Configurații ORDONATE

Mihai Oltean, Crina Groșan

Exsită numeroase probleme în care se dau anumite configurații între care poate fi stabilită o relație de ordine. În cadrul acestui articol vă vom prezenta câteva exemple în care trebuie să determinăm numărul de ordine al unei configurații sau configurația care are un anumit număr de ordine.

Exemplele care urmează au toate același enunț general: *se consideră o mulțime de configurații înzestrate cu o relație de ordine totală (oricare două configurații - elemente ale mulțimii - pot fi comparate).*

Fie *K* cardinalul mulțimii de configurații. Din motivul mai sus enunțat se poate stabili o relație bijectivă între mulțimea configurațiilor și mulțimea `{1,2,...,K}`.

Numărul *i* asociat unei configurații se numește număr de ordine al configurației respective în cadrul mulțimii de configurații.

Apar două probleme:

- Se dă o configurație și se cere numărul ei de ordine.
- Se dă un număr de ordine și se cere configurația atașată.

Nume din cartea de telefon

Se consideră numele persoanelor din cartea de telefon. Pentru simplificare se presupune că numele sunt formate dintr-un singur cuvânt de lungime *m,* care conține litere doar din primele *n* litere ale alfabetului.

Observație

Toate numele, care pot fi formate cu literele date, există în cartea de telefon. Nu există două persoane cu același nume.

Se dă un nume de persoană și se cere numărul lui de ordine din cartea de telefon.

Soluție

Numărul cuvintelor de lungime *m* peste un alfabet format din *n* litere este n^m. Cu o literă oarecare *x* încep n^{m-1} cuvinte.

Dacă prima literă a cuvântului este a *k*-a din cele *n*, atunci numărul de ordine al cuvântului este cel puțin $k \cdot n^{m-1}$ și cel mult $(k + 1) \cdot n^{m-1} - 1$.

Pentru cuvântul rămas după eliminarea primei litere se aplică aceleași raționamente.

Funcția `Numar(i)` returnează numărul de ordine al unui cuvânt format din ultimele $(m - i + 1)$ litere ale cuvântului inițial, în cadrul mulțimii cuvintelor de lungime $(m - i + 1)$ peste un alfabet care conține primele *n* litere ale alfabetului.

Listing TELEFON.PAS

```
Program Telefon;
var a:string;
    n,m:Byte;

function N_la_M(i:Byte):Longint;
begin
  if i=0 then N_La_M:=1
         else N_La_M:=n*N_La_M(i-1)
end;

function Numar(i:Byte):Longint;
var j:Byte;
begin
  if i=m+1 then Numar := 1
  else Numar:=(Ord(a[i])-Ord('a'))*
                      N_la_M(m-i)+Numar(i+1)
end;

Begin
  Write('n='); Readln(n);
  Write('m='); Readln(m);
  Write('Numele persoanei:'); Readln(a);
  Writeln('Persoana are numarul de ordine:',
                                    Numar(1))
End.
```

Exemplu

Intrare

```
n=3
m=2
Numele persoanei:bc
```

Ieșire

```
Persoana are numarul de ordine:6
```

Permutări ordonate

Peste mulțimea permutărilor de *n* elemente se consideră relația de ordine lexicografică. Se dă o permutare cu *n* elemente și se cere numărul ei de ordine relativ la relația considerată.

Soluție

Pentru o mulțime de *n* elemente avem în total *n*! permutări ale elementelor. Considerăm permutările mulțimii {1,2,...,n}.

Numărul permutărilor de lungime *n*, care încep cu un număr oarecare *x* din mulțimea {1,2,...,n} este (*n* - 1)!.

Numărul de ordine al unei permutări de lungime *n*, care începe cu *x* este cel puțin (*x* - 1) · (*n* - 1)! + 1 și cel mult *x* · (*n* - 1)!.

Aceleași raționamente se aplică după eliminarea primei cifre a unei permutări. Trebuie avut grijă, ca permutarea rămasă să fie consistentă. Pentru a realiza acest lucru se scade o unitate din fiecare număr rămas, dacă acesta este mai mare decât cel eliminat.

Listing PERMUTARI.PAS

```
program Permutari;
const MaxN = 10;
var a:array[1..MaxN] of 1..MaxN;
    n,i:Byte;

function factorial(i:Byte):Longint;
begin
  if i=0 then factorial:=1
         else factorial:=i*factorial(i-1)
end;

function Numar(i:Byte):Longint;{ returneaza }
var j:Byte;          { numarul de ordine al }
begin        { permutarii consistente care }
  if i=n+1           { incepe pe pozitia i }
  then Numar:=1
  else
    begin
      for j:=i+1 to n do
        if a[j] > a[i] then Dec(a[j]);
      Numar:=(a[i]-1)*factorial(n-i)+
                                  Numar(i+1)
    end
end;

Begin
  Write('n='); Readln(n);
  Write('Permutarea:');
  for i:=1 to n do
    Readln(a[i]);
  Writeln('Permutarea are numarul de ',
                        'ordine:',Numar(1));
End.
```

Exemplu

Intrare

```
n=3
Permutarea:2 3 1
```

Ieșire

```
Permutarea are numarul de ordine:4
```

Submulțimi ordonate

Fie submulțimile mulțimii {1,2,...,n}. Acestea se consideră ordonate lexicografic. Se dă o submulțime și se cere numărul ei de ordine.

Soluție

Numărul submulțimilor mulțimii {1,2,...,n} este 2^n (dacă se contorizează și mulțimea vidă și submulțimea maximă, formată din toate elementele mulțimii).

Numărul submulțimilor mulțimii {1,2,...,n} care încep cu un număr oarecare *x* sunt în număr de 2^{n-x}. Numărul de ordine al unei submulțimi a mulțimii {1,2,...,n} care începe cu *x* este cel puțin $2^{n-1} + 2^{n-2} + \ldots + 2^{n-x+1}$ și cel mult $2^{n-1} + 2^{n-2} + \ldots + 2^{n-x} - 1$. Aceleași raționamente se aplică pentru submulțimea rămasă. Aceasta va trebui să fie o submulțime validă a mulțimii {1,2,...,n-1}. Pentru a realiza acest lucru scădem câte o unitate din fiecare număr al submulțimii rămase. Această operație este posibilă în cazul în care considerăm că elementele unei submulțimi sunt ordonate crescător.

Listing SUBM.PAS

```
program Submultimi;
const MaxN=10;
var a:array[1..MaxN] of 1..MaxN;
    n,m,i:Byte;

function DoiLaN(i:Byte):Longint;
begin          { calculeaza 2 la puterea i }
  DoiLaN:=1 shl i
end;

function Numar(i:Byte):Longint; { returneaza }
var j:Byte;          { numarul de ordine al }
    nr:Longint;  { submultimii consistente }
begin           { care incepe pe pozitia i }
  if i=m+1       { in submultimea initiala }
    then Numar:=1
```

```
    else
      begin
        for j:=i+1 to m do
          Dec(a[j],a[i]);
        nr:=1;
        for j:=1 to a[i]-1 do
          nr:=nr+DoiLaN(n-i-j+1);
        Numar:=nr+Numar(i+1)
      end
end;

Begin
  Write('n='); Readln(n);
  Write('m='); Readln(m);
  Writeln('Submultimea:');
  for i:=1 to m do
    Read(a[i]);
  Writeln('Submultimea are numarul de ',
                          'ordine:',Numar(1))
End.
```

Exemplu

Intrare

```
n=3
m=2
Submultimea:2 3
```

Ieșire

```
Submultimea are numarul de ordine:6
```

Parantezări ordonate

Se consideră mulțimea șirurilor de *n* perechi de paranteze rotunde împerecheate corect. Această mulțime se consideră ordonată invers după poziția pe care se închide paranteza deschisă pe prima poziție.

Regula de ordonare se repetă pentru parantezele rămase.

Se dă un șir de paranteze și se cere numărul său de ordine.

Soluție

Numărul șirurilor de *n* perechi de paranteze împerecheate corect este numărul lui *Catalan*. Acesta are formula

$$T_n = T_0 \cdot T_{n-1} + T_1 \cdot T_{n-2} + ... + T_{n-1} \cdot T_0,$$

unde $T_0 = T_1 = 1$.

Paranteza care se deschide pe prima poziție se închide pe una dintre pozițiile pare (2, 4, 6, ..., 2*n*).

Dacă prima pereche de paranteze se închide pe poziția *k*, atunci numărul de ordine al șirului care conține această configurație este cel puțin $T_{n-1} \cdot T_0 + ... + T_{n-[k/2]-1} \cdot T_{[k/2]}$ și cel mult $T_{n-1} \cdot T_0 + ... + T_{n-[k/2]} \cdot T_{[k/2]+1}$.

În interiorul acestei paranteze se află [*k*/2]-1 paranteze care se închid corect. La dreapta mai rămân [(*n*-*k*)/2] perechi de paranteze care, de asemenea, se închid corect. Pentru aceste două secvențe de paranteze se aplică aceleași raționamente.

