Principiul lui Dirichlet

Capitolul

8

- **Principiul cutiei lui Dirichlet**
- **Implementări sugerate**
- **Probleme propuse**
- **Soluțiile problemelor**

8.1. Principiul cutiei lui Dirichlet

Cunoscutul matematician ***Peter Gustav Lejeune Dirichlet*** a enunțat în cartea sa *„Recherches sur les formes quadratiques à coefficients et à indéterminées complexes”* următorul principiu:

„Se consideră n obiecte care trebuie dispuse în k cutii, cu $n > k$. Atunci există o cutie care va conține cel puțin două obiecte.”

Forma generalizată a principiului:

Se consideră p obiecte care trebuie plasate în c cutii şi un număr natural n care verifică proprietatea: $p > n \cdot c$. Atunci, indiferent de dispunerea obiectelor, va exista o cutie care va conține cel puțin $n + 1$ obiecte.

Plecând de la acest principiu se pot rezolva uşor o serie de probleme.

Exemplul 1

Într-o mare bibliotecă, există întotdeauna două cărți care conțin acelaşi număr de cuvinte (Abraham, 1933).

Soluție

Un rând al cărții poate conține cel mult 20 de cuvinte. O pagină conține cel mult 50 de rânduri. O carte conține cel mult 1000 de pagini. Prin urmare o carte conține cel mult un milion de cuvinte. Deci o bibliotecă în care există cel puțin 1.000.001 cărți are două cărți cu acelaşi număr de cuvinte.

Exemplul 2

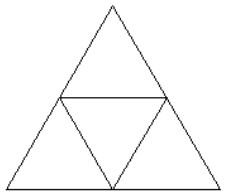
Într-un magazin de încălțăminte se află p perechi de pantofi de măsuri diferite. Din păcate pantofii nu sunt împerecheați din neglijența vânzătorului care nu i-a pus la loc în cutiile lor după ce au fost probați de clienți. Vânzătorul dorește să găsească doi pantofi care formează o pereche. Care este numărul minim de pantofi care trebuie cercetați pentru a fi siguri că s-a obținut o pereche?

Soluție

Vânzătorul va lua pe rând câte un pantof şi îl va pune fie în cutia cu perechea lui, fie într-o cutie nouă, dacă perechea lui încă nu s-a pus în cutie. În cazul cel mai defavorabil, va lua p pantofi diferiți şi îi va aşeza în cutii, iar al $p + 1$-lea pantof va fi perechea unuia deja ales.

Exemplul 3

Dându-se cinci puncte în interiorul unui triunghi echilateral de latură 1, să se arate că există două puncte cu distanța dintre ele mai mică de ½.



Soluție

Ducând liniile mijlocii în triunghi, vom obține patru triunghiuri echilaterale de latură ½. Conform principiului, există un triunghi care conține cel puțin două puncte. Aceste puncte verifică proprietatea cerută.

Exemplul 4

Se dau $n + 1$ numere naturale diferite, mai mici ca $2n$. Să se verifice dacă există printre ele trei numere, astfel încât unul dintre ele să fie egal cu suma celorlalte două.

Soluție

Fie numerele date $a_1 < a_2 < ... < a_{n+1}$. Atunci se verifică relația:
$0 < a_2 - a_1 < a_3 - a_1 < \ldots < a_{n+1} - a_1 < 2n$

Aplicăm principiul cutiei pentru numerele $a_1, a_2, ..., a_{n+1}, a_2 - a_1, ..., a_{n+1} - a_1$, fiecare având valoarea în mulțimea $\{1, 2, \ldots, 2n - 1\}$, deci cel puțin două numere sunt egale. Dar datorită condițiilor problemei, unul dintre aceste două numere va aparține mulțimii $\{a_1, a_2, ..., a_{n+1}\}$ şi celălalt mulțimii $\{a_2 - a_1, ..., a_{n+1} - a_1\}$. Fie aceste numere a_p şi $a_q - a_1$. Deci $a_p = a_q - a_1$ şi atunci $a_q = a_p + a_1$.

Exemplul 5

Se consideră m calculatoare notate cu $c_1, \ldots, c_m$ şi n imprimante notate cu $p_1, \ldots, p_n$, unde $m > n$. Se cere să se determine numărul minim de legături calculator-imprimantă, astfel încât dacă la un moment dat n calculatoare doresc simultan să scrie la imprimante, acest lucru să fie posibil (legăturile stabilite să permită celor n calculatoare să folosească legături la imprimante diferite).

