Algoritmi care lucrează cu tablouri unidimensionale
26.11.2022

1). Fie șirul
X=(1,1,2,2,2,2,3,3,3,3,3,3,4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,6,6,7, …).
a) Considerând că primul element din șir este pe poziția 1, determinați pozițiile pe care va apărea o valoare ***n*** dată (1<=n<=1000) .
b) Pentru o pozitie ***m*** dată, scrieti un subalgoritm care determina câte numere prime distincte apar în șirul X până la poziția ***m*** (inclusiv).
a)

Nr. nat.	1	2	3	4		n-1	n
Nr. aparitii	2=2*1	4=2*2	6=2*3	8=2*4		2*(n-1)	2*n

Din modul de generare a elementelor sirului, se observă că fiecare valoare naturala **n** (n>=1) apare de 2*n în șir, iar inainte de apariția valorii **n** vor fi 2*1+2*2+...+2*(n-1) elemente.
=> 2*1+2*2+...+2*(n-1)=2*(1+2+...+(n-1))=2*((n-1)(1+n-1))/2=(n-1)*n
Prima poziție pe care va apărea valoarea **n** va fi : n*(n-1)+1
Ultima poziție pe care va apărea valoarea **n** va fi n*(n-1)+2*n=n*(n-1+2)=n*(n+1)
b)
Pentru a putea determina numărul de numere prime distincte care apar în șir până la poziția **m,** întâi determinăm valoarea **val** (număr natural), care apare pe poziția **m:**

Pozitia (**m**)	1	5	10	15	20	21	30	31
Valoarea (**val**)	1	2	3	4	4	5	5	6

Determinarea valorii **val** care apare pe poziția **m** se poate face în cel puțin două moduri:
1. Incepem să generăm elementele șirului și să numărăm pozițiile la care am ajuns. Când ultima poziție obținută va fi mai mare decât **m,** am determinat valoarea val. Pentru această soluție, ne folosim de observația că pentru un număr natural ***s,*** în șir vor exista ***2*s*** elemente care vor avea această valoare. Subalgoritmul corespunzator este:

```
Functia getVal(m) este:
val=1; //păstrează valoarea la care s-a ajuns, prima valoarea va fi 1
contor=0; //numără câte elemente din șir am "generat" până atunci, la început va fi 0
Câttimp ((contor<m) AND (contor+2*val)<m) execută:
        contor=contor+2*val;
        val=val+1;
sfCâttimp
returnează val;
sfFunctie
```

2. Ne folosim de formula obținută la punctul anterior, care spune că un numar natural **n** dat, va apărea în șirul X pe pozițiile n*(n-1)+1, … ,n*(n+1). Incercăm să determinăm *cel mai mic număr natural* ***i***, cu proprietatea că i*(i-1) >=m. După determinarea acestui numar, **val** va fi egal cu i-1.
Exemple:

m	2	5	10	15	20	21	30	31
i	2	3	4	5	5	6	6	7
i*(i-1)	2*1=2>=2	3*2=6>=5	4*3=12>=10	5*4=20>=15	5*4=20>=20	6*5=30>=21	6*5=30>=30	7*6=42>=31
val	1	2	3	4	4	5	5	6

Subalgoritmul corespunzător este:

```
Functia getVal(m) este:
        i=1;
        contor=0;
        Câttimp (i*(i-1)<m) executa
                i=i+1
        sfCâttimp
        returnează i-1;
sfFunctie
```

Pentru a determina numărul de numere prime distincte din șirul X până la poziția **m,** parcurgem numerele naturale de la 2 până la valoarea **val** care apare pe pozitia **m,** verificăm care sunt numere prime și le contorizam.
In functia de mai jos, **getVal(m)** determină valoarea care apare pe pozitia **m (**folosind oricare dintre soluțiile descrise anterior)

```
Functie prime_distincte(m) este:
        prime=0;
        Pentru i=2, getVal(m), 1 este:
                Daca prim(i) atunci prime=prime+1;
                sfDaca
        SfPentru
        returnează prime;
SfFunctie;
```

Observație! Dacă se dorește determinarea numărului de numere prime care apar în șir până la poziția **m,** soluția este asemănătoarea cu cea descrisă mai sus, dar trebuie să luam în considerare cazul în care valoarea **val** care apare pe pozitia **m** este număr prim și să o adăugam la contor doar de m-val*(val-1) ori.
De exemplu, pentru **m**=25, valoarea care apare pe aceasta pozitie este **val=5.** Numerele prime până la 5 sunt 2 și 3 care apar de 2*2+2*3=10 ori. Valoarea 5, apare doar de 5 ori până la poziția

25 și nu mai putem folosi formula 2*5 pentru a calcula numărul de apariții a valorii 5. Pentru acest caz, folosim formula 25-5*(4-1)=25-20=5 pentru calcularea numărului de apariții până la poziția **m**.

