

# Portofoliu Algoritmi și complexitate - Probleme rezonabile

Anul I, semestrul 2

Numele tău aici

1. Dacă vreți probleme rezolvate mai grele intrați aici;
2. Dacă nu înțelegeți ceva întrebați-mă;
3. Vă invit să mai adăugați și problemele voastre (da, știu);
4. Dacă găsiți ceva greșit spuneți-mi;
5. Nu scrieți comentariile, sunt mai mult pentru amuzamentul meu.

## Laborator 1

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

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

```
1  #include <iostream>
2  #include <math.h>
3  using namespace std; // urgh
4  int main() {
5      double eps;
6
7      double x0;
8      double x1 = 0;
9      double delta = 10000;
10     int n = 1;
11     cin << eps;
12     while (delta > eps) {
13         x0 = x1;
14         x1 = pow(1 + 1.0 / n, n);
15         n++;
16         delta = x1 - x0;
17     }
18     cout << x1 << endl;
19
20     return 0;
21 }
```

## Laborator 2

### 2. Suma cifrelor - corectitudinea

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n, s = 0;
6      cin << n;
7      int i = 0;
8      while (n > 0) {
9          // se putea și cu s += n%10; și n /= 10;
10         s = s + n % 10;
11         n = n / 10;
12         i++;
13     }
14     cout << "suma = " << n << endl;
15     return 0;
16 }
```

#### 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  $I$  adevărată la începutul iterației și  $n \neq 0$ ; demonstrăm  $I$  adevărată 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

$\mathbf{n} = \mathbf{n} / 10;$

$$n = \left[ \left( \sum_{j=0}^{k-i} c_{i+j} 10^j \right) / 10 \right] = \left[ \sum_{j=0}^{k-i} c_{i+j} 10^{j-1} \right] = \left[ \sum_{j=1}^{k-i} c_{i+j} 10^{j-1} \right] + [c_i 10^{-1}]$$

$$\text{Cum } 0 \leq c_i \leq 9 \implies 0 \leq c_i 10^{-1} \leq 0.9 \implies [c_i 10^{-1}] = 0.$$

$$\text{Deci } n = \left[ \sum_{j=1}^{k-i} c_{i+j} 10^{j-1} \right] = \sum_{j=1}^{k-i} c_{i+j} 10^{j-1} = \sum_{j=0}^{k-i-1} c_{i+j+1} 10^j.$$

12

$\mathbf{i}++;$

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  $I$  adevărată și la sfârșitul iterației.

La ieșirea din buclă:

$$i = k + 1$$

$$n = \sum_{j=0}^{k-(k+1)} c_{k+1+j} 10^j = \sum_{j=0}^{-1} c_{k+1+j} 10^j = 0$$

$$s = \sum_{j=0}^{k+1-1} c_{k+1-1-j} = \sum_{j=0}^k c_{k-j}$$

$$\text{Deci } P_{out} = \left\{ s = \sum_{j=0}^k c_{k-j} \right\} \text{ adevărată.}$$

În concluzie algoritmului este parțial corect.

## II. Total corectitudinea

Considerăm funcția  $t : \mathbb{N} \rightarrow \mathbb{N}$ ,  $t(i) = k + 1 - i$ ;

$t(i + 1) - t(i) = k + 1 - (i + 1) - (k + 1 - i) = -1 < 0$ , deci  $t$  monoton strict descrescătoare.

$$t(i) = 0 \iff i = k + 1 \iff n = \sum_{j=0}^{-1} c_{k+1+j} 10^j = 0 \iff \text{condiția de ieșire din buclă.}$$

În concluzie algoritmului este total corect.

## Laborator 3

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

```
1  #include <iostream>
2  using namespace std;
3  // e important sa numerotați liniile aici
4  int main() {
5      int n, x[100], y[100];
6      cin >> n;
7      cout << "x = ";
8      for (int i = 0; i < n; ++i)
9          cin >> x[i];
10     cout << "y = ";
11     for (int i = 0; i < n; ++i)
12         cin >> y[i];
13
14     int ps = 0;
15     // sa spunem ca linia asta e z (15, sau cat e ea)
16     for (int i = 0; i < n; ++i) // z+1
17         ps = ps + x[i] * y[i]; // z+2
18
19     // si endl e oribil
20     cout << ps << endl;
21     return 0;
22 }
```

operația	cost	nr repetari	cost total
$z+2$	3	n	$3n$

$$T(n) = 3n$$

## Laborator 4

### 4. Cele mai mici 2 elemente dintr-o secvența

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n, x[100];
6      cin >> n;
7      for (int i = 0; i < n; ++i)
8          cin >> x[i];
9
10     // nu mai este 1989, putem declara variabile si la mijlocul functiei
11     int m1, m2;
12     // presupunem ca utilizatorul e rezonabil (nu e) si spunem ca n >= 2
13     m1 = x[0];
14     m2 = x[1];
15
16     // m1 < m2
17     if (m2 < m1) {
18         int t = m1;
19         m1 = m2;
20         m2 = t; //in loc de t putem folosi x[0]
21     }
22     for (int i = 2; i < n; i++) {
23         if (m1 >= x[i]) { // linia z
24             m2 = m1;
25             m1 = x[i];
26         }
27         else if (m2 > x[i]) { // linia z'
28             m2 = x[i];
29         }
30     }
31
32     cout << m1 << " " << m2 << endl;
33
34     return 0;
35 }
```

Cazul cel mai favorabil ( $m1 \geq x[i] \forall i \in \{2, \dots, n-1\}$ ).  $T(n) = n-2$  (se execută doar linia z).  
Cazul cel mai defavorabil ( $m1 < x[i] \forall i \in \{2, \dots, n-1\}$ ) :  $T(n) = 2n-4$  (se execută linia z și z').

Mereu  $T(n) \in \Theta(n)$ .

## Laborator 5

1. La o stație meteo ....

```
1  #include <iostream>
2  using namespace std;
3  struct Zi {
4      int temp;
5      int presiune;
6  };
7
8  int main() {
9      int n;
10     cin >> n;
11
12     Zi x[100];
13     for (int i = 0; i < n; ++i)
14         cin >> x[i].temp >> x[i].presiune;
15
16     // daca vreți alt algoritm de sortare înlocuiți aici:
17     for (int i = 0; i < n; ++i) {
18         for (int j = i+1; j < n; ++j) {
19             if (x[i].temp > x[j].temp || (x[i].temp == x[j].temp &&
20                 x[i].presiune < x[j].presiune)) {
21                 Zi t = x[i];
22                 x[i] = x[j];
23                 x[j] = t;
24             }
25         }
26     }
27     for (int i = 0; i < n; ++i)
28         cout << "t:" << x[i].temp << ", p: " << x[i].presiune << endl;
29
30     return 0;
31 }
```

## Laborator 6

### 5. Ackermann

```
1 int A(int m, int n) {
2     if (m == 0) {
3         return n + 1;
4     }
5     else {
6         if (n == 0) return A(m-1, 1);
7         else return A(m-1, A(m, n-1));
8     }
9 }
```

### 8. Baza 2

```
1 #include <iostream>
2 using namespace std;
3
4 // 47.145518,27.6036255 bdm tss
5 int baza2(int n, char s[], int i) {
6     if (n == 0) return i;
7     // se poate si mai scurt ( s[i] = '0' + n%2;)
8     if (n %2 == 0)
9         s[i] = '0';
10    else s[i] = '1';
11    return baza2(n/2, s, i+1);
12 }
13
14 int main() {
15     char s[100];
16     int n = 10;
17     //cin >> n;
18
19     int len = baza2(n, s, 0);
20     for (int i = len -1; i >= 0; i--)
21         cout << s[i];
22     cout << endl;
23     return 0;
24 }
```



## Laborator 7

### 2. Fibonacci

```
1  #include <iostream>
2  #include <math.h>
3  using namespace std;
4
5  double putere(double x, int n) {
6      if (n == 1) return x;
7      double r = putere(x, n / 2);
8      r = r * r;
9      if (n % 2 == 1) r = r * x;
10     return r;
11 }
12 int fib(int n) {
13     // daca vreti sa mearga si fib(0) (momentan da stack overflow),
14     // scrieti si linia asta:
15     //if (n==0) return 0;
16
17     // puteti defini o variabila 'double sqrt5 = sqrt(5);'
18     return round(1.0 / sqrt(5) *
19                 (putere((1+sqrt(5))/2, n) - putere(1-sqrt(5)/2, n)));
20     // varianta mai eficienta
21     // return round(1.0 / sqrt(5) * (putere((1+sqrt(5))/2, n)));
22 }
23 int main() {
24     int n;
25     cin >> n;
26     cout << fib(n) << endl;
27     return 0;
28 }
```

## Laborator 8

### 6. Problema aia lunga cu integrala definita

```
1 // puteti sa nu scrieti liniile astea 2 si functia main
2 #include <iostream>
3 using namespace std;
4
5 double f(double x) { // f(x) = x^2
6     // mai schimbati si voi putin expresia asta, cum ar fi
7     // return x; return 1; return sin(x); ....
8     return x * x;
9 }
10
11 // Se poate face si ceva de genu asta ca f sa fie parametru:
12 // template<typename F>
13 // double integrate(F f, double a, double b, double eps = 1e-5) {
14 // sau asa:
15 // double integrate(double (*f)(double), double a, ...
16
17 // daca vreti sa va simtiti mai romani puteti scrie "integreaza"
18 double integrate(double a, double b, double eps) {
19     double delta = b - a;
20     if (delta < eps) {
21         return delta * (f(a) + f(b)) / 2;
22     }
23     else {
24         double c = (a+b)/2;
25         return integrate(a, c, eps) + integrate(c, b, eps);
26     }
27 }
28 int main() {
29     double a, b, eps;
30     // cin >> a >> b >> eps;
31     // cu varianta mai faina merge si ceva de genu asta:
32     // cout << integrate(sin, a, b) << endl;
33     // doar ca n-ar trebui sa stim d-astea
34
35     cout << integrate(a, b, eps) << endl;
36
37     return 0;
38 }
```

## Laborator 9

### 5. numere de 4 cifre cu suma cifrelor 11

```
1  #include <iostream>
2  using namespace std;
3
4  // blasfemie, x ar trebui sa fie parametru pt functii,
5  // dar asa ne spune in curs...
6  int x[4], n=4;
7
8  // am putea declara si asa:
9  //constexpr int n = 4;
10 //int x[n];
11 // dar nici asta n-ar trebui sa stim, si tot blasfemie e
12
13 void afiseaza() {
14     for (int i = 0; i < n; i++)
15         cout << x[i];
16     cout << " ";
17 }
18 bool valid() {
19     int s = 0;
20     for (int i = 0; i < n; i++)
21         s += x[i];
22     return s == 11;
23 }
24
25 void btr(int k) {
26     //int i = 0;
27     //if (k == 0) i = 1;
28     //for (; i < 10; i++) {...
29
30     //varianta mai hardcore ar fi:
31     //for (int i = k == 0; i < 10; i++) {...
32
33     for (int i = 0; i < 10; i++) {
34         // numerele nu incep cu 0
35         // sunt metode mult mai eficiente pt verificarea asta
36         // (vezi inceputul functiei)
37         if (k == 0 && i == 0) continue;
38
39         // putem verifica pe parcurs daca suma e mai mare decat 11,
40         // si daca e, trecem la urmatorul
41         // "exercitiu cititorului"
42         x[k] = i;
43         if (k == n - 1) {
44             // valid() == true e redundant
45             // 'valid()' deja e bool si 'valid() == true' e tot bool
46             // si nu face absolut nimic in plus
```

```
47         // if (valid()) e mai normal
48         if (valid() == true)
49             afiseaza();
50     }
51     else {
52         btr(k+1);
53     }
54 }
55 }
56
57 int main() {
58     btr(0);
59     return 0;
60 }
```