Soluție

Pentru a fi îndeplinită condiția cerută trebuie ca la oricare dintre imprimante să se scrie, oricum s-ar alege n calculatoare. Fie p_i o imprimantă oarecare. Atunci numărul minim de legături necesare spre imprimanta p_i este $m - n + 1$. Astfel rămân doar $m - (m - n + 1) = n - 1$ calculatoare nelegate la p_i şi atunci oricum am alege n calculatoare, unul va scrie la ea.

Cum există n imprimante, vor fi necesare pentru fiecare imprimantă $m - n + 1$ legături şi deci în total $n(m - n + 1)$ legături.

8.2. Implementări sugerate

Pentru a vă familiariza cu această clasă de probleme, vă recomandăm să scrieți programele aferente exemplelor prezentate.

8.3. Probleme propuse

8.3.1. Camioane

Considerăm un depozit care are n încăperi, conținând diferite cantități de marfă notate cu $c_1, c_2, ..., c_n$.

Să se scrie un program care determină un grup de încăperi, având proprietatea că suma cantităților de marfă pe care le conțin se poate împărți exact la cele n camioane identice care le transportă.

Date de intrare

Prima linie a fișierului de intrare **`CAMION.IN`** conține numărul natural n. Următoarea linie conține cele n valori c_i, ($i = 1, 2, \ldots, n$) separate prin spații.

Date de ieşire

Fişierul de ieşire **`CAMION.OUT`** va conține numerele de ordine ale camerelor care aparțin soluției.

Restricții şi precizări

- $1 \le n \le 1000$;
- $1 \le c_i \le 30000$, $i = 1, 2, \ldots, n$ (c_i sunt numere naturale distincte);
- Dacă există mai multe soluții, în fişier se va scrie una singură.

Exemplu

`CAMION.IN`	**`CAMION.OUT`**
`7` `31 3 6 17 12 1 8`	`3 4 5`

8.3.2. Multiplu

Se dă un număr natural n. Să se găsească un multiplu al lui care să conțină doar cifrele 0 şi 1. Fiecare cifră va apărea în multiplu cel puțin o dată.

Date de intrare
Prima linie a fişierului de intrare **MULTIPLU.IN** conține numărul natural n.

Date de ieşire
Fişierul de ieşire **MULTIPLU.OUT** va conține multiplul cerut.

Restricții şi precizări
- $1 \le n \le 10000$.

Exemplu

MULTIPLU.IN	MULTIPLU.OUT
7	1111110

8.3.3. Numere

Se dau $(m-1)(n-1)+1$ numere naturale nenule distincte, memorate în şirul x. Să se arate că există cel puțin m numere în şir care se divid succesiv unul pe altul sau există cel puțin n care nu se divid între ele. Să se afişeze numerele care verifică aceste condiții. Se cere reprezentarea grafică a rezolvării problemei.

Date de intrare
Prima linie a fişierului de intrare **NUMERE.IN** conține numerele naturale m şi n, separate printr-un spațiu. Următoarea linie conține cele $(m-1)(n-1)+1$ numere x_i, separate prin câte un spațiu.

Date de ieşire
Numerele se vor introduce treptat în căsuțele corespunzătoare reprezentării grafice, conform exemplului. La găsirea soluției, aceasta se va afişa cu o culoare diferită.

Restricții şi precizări
- $1 \le n, m \le 100$;
- $0 < x_i < 30000$, $i = 1, 2, \ldots, (n-1)(m-1)+1$.

Exemplu

NUMERE.IN
```
5 4
9 22 11 18 15 36 99 220 180 880 440 110 33
```

O imagine posibilă pentru datele conținute în fișierul de intrare de mai sus ar fi:

9	**11**	15
18	**22**	33
36	99	**110**
180	**220**	

440

8.3.4. Nunta

La o nuntă se întâlnesc mai multe persoane, numerotate de la 1 la *n*. Să se afișeze, dacă există, două persoane care au același număr de rudenii.

Date de intrare

Prima linie a fișierului de intrare **NUNTA.IN** conține un număr natural *n*, reprezentând numărul nuntașilor. Următoarele linii conțin perechi de numere întregi *i, j* cu *i* și *j* rudenii.

Date de ieșire

Fișierul de ieșire **NUNTA.OUT** va conține două numere naturale, reprezentând două persoane cu același număr de rudenii.

Restricții și precizări

- $1 \le n \le 1000$;
- în fișier se va scrie o singură soluție.

Exemplu

NUNTA.IN	**NUNTA.OUT**
5 1 3 2 3 1 4 2 5 1 5	2 3

8.3.5. Cifre

Să se găsească un multiplu al numărului natural dat *n* care conține doar cifrele 0 și 5.