Un subalgoritm care rezolvă această cerință este:

```
Functie prime_toate(m) este:
        prime=0; //păstrează numărul de numere prime din șir până la poziția curentă

        n=getVal(m); //obtinem valoarea n care apare pe pozitia m

        //determinăm câte numere prime strict mai mici  decât n apar în șir
        Pentru i=2, n-1, 1 este:
                Dacă prim(i) atunci prime=prime+2*i;
                sfDacă
        SfPentru

        //verificăm dacă n este număr prim.
        //Dacă este prim adăugăm numărul de apariții până la pozitia m
        Dacă (prim(n)) atunci
                        prime=prime+m-n*(n-1)
        sfDacă
        returnează prime;
SfFunctie;
```

17. Fie şirul $x = (1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, ...)$, care se continuă conform regulii care se poate observa din elementele enumerate.

Considerând că primul element din şir este pe poziția 1, în care dintre următoarele subsecvențe va apărea doar valoarea **11**?

A. $x[100], ..., x[109]$ B. $x[113], ..., x[120]$ C. $x[140], ..., x[152]$ D. $x[123], ..., x[132]$

(Admitere licenta, sesiunea septembrie 2022)

Folosind formulele obținute la problema anterioara, subpunctul a), obținem că valoarea n=11 va apărea în șir între pozițiile: n*(n-1)+1=11*10+1=**111** și n*(n+1)=11*12=**132**

Răspuns corect: B si D

18. Câte din primele 100 de elemente ale şirului x descris în subiectul **17** sunt numere prime?

A. 4 B. 34 C. 36 D. 30

(Admitere licenta, sesiunea septembrie 2022)

Folosind rezultatul obținut la subiectul 17, și anume că prima poziție pe care apare valoarea 11 este 111, și că o valoare **n** apare în sirul X de **2*n** ori, pentru n=10 , obținem că pe poziția 100 va apărea valoarea 10.

Numarele prime până la 10 sunt 2,3,5 și 7 care apar de 2*2+2*3+2*5+2*7=2*(2+3+5+7)=2*17=34 ori. Numarul 10 nu este prim, rezultă că până la poziția 100 sunt 34 de numere prime.
Răspuns corect B

2) Se consideră expresia $E(x)=a_0+a_1*x^1+a_2*x^2+...+a_n*x^n$, unde a_0, a_1, a_2, …, a_n și x sunt numere reale nenule date.
a) Scrieți un algoritm care calculează valoarea expresiei E(x).
b) Care este numarul minim de inmultiri necesare pentru a calcula expresia E(x)?

a) Pentru a calcula valoarea expresiei E(x) unde șirul a, numărul n și x sunt considerate date de intrare, există mai multe soluții.
1. Parcurgem șirul **a** de la poziția 1 până la n, calculăm valoarea $\mathbf{x}^i$ și adăugăm termenul corespunzător unui rezultat temporar, conform subalgoritmului ***CalculVal1***

Functia **CalculVal1**(a,n,x) este:
```
    Val=a[0];
    p=1;
    Pentru i=1,n, 1 este:
        p=p*x;
        Val=Val+a[i]*p;
    sfPentru
    returnează Val;
sfFunctie
```

Observăm că pentru această soluție sunt necesare **2*n** înmulțiri, deoarece pentru fiecare valoarea ***i***, (1<=***i***<=n), efectuăm 2 înmulțiri: una pentru a calcula valoarea lui $\mathbf{p=x^i}$ si una pentru a calcula valoarea $\mathbf{a_i*p}$.

2. Putem reduce numărul de înmulțiri necesare pentru a calcula expresia, dacă nu efectuăm nicio înmulțire pentru a calcula valoarea x^1.
Atentie! Aceasta soluție este corectă doar dacă n>=1.
Functia **CalculVal2**(a,n,x) este:
```
    Val=a[0]+a[1]*x;
    p=x;
    Pentru i=2,n, 1 este:
        p=p*x;
        Val=Val+a[i]*p;
    sfPentru
    returnează Val;
sfFunctie
```

Pentru această soluție, numărul de înmulțiri necesare va fi **2*(n-1)+1.**

3. Expresia $E(x)=a^nx^n+a_{n-1}*x^{n-1}+ \ldots+a_1*x+a_0$ poate fi rescrisă astfel
$E(x)=(...(((a_n*x+a_{n-1})*x+a_{n-2})*x+a_{n-3})...+a_1)*x+a_0$. Cele două expresii sunt echivalente și nu mai este necesar să calculăm pentru fiecare ***i*** (1<=***i***<=n) valoarea lui x^i. Subalgoritmul pentru aceasta solutie este:

Functia **CalculVal3**(a,n,x) este:
```
    Val=a[n];
    Pentru i=n-1,0, -1 execute
        Val=Val*x+a[i]
    sfpentru
    returnează Val
sfFunctia
```

Se observă că pentru această soluție, numărul de înmulțiri necesare este **n**.

b) Considerând cele 3 soluții prezentate anterior, numărul minim de înmulțiri necesare pentru a calcula expresia **$E(x)=a^nx^n+a_{n-1}*x^{n-1}+ \ldots+a_1*x+a_0$** va fi **n.**

24. Se considera expresia: $E(x) = a_0 + a_1*x + a_2*x^2 + a_3*x^3 + a_4*x^4$, unde a_0, a_1, a_2, a_3, a_4 și x sunt numere reale nenule. Numărul minim de înmulțiri necesare pentru a calcula valoarea expresiei $E(x)$ este:

A. 4 B. 5 C. 7 D. 3

(Concursul Mate-Info UBB, martie 2022)

Pentru expresia $E(x)=a_0+a_1*x^1+a_2*x^2+a_3*x^3+a_4*x^4$, **n** va fi egal cu 4. Conform problemei prezentate anterior, numărul minim de înmulțiri necesare va fi n=4.
Răspuns A

3) Fie ***a*** un șir de n numere naturale $(a_1,a_2, \ldots, a_n)$ ordonat strict crescător și un număr S. Scrieți un subalgoritm care determină numărul perechilor (a_i, a_j) care îndeplinesc condițiile $S=a_i+a_j$ și i<j.

Exemplu:
Pentru a=(2,3,4,5,8,10) și S=12 => există 2 perechi (2,10) și (4,8).

O soluție posibilă pentru această problemă este parcurgerea sirului **a** folosind două instrucțiuni repetitive (ex. pentru) și verificarea dacă elementele la care s-a ajuns au suma egala cu **S**, conform subalgoritmului următor.
Functia nr_perechi(a,n,S) este:
```
    contor=0
    Pentru i=1, n-1, 1 executa
        Pentru j=i+1, n, 1 executa:
            Daca (a[i]+a[j]=S) atunci
                contor=contor+1;
            sfDaca
        SfPentru
    SfPentru
```

```
        returneaza contor;
sfFunctie
```

Această soluție are complexitatea $O(n^2)$ și nu ține cont de condiția din enunț care precizează că șirul este ordonat strict crescător.

O altă soluție posibilă, care ține cont de condiția că șirul ***a*** este ordonat crescător și încearcă să obțină numărul perechilor care satisfac condiția, este să parcurgem șirul o singură dată, dar să folosim două contoare: un contor **i** parcurge șirul de la pozitia 1 spre n, iar un contor **j** de la poziția n spre 1. Pentru fiecare pereche (i,j), unde i<j, astfel obținută, verificăm condiția ca $a_i+a_j=S$. Dacă perechea îndeplinește condiția, incrementăm contorul ***i*** și decrementăm contorul ***j*** pentru a căuta o altă pereche care îndeplinește condițiile.
Dacă $a_i+a_j>S$ înseamnă că trebuie să scădem valoarea sumei a_i+a_j și încercăm cu elementul aflat pe poziția j-1, iar dacă $a_i+a_j<S$ înseamnă că trebuie să mărim valoarea sumei a_i+a_j și încercăm cu elementul aflat pe pozitia i+1. Incercăm cu aceste valori pentru că în enunțul problemei s-a precizat că șirul este ordonat strict crescator și atunci $a_i<a_{i+1}$, respectiv $a_{j-1}<a_j \Rightarrow a_i+a_j<a_{i+1}+a_j$ și $a_i+a_{j-1}<a_i+a_j$.

Subalgoritmul pentru această soluție este:

```
Functia nr_perechi(a,n,S) este:
        contor=0
        i=1;
        j=n;
        Cattimp (i<=j) executa:
                Daca (a[i]+a[j]==S) atunci
                                contor=contor+1;
                                i=i+1;
                                j=j-1;
                altfel Daca (a[i]+a[j]<S) atunci i=i+1
                                altfel j=j-1;
                        sfdaca
                sfdaca
        sfcattimp
        returneaza contor;
sfFunctie
```

