{i-1} c_{i-1-j} \right\}.$

La intrarea in buclă:

$i = 0$

$n = \sum_{j=0}^{k} c_j 10^j$

Deci propoziția $I = \left\{ n = \sum_{j=0}^{k} c_j 10^j; s = \sum_{j=0}^{-1} c_{-1-j} = 0 \right\}$ este adevărată.

Arătăm că propoziția I este invariantă.

Presupunem I adevărata la începutul iterației și $n \neq 0$; demonstrăm I adevărata la sfârșitul iterației.

$n = \sum_{j=0}^{n-i} c_{i+j} 10^j;\ s = \sum_{j=0}^{i-1} c_{i-1-j}$

10
```
s = s + n % 10;
```

$s = \left(\sum_{j=0}^{i-1} c_{i-1-j}\right) + c_i = \sum_{j=0}^{i} c_{i-1-j}$

11
```
n = n / 10;
```

$n = \left[\left(\sum_{j=0}^{k-i} c_{i+j}10^j\right) / 10\right] = \left[\sum_{j=0}^{k-i} c_{i+j}10^{j-1}\right] = \left[\sum_{j=1}^{k-i} c_{i+j}10^{j-1}\right] + \left[c_i 10^{-1}\right]$
Cum $0 \le c_i \le 9 \implies 0 \le c_i 10^{-1} \le 0.9 \implies \left[c_i 10^{-1}\right] = 0.$
Deci $n = \left[\sum_{j=1}^{k-i} c_{i+j}10^{j-1}\right] = \sum_{j=1}^{k-i} c_{i+j}10^{j-1} = \sum_{j=0}^{k-i-1} c_{i+j+1}10^j.$

12
```
i++;
```

Scriem *res* și n în funcție de noul i. Deci i devine $i-1$.
$s = \sum_{j=0}^{i-1} c_{i-1-j}$
$n = \sum_{j=0}^{k-(i-1)-1} c_{i-1+j+1}10^j = \sum_{j=0}^{k-i} c_{i+j}10^j$
Deci I adevărata și la sfârșitul iterației.

La ieșirea din buclă:
$i = k+1$
$n = \sum_{j=0}^{k-(k+1)} c_{k+1+j}10^j = \sum_{j=0}^{-1} c_{k+1+j}10^j = 0$
$s = \sum_{j=0}^{k+1-1} c_{k+1-1-j} = \sum_{j=0}^{k} c_{k-j}$
Deci $P_{out} = \left\{ s = \sum_{j=0}^{k} c_{k-j} \right\}$ adevărată.
În concluzie algoritmului este parțial corect.

II. Total corectitudinea

Considerăm funcția $t : \mathbb{N} \to \mathbb{N}$, $t(i) = k+1-i$;
$t(i+1) - t(i) = k+1-(i+1)-(k+1-i) = -1 < 0$, deci t monoton strict descrescătoare.
$t(i) = 0 \iff i = k+1 \iff n = \sum_{j=0}^{-1} c_{k+1+j}10^j = 0 \iff$ condiția de ieșire din buclă.
În concluzie algoritmului este total corect.

Laborator 3

2. (Inspirat de Sabina) Descrieți un algoritm pentru calculul produsului scalar a doi vectori din $\mathbb{R}^n$...

(1) $ps = \langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$

```cpp
#include <iostream>
using namespace std;

// e important sa numerotați liniile aici
int main() {
    int n, x[100], y[100];
    // citim marimea vectorului
    cin >> n;
    // citim vectorul, chestie destul de standard
    cout << "x = ";
    for (int i = 0; i < n; ++i)
        cin >> x[i];
    cout << "y = ";
    for (int i = 0; i < n; ++i)
        cin >> y[i];

    // in variabila asta memoram produsul scalar
    int ps = 0;
    // (vezi formula de sus)
    // sa spunem ca linia asta e ż (15, sau cat e ea)
    for (int i = 0; i < n; ++i) // ż+1
        ps = ps + x[i] * y[i];  // ż+2

    // nici 'endl' nu e cea mai grozava chestie:
    cout << ps << endl;
    return 0;
}
```