Date de intrare

Prima linie a fișierului de intrare **CIFRE.IN** conține numărul natural *n*.

Date de ieşire
Fişierul de ieşire **CIFRE.OUT** va conține multiplul cerut.

Restricții şi precizări
- $1 \le n \le 10000$.

Exemplu

CIFRE.IN	CIFRE.OUT
7	50505

8.4. Soluțiile problemelor propuse

8.4.1. Camioane

În enunț s-a precizat că avem *n* încăperi şi *n* camioane. În încăperi avem diferite cantități de marfă care ar trebui transportate cu cele *n* camioane. Nu cunoaştem capacitatea de transport a camioanelor, în schimb ştim că ele sunt identice. În concluzie, vom căuta să determinăm o sumă de cantități care este multiplu de *n* (şi astfel se poate încărca fiecare cea de a *n*-a parte în câte un camion). Astfel problema s-a redus la depistarea acelor încăperi în care suma cantităților este multiplu de *n*.

Vom considera următoarele sume:

$s_1 = c_1$
$s_2 = c_1 + c_2$
$s_3 = c_1 + c_2 + c_3$
…
$s_n = c_1 + c_2 + c_3 + \ldots + c_n$

Distingem două situații:

a) *Există i pentru care s_i este multiplu de n*. Atunci grupul de încăperi va conține încăperile având numerele de ordine: 1, 2,…, *i*.

b) *Nu există nici o sumă din cele de mai sus care să fie multiplu de n*. În acest caz, vom calcula şirul de resturi $r_i = s_i$ **mod** n, $i = 1, 2, ..., n$. Cum toate cele *n* resturi sunt nenule şi valorile lor se reduc la cele $n - 1$ elemente ale mulțimii $\{1, 2, \ldots, n - 1\}$, aplicând principul cutiei lui Dirichlet, *vor exista două resturi egale*. Fie *p* şi *q* doi indici ($p < q$) pentru care $r_p = r_q$. Atunci diferența dintre sumele s_q şi s_p este multiplu de *n*, deci soluția problemei o constituie încăperile $p + 1, p + 2, \ldots, q$. Adică ştiind că

$s_p = c_1 + c_2 + ... + c_p$ şi $s_q = c_1 + c_2 + ... + c_p + c_{p+1} + ... + c_q$, calculăm diferența

$s_q - s_p = c_1 + c_2 + ... + c_p + c_{p+1} + ... + c_q - (c_1 + c_2 + ... + c_p) = c_{p+1} + ... + c_q$.

```
Subalgoritm Determină_încăperi(n,c):
  pentru i=1,n execută:
      { calculăm şirul de resturi a sumelor parţiale a cantităţilor din cele n camere }
    rest[i] ← 0
  sfârşit pentru
  r ← 0                                   { sumele parţiale le generăm în r }
  pentru i=1,n execută:
    r ← (r + c[i]) mod n                  { calculăm sumele parţiale }
    dacă r = 0 atunci                     { dacă o sumă parţială se divide cu n }
      scrie 1,2,...,i  { cantitatea din încăperile 1, 2, ..., i este multiplu de n }
      ieşire forţată din subalgoritm
    altfel
      dacă rest[r] ≠ 0 atunci
        număr_încăperi ← i - rest[r]
        prima_încăpere ← rest[r] + 1
        pentru j=1,număr_încăperi execută:
          scrie prima_încăpere + j - 1
        sfârşit pentru
        ieşire forţată din subalgoritm
      altfel
        rest[r] ← i
      sfârşit dacă
    sfârşit dacă
  sfârşit pentru
sfârşit subalgoritm
```

8.4.2. Multiplu

Considerăm şirul numerelor: 1, 11, 111, 1111, … 11..1, unde în ultimul număr avem n cifre de 1.

Vom calcula resturile împărţirii acestor numere la n şi le vom memora în şirul $rest_i$, $i = 1, 2, ..., n$. Dacă în acest şir există un element $rest_k = 0$, atunci numărul cerut se formează din k cifre de 1 urmate de o cifră 0.

Dacă toate elementele şirului sunt nenule, ele aparţin mulţimii $\{1, 2, \ldots, n-1\}$. Conform principiului cutiei lui Dirichlet, cel puţin două dintre aceste resturi sunt egale.

