Algoritmi care lucrează cu tablouri unidimensionale
Partea I
13.11.2020
Grigoreta Cojocar

Subiecte grila:

1.Se consideră subalgoritmul `calcul(a, n)`, care primeste ca parametru un sir a cu ***n*** numere naturale `(a[1],a[2],...,a[n])` si numarul intreg `n (1 ≤ n ≤ 10000)`.

```
Subalgoritm calcul(a, n):
    Dacă n=0 atunci
              returnează 0;
    altfel
              returnează  a[n]*(a[n] mod 2)+calcul(a,n-1);
    sfDaca
SfFunctie
```

Pentru ce valori a numarului *n* si a sirului *a* functia *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 `calcul (v,n)`, care primeste ca parametru un sir v cu n numere naturale `(v[1],v[2], …, v[n])` si numarul intreg `n (1 <= n <= 10000).`

```
Subalgoritm calcul(v,n):
    m←0
    x←0
    s←0
    Pentru i=1,n executa
        s ← s + v[i]
       m ← m + (s MOD 2 + x) MOD 2
       x ← s MOD 2
   sfPentru
   returneaza m;
SfSubalgoritm
```

Precizati care dintre urmatoarele afirmatii sunt adevarate:
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], ..., v[n])` și numărul întreg `n (1 ≤ n ≤ 10000)`.

```
Subalgoritm f(v, n):
    x←0
    Pentru i ← 1, n execută
        c ← v[i]
        Câttimp c MOD 3 = 0 execută
            x←x+1
            c ← c DIV 3
        SfCâttimp
    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] * ... * v[n]` este divizibil cu 3^k
C. Subalgoritmul returnează cel mai mare număr `k` astfel încât `v[1] + v[2] + ... + 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 *xlxe* sunt antipalindroame, dar *asdsg* nu este.

Fie subalgoritmul `antipalindrom(s, stânga, dreapta)` care primește ca și parametru un șir **s** cu **n** (1 ≤ **n** ≤ 10000) caractere (s[1], s[2], ..., 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?

A.
```
Subalgoritm antipalindrom(s, stânga, dreapta):
     Dacă stânga = dreapta atunci
       returnează adevărat
     altfel
       prim ← s[stânga]
       ultim ← s[dreapta]
      Dacă prim = ultim atunci
           returnează fals
     altfel
returnează antipalindrom(s, stânga + 1, dreapta - 1)
       SfDacă
  SfDacă
SfSubalgoritm
```

B.
```
Subalgoritm antipalindrom(s, stânga, dreapta):
        Dacă stânga ≥ dreapta atunci
              returnează adevărat
        SfDacă
       prim ← s[stânga]
       ultim ← s[dreapta]
       Dacă prim = ultim atunci
              returnează fals
        altfel
returnează antipalindrom(s, stânga + 1, dreapta - 1)
      SfDacă
SfSubalgoritm
```

C.
```
Subalgoritm antipalindrom(s, stânga, dreapta):
    Dacă stânga > dreapta atunci
          returnează adevărat
    altfel
          prim ← s[stânga]
         ultim ← s[dreapta]
         Dacă prim ≠ ultim atunci
              returnează fals
         altfel
returnează antipalindrom(s, stânga + 1, dreapta - 1)
         SfDacă
     SfDacă
SfSubalgoritm
```

D.
```
Subalgoritm antipalindrom(s, stânga, dreapta):
    Dacă stânga > dreapta atunci
       returnează adevărat
    SfDacă
    prim ← s[stânga]
    ultim ← s[dreapta]
    Dacă prim ≠ ultim atunci
       returnează adevărat
     SfDacă
returnează antipalindrom(s, stânga + 1, dreapta - 1)
SfSubalgoritm
```

Răspuns corect: B

Probleme rezolvate:

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

Exemple:
X=(2, 2, 2) => poziția ?, lungimea=0

X=(2, 3, 5) => poziția=1, lungimea=3

X=(20, 21, 121) => poziția=1, lungimea=3

X=(4, 2, 3, 8, 11, 13, 17, 9, 23, 29, 31, 20) =>
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) => lungimea = 8, poziția=4

Observație: În exemple șirurile sunt indexate începând cu 1.

Analiză

Se parcurge sirul X si pentru fiecare pozitie *i* se obtine lungimea maxima a subsecventei de numere prime intre ele care incepe pe pozitia *i.* Daca lungimea obtinuta este cel putin 2 si este mai mare decat lungimea maxima obtinuta anterior se retine noua lungime si pozitia de inceput.

Specificarea funcțiilor

Subalgoritmul **citire_sir(x,n)**
Descriere: citeste sirul **x** cu **n** elemente
Date: -
Rezultate: **n**- numarul de elemente, **x**-elementele sirului

Functia **cmmdc(a, b)**
Descriere: determina cel mai mare divizor comun al numerelor **a** si **b**
Date: **a, b** (a>0, b>0)
Rezultate: cmmdc(a,b)

Functia **secv_poz(x, n, i)**
Descriere: determina lungimea maxima a subsecventei de numere prime intre ele din sirul **x** care incepe pe pozitia **i**
Date: x- elementele sirului, n - numarul de elemente a sirului x, i -pozitia de inceput
Rezultate: l - lungimea secventei, sau 0

Subalgoritmul **secv_max(x, n, poz_inceput, lungime)**
Descriere: determina pozitia de inceput si lungimea subsecventei maximale de numere prime intre ele din sirul **x.** Daca exista mai multe subsecvente de lungime maxima, va determina pozitia primei subsecvente
Date: **x**- elementele sirului, **n** - numarul de elemente din x
Rezultate: **poz_inceput**- pozitia de inceput a subsecventei
lungime - lungimea maxima, sau 0 daca nu exista numere prime intre ele in sirul x
Subalgoritmul **tiparire_sir(x,n, start, lungime)**;
Descriere: Tipareste subsecventa din sirul **x** de lungime **n** care incepe pe pozitia **start** si are **lungime** elemente
Date: **x** - elementele sirului, **n** - numarul de elemente ale sirului x, **start** - pozitia de inceput a subsecventei, **lungime**- lungimea subsecventei
Rezultate: -

In continuare sunt prezentati in limbajul pseudocod doar cei mai importanți (sub)algoritmi folositi in rezolvarea acestei probleme: algoritmul **SecventaPrimeIntreEle** si subalgoritmii **secv_max** si **secv_poz.** Ceilalti subalgoritmi: cmmdc, citire_sir si tiparire_sir sunt inclusi doar in solutiile prezentate in limbajele C++ si Pascal.

Algoritm **SecventaPrimeIntreEle** este:
    citire_sir(x,n);
    secv_max(x,n, poz_inceput, lungime);
    tiparire_sir(x,n,poz_inceput, lungime);
Sfalgoritm

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
                sfdaca
         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 cmmdc(x[j], x[poz+lungime])=1 atunci j=j+1
                                altfel ok=false
                        sfdaca
                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 maximale obtinute pana in acel moment, stim deja ca nu vom putea obtine o lungime mai mare

pentru subsecventele respective. Tinand cont de aceasta observatie, 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;
```

```
      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;


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.
```

