

Portofoliu Algoritmi și complexitate - Probleme rezonabile

Anul I, semestrul 2

Numele tău aici

Contribuitori:

- Sabina
- Carla

- Dacă vreți probleme rezolvate mai grele intrați aici;
 - Dacă nu înțelegeți ceva întrebați-mă;
 - Vă invit să 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;
 - Nu scrieți comentariile, sunt mai mult pentru amuzamentul meu;
 - Voi mai actualiza acest pdf când mai primim fișe de laboratoare.
- Deci intrați din nou aici după duminica viitoare (19 mai).

z sercem, Paбло.

Laborator 1

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

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

```
1 #include <iostream>
2 #include <cmath>
3
4 // 'using namespace std;' nu e cea mai bună practică,
5 // în special pentru proiecte mai mari.
6 // De exemplu, dacă definim o funcția asta:
7 //double riemann_zeta(double n) {
8 //    std::cout << "majie";
9 //    return 0;
10 //}
11 // și în main avem asta:
12 //cout << riemann_zeta(-1) << endl;
13 // Când rulăm programul, pe ecran se afișează "majie0", are sens nu?
14 // Asta se întâmplă dacă compilăm cu versiuni mai vechi de c++17
15 // (o dată cam la 3 ani mai apare o versiune de c++).
16 // Dar dacă compilăm cu c++17 (sau mai recent), pe ecran apare
17 // "-0.0833333". De ce?
18 // Păi din c++17 în "cmath" mai este definită și funcția "riemann_zeta",
19 // care sigur nu e definită ca funcția noastră (deși tot 'majie' e și
20 // acolo).
21 // Iar dacă nu știți asta și încercați ceva de genul ăsta, rămâneți
22 // fără păr în cap (sau fără tastatură) până aflați de ce nu va merge
23 // programul cum trebuie.
24 // Dar pentru ce facem noi la info, să zicem că e ok să scriem asta:
25 using namespace std;
26
27 int main() {
28     double eps, x0 = 0, x1 = 0;
29     double delta = 10000;
30     cin << eps;
31
32     int n = 1;
33     while (delta > eps) {
34         x0 = x1;
35         x1 = pow(1 + 1.0 / n, n);
36         n++;
37         delta = x1 - x0;
38     }
39     cout << x1 << endl;
40
41     return 0;
42 }
```

1. (Carla) BMI

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      unsigned kg, m;
6      float ma = 0;
7      cin >> kg >> m;
8      ma=kg/m^2;
9      if (ma<18.5)
10         cout << "risc minim pentru sanatate" << endl;
11     else if (ma>=18.5 && ma<25)
12         cout << "risc minim/scazut pentru sanatate" << endl;
13     else if (ma>=25 && ma<30)
14         cout << "risc scazut/moderat pentru sanatate" << endl;
15     else if (ma>=30 && ma<35)
16         cout << "risc moderat/ridicat pentru sanatate" << endl;
17     else cout << "risc ridicat pentru sanatate" << endl;
18     return 0;
19 }
```

2. (Carla) functia

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main() {
6      float x, f=0;
7      cin >> x;
8      if (x>=-5 && x<=5)
9          f=sqrt(abs(x))+6;
10     else f=5*x*x+7;
11     cout << f << endl;
12     return 0;
13 }
```

3. (Carla) alta functie

```
1  int functie(int x, int y, int a, int b) {
2      int c=0;
3      if (abs(x)<=8 && abs(y)<=8)
4          c=a*b*sqrt(x*x+y*y)*sin(x*x+y*y);
5      return c;
6  }
```

4. (Carla) cifra destinului (uau)

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int nastere(int zi, int luna, int an) {
6      int x = 0;
7      while (zi) {
8          x=x+zi%10;
9          zi=zi/10;
10     }
11     while (luna) {
12         x=x+luna%10;
13         luna=luna/10;
14     }
15     while (an) {
16         x=x+an%10;
17         an=an/10;
18     }
19     return x;
20 }
21
22 int destin(int x) {
23     unsigned s=0;
24     if (x<10) return x;
25     else while (x) {
26         s=s+x%10;
27         x=x/10;
28     }
29     return s;
30 }
31 int main() {
32     int a,b,c;
33     cin >> a >> b >> c;
34     int x=nastere(a,b,c);
35     cout << destin(x) << endl;
36     return 0;
37 }
```

5. (Carla) ăla cu multe subpuncte

a)

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main () {
6      float a, b, c, d, x1, x2;
```

```

7      cin >> a >> b >> c; // coeficientii
8      d=b*b-4*a*c; //delta
9      x1=(-b-sqrt(d))/2*a; //prima solutie
10     x2=(-b+sqrt(d))/2*a; // a doua solutie
11     cout << x1 << ' ' << x2 << endl;
12     return 0;
13 }

```

b)

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main() {
6      float x, n, max=-100000, min=100000;
7      cin >> n; //numarul de elemente
8      while (n) {
9          cin >> x; // element cu element
10         if (max<x)
11             max=x;
12         if (min>x)
13             min=x;
14         n--;
15     }
16     cout << max << ' ' << min << endl;
17     return 0;
18 }

```

k)

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      unsigned n,d;
6      cin>>n;
7      for(d=2;d<=n/2;d++)
8          if(n%d==0)
9              cout<<d<<" ";
10     cout << endl;
11     return 0;
12 }

