



Algoritmi care lucrează cu tablouri unidimensionale

Partea I

13.11.2020

Grigoreta Cojocar

Subiecte grila:

1. Se consideră subalgoritmul $\text{calcul}(a, n)$, 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 $\text{calcul}(a, n)$:

Dacă $n=0$ **atunci**

returnează 0;

altfel

returnează $a[n] * (a[n] \bmod 2) + \text{calcul}(a, n-1)$;

sfDacă

SfFuncție

Pentru ce valori a numărului n și a sirului a funcția $\text{calcul}(a, n)$ 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. $n = 7, a = (1, 1, 2, 1, 1, 1, 3)$

Răspuns corect: B

2. Se considera subalgoritmul $\text{calcul}(v, n)$, 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 $\text{calcul}(v, n)$:

$m \leftarrow 0$

$x \leftarrow 0$

$s \leftarrow 0$

Pentru $i=1, n$ **execută**

$s \leftarrow s + v[i]$

$m \leftarrow m + (s \bmod 2 + x) \bmod 2$

$x \leftarrow s \bmod 2$

sfPentru

returnează m ;

SfSubalgoritm



Precizați care dintre următoarele afirmații sunt adevărate:

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

Răspuns corect: C

3. Subalgoritmul de mai jos are ca parametri de intrare un șir 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]  
        Cât timp c MOD 3 = 0 execută  
            x ← x + 1  
            c ← c DIV 3  
        SfCât timp  
    SfPentru  
    returnează x  
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

- A. Subalgoritmul returnează numărul numerelor divizibile cu 3 din șirul 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 șirul v

Răspuns corect: B

4. Spunem că un șir având n caractere este antipalindrom dacă toate perechile de caractere egal depărtate de extremități sunt distincte două câte două (cu excepția celui din mijloc dacă n este impar). De exemplu, *asdfg* și *x/x* sunt antipalindroame, dar *asds* nu este.

Fie subalgoritmul `antipalindrom(s, stânga, dreapta)` care primește ca și parametru un șir s cu n ($1 \leq n \leq 10000$) caractere ($s[1], s[2], \dots, s[n]$), și numerele naturale $stânga$ și $dreapta$.

Care din următoarele implementări vor returna adevărat pentru apelul `antipalindrom(s, 1, n)` dacă și numai dacă șirul s este antipalindrom?



<p>A. Subalgoritm antipalindrom(s, stânga, dreapta): Dacă stânga = dreapta atunci returnează <i>adevărat</i> altfel prim \leftarrow s[stânga] ultim \leftarrow s[dreapta] Dacă prim = ultim atunci returnează <i>fals</i> altfel returnează antipalindrom(s, stânga + 1, dreapta - 1) SfDacă SfSubalgoritm</p>	<p>B. Subalgoritm antipalindrom(s, stânga, dreapta): Dacă stânga \geq dreapta atunci returnează <i>adevărat</i> SfDacă prim \leftarrow s[stânga] ultim \leftarrow s[dreapta] Dacă prim = ultim atunci returnează <i>fals</i> altfel returnează antipalindrom(s, stânga + 1, dreapta - 1) SfDacă SfSubalgoritm</p>
<p>C. Subalgoritm antipalindrom(s, stânga, dreapta): Dacă stânga > dreapta atunci returnează <i>adevărat</i> altfel prim \leftarrow s[stânga] ultim \leftarrow s[dreapta] Dacă prim \neq ultim atunci returnează <i>fals</i> altfel returnează antipalindrom(s, stânga + 1, dreapta - 1) SfDacă SfSubalgoritm</p>	<p>D. Subalgoritm antipalindrom(s, stânga, dreapta): Dacă stânga > dreapta atunci returnează <i>adevărat</i> SfDacă prim \leftarrow s[stânga] ultim \leftarrow s[dreapta] Dacă prim \neq ultim atunci returnează <i>adevărat</i> SfDacă returnează antipalindrom(s, stânga + 1, dreapta - 1) SfSubalgoritm</p>

Răspuns corect: B

Probleme rezolvate:

1. Fiind dat un șir $X=(x_1, x_2, \dots, x_n)$ de numere naturale nenule ($1 \leq x_i \leq 3000, 1 \leq n \leq 500$), să se determine poziția de început și lungimea celei mai lungi subsecvențe de numere *prime între ele* din șir.

