

CURSUL 5 - COMPLETARI

Curs Pregatire Admitere UBB 2022

CE CONTINUE ACEST CURS?

- In urma parcurgerii materiei, am reanalizat tot ce s-a predat si am concluzionat ca exista anumite aspecte care trebuie reluate sau explicate.
- In acest curs vom vedea mai multi astfel de algoritmi care consider ca sunt mult mai usor de inteles daca s-a fixat tot ce am discutat pana acum.
- Desigur, in cea de-a doua partea a cursului vom rezolva cateva grile special alese si in cadrul acestui curs vom discuta toata strategia si modalitatiile prin care va maximizati sansele la un rezultat foarte bun.

QUICKSORT

- Metoda de sortare QuickSort este o metoda foarte buna de sortare a sirurilor. Aceasta este de multe ori cea care sta in spatele tuturor functiilor de sortare implementate in limbajele de programare (<algorithm> - sort())
- Aceasta metoda utilizeaza procedeul divide et impera pentru a sorta sirurile, dar cel mai important este ca la o prima analiza, complexitatea ei poate parera foarte mare, dar de fapt, in realitate, pe cazurile generale de sortari, este cea mai buna complexitate posibila.

QUICKSORT

- Cum functioneaza?
 - Fiind vorba de o metoda de sortare de tipul Divide Et Impera, este evident faptul ca vom sorta intervale din sirul nostru. Dar cum alegem intervalele? Ei bine, daca la Merge Sort sortam mereu cele 2 jumatati de interval, aici vom proceda usor diferit.
 - Pentru a sorta sirul, selectam un element total arbitrar ales pe care il vom numi PIVOT!!
 - Dupa ce am luat pivotal, vom incerca sa il ducem la pozitia lui finala. Care e pozitia lui finala? Ei bine, un PIVOT este la pozitia lui corecta daca TOATE elementele situate la stanga lui sunt mai mici decat el si toate cele situate la dreapta sunt mai mari decat el.
 - Dupa ce am determinat pivotul, vom apela functia de Quicksort pentru cele 2 intervale din stanga si, respectiv dreapta pivotului.

QUICKSORT

- Variabila ‘pindex’ va memora pozitia unde este situat pivotul intervalului [st, dr]

```
19 void QuickSort(int st, int dr){  
20     if(st <= dr){  
21         int pindex = partitie(st, dr);  
22         QuickSort(st, pindex - 1);  
23         QuickSort(pindex + 1, dr);  
24     }  
25 }
```

QUICKSORT

- Pentru a lamuri intregul algoritm, ce trebuie sa facem este sa analizam functia ‘paritie’.
- Avand in vedere ca pivotal poate fi selectat intr-un mod total arbitrar, functia ‘partitie’ va selecta pivotul ca fiind elementul $a[dr]$. Va aranja elementele intervalului in asa fel incat toate elementele din stanga pozitiei finale a valorii $a[dr]$ sa fie mai mici ca el si toate cele din dreapta mai mari ca el.
- De asemenea, va returna indicele unde $a[dr]$ va fi pozitionat in noul sir paritionat dupa regula mai sus explicata

QUICKSORT

```
6  int partitie(int st, int dr){  
7      int pivot = a[dr];  
8      int index_curent = st;  
9      for(int i = st; i < dr; ++i){  
10          if(a[i] <= pivot){  
11              swap(a[i], a[index_curent]);  
12              index_curent++;  
13          }  
14      }  
15      swap(a[dr], a[index_curent]);  
16      return index_curent;  
17 }
```

QUICKSORT

- Cum functioneaza aceasta functie? Are o pozitie care indica la inceputul vectorului, si parcurgand tot vectorul, toate elementele mai mici decat pivotal sunt interschimbate cu primele k pozitii, unde k este numarul de numere mai mici decat pivotul.
- La final unde a ramas indicele, este dusa valoarea lui a[dr] (pivotul).
- Se returneaza valoarea si am rezolvat problema in sine.
- IATA CUM ARATA PROGRAMUL COMPLET:

QUICKSORT

```
1 #include <iostream>
2 using namespace std;
3
4 int a[100001], n;
5
6 int partitie(int st, int dr){
7     int pivot = a[dr];
8     int index_curent = st;
9     for(int i = st; i < dr; ++i){
10         if(a[i] <= pivot){
11             swap(a[i], a[index_curent]);
12             index_curent++;
13         }
14     }
15     swap(a[dr], a[index_curent]);
16     return index_curent;
17 }
18 }
```

```
19 void QuickSort(int st, int dr){
20     if(st <= dr){
21         int pindex = partitie(st, dr);
22         QuickSort(st, pindex - 1);
23         QuickSort(pindex + 1, dr);
24     }
25 }
26
27 int main(){
28     cin >> n;
29     for(int i = 1; i <= n; ++i)
30         cin >> a[i];
31     QuickSort(1, n);
32     for(int i = 1; i <= n; ++i)
33         cout << a[i] << ' ';
34     return 0;
35 }
```

QUICKSORT

- Complexitate SPATIU?
 - Evident: $O(n)$ – lungimea vectorului
- Complexitatea TIMP?
 - $O(n * \log_2(n))$ – cazul general
 - $O(n^2)$ – in cel mai rau caz!!
- Comparatie intre MergeSort si QuickSort: MergeSortul in general este mai rapid, dar QuickSort reuseste sa-l intreaca din multe alte aspecte (Memorie utilizata, Timp pe cazurile medii, etc)

GRILE

1. Se consideră expresia următoare, în care a este un număr natural.

$$((a < 4) \text{ SAU } (a < 5)) \text{ SI } (a > 2)$$

Pentru ce valori ale lui a va avea expresia valoarea **ADEVĂRAT**?

- A. $a = 3$
- B. $a = 4$
- C. $a = 2$
- D. Expresia nu va avea niciodată valoarea **ADEVĂRAT**

Raspuns: A, B

2. Subalgoritmul de mai jos are ca parametri de intrare un sir v cu n numere naturale nenule ($v[1], v[2], \dots, v[n]$) și numărul întreg n ($1 \leq n \leq 10000$).

```
Subalgoritm f(v, n):
    x ← 0
    Pentru i ← 1, n execută
        c ← v[i]
        Cătimp c MOD 3 = 0 execută
            x ← x + 1
            c ← c DIV 3
        SfCătimp
    SfPentru
    returnează x
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

- A. Subalgoritmul returnează numărul numerelor divizibile cu 3 din sirul v
- B. Subalgoritmul returnează cel mai mare număr k astfel încât $v[1] * v[2] * \dots * v[n]$ este divizibil cu 3^k
- C. Subalgoritmul returnează cel mai mare număr k astfel încât $v[1] + v[2] + \dots + v[n]$ este divizibil cu 3^k
- D. Subalgoritmul returnează suma numerelor divizibile cu 3 din sirul v

Raspuns: B

3. Se consideră expresia următoare, în care x este un număr natural pozitiv.

$$(x \text{ MOD } 2) + ((x + 1) \text{ MOD } 2)$$

Care din afirmațiile de mai jos sunt adevărate?

- A. Expresia are valoarea 1 pentru orice număr natural pozitiv x .
- B. Expresia are valoarea 1 dacă și numai dacă x este un număr par.
- C. Expresia are valoarea 1 dacă și numai dacă x este un număr impar.
- D. Există număr natural x pentru care expresia are o valoare strict mai mare decât 1.

Raspuns: A

4. Fie subalgoritmul **prelucrare(x, n)** definit mai jos, care primește ca și parametru un sir **x** cu **n** numere reale nenule ($x[1], x[2], \dots, x[n]$) și numărul întreg **n** ($1 \leq n \leq 10000$). Operatorul / reprezintă împărțirea reală (ex. $3/2=1,5$).