```

6. (Carla) nr perfect

```

1  #include <iostream>
2
3  using namespace std;

```

```

4
5
6 // Pablo: poate fi bool, avem și d-ăștea in c++
7 int perfect(int n) {
8     int d;
9     int s = 1;
10    if (n<=2) return 0;
11    else {
12        for (d=2; d*d<=n; d++)
13            if (n%d==0) {
14                s=s+d;
15                if (n/d!=d)
16                    s=s+(n/d);
17            }
18        if (n==s)
19            return 1;
20    }
21    return 0;
22 }
23 int main() {
24     int n;
25     cin >> n;
26     if (perfect(n))
27         cout << n << " este perfect" << endl;
28     else cout << n << " nu este perfect" << endl;
29
30     return 0;
31 }

```

10. (Carla) Operatii cu multimi

Apartenenta

```

1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     int n, ok=0, c;
7     cin >> n;
8     int v[n+1];
9     for (int i=1; i<=n; i++)
10         cin >> v[i];
11     cin >> c;
12     for (int i=1; i<=n; i++)
13         if (v[i]==c)
14             ok=1;
15     if (ok==1)
16         cout << "apartine" << endl;
17     else cout << "nu apartine" << endl;

```

```
18
19     return 0;
20 }
```

Reuninunea

```
1  #include<iostream>
2  using namespace std;
3
4  int main() {
5      int a[20], b[20], c[50], i, j, m, n, k, gasit;
6      cin>>n;
7      for (i=0; i< n; i++) {
8          cin>>a[i];
9      }
10     cin>>m;
11
12     for(i=0; i<=m-1; i++) {
13         cin>>b[i];
14     }
15
16     k=0;
17     for(i=0; i<=n-1; i++) {
18         gasit=0;
19         for(j=0; j<=m-1 && !gasit; j++)
20             if (a[i]==b[j])
21                 gasit=1;
22         if(!gasit)
23             c[k++]=a[i];
24     }
25
26     for (i=0; i<=m-1; i++)
27         cout << b[i] << " ";
28     for (i=0; i<k; i++)
29         cout<<c[i]<<" ";
30
31     return 0;
32 }
```

Laborator 2

2. (Inspirat de Sabina) 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; n /= 10;'
10        s = s + n % 10;
11        n = n / 10;
12        i++;
13    }
14    cout << "suma = " << s << 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. (Inspirat de Sabina) 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     // nici 'endl' nu e cea mai grozava chestie:
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. (Inspirat de Sabina) 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 ž
24             m2 = m1;
25             m1 = x[i];
26         }
27         else if (m2 > x[i]) { // linia ž'
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 ž).

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

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 <cmath>
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)
14     // (momentan pt pow(x, 0), se blocheaza până dă stack overflow),
15     // scrieti si linia asta:
16     //if (n==0) return 0;
17
18     // puteti defini o variabila 'double sqrt5 = sqrt(5);'
19     return round(1.0 / sqrt(5) *
20                 (putere((1+sqrt(5))/2, n) - putere(1-sqrt(5)/2, n)));
21     // varianta mai eficienta
22     // return round(1.0 / sqrt(5) * (putere((1+sqrt(5))/2, n)));
23 }
24 int main() {
25     int n;
26     cin >> n;
27     cout << fib(n) << endl;
28     return 0;
29 }
```

Laborator 8

6. Problema aia lunga cu integrala definita

```
1 // puteți sa nu scrieți 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 așa:
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 faină merge și ceva de genu ăsta:
32     // cout << integrate(sin, a, b) << endl;
33     // doar că n-ar trebui să 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 de bun simț
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 }
```

Laborator 10-11

5. Gidul turistic și drumetețiile sale

```
1 #include <iostream>
2 using namespace std;
3
4 struct Drumetie {
5     int nr, s;
6 };
7
8 // am putea pune codul de aici in 'planificare', dar banuiesc ca
9 // asa ați face majoritatea. (ceea ce nu e așa de rău)
10 void sortare(Drumetie v[], int m) {
11     // aici putem sa le sortam după ziua de incepere, pt ca ziua
12     // de final e mereu 's+n'
13     for (int i = 0; i < m; i++) {
14         for (int j = i+1; j < m; j++) {
15             if (v[j].s < v[i].s) {
16                 Drumetie t = v[i];
17                 v[i] = v[j];
18                 v[j] = t;
19             }
20         }
21     }
22 }
23
24 // funcția returneaza indicele ultimului spectacol, ceea ce e destul
25 // de idiot. (am putea returna k+1, adica lungimea)
26 int planificare(Drumetie a[], int m, int b[], int n) {
27     sortare(a, m);
28     b[0] = a[0].nr;
29     // in loc sa memoram acest u, am putea memora
30     // v[u].s+2
31     int u = 0;
32     int k = 0;
33     for (int i = 1; i < m; i++) {
34         if (a[i].s >= a[u].s+n) {
35             k += 1; // sau 'k++', ori și mai bine '++k'
36             b[k] = a[i].nr;
37             u = i;
38         }
39     }
40     return k;
41 }
42
43 int main() {
44     Drumetie v[100];
45     int m, n;
46     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 }

```

```

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

```

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 să compare cel puțin două elemente între ele.

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