Listing PARANT.PAS

```
Program Paranteze;
const MaxN=10;
var a:string;
    catalan:array[0..MaxN] of Longint;
    n:Byte;

procedure Calcul_Catalan;
var k,j:Byte;
begin
  catalan[0]:=1; catalan[1]:=1;
  for k:=2 to n do
  begin
    catalan[k]:=0;
    for j:=0 to k-1 do
      catalan[k]:=catalan[k]+
                  catalan[j]*catalan[k-1-j]
  end
end;

function Numar(i,n:Byte):Longint;
var nr:Longint;       { returneaza numarul }
    dif,k,j:Byte;   { de ordine al sirului }
begin          { corect de paranteze care }
  if n>0      { incepe pe pozitia i si are}
  then          { n perechi de paranteze }
    begin
      nr:=0; dif:=1; k:=1;
      while dif>0 do
      begin
        Inc(k);
        if a[i+k-1]='(' then Inc(dif)
                        else Dec(dif)
      end;
      for j:=n-1 downto k div 2 do
        Inc(nr,catalan[j]*catalan[n-1-j]);
      nr:=nr+Numar(i+1,k div 2 - 1)*
                          catalan[n-k div 2];
      Numar:=nr+Numar(i+k,n-k div 2)
    end
  else Numar:=0
end;

begin
  Write('Parantezarea:'); Readln(a);
  n:=Length(a) div 2;
  Calcul_Catalan;
  Writeln('Parantezarea are numarul de ', '
                    ordine: ',Numar(1,n)+1)
End.
```

Exemplu

Intrare

```
Parantezarea:()(())
```

Ieșire

```
Parantezarea are numarul de ordine:3
```

Reconstituirea unei configurații

Vom exemplifica acum și cazul invers, adică reconstituirea configurației atașate unui număr de ordine dat.

Raționamentele prezentate mai sus se păstrează și în această situație.

Vom trata în continuare cazul permutărilor.

Enunț

Peste mulțimea permutărilor de n elemente se consideră relația de ordine lexicografică. Se dă un număr de ordine și se cere permutarea atașată.

Soluție

Pentru o mulțime de n elemente avem în total $n!$ permutări ale elementelor. Considerăm că permutările sunt din mulțimea `{1,2,...,n}`. Fie x numărul de ordine al permutării pe care trebuie să o determinăm.

Numărul permutărilor de lungime n, care încep cu un număr oarecare k din `{1,2,...,n}` este $(n - 1)!$. Trebuie să determinăm numărul k care satisface condiția:

$$(k - 1) \cdot (n - 1)! + 1 \le x < k \cdot (n - 1)!.$$

Pe prima poziție a permutării vom amplasa numărul k astfel determinat. Aceleași raționamente se aplică pentru restul pozițiilor permutării. Pe acestea vor trebui amplasate numere din mulțimea `{1,...,k-1,k+1,...,n}`.

Listing RECONST.PAS

```
Program ReconstituiePermutare;
const MaxN=10;
var a:array[1..MaxN] of Byte;
    n,i,j,k:Byte;
    x:Longint;
    perm:array[0..MaxN] of Longint;
    folosite:set of Byte;

procedure CalculeazaPerm(n:Byte);
      { calculeaza numarul permutarile de }
      {                1,2,...,n elemente }
begin
  perm[0]:=1;
  for i:=1 to n do
    perm[i]:=perm[i-1]*i
end;

Begin
  Write('n='); Readln(n);
  Write('Numarul de ordine:'); Readln(x);
  CalculeazaPerm(n);
  folosite:=[];
  for i:=1 to n do
  begin
    k:=1;
    while k*perm[n-i]<x do Inc(k);
     { determinam numarul de pe pozitia i }
    Dec(x,(k-1)*perm[n-i]);
    a[i]:=1;
    for j:=1 to n do
    begin
      if not (j in folosite)
        then Dec(k);
      if k=0 then Break
    end;              { permutarea trebuie }
                      { sa fie consistenta }
    a[i]:=j;
    folosite:=folosite+[j];
  end;
  Write('Permutarea este:');
  for i:=1 to n do
    Write(a[i],' ')
End.
```

Exemplu

Intrare

```
n=3
x=4
```

Ieșire

```
Permutarea este: 2 1 3
```

Probleme propuse

Să se reconstituie configurațiile, plecând de la numerele de ordine pentru celelalte probleme prezentate în cadrul acestui articol.

Combinări

Fie mulțimea combinărilor de n elemente luate câte k, peste care se consideră relația de ordine lexicografică. Se dă o combinare și se cere să se determine numărul ei de ordine. Să se reconstituie combinarea care are un număr de ordine dat.

Arbori

Fie mulțimea arborilor parțiali ai unui graf neorientat complet cu n vârfuri, peste care se consideră relația de ordine lexicografică relativă la muchii. Muchia (k, j) este mai mică decât muchia (g, f), dacă $k < g$ sau $k = g$ și $j < f$. Se dă un arbore și se cere să se determine numărul lui de ordine. Să se reconstituie un arbore având un număr de ordine dat.

Parantezări ordonate lexicografic

Se consideră mulțimea șirurilor de n perechi de paranteze rotunde împerecheate corect. Această mulțime este ordonată lexicografic considerând `'(' < ')'`. Se dă un șir de paranteze și se cere numărul lui de ordine. Să se reconstituie un șir având numărul de ordine dat.

Mihai Oltean și Crina Groșan sunt doctoranzi ai Universității Babeș-Bolyai din Cluj. Pot fi contactați prin e-mail la molteanu@cs.ubbcluj.ro, *respectiv* cgrosan@cs.ubbcluj.ro.