operația	cost	nr repetari	cost total
ż+2	3	n	3n

$T(n) = 3n$

Laborator 4

4. (Inspirat de Sabina) Cele mai mici 2 elemente dintr-o secvență

```cpp
#include <iostream>
using namespace std;

int main() {
    int n, x[100];
    cin >> n;
    //
    for (int i = 0; i < n; ++i)
        cin >> x[i];

    // nu mai este 1989, putem declara variabile si la mijlocul functiei
    int m1, m2;
    // presupunem ca utilizatorul e rezonabil (nu e) si spunem ca n >= 2

    // in m1 memoram cel mai mic element si in m2 al 2-lea cel mai mic,
    // deci: m1 ≤ m2

    // initializam cel m1, m2 cu primele 2 elemente din secvența
    m1 = x[0];
    m2 = x[1];


    // ne asiguram ca m1 si m2 sunt in ordinea asteapata
    // (adica m1 ≤ m2), iar daca nu le interschimbăm
    if (m2 < m1) {
        int t = m1;
        m1 = m2;
        m2 = t; //in loc de t putem folosi x[0]
    }
    for (int i = 2; i < n; i++) {
        // daca x[i] e mai mic decat m1, atunci sigur e mai mic si decat
        // m2, asadar curentul m1 e al doilea cel mai mic nr din secvența
        // (pana acum), deci il punem in m2
        // deoarece x[i] este cel mai mic, il punem in m1

        if (x[i] <= m1) { // linia ż
            m2 = m1;
            m1 = x[i];
        }
        // aici am pus '<' si mai sus '<=' deoarece e posibil sa avem
        // de doua ori cea mai mica valoare, si atunci m1 = m2 = min{x}
        // daca ajungem sa comparam x[i] cu m2, inseamna ca x[i] > m1
        else if (x[i] < m2) { // linia λ
              // daca ajungem aici, x[i] este al doilea cel mai mic
              // element din secventa, si il punem in m2
              m2 = x[i];
```

```
47          }
48      }
49      // afisam ce trebuie (primele 2 elemente)
50      cout << m1 << " " << m2 << endl;
51
52      // o alta metoda de rezolvare a problemei ar fi sa sortam secvența
53      // pana la al 2-lea element si sa afisam primele 2 elemente:
54      // (tot Θ(n))
55      //for (int i = 0; i < 2; i++)
56      //    for (int j = 0; j < n; j++)
57      //        if (x[j] < x[i]) {
58      //            int t = x[i]; x[i]= x[j]; x[i] =t;
59      //        }
60      // cout << x[0] << " " << x[1];
61
62      return 0;
63  }
```

Cazul cel mai favoriabil (m1 $>=$ x[i]) $\forall i \in \{2, ...n-1\}$: T(n) = n-2 (se execută doar linia ż).
Cazul cel mai defavorabil (m1 $<$ x[i]) $\forall i \in \{2, ...n-1\}$: T(n) = 2n-4 (se execută linia ż și λ).
Mereu $T(n) \in \Theta(n)$.

Laborator 5

1. La o stație meteo ....

Notam cu p_i presiunea din ziua i si cu t_i temperatura din ziua i, $i \in \overline{1,n}$.
Notam $i \preceq j$ daca ziua i ar trebui sa apara inaitea zilei j:

$$i \preceq j \iff (t_i < t_j) \vee ((t_i = t_j) \wedge (p_i > p_j))$$

```cpp
#include <iostream>
using namespace std;
struct Zi {
    int temp;
    int presiune;
};

int main() {
    int n;
    Zi x[100];


    cin >> n;

    //citim elementele tabloului
    for (int i = 0; i < n; ++i)
        cin >> x[i].temp >> x[i].presiune;

    // daca vreți alt algoritm de sortare inlocuiți aici:
    for (int i = 0; i < n; ++i) {
        for (int j = i+1; j < n; ++j) {
            // sortam cu relatia de ordine ≼ de mai sus
            // (practic sortam cum spune cerința: mai intai crescator
            // dupa temperatura, iar daca temperaturile sunt egale,
            // descrescator dupa presiune)
            if (x[i].temp > x[j].temp || (x[i].temp == x[j].temp &&
                x[i].presiune < x[j].presiune)) {
               Zi t = x[i];
               x[i] = x[j];
               x[j] = t;
            }
        }
    }
    // afisam vectorul dupa ce l-am sortat
    for (int i = 0; i < n; ++i)
       cout << "t:" << x[i].temp << ", p: "<< x[i].presiune << endl;

    return 0;
}
```

Laborator 6

5. Ackermann

```
// Asta scrie in definitie, asta am scris.
int A(int m, int n) {
    if (m == 0) {
        return n + 1;
    }
    else {
        if (n == 0) return A(m-1, 1);
        else return A(m-1, A(m, n-1));
    }
}
```