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

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

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

Observație: Consideram 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 } { 93, 127 } { 86, 131 } { 76, 139 } { 66, 70, 94, 115, 119 } { 111, 151 } { 99, 125 } { 60, 78, 92, 123, 143, 167 } { 88, 118, 145, 179 } { 117, 181 } { 80, 122 } { 105, 141, 155, 161, 191 }	126 128 132 140 144 152 156 168 180 182 186 192

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

Solutia 1:

Se calculeaza suma divizorilor pentru fiecare element din sir si se retine valoarea intr-un nou sir, numit *sd*. Se ordoneaza valorile din sirul *sd* in ordine crescatoare interschimband si elementele din sirul intial, iar apoi in sirul *sd* se determina toate subsecventele de valori egale. Daca lungimea unei subsecvente este mai mare sau egala cu 2, inseamna ca s-a gasit o submultime de numere prietene (conform cerintei).

Exemplu:

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]=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 ordoneaza crescator elementele sirului *sd*, interschimband corespunzator si 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

Dupa ordonare, se determina in sirul *sd* toate secventele de valori egale, care au cel putin 2 elemente. Fiecarei secvente de numere egale ii va corespunde o submultime din sirul *a* care satisface cerinta problemei:

- pentru secventa (126, 126) din sirul *sd*, se obtine submultimea {68,82}.
- pentru secventa (128, 128) din sirul *sd*, se obtine submultimea {93,127}.
- pentru secventa (132, 132) din sirul *sd*, se obtine submultimea {86,131}.
- pentru secventa (180, 180,180,180) din sirul *sd*, se obtine submultimea {179,145,88,118}.
- pentru secventa (186, 186) din sirul *sd*, se obtine submultimea {80,122}.
- pentru secventa (192) din sirul *sd,* nu se obtine o multime deoarece secventa este formata 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 SubmultimiNrPrietene;
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;

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 multimile. 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]=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 construieste multimea corespunzatoare valorii lui sd_i parcurgand elementele din *sd* aflate pe pozitiile *i+1*, *i+2*, …,*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, …, 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, …, sd_{13} si nu se mai construieste o submultime. Se continua procedeul cu sd_6, sd_7,..., 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 SubmultimiNrPrietene;
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;

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, prelucrate, multime:vector;
     i,j,lung_multime,lung_prelucrate:integer;
begin
CreazaSirSd(n,x,sd);
lung_prelucrate:=0;
for i:=1 to n-1 do
   begin
     lung_multime:=0;
     if (DeterminaPozitie(lung_prelucrate,prelucrate,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;

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