Fie $p < q$ şi $rest_p = rest_q$. Atunci diferenţa dintre numărul 11…1 (având q cifre) şi numărul 11…1 (având p cifre) se împarte exact la n şi este deci multiplu de n. Numărul cerut va avea în acest caz primele $q - p$ cifre egale cu 1, iar ultimele p cifre egale cu 0.

```
Subalgoritm Determină_multiplu(n):
  rest[1] ← 1
  pentru i=2,n execută:
    rest[i] ← (rest[i-1] * 10 + 1) mod n
    dacă rest[i] = 0 atunci
      scrie i bucăți de 1 şi un singur 0
      ieşire forțată din subalgoritm
    altfel
      j ← 1
      repetă
        dacă rest[j] = rest[i] atunci
          scrie i-j bucăți de 1 şi j bucăți de 0
          ieşire forțată din subalgoritm
        altfel
          j ← j + 1
      până când j = i
    sfârşit dacă
  sfârşit pentru
sfârşit subalgoritm
```

8.4.3. Numere

Ordonăm crescător şirul x, apoi îl parcurgem şi repartizăm numerele într-o matrice de dimensiuni $(m-1) \times (n-1)$ astfel:

- Scriem un număr pe linia i dacă x se divide printr-un număr situat pe linia $i-1$ şi i este maxim având această proprietate.
- Repartizăm un număr pe o linie în prima poziție liberă.

Vom considera că în matrice există şi linia 0 (care conține elementul 1), dar pe această linie nu putem repartiza numere.

Exemplu

Fie $m = 4$, $n = 4$ şi numerele: 2, 6, 10, 16, 50, 70, 99, 100, 123, 150. Completăm matricea respectând paşii specificați anterior.

2		

2		
6		

2	99	
6	10	16
50	70	

100

Am „ieşit" din matrice pe coloana 1. 100 are divizorul 50 pe linia 3, 50 are divizorul 10 pe linia 2, 10 are divizorul 2 pe linia 1. Am găsit deci numerele 2, 10, 50 şi 100 care verifică prima proprietate.

Datorită modului de construire a matricei, în momentul în care se încearcă adăugarea unui element x_k care are un divizor pe linia $m - 1$, acesta ar trebui repartizat pe coloana 1 a liniei m. Vom putea preciza m numere care se divid reciproc. Unul dintre acestea este x_{k_i} de pe linia m. Pe linia precedentă $(m - 1)$ vom găsi în mod sigur un divizor al lui. Îl selectăm şi continuăm procedeul avansând cu încă o linie în matrice $(m - 2)$. Repetăm procedeul până la linia 1 inclusiv.

Exemplu

Fie $m = 5$, $n = 4$ şi numerele: 2, 6, 10, 12, 16, 18, 38, 56, 80, 90, 101, 300, 245.

2		

2		
6		

2		
6	10	

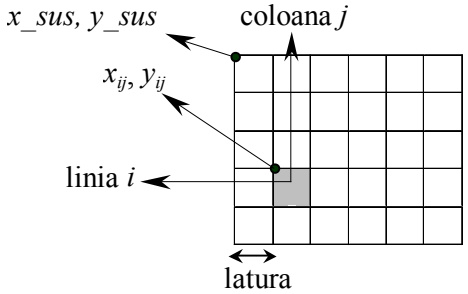
2		
6	10	
12		

2			
6	10	16	**38**
12	18		

Am „ieşit" din matrice pe linia 2. Din cauză că numerele sunt situate pe aceeaşi linie, ele nu se divid unele pe altele. Deci în acest caz soluția problemei este: 6, 10, 16, 38.

Întotdeauna se vor găsi numerele cu proprietatea cerută pentru că în matrice se pot repartiza doar $(n - 1)(m - 1)$ numere. Dar trebuie repartizate $(n - 1)(m - 1) + 1$. Deci, în cel mai defavorabil caz, ultimul număr repartizat în matrice va completa o soluție.

În vederea afişării grafice, lungimea şi lățimea rețelei de pătrate se calculează în funcție de rezoluția ecranului, stabilindu-se dimensiunea laturii unui pătrat din această rețea. Cunoscând această valoare şi stabilind coordonatele colțului stânga-sus al rețelei, se desenează efectiv rețeaua de $(n - 1)(m - 1)$ pătrate în procedura `desen_retea`, pentru fiecare pătrat calculându-se coordonatele colțurilor stânga-sus şi dreapta-jos.