8. Baza 2

```
#include <iostream>
using namespace std;
// 47.145518,27.6036255  bdm tss

// in tabloul s stocam caracterele corespunzatoare cifrelor lui n in
// baza 2 in ordine inversa, daca n are z cifre in baza 2 atunci,
// s[0] = a (z-1)-a cifră, s[1] = a (z-2)-a cifră... s[z-1] - prima
// cifra, daca citim de la dreapta la stanga

// functia returneaza cate cifre are n in baza 2 (z de mai sus), si
// implicit, pana unde am modificat vectorul s
int baza2(int n, char s[], int i) {
    if (n == 0) return i;
    // se poate si mai scurt ( s[i] = '0' + n%2;)
    if (n % 2 == 0)
        s[i] = '0';
    else s[i] = '1';
    return baza2(n/2, s, i+1);
}

int main() {
    char s[100];
    int n;
    cin >> n;

    int len = baza2(n, s, 0);
    // afisam cifrele in ordinea corecta
    for (int i = len - 1; i >= 0; i--)
        cout << s[i];
    cout << endl;
    return 0;
}
```

Laborator 7

2. Fibonacci

```cpp
#include <iostream>
#include <cmath>
using namespace std;


// pe scurt: "asta zîce în curs"
double putere(double x, int n) {
    if (n == 1) return x;
    double r = putere(x, n / 2);
    r = r * r;
    if (n % 2 == 1) r = r * x;
    return r;
}
// implementam formula data in cerinta
int fib(int n) {
    // daca vreti sa mearga și fib(0)
    // (momentan pt pow(x, 0), se blocheaza până dă stack oveflow),
    // scrieti si linia asta:
    //if (n==0) return 0;

    // puteti defini o variabila 'double sqrt5 = sqrt(5);'
    return round(1.0 / sqrt(5) *
                 (putere((1+sqrt(5))/2, n) - putere(1-sqrt(5)/2, n)));
    // varianta mai eficienta
    // return round(1.0 / sqrt(5) * (putere((1+sqrt(5))/2, n)));
}
int main() {
    int n;
    cin >> n;
    cout << fib(n) << endl;
    return 0;
}
```

5. (inspirat de Carla) exista v[x] == x

```cpp
#include <iostream>
using namespace std;
/* m = mijloc
pe scurt: ne utitam la mijloc si daca v[m] = mi, gata, returnam mjlocul
daca v[m] < m ne uitam in stanga, iar daca v[m] > m ne uitam in dreapta
cazul trivial: daca avem doar un element (stanga == dreapta)
verificam daca v[stanga] == stanga iar daca nu e, atunci returnam -1,
valoare imposibila ca index (care ar trebui sa fie mai mare de 0)
*/
int cauta(int v[], int stanga, int dreapta) {
    if (stanga == dreapta) {
        if (v[stanga] == stanga) return stanga;
        // am putea omite else-ul
        else return -1;
    }
    int mijloc = (stanga+dreapta)/2;
    if (v[mijloc] == mijloc)
        return mijloc;
    if (v[mijloc] < mijloc)
        return cauta(v, mijloc, stanga);
    else return cauta(v, dreapta, mijloc);

}
int main() {
    int n, v[100];
    cin << n;
    for (int i = 0; i < n; i++)
        cin << v[i];
    cout << cauta(v, 0, n) << endl;
    return 0;
}
```

Laborator 8

6. Problema aia lunga cu integrala definita

```cpp
// puteți sa nu scrieți liniile astea 2 si functia main
#include <iostream>
using namespace std;


//aici definim functia ce va fi integrata
double f(double x) {//f(x) = x^2
     // mai schimbati si voi putin expresia asta, cum ar fi
     // return x; return 1; return sin(x); ....
     return x * x;
}

// Se poate face si ceva de genu asta ca f sa fie parametru:
//template<typename F>
//double integrate(F f, double a, double b, double eps = 1e-5) {
// sau așa:
//double integrate(double (*f)(double), double a, ...

// daca vreti sa va simtiti mai romani puteti scrie "integreaza"

// asta e formula data in cerinta
double integrate(double a, double b, double eps) {
    double delta = b - a;
    if (delta < eps) {
        return delta * (f(a) + f(b)) / 2;
    }
    else {
        double c = (a+b)/2;
        return integrate(a, c, eps) + integrate(c, b, eps);
    }
}
int main() {
     double a, b, eps;
     cin >> a >> b >> eps;
     // cu varianta mai fainã merge și ceva de genu ăsta:
     // cout << integrate(sin, a, b) << endl;
     // doar că n-ar trebui să stim d-astea

     cout << integrate(a, b, eps) << endl;

     return 0;
}
```