Exemple:

$X=(2, 2, 2) \Rightarrow$ poziția ?, lungimea=0

$X=(2, 3, 5) \Rightarrow$ poziția=1, lungimea=3

$X=(20, 21, 121) \Rightarrow$ poziția=1, lungimea=3

$X=(4, 2, 3, 8, 11, 13, 17, 9, 23, 29, 31, 20) \Rightarrow$
 poziția=4, lungimea = 8, (8, 11, 13, 17, 9, 23, 29, 31)
 sau pozitia=5, lungimea = 8 (11, 13, 17, 9, 23, 29, 31, 20)



$X=(4, 2, 3, 8, 11, 13, 17, 9, 23, 29, 31) \Rightarrow$ lungimea = 8, poziția=4

Observație: În exemple șirurile sunt indexate începând cu 1.

Analiză

Se parcurge șirul X și pentru fiecare poziție i se obține lungimea maximă a subsecvenței de numere prime între ele care începe pe poziția i . Dacă lungimea obținută este cel puțin 2 și este mai mare decât lungimea maximă obținută anterior se retine noua lungime și poziția de început.

Specificarea funcțiilor

Subalgoritmul **citire_sir(x,n)**

Descriere: citește șirul x cu n elemente

Date: -

Rezultate: n - numărul de elemente, x -elementele șirului

Funcția **cmmdc(a, b)**

Descriere: determină cel mai mare divizor comun al numerelor a și b

Date: a, b ($a > 0, b > 0$)

Rezultate: $\text{cmmdc}(a,b)$

Funcția **secv_poz(x, n, i)**

Descriere: determină lungimea maximă a subsecvenței de numere prime între ele din șirul x care începe pe poziția i

Date: x - elementele șirului, n - numărul de elemente a șirului x , i -poziția de început

Rezultate: l - lungimea secvenței, sau 0

Subalgoritmul **secv_max(x, n, poz_inceput, lungime)**

Descriere: determină poziția de început și lungimea subsecvenței maxime de numere prime între ele din șirul x . Dacă există mai multe subsecvențe de lungime maximă, va determina poziția primei subsecvențe

Date: x - elementele șirului, n - numărul de elemente din x

Rezultate: **poz_inceput**- poziția de început a subsecvenței

lungime - lungimea maximă, sau 0 dacă nu există numere prime între ele în șirul x

Subalgoritmul **tiparire_sir(x,n, start, lungime);**

Descriere: Tipărește subsecvența din șirul x de lungime n care începe pe poziția **start** și are **lungime** elemente

Date: x - elementele șirului, n - numărul de elemente ale șirului x , **start** - poziția de început a subsecvenței, **lungime**- lungimea subsecvenței

Rezultate: -

În continuare sunt prezentați în limbajul pseudocod doar cei mai importanți (sub)algoritmi folosiți în rezolvarea acestei probleme: algoritmul **SecventaPrimeIntreEle** și subalgoritmii **secv_max** și **secv_poz**. Ceilalți subalgoritmi: **cmmdc**, **citire_sir** și **tiparire_sir** sunt incluși doar în soluțiile prezentate în limbajele C++ și Pascal.

Algoritm **SecventaPrimeIntreEle** este:

citire_sir(x,n);

secv_max(x,n , **poz_inceput**, **lungime**);

tiparire_sir(x,n ,**poz_inceput**, **lungime**);

Sfialgoritm



Date intrare: x,n

Rezultate: poz_inceput, lungime

Subalgoritm `secv_max(x,n, poz_inceput, lungime)` este:

```
poz_inceput=0;
lungime=0;
Pentru i=1,n, 1 este
    lung=secv_poz(x,n,i);
    Daca lung>lungime AND lung >1 atunci
        lungime=lung;
        poz_inceput=i
    sfidaca
sfpentru
```

sfsubalgoritm

Date intrare: x,n, poz

Rezultate: lungime

Functie `secv_poz(x,n,poz)` este:

```
lungime=1;
Cattimp (poz+lungime<=n) executa
    j=poz; ok=adevarat
    cattimp (ok=adevarat) and (j<=poz+lungime-1) executa:
        Daca  $\text{cmmdc}(x[j], x[\text{poz}+\text{lungime}])=1$  atunci  $j=j+1$ 
        altfel ok=false
    sfidaca
    sfcattimp
    Daca ok=adevarat atunci lungime=lungime+1
    altfel returneaza lungime;
sfdaca
sfcattimp
returneaza lungime;
```

sffunctie

Observatie:

Subalgoritmul `secv_max` poate fi optimizat, daca tinem cont ca in anumite situatii nu este necesar sa parcurgem tot sirul si sa obtinem lungimea subsecventei care respecta proprietatea ceruta pentru fiecare pozitie din sir. De exemplu, daca avem sirul $X=(2, 3, 5)$, dupa ce am determinat lungimea subsecventei care incepe pe pozitia 1 (lungimea va fi 3), nu mai este necesar sa determinam lungimea subsecventei care incepe pe pozitia 2 sau pe pozitia 3, deoarece lungimile acestor subsecvente nu pot fi mai mari decat lungimea obtinuta anterior (nu mai exista elemente suficiente in sir, astfel incat lungimea sa fie mai mare decat 3). Daca numarul de pozitii pentru care nu s-a determinat inca lungimea subsecventei este mai mic decat lungimea subsecventei maxime obtinute pana in acel moment, stim deja ca nu vom putea obtine o lungime mai mare



pentru subsecvențele respective. Ținând cont de această observație, subalgoritmul **secv_max** poate fi rescris astfel:

Date intrare: x,n

Rezultate: poz_inceput, lungime

Subalgoritm secv_max(x,n, poz_inceput, lungime) este:

```
poz_inceput=0;
lungime=0;
i=1;
Cattimp (i<=n) AND (n-i>=lungime) executa
    lung=secv_poz(x,n,i);
    Daca lung>lungime AND lung >1 atunci
        lungime=lung;
        poz_inceput=i
    sfdaca
    i=i+1;
sfcattimp
sfsubalgoritm
```

Programul in C++

```
#include<iostream>
using namespace std;
#define MAX 50
typedef int sir[MAX];

void citire_sir(sir a, int &n){
    cout<<"Introduceti lungimea vectorului ";
    cin>>n;
    cout<<"Introduceti elementele vectorului ";
    for(int i=1;i<=n;i++)
        cin>>a[i];
}

void tipareste_sir(sir a, int n, int start, int lung){
    for(int i=start;i<=start+lung-1;i++)
        cout<<a[i]<<" ";
    cout<<endl;
}

int cmmdc(int a, int b){
    //verificam daca a=0 sau b=0
    if (a*b==0)
        return a+b;
    while (a!=b){
        if (a>b) a=a-b;
        else b=b-a;
    }
    return a;
}
```



```
}
int secv_poz(sir x, int n, int poz){
    int lungime=1;
    while(poz+lungime<=n){
        int j=poz;
        bool ok=true;
        while (ok && (j<poz+lungime))
            if (cmmdc(x[j],x[poz+lungime])==1) j++;
            else
                ok=false;
        if (ok) lungime++;
        else
            return lungime;
    }
    return lungime;
}

/* Variabila booleana poate fi declarata si initializata si inaintea instructiunii
while, fara a modifica rezultatul
int secv_poz(sir a, int n, int poz){
    int lungime=1;
    bool ok=true;
    while ((poz+lungime<=n)&& ok){ //cat timp nu am ajuns la finalul sirului si toate
numerele sunt prime intre ele
        for(int i=poz;(i<poz+lungime)&&ok; i++){
            if (cmmdc(a[i],a[poz+lungime])!=1)
                ok=false;
        }
        if (ok) lungime++;
    }
    return lungime;
}

*/

void secv_max(sir a, int n, int& start, int& lung){
    lung=0;
    start=1;
    for(int i=1;i<=n;i++)
//determinam lungimea maxima a subsecventei de numere prime intre ele care incepe pe
//pozitia i
        int l=secv_poz(a,n,i);
        if (l>=2 && l>lung){
            start=i;
            lung=l;
        }
    }
}

/*Versiunea optimizata a subalgoritmului secv_max
```



```
void secventa_maxima(sir a, int n, int& start, int& lung){
    lung=0;
    start=1;
    int i=1;
    while ((i<=n)&& (n-i>=lung)){
        int l=secv_poz(a,n,i);
        if (l>=2 && l>lung){
            start=i;
            lung=l;
        }
        i++;
    }
}

int main(){
    sir x;
    int n, start, lung;
    citire_sir(x,n);
    secv_max(x,n,start,lung);
    if (lung<2)
        cout<<"Nu exista nicio secventa de numere prime intre ele"<<endl;
    else
        tipareste_sir(x,n,start,lung);
    return 0;
}
```

Programul in Pascal

```
Program SecventaMaximaPrimeIntreEle;
type vector=array[1..100] of integer;

procedure citireSir(var n:integer; var x:vector);
var i:integer;
begin
    writeln('Introduceti nr. de elemente');
    readln(n);
    writeln('Introduceti elementele');
    for i:=1 to n do
        begin
            read(x[i]);
        end;
end;

Function cmmdc(a,b:integer):integer;
begin
    if (a>0) AND (b>0) then
        begin
            while (a<>b) do
                if (a>b) then a:=a-b
                else b:=b-a;
        end;
end;
```




```
cmmdc:=a;
end
else
cmmdc:=a+b;

end;

Function secv_poz(n:integer;x:vector;poz:integer):integer;
var l,k:integer;
    ok:boolean;
begin
    l:=1;
    ok:=true; {presupunem ca numerele sunt prime intre ele}
    while (poz+l<=n) and (ok) do
        begin
            k:=poz;
            while (k<poz+l) and (ok) do
                begin
                    if (cmmdc(x[k],x[poz+l])<>1) then ok:=false;

                    inc(k);
                end;
                if (ok) then inc(l);
            end;
            secv_poz:=l;
        end;
    end;

Procedure secv_max(n:integer;x:vector; var start,lungime:integer);
var i,lung_secv:integer;
begin
    lungime:=0;
    i:=1;
    while (i<=n) and (n-i>=lungime) do
        begin
            lung_secv:=secv_poz(n,x,i);
            if (lung_secv>lungime) then
                begin
                    lungime:=lung_secv;
                    start:=i;
                end;
            i:=i+1;
        end;
    end;

Procedure TiparireSecv(n:integer; x:vector; start,lungime:integer);
var i,j:integer;
begin
    for i:=start to start+lungime-1 do
        begin
```



```
        write(x[i], ' ');  
    end;  
    writeln;  
  
end;  
  
var n,start,lung:integer;  
    x:vector;  
begin  
    citireSir(n,x);  
    secv_max(n,x,start,lung);  
    if (lung>1) then  
        TiparireSecv(n,x,start, lung)  
    else  
        writeln('Nu exista nicio secventa de numere prime intre ele');  
    end.  
end.
```

2. Se dă o mulțime de maxim 60 de numere naturale.

Se cer toate submulțimile disjuncte de numere *prietene* având cel puțin 2 elemente.

Spunem că două numere naturale p și q sunt *prietene* dacă suma tuturor divizorilor lui p este egală cu suma tuturor divizorilor lui q .

Observație: Considerăm ca șirurile sunt indexate începând cu 1.

Exemplu

$M=\{68, 82, 64, 93, 127, 86, 131, 121, 137, 76, 139, 66, 70, 94, 115, 119, 149, 111, 151, 99, 125, 157, 133, 106, 163, 60, 78, 92, 123, 143, 167, 98, 173, 129, 88, 118, 145, 179, 117, 181, 169, 80, 122, 105, 141, 155, 161, 191, 193, 57\}$

Rezultat:

Submultime	Suma divizori (sd) (aceasta valoare nu trebuie afisata)
{ 68, 82 }	126
{ 93, 127 }	128
{ 86, 131 }	132
{ 76, 139 }	140
{ 66, 70, 94, 115, 119 }	144
{ 111, 151 }	152
{ 99, 125 }	156
{ 60, 78, 92, 123, 143, 167 }	168
{ 88, 118, 145, 179 }	180
{ 117, 181 }	182
{ 80, 122 }	186
{ 105, 141, 155, 161, 191 }	192

Suma divizorilor lui 68 este $126 = 1+2+4+17+34+68$



Solutia 1:

Se calculează suma divizorilor pentru fiecare element din sir și se reține valoarea într-un nou sir, numit sd . Se ordonează valorile din sirul sd în ordine crescătoare interschimbând și elementele din sirul inițial, iar apoi în sirul sd se determină toate subsecvențele de valori egale. Dacă lungimea unei subsecvențe este mai mare sau egală cu 2, înseamnă că s-a găsit o submulțime de numere prietene (conform cerinței).

Exemplu:

Fie mulțimea {68, 179, 93, 105, 127, 86, 80, 131, 145, 122, 82, 88, 118} pastrată în sirul $a = (68, 179, 93, 105, 127, 86, 80, 131, 145, 122, 82, 88, 118)$.

Pentru început se va construi sirul după regula $sd[i] = \text{suma divizorilor lui } a[i]$.

i	1	2	3	4	5	6	7	8	9	10	11	12	13
a_i	68	179	93	105	127	86	80	131	145	122	82	88	118
sd_i	126	180	128	192	128	132	186	132	180	186	126	180	180

După construire, se ordonează crescător elementele sirului sd , interschimbând corespunzător și elementele din sirul a .

i	1	2	3	4	5	6	7	8	9	10	11	12	13
a_i	68	82	93	127	86	131	179	145	88	118	80	122	105
sd_i	126	126	128	128	132	132	180	180	180	180	186	186	192

După ordonare, se determină în sirul sd toate secvențele de valori egale, care au cel puțin 2 elemente. Fiecărei secvențe de numere egale îi va corespunde o submulțime din sirul a care satisface cerința problemei:

- pentru secvența (126, 126) din sirul sd , se obține submulțimea {68, 82}.
- pentru secvența (128, 128) din sirul sd , se obține submulțimea {93, 127}.
- pentru secvența (132, 132) din sirul sd , se obține submulțimea {86, 131}.
- pentru secvența (180, 180, 180, 180) din sirul sd , se obține submulțimea {179, 145, 88, 118}.
- pentru secvența (186, 186) din sirul sd , se obține submulțimea {80, 122}.
- pentru secvența (192) din sirul sd , nu se obține o mulțime deoarece secvența este formată dintr-un singur element.

Program C++

```
#include <iostream>
using namespace std;
#define MAX 60
typedef int sir[MAX];

void citeste_sir(sir a, int& n){
    cout<<"Introduceti numarul de elemente ";
    cin>>n;
    cout<<"Introduceti elementele: ";
    for(int i=1; i<=n; i++)
```



```
        cin>>a[i];
    }

    void tipareste_sir(sir a, int n){
        for(int i=1;i<=n;i++)
            cout<<a[i]<<" ";
        cout<<endl;
    }
    int sumaDivizori(int n){
        int sum=1;
        for(int i=2;i<=n/2;i++)
            if (n%i==0)
                sum+=i;
        return sum+n;
    }
    void adaugaElem(sir a, int &n, int val){
        a[++n]=val;
    }

    void swap(int& x,int& y){
        int aux=x;
        x=y;
        y=aux;
    }
    void determina_multimi(sir a, int n){
        sir sd, multime;
        int lmultime;
        for(int i=1;i<=n;i++)
            sd[i]=sumaDivizori(a[i]);
        //rearanjam valorile din a, astfel incat valorile din sd sa fie
        //ordonate crescator
        bool sw=true;
        while(sw){
            sw=false;
            for(int i=1;i<n;i++){
                if (sd[i]>sd[i+1]){
                    sw=true;
                    swap(sd[i],sd[i+1]);
                    swap(a[i],a[i+1]);
                }
            }
        }
        //determinam in sd subsecventele care contin doar valori egale
        int i=1;
        while (i<=n){
            int j=i+1;
            lmultime=0;
            adaugaElem(multime, lmultime, a[i]);
            while((j<=n)&&(sd[i]==sd[j])){
                adaugaElem(multime, lmultime, a[j]);
                j++;
            }
        }
    }
}
```



```
    }  
    if (lmultime>1)  
        tipareste_sir(multime, lmultime);  
    i=i+lmultime;  
}  
}  
  
int main(){  
    // int m=50;  
    // sir x={0,68, 82, 64, 93, 127, 86, 131, 121, 137, 76, 139, 66, 70, 94, 115, 119, 149, 111, 151, 99, 125,  
    // 157, 133, 106, 163, 60, 78, 92, 123, 143, 167, 98, 173, 129, 88, 118, 145, 179, 117, 181, 169, 80, 122, 105,  
    // 141, 155, 161, 191,193, 57};  
    citeste_sir(x,m);  
    determina_multimi(x,m);  
    return 0;  
}
```

Program Pascal

```
Program SubmultimiNrPrietenie;  
type vector=array[1..100] of integer;  
  
procedure citireMultime(var n:integer; var x:vector);  
var i:integer;  
begin  
    writeln('Introduceti nr. de elemente');  
    readln(n);  
    writeln('Introduceti elementele');  
    for i:=1 to n do  
        begin  
            read(x[i]);  
        end;  
end;  
  
procedure tiparireMultime(n:integer;x:vector);  
var i:integer;  
begin  
    if n=0 then  
        writeln('Multimea vida')  
    else  
        begin  
            write('Multimea: ');  
            for i:=1 to n do  
                write(x[i], ' ');  
            end;  
            writeln;  
        end;  
end;  
  
function SumaDivizori(n:integer):integer;  
var suma,i:integer;  
begin  
    suma:=1;
```



```
for i:=2 to n div 2 do
    if (n mod i=0) then suma:=suma+i;
if (n>1) then suma:=suma+n;
SumaDivizori:=suma;
end;

Procedure CreazaSirSd(n:integer; x:vector;var sd:vector);
var i:integer;
begin
    for i:=1 to n do
        sd[i]:=SumaDivizori(x[i]);
    end;

Procedure AdaugaElem(var a:vector; var n:integer; val:integer);
begin
    inc(n);
    a[n]:=val;
end;

Procedure swap(var x,y:integer);
var aux:integer;
begin
    aux:=x;
    x:=y;
    y:=aux;
end;

Procedure DeterminaMultimi(n:integer;x:vector);
var sd, multime:vector;
    i,j,lung_multime:integer;
    sw:boolean;
begin
    CreazaSirSd(n,x,sd);
    {rearanjam valorile din x, astfel incat valorile din sd sa fie
    ordonate crescator }
    sw:=true;
    while (sw) do
        begin
            sw:=false;
            for i:=1 to n-1 do
                if sd[i]>sd[i+1] then
                    begin
                        sw:=true;
                        swap(sd[i],sd[i+1]);
                        swap(x[i],x[i+1]);
                    end;
            end;
        {determinam in sd subsecventele care contin doar valori egale}
        i:=1;
        while (i<=n) do
            begin
                j:=i+1;
```



```

lung_multime:=0;
AdaugaElem(multime, lung_multime,x[i]);
while (j<=n) and (sd[i]=sd[j]) do
    begin
        AdaugaElem(multime,lung_multime,x[j]);
        inc(j);
    end;

    if (lung_multime>1) then tiparireMultime(lung_multime, multime);
    i:=i+lung_multime;
end;
end;
var n:integer;
    x:vector;
begin
    citireMultime(n,x);
    determinaMultimi(n,x);
end.

```

Solutia 2:

Se calculeaza suma divizorilor pentru fiecare element din sir si se retine valoarea intr-un nou sir, numit sd . Pentru determinarea submultimilor disjuncte, se foloseste un vector aditional care va pastra valorile din sirul sd pentru care s-au determinat deja multimele. Multimea corespunzatoare unei valori din sd se obtine prima data cand se intalneste valoarea respectiva in sd .

Fie multimea {68, 179, 93, 105,127, 86, 80, 131, 145,122, 82, 88, 118} pastrata in sirul

$a = (68, 179, 93, 105,127, 86, 80, 131, 145,122, 82, 88, 118)$.

Pentru inceput se va construi sirul dupa regula $sd[i]=\text{suma divizorilor lui } a[i]$.

i	1	2	3	4	5	6	7	8	9	10	11	12	13
a_i	68	179	93	105	127	86	80	131	145	122	82	88	118
sd_i	126	180	128	192	128	132	186	132	180	186	126	180	180

Dupa construire, se initializeaza vectorul *prelucrate* care va pastra valorile din sd pentru care s-a determinat deja multimea corespunzatoare de numere prietene. Apoi, se parcurge sirul sd si pentru fiecare element sd_i se verifica daca exista si in vectorul *prelucrate*. Daca nu exista, se adauga valoare sd_i in *prelucrate* si se construiesc multimea corespunzatoare valorii lui sd_i parcurgand elementele din sd aflate pe pozitiile $i+1, i+2, \dots, n$ si verificand daca sunt egale cu sd_i . Daca gasim un element egal pe o pozitie k , inseamna ca a_i si a_k fac parte din aceeasi submultime si vor fi adaugate multimii respective.

De exemplu, pentru $sd_1=126$, se adauga valoarea 126 in vectorul *prelucrate*, iar apoi se cauta printre elementele $sd_2, sd_3, \dots, sd_{13}$ valori egale cu $sd_1=126$. Elementul de pe pozitia 11, are aceeași valoare $sd_1=sd_{11}=126$, si se obtine submultimea formata din elementele a_1 si a_{11} , adica {68,82}. Se procedeaza asemanator pentru valorile 180, 128,192. Cand se ajunge la pozitia 5,



deoarece valoarea 128 ($sd_5=128$) exista deja in vectorul *prelucrate*, nu se mai parcurg elementele $sd_6, sd_7, \dots, sd_{13}$ si nu se mai construiesc o submultime. Se continua procedeul cu $sd_6, sd_7, \dots, sd_{13}$.

Program C++

```
#include <iostream>
using namespace std;
#define MAX 60
typedef int sir[MAX];

void citeste_sir(sir a, int& n){
    cout<<"Introduceti numarul de elemente ";
    cin>>n;
    cout<<"Introduceti elementele: ";
    for(int i=1;i<=n;i++)
        cin>>a[i];
}

void tipareste_sir(sir a, int n){
    for(int i=1;i<=n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
}

int sumaDivizori(int n){
    int sum=1;
    for(int i=2;i<=n/2;i++)
        if (n%i==0)
            sum+=i;
    return sum+n;
}

//verifica daca valoarea val apare in sirul a cu n elemente
//daca apare, returneaza pozitia pe care apare, altfel returneaza -1
int pozitie(sir a, int n, int val){
    for(int i=1;i<=n;i++)
        if (a[i]==val)
            return i;
    return -1;
}

//adauga valoarea val la sfarsitul sirului a cu n elemente
void adaugaElem(sir a, int &n, int val){
    a[++n]=val;
}

void determina_multimi(sir a, int n){
    sir sd, prelucrate, multime;
    int lung_prelucrate, lmultime;
    for(int i=1;i<=n;i++)
        sd[i]=sumaDivizori(a[i]);
    lung_prelucrate=0;
    for(int i=1;i<=n;i++){
        int poz=pozitie(prelucrate, lung_prelucrate, sd[i]);
```




```
        if (poz<0){
            //inca nu a fost tiparita multimea corespunzatoare valorii sd[i]
            adaugaElem(prelucrate, lung_prelucrate, sd[i]);
            lmultime=0;
            adaugaElem(multime, lmultime, a[i]);
            for(int j=i+1; j<=n;j++)
                if (sd[j]==sd[i])
                    adaugaElem(multime, lmultime, a[j]);
            if (lmultime>=2){
                cout<<"Submultimea: "<<endl;
                tipareste_sir(multime, lmultime);
            }
        }
    }
}

int main(){
    //    int m=50;
    //    sir x={0,68, 82, 64, 93, 127, 86, 131, 121, 137, 76, 139, 66, 70, 94, 115, 119,
    //    149, 111, 151, 99, 125, 157, 133, 106, 163, 60, 78, 92, 123, 143, 167, 98, 173, 129,
    //    88, 118, 145, 179, 117, 181, 169, 80, 122, 105, 141, 155, 161, 191,193, 57};
    citeste_sir(x,m);
    determina_multimi(x,m);
    return 0;
}
```

Program Pascal

```
Program SubmultimiNrPrietenie;
type vector=array[1..100] of integer;

procedure citireMultime(var n:integer; var x:vector);
var i:integer;
begin
    writeln('Introduceti nr. de elemente');
    readln(n);
    writeln('Introduceti elementele');
    for i:=1 to n do
        begin
            read(x[i]);
        end;
end;

procedure tiparireMultime(n:integer;x:vector);
var i:integer;
begin
    if n=0 then
        writeln('Multimea vida')
    else
        for i:=1 to n do
            begin
                write(x[i], ' ');
            end;
        writeln;
    end;
end;
```



```
end;
writeln;
end;

function SumaDivizori(n:integer):integer;
var suma,i:integer;
begin
    suma:=1;
    for i:=2 to n div 2 do
        if (n mod i=0) then suma:=suma+i;
        if (n>1) then suma:=suma+n;
    SumaDivizori:=suma;
end;

function DeterminaPozitie(n:integer; x:vector;val:integer):integer;
var i,poz:integer;
begin
    poz:=-1;
    i:=1;
    while (i<=n) and (poz<0) do
        begin
            if (x[i]=val) then
                poz:=i;
                inc(i);
            end;
        DeterminaPozitie:=poz;
    end;

Procedure CreazaSirSd(n:integer; x:vector;var sd:vector);
var i:integer;
begin
    for i:=1 to n do
        sd[i]:=SumaDivizori(x[i]);
    end;
Procedure AdaugaElem(var a:vector; var n:integer; val:integer);
begin
    inc(n);
    a[n]:=val;
end;

Procedure DeterminaMultimi(n:integer;x:vector);
var sd, prelutrate, multime:vector;
    i,j,lung_multime,lung_prelutrate:integer;
begin
    CreazaSirSd(n,x,sd);
    lung_prelutrate:=0;
    for i:=1 to n-1 do
        begin
            lung_multime:=0;
            if (DeterminaPozitie(lung_prelutrate,prelutrate,sd[i])<0) then
                begin
```



```
AdaugaElem(prelucrate, lung_prelucrate, sd[i]);
AdaugaElem(multime, lung_multime, x[i]);
for j:=i+1 to n do
    begin
        if sd[i]=sd[j] then
            begin
                AdaugaElem(multime, lung_multime, x[j]);
            end;
        end;
    if (lung_multime>1) then tiparireMultime(lung_multime, multime);
end;

end;
var n:integer;
    x:vector;
begin
    citireMultime(n,x);
    determinaMultimi(n,x);
end.
```