```
Subalgoritm prelucrare(x, n):
    p ← 1
    Pentru k ← 1, n - 1 execută
        p ← p + 1
        Pentru i ← 1, n - 1 execută
            Dacă x[i] > x[i + 1] atunci
                x[i] ← x[i] * x[i + 1]
                x[i + 1] ← x[i] / x[i + 1]
                x[i] ← x[i] / x[i + 1]
            SfDacă
        SfPentru
    SfPentru
    n ← p
SfSubalgoritm
```

Care dintre următoarele afirmații descriu modificarea aplicată sirului **x** în urma apelului subalgoritmului **prelucrare(x, n)**?

- A. Elementele sirului **x** vor rămâne nemodificate
- B. Elementele sirului **x** vor fi în ordine descrescătoare
- C. Elementele sirului **x** vor fi în ordine crescătoare
- D. Numărul **n** este decrementat cu o unitate

Raspuns: C

5. Se consideră subalgoritmul `calcul(a, n)`, care primește ca parametru un sir **a** cu **n** numere naturale ($a[1], a[2], \dots, a[n]$) și numărul întreg **n** ($1 \leq n \leq 10000$).

```
Subalgoritm calcul(a, n):
    Dacă n = 0 atunci
        returnează 0
    altfel
        returnează a[n] * (a[n] MOD 2) + calcul(a, n - 1)
    SfDacă
SfSubalgoritm
```

Pentru ce valori a numărului **n** și a sirului **a** funcția `calcul(a,n)` va returna valoarea 10?

- A. $n = 4, a = (2, 4, 7, 5)$
- B. $n = 6, a = (3, 1, 2, 5, 8, 1)$
- C. $n = 6, a = (2, 4, 5, 3, 8, 5)$
- D. $n = 7, a = (1, 1, 2, 1, 1, 1, 3)$

Raspuns: B

6. Se consideră subalgoritmul `calcul(v, n)`, care primește ca parametru un sir **v** cu **n** numere naturale ($v[1], v[2], \dots, v[n]$) și numărul întreg n ($1 \leq n \leq 10000$).

```
Subalgoritm calcul(v, n):
    m ← 0
    x ← 0
    s ← 0
    Pentru i ← 1, n execută
        s ← s + v[i]
        m ← m + (s MOD 2 + x) MOD 2
        x ← s MOD 2
    SfPentru
    returnează m
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

- A. Subalgoritmul calculează și returnează suma numerelor impare din sirul **v**
- B. Subalgoritmul calculează și returnează suma numerelor pare din sirul **v**
- C. Subalgoritmul calculează și returnează numărul de numere impare din sirul **v**
- D. Subalgoritmul calculează și returnează numărul de numere pare din sirul **v**

Raspuns: C

7. Se consideră subalgoritmul $\text{magic}(x)$, unde x este un număr natural ($1 \leq x \leq 32000$).

```
Subalgoritm magic(x):
    st ← 1
    dr ← x
    Câtimp st ≤ dr execută
        mj ← (st + dr) DIV 2
        Dacă mj * mj = x atunci
            returnează adevărat
        SfDacă
        Dacă mj * mj < x atunci
            st ← mj + 1
        altfel
            dr ← mj - 1
        SfDacă
        SfCâtimp
        returnează fals
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

- A. Subalgoritmul verifică dacă există un pătrat perfect mai mic decât x .
- B. Subalgoritmul numără divizorii primi ai numărului x .
- C. Subalgoritmul verifică dacă numărul x este prim.
- D. Subalgoritmul verifică dacă numărul x este pătrat perfect.

Raspuns: D

8. Se consideră subalgoritmul ceFace(n), unde n este un număr natural ($1 \leq n \leq 10000$).

```
Subalgoritm ceFace(n):
    a ← n
    b ← 0
    Cât timp a ≠ 0 execută
        b ← b * 10 + a MOD 10
        a ← a DIV 10
    SfCâtimp
    Dacă n = b atunci
        returnează adevărat
    altfel
        returnează fals
    SfDacă
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

- A. Subalgoritmul verifică dacă numărul n este prim.
- B. Subalgoritmul verifică dacă numărul n este palindrom.
- C. Subalgoritmul returnează întotdeauna adevărat.
- D. Subalgoritmul verifică dacă numărul n este divizibil cu 10.