Laborator 9

5. numere de 4 cifre cu suma cifrelor 11

```cpp
#include <iostream>
using namespace std;

// blasfemie, x ar trebui sa fie parametru pt functii,
// dar asa ne spune in curs...
int x[4], n4;

// am putea declara si asa:
//constexpr int n = 4;
//int x[n];
// dar nici asta n-ar trebui sa stim, si tot blasfemie e

// "asa spune si in curs"
void afiseaza() {
    for (int i = 0; i < n; i++)
       cout << x[i];
    cout << " ";
}
bool valid() {
    int s = 0;
    for (int i = 0; i < n; i++)
         s += x[i];
    return s == 11;
}

// "asa spune in curs"
void btr(int k) {
    //int i = 0;
    //if (k == 0) i = 1;
    //for (; i < 10; i++) {...

    //varianta mai hardcore ar fi:
    //for (int i = k == 0; i < 10; i++) {...

    for (int i = 0; i < 10; i++) {
        // numerele nu incep cu 0
        // sunt metode mult mai eficiente pt verificarea asta
        // (vezi inceputul functiei)
        if (k == 0 && i == 0) continue;

        // putem verifica pe parcurs daca suma e mai mare decat 11,
        // si daca e, trecem la urmatorul
        // "exercitiu cititorului"
        x[k] = i;
        if (k == n - 1) {
            // valid() == true e redundant
```

```
            // 'valid()' deja e bool si 'valid() == true' e tot bool
            // si nu face absolut nimic in plus
            // if (valid()) e mai de bun simț
            if (valid() == true)
                afiseaza();
        }
        else {
            btr(k+1);
        }
    }
}

int main() {
    btr(0);
    return 0;
}
```

10. suma numerelor = n

```
#include<iostream>
using namespace std;
int n, nr_sol, sol[20];

void afis(int l){
    nr_sol++;
    cout<<"Solutia nr. "<< nr_sol<<" : ";
    for(int i = 0; i < l; i++)
        cout<<sol[i]<<" ";
    cout<<endl;
}

void back(int i, int n) {
    if (n == 0) {
        afis(i);
        return;
    }
    for(int j = 1; j<= n; j++) {
        sol[i]=j;
        back(i+1, n-j);
    }
}

int main(){
    cin>>n;
    nr_sol=0;
    back(0,n);
    cout<<nr_sol<<" solutii"<< endl;
    return 0;
}
```

Laborator 10-11

5. Gidul turistic și drumețiile sale

```cpp
#include <iostream>
using namespace std;

struct Drumetie {
    int nr, s;
};
// "asa ne spune in curs", dar f = s+n
// (final = start + nr de zile per calatorie)

// am putea pune codul de aici in 'planificare', dar banuiesc ca
// asa ați face majoritatea. (ceea ce nu e așa de rău)
void sortare(Drumetie v[], int m) {
    // aici putem sa le sortam după ziua de incepere, pt ca ziua
    // de final e mereu 's+n'
    for (int i = 0; i < m; i++) {
        for (int j = i+1; j < m; j++) {
            if (v[j].s < v[i].s) {
                Drumetie t = v[i];
                v[i] = v[j];
                v[j] = t;
            }
        }
    }
}

// funcția retuneaza indicele ultimului spectacol, ceea ce e destul
// de idiot. (am putea returna k+1, adica lungimea)
int planificare(Drumetie a[], int m, int b[], int n) {
    sortare(a, m);
    b[0] = a[0].nr;
    // in loc sa memoram acest u, am putea memora
    // v[u].s+2
    int u = 0;
    int k = 0;
    for (int i = 1; i < m; i++) {
         if (a[i].s >= a[u].s+n) {
            k += 1;// sau 'k++', ori și mai bine '++k'
            b[k] = a[i].nr;
            u = i;
         }
    }
    return k;
}

int main() {
    Drumetie v[100];
```

```
    int m, n;
    int b[100];

    // ar fi indicat sa adaugati si ceva cum ar fi
    //'cout << "introduce-ti m:";....'
    cin >> m >> n;
    for (int i = 0; i < m; i++) {
        int s;
        cin >> s;// sau 'cin >> v[i].s;'
        v[i].s =s;
        v[i].nr = i+1;
    }
    int k = planificare(v, m, b, 2);

    // daca returnam k+1, mergem până la k, ca oamenii normali
    for (int i = 0; i < k+1; i++)
        cout << b[i] << " ";
    //int k = planificare();
    return 0;
}
```