Coordonatele pătratului de pe linia i şi coloana j sunt:

x_{ij} = x_sus+latura*(j-1)
y_{ij} = y_sus+latura*(i-1)

În paralel cu rezolvarea propriu-zisă a problemei se realizează desenarea numerelor în pătratele corespunzătoare. Când se completează un element `a[i,j]` al matricei, se afişează pe ecran numărul respectiv. La umplerea unei linii sau coloane, se scriu din nou pe ecran numerele care fac parte din soluție, dar colorate diferit.

```
procedure repartizeaza_deseneaza;
var i,l,j,t,y:Integer;
    max,xmax,ymax,ErrCode,gd,gm:Integer;
begin
  gd:=Detect;                          { initializare mod grafic }
  InitGraph(gd,gm,'C:\bp\bgi');
  desen_retea;
  for t:=1 to nr do begin
    l:=cauta_linie(x[t]);
    if l=m then begin     { s-a iesit din matrice pe coloana 1 }
      y:=x[t];
      pune_numar(m,1,x[t]);
      Delay(timp);
      SetColor(Red);
               { reafiseaza numerele solutiei cu alta culoare }
      pune_numar(m,1,x[t]);
      while l>1 do begin
        Dec(l);
        for j:=1 to a[l,0] do
          if y mod a[l,j]=0 then begin
            pune_numar(l,j,a[l,j]);
            y:=a[l,j];
            Break
          end;
        Delay(timp)
      end;
      ReadKey;
      CloseGraph;
      Exit
    end else
                           { s-a iesit din matrice pe o linie }
      if a[l,0]=n-1 then begin
        pune_numar(l,n,x[t]);
        Delay(1000);
             { reafiseaza numerele din solutie cu alta culoare }
        SetColor(Red);
```

```
        for i:=1 to n-1 do
          pune_numar(l,i,a[l,i]);
          pune_numar(l,n,x[t]);
          ReadKey;
          CloseGraph;
          Exit
        end else begin  { se repartizeaza elementul in matrice }
          Inc(a[l,0]);
          a[l,a[l,0]]:=x[t];
          pune_numar(l,a[l,0],x[t]);
          Delay(timp);
          if a[l,0]=1 then
            ultima_linie:=l
        end
  end
end;
```

8.4.4. Nunta

Fie gr_1, gr_2,…gr_n cu gr_i reprezentând numărul de rudenii ale persoanei i. Cum există n persoane, vom avea $0 \le gr_i \le n-1$, $i = 1,\ldots n$. Distingem două situații:

a) dacă există o cel puțin o persoană care are $n-1$ rudenii (toată lumea) rezultă că toate persoanele au cel puțin o rudenie. Atunci cele n valori din șir sunt numere din mulțimea $\{1, \ldots, n-1\}$. Conform principiului cutiei, cel puțin două valori sunt egale.
b) dacă nu există nici o persoană cu $n-1$ rudenii, cele n valori ale șirului sunt cuprinse în mulțimea $\{0, 1, 2, \ldots, n-2\}$ care are $n-1$ valori distincte. Şi în acest caz, conform principiului cutiei, există două valori care sunt egale.

În termenii din teoria grafurilor, considerând persoanele nodurile unui graf neorientat, iar relațiile de rudenie muchiile acestuia, problema se poate enunța astfel:

"În orice graf neorientat există două noduri care au acelaşi grad".

Presupunând că s-au determinat elementele şirului *gr*, algoritmul de determinare a soluției este:

```
Subalgoritmul Determină_pereche(n,gr,p,q):
  pentru i=1,n execută:
    x[i] ← 0
  sfârşit pentru
  pentru i=1,n execută:
    dacă x[gr[i]] ≠ 0 atunci
      p ← i              { mai există o persoană cu acelaşi număr de rudenii }
      q ← x[gr[i]]
      scrie p,q
      ieşire forțată din pentru
```

```
    altfel x[gr[i]] ← i
    sfârşit dacă
  sfârşit pentru
sfârşit subalgoritm
```

8.4.5. Cifre

Considerăm şirul numerelor 50, 5050, 505050, …, 5050..50.
(*n* perechi de '50' alipite)

Vom calcula resturile împărţirii acestor numere la *n* şi le vom memora în şirul $rest_i$, $i = 1, 2, ..., n$. Dacă în acest şir există un element $rest_k = 0$, atunci numărul care dă acest rest prin împărţirea la *n* este numărul cerut în problemă. Dacă toate elementele şirului sunt nenule, ele aparţin mulţimii $\{1, 2, \ldots, n-1\}$.

Conform principiului cutiei lui *Dirichlet*, cel puţin două dintre ele sunt egale.

Fie $j < i$ cu $rest_i = rest_j$. Atunci diferenţa dintre numerele care prin împărţirea la *n* au dat cele două resturi egale va constitui numărul cerut. Acesta va fi de forma: 5050…50000000..00 (*i* – *j* perechi '50' urmate de *j* perechi '00').

```
Subalgoritm Determină_multiplu(n):
  rest[1] ← 50 mod n
  pentru i=2,n execută:
    rest[i] ