Portofoliu Algoritmi și complexitate - Probleme rezonabile

Anul I, semestrul 2

Numele tău aici

- Dacă vreți probleme rezolvate mai grele intrati aici;
- Dacă nu înțelegeți ceva întrebați-mă;
- Vă invit sa mai scrieți și problemele voastre (da, știu);
- Dacă găsiți ceva greșit spuneți-mi;
- Nu scrieți comentariile, sunt mai mult pentru amuzamentul meu;
- Voi mai actualiza acest pdf cand mai primim fișe de laboratoare.

Deci intrați din nou aici după duminica viitoare (19 mai).

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>
#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;

int main() {
    double eps, x0 = 0, x1 = 0;
    double delta = 10000;
    cin << eps;

    int n = 1;
    while (delta > eps) {
        x0 = x1;
        x1 = pow(1 + 1.0 / n, n);
        n++;
        delta = x1 - x0;
    }
    cout << x1 << endl;

    return 0;
}
```

Laborator 2

2. Suma cifrelor - corectitudinea

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

int main() {
    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.

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

```cpp
    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 $0 \leq c_i \leq 9 \implies 0 \leq c_i 10^{-1} \leq 0.9 \implies \left[c_i 10^{-1}\right] = 0.$
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. Descrieți un algoritm pentru calculul produsului scalar a doi vectori din $\mathbb{R}^n$...

```cpp
#include <iostream>
using namespace std;
// e important sa numerotați liniile aici
int main() {
    int n, x[100], y[100];
    cin >> n;
    cout << "x = ";
    for (int i = 0; i < n; ++i)
        cin >> x[i];
    cout << "y = ";
    for (int i = 0; i < n; ++i)
        cin >> y[i];

    int ps = 0;
    // 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

    // și endl e oribil
    cout << ps << endl;
    return 0;
}
```

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

$T(n) = 3n$

Laborator 4

4. Cele mai mici 2 elemente dintr-o secvența

```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
    m1 = x[0];
    m2 = x[1];

    // m1 < m2
    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++) {
        if (m1 >= x[i]) { // linia ż
            m2 = m1;
            m1 = x[i];
        }
        else if (m2 > x[i]) { // linia ż'
              m2 = x[i];
        }
    }

    cout << m1 << " " << m2 << endl;

    return 0;
}
```

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 ż').
Mereu $T(n) \in \Theta(n)$.

Laborator 5

1. La o stație meteo ....

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

int main() {
    int n;
    cin >> n;

    Zi x[100];
    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) {
            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;
            }
        }
    }
    for (int i = 0; i < n; ++i)
       cout << "t:" << x[i].temp << ", p: "<< x[i].presiune << endl;

    return 0;
}
```

Laborator 6

5. Ackermann

```
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
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 = 10;
    //cin >> n;

    int len = baza2(n, s, 0);
    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;

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;
}
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;
}
```

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;

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"
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 cã 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], n=4;

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

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;
}

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;
}
```

Laborator 10-11

5. Gidul turistic și drumețiile sale

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

struct Drumetie {
    int nr, s;
};

// 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];
```

```
47
48      // ar fi indicat sa adaugati si ceva cum ar fi
49      //'cout << "introduce-ti m:";....'
50      cin >> m >> n;
51      for (int i = 0; i < m; i++) {
52          int s;
53          cin >> s;// sau 'cin >> v[i].s;'
54          v[i].s =s;
55          v[i].nr = i+1;
56      }
57      int k = planificare(v, m, b, 2);
58
59      // daca returnam k+1, mergem până la k, ca oamenii normali
60      for (int i = 0; i < k+1; i++)
61          cout << b[i] << " ";
62      //int k = planificare();
63      return 0;
64  }
```

9. interclasarea sirurilor (nu scrieti)

```
1   #include <iostream>
2   #include <vector>
3   #include <chrono>
4   #include <thread>
5
6   // ieseau liniile din pagina
7   using namespace std;
8   using namespace std::chrono;
9
10  // cel mai bun algoritm de sortare
11  // nu se poate mai bine de 0 comparatii
12  vector<unsigned> sleep_sort(const vector<vector<unsigned>>& x) {
13      // sleep_sort nu merge pt numere negative
14      // asa ca nu exista numere negative ;)
15      vector<unsigned> res;
16      std::vector<std::thread> threads;
17
18      for (auto& vec : x) {
19          for (auto& val : vec) {
20               threads.emplace_back([val, &res]() {
21                   std::this_thread::sleep_for(milliseconds(val));
22                   res.push_back(val);
23                   });
24          }
25      }
26      for (auto& t : threads)
27           t.join();
28      return res;
29  }
```

```cpp

using Clock = high_resolution_clock;
int main() {
    std::vector<std::vector<unsigned>> x = {
         { 1, 2, 3, 4, 6, 8, 10, 213, 100000 },
         { 4, 7, 8, 12, 32 , 41, 57, },
         { 1, 6, 7, 9, 14, 51},
    };
    auto t1 = Clock::now();
    auto res = sleep_sort(x);
    auto t2 = Clock::now();
    // ne asteptam ca timpul de executie sa fie cea mai mare valoare
    // din x adica doar 100000 ms (1 min 40 sec), in cazul nostru
    // ceea ce e perfect rezonabil in vedere ca avem 0 comparatii
    std::cout << "Timp de executie: "
              << duration_cast<milliseconds>(t2 - t1).count()
              << " ms\n";

    for (auto& a : res)
         std::cout << a << ", ";
    std::cout << "\n";
    std::cout << "nr de comparații = 0\n";

    return 0;
}
```

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.

Și pentru a răspunde la posibila întrebare a Ionelei: da, m-am distrat.