Intrebari grila

24. Se consideră algoritmul det(a, n, m), unde ***a*** este un șir de ***n*** numere naturale (***a***[1], ***a***[2], ..., ***a***[***n***] dacă ***n*** ≥ 1) sau șir vid dacă ***n*** = 0. ***n*** și ***m*** sunt numere naturale ($0 \le n \le 100$, $0 \le m \le 10^6$).

```
1.  Algorithm det(a, n, m):
2.      For i ← 1, n – 1 execute
3.          For j ← i + 1, n execute
4.              If a[i] > a[j] then
5.                  tmp ← a[i]
6.                  a[i] ← a[j]
7.                  a[j] ← tmp
8.              EndIf
9.          EndFor
10.     EndFor
11.     i ← 1
12.     j ← n
13.     b ← False
14.     While i < j execute
15.         If a[i] + a[j] = m then
16.             b ← True
17.         EndIf
18.         If a[i] + a[j] < m then
19.             i ← i + 1
20.         else
21.             j ← j - 1
22.         EndIf
23.     EndWhile
24.     return b
25. EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate:
A. Algoritmul returnează *True* dacă în șirul **a** există o pereche de numere care au suma egala cu **m.**
B. Algoritmul returnează întotdeauna *False.*
C. Algoritmul returnează *False* dacă **n**=0.
D. În liniile 2, …, 10 algoritmul sortează crescător șirul **a**.

(Admitere licenta, sesiunea iulie 2022)

Răspuns corect: A, C,D

Se consideră algoritmul *ceFace(**T**,**n**,**e**)* care primește ca și parametru un șir **T** cu **n** numere naturale ordonate crescator (**T**[1], **T**[2], …, **T**[n]) și numerele naturale **n** și **e** (1<=**n**,**e**, <=10000).

```
Algorithm ceFace(T, n, e):
            If e MOD 2 = 0 then
                a ← 1
                b ← n
                While a ≤ b execute
                    m ← (a + b) DIV 2
                    If e < T[m] then
                        b ← m - 1
                    else
                        If e > T[m] then
                            a ← m + 1
                        else
                            return m
                        EndIf
                    EndIf
                EndWhile
                return 0
            else
                c ← 1
                g ← 0
                While (c ≤ n) AND (g = 0) execute
                    If e = T[c] then
                        g = c
                    EndIf
                    c ← c + 1
                EndWhile
                return g
            EndIf
        EndAlgorithm
```

Precizati care dintre următoarele afirmații sunt adevărate:
A. Algoritmul returnează 0 dacă numărul **e** nu se află în șirul **T**.
B. Dacă numărul **e** este impar și se află în șirul **T**, algoritmul returnează poziția din sirul **T** pe care se află **e** folosind algoritmul de căutare binară.
C. Dacă numărul **e** este impar și se află în șirul **T**, algoritmul returnează poziția din șirul T pe care se află **e** folosind algoritmul de căutare secventiala.
D. Algoritmul returnează poziția din șirul **T** pe care se află numărul **e**.

(Concursul Mate-Info UBB, martie 2022)

Răspuns corect: A, C

Observatie! Răspunsul D nu este considerat corect, deoarece enunțul variantei D nu precizează ce returnează algoritmul, dacă numărul **e** nu se află în șirul T.

Algoritmul *magic (**s**,**n**)* are ca parametri de intrare un șir **s** cu **n** caractere **(s**[1], **s**[2], …, s[n]) și numărul întreg **n** (1<=**n**<=10000)

```
Algorithm magic(s, n):
    i ← n
    While 1 ≤ i execute
        If s[i] ≠ s[n - i + 1] then
            return 0
        EndIf
        i ← i - 1
    EndWhile
    return 1
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevarate:
A. Algoritmul returnează 1 dacă **s** are un număr par de caractere.
B. Algoritmul returnează 1 daca **s** este palindrom.
C. Algoritmul are o eroare deoarece expresia n-i+1 poate avea valori negative în cursul execuției.
D. Algoritmul returnează 1 dacă **s** conține doar caractere alfanumerice.

(Concursul Mate-Info UBB, martie 2022)

Răspuns corect: B

10. Se consideră algoritmul `f(v, n)`, unde ***n*** este număr natural nenul ($1 \leq n \leq 10000$) şi ***v*** este un vector cu ***n*** numere naturale pozitive (***v***[1], ***v***[2], ..., ***v***[***n***]). Presupunem că algoritmul `prim(d)` returnează *True* dacă ***d*** (număr natural) este prim şi *False* în caz contrar.

```
Algorithm f(v, n):
    x ← 1
    a ← 0
    For i ← 1, n execute
        For d ← 2, (v[i] DIV 2) execute
            If (prim(d) = True) AND (v[i] MOD d = 0) then
                x ← x * d
            EndIf
        EndFor
    EndFor
    For d ← 2, (x DIV 2) execute
        If (x MOD d = 0) AND (prim(d) = True) then
            a ← a + 1
        EndIf
    EndFor
    return a
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Algoritmul returnează numărul divizorilor proprii primi distincți ai tuturor numerelor din vectorul ***v***.
B. Algoritmul returnează produsul divizorilor primi ai numerelor din vectorul ***v***.
C. Algoritmul returnează numărul numerelor prime din vectorul ***v***.
D. Algoritmul returnează numărul total al tuturor divizorilor numerelor din vectorul ***v***.

(Admitere licenta, sesiunea septembrie 2022)

Răspuns corect: A