Raspuns: B

9. Se consideră subalgoritmul calculeaza(**a**,**b**), unde **a** și **b** sunt numere naturale ($1 \leq a, b \leq 10000$).

```
Subalgoritm calculeaza(a, b):
    x ← 1
    Pentru i ← 1, b execută
        x ← (x MOD 10) * a
    SfPentru
    returnează x
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

- A. Dacă **a** = 2021 și **b** = 2021, valoarea returnată de subalgoritm este 2021.
- B. Pentru toate apelurile subalgoritmului cu **a** = 2021 și $1 \leq b \leq 10000$, valoarea returnată este 2021.
- C. Dacă **a** = 7777 și **b** = 2021, valoarea returnată este 7777.
- D. Pentru toate apelurile subalgoritmului cu $1 \leq a \leq 10000$ și **b** = 2021, valoarea returnată este valoarea lui **a**.

Raspuns: A, B, C

10. Câte elemente se găsesc pe cele două diagonale ale unei matrice pătratice cu n linii și n coloane ($10 \leq n \leq 1000$)? Se numără elementele de pe poziții distințe.

- A. $2 * n$
- B. $n * n$
- C. $2 * n - 1$
- D. $2 * n - (n \text{ MOD } 2)$

Raspuns: D

11. Care dintre expresiile logice următoare au valoarea ADEVĂRAT pentru $a = 1$ și $b = 0$?

- A. NU ((($a > 0$) **ŞI** ($b < 1$)) **SAU** ($a > 1$))
- B. (($b > 0$) **ŞI** ($b < 1$)) **SAU** (($a > 0$) **ŞI** ($a < 2$))
- C. (NU ($a > b$)) **SAU** (NU ($b > 0$))
- D. ($a > 0$) **SAU** (($b > 0$) **ŞI** ($b < 0$)) **SAU** ($a < 1$)

Raspuns: B, C, D

13. Se consideră subalgoritmul `ceFace(a,b)`, unde **a** și **b** sunt numere naturale ($1 \leq a < b \leq 10000$).

Subalgoritm ceFace(a, b):

```
m ← a
Cât timp b MOD m > 0 execută
    m ← m + 1
SfCâtimp
    returnează m
SfSubalgoritm
```

Ce va returna apelul `ceFace(47, 100)?`

- A. 48
- B. 50
- C. 3
- D. 100

Raspuns: B

14. Se consideră subalgoritmul **afis(n)**, unde **n** este un număr natural ($0 \leq n \leq 10000$).

Subalgoritm afis(n):

```
    Scrie n
    Dacă n > 0 atunci
        afis (n - 1)
        Scrie n
    SfDacă
    SfSubalgoritm
```

Ce se va afișa la apelul **afis(4)**?

- A. 432100123
- B. 123401234
- C. 1234004321
- D. 432101234

Raspuns: D

15. Care dintre următoarele baze de numerație x satisfac condiția $232_{(x)} \leq 67_{(10)}$?

- A. $x = 5$
- B. $x = 3$
- C. $x = 4$
- D. $x = 6$

Raspuns: A, C

18. Se consideră subalgoritmul $f(a, b)$, unde a și b sunt numere întregi ($-10000 \leq a, b \leq 10000$).

Subalgoritm $f(a, b)$:

Scrie "FMI"

Dacă $(a = 0)$ SAU $(b = 0)$ atunci
 returnează 1

SfDacă

Dacă $a > b$ atunci
 returnează $f(a - b * b, a * (a - b) - b * (a - b))$

SfDacă

Dacă $a \leq b$ atunci
 returnează $f(b - a * a, a * (a - b) - b * (a - b))$

SfDacă

SfSubalgoritm

Precizați de câte ori se scrie textul *FMI* la executarea secvenței de cod:

$f(f(3, 2), f(2, 3))$

- A. De 8 ori
- B. De 6 ori
- C. De 3 ori
- D. De o infinitate de ori

Raspuns: A

VA MULTUMESC!