9. interclasarea sirurilor (nu scrieti)

```
//nu scrieti
#include <iostream>
#include <vector>
#include <chrono>
#include <thread>

// ieseau liniile din pagina
using namespace std;
using namespace std::chrono;

// cel mai bun algoritm de sortare
// nu se poate mai bine de 0 comparatii
vector<unsigned> sleep_sort(const vector<vector<unsigned>>& x) {
    // sleep_sort nu merge pt numere negative
    // asa ca nu exista numere negative ;)
    vector<unsigned> res;
    std::vector<std::thread> threads;

    for (auto& vec : x) {
        for (auto& val : vec) {
             threads.emplace_back([val, &res]() {
                 std::this_thread::sleep_for(milliseconds(val));
                 res.push_back(val);
                 });
        }
    }
    for (auto& t : threads)
```

```
28          t.join();
29      return res;
30      }
31
32  using Clock = high_resolution_clock;
33  int main() {
34      std::vector<std::vector<unsigned>> x = {
35          { 1, 2, 3, 4, 6, 8, 10, 213, 100000 },
36          { 4, 7, 8, 12, 32 , 41, 57, },
37          { 1, 6, 7, 9, 14, 51},
38      };
39      auto t1 = Clock::now();
40      auto res = sleep_sort(x);
41      auto t2 = Clock::now();
42      // ne asteptam ca timpul de executie sa fie cea mai mare valoare
43      // din x adica doar 100000 ms (1 min 40 sec), in cazul nostru
44      // ceea ce e perfect rezonabil in vedere ca avem 0 comparatii
45      std::cout << "Timp de executie: "
46                << duration_cast<milliseconds>(t2 - t1).count()
47                << " ms\n";
48
49      for (auto& a : res)
50          std::cout << a << ", ";
51      std::cout << "\n";
52      std::cout << "nr de comparații = 0\n";
53
54      return 0;
55  }
```

O rezolvare mai serioasă ar fi concatenarea subșirurilor și sortarea cu radix sort (sau alt algoritm care nu compară elementele între ele). Asta face rezolvarea noastră mai bună decât rezolvarea vrută de autoarea problemei (metoda greedy), care trebuie sa compare cel puțin două elemente între ele.

Laborator 12

1. Fibonacci

```cpp
#include <iostream>
using namespace std;

int v[100];
//varianta descendenta
int F(int n) {
    if (n == 0 || n == 1) return n;
    if (v[n] == 0)
        v[n] = F(n-1)+F(n-2);
    return v[n];
}
//varianta ascendenta
int F(int n) {
    // am putea pastra, doar ultimele 2 elemente ale sirului,
    // nu tot sirul
    v[0] = 0;
    v[1] = 1;
    for (int i = 2; i <= n; i++) {
         v[i] = v[i-1]+v[i-2];
    }
    return v[n];
}

int main() {
    int n;
    cin >> n;
    cout << F(n) << endl;
    return 0;
}
```

2. Functia magica

```cpp
#include <iostream>
using namespace std;

double v[100];
// a) varianta descendenta
double P(int n, double x) {
    if (n == 0) return 1;
    if (n == 1) return x;
    if (v[n] == 0)
        v[n] = ((2.0*n-1)/n) * x * P(n-1, x) + ((n-1.0)/n) * P(n-2, x);
    return v[n];
}

// b) varianta iterativa
```

```cpp
double P(int n, double x) {
    if (n == 0) return 1;
    if (n == 1) return x;
    double p1 = 1, p2 = x;
    for (int i = 2; i <= n; i++) {
        double p = ((2.0*i-1) / i) * x * p2 + ((i-1.0) /i) * p1;
        p1 = p2;
        p2 = p;
    }
    return p2;
}

int main() {
    int n;
    double x;
    // presupunem ca functia nu ar returna niciodata 0
    // (desi ea poate) pentru ca simplifica putin lucrurile:
    // ar trebui sa initializam vectorul v cu NAN (not a number)
    // si sa verificam in P cu isnan in loc de 0

    cin >> n >> x;
    cout << P(n, x) << endl;
    return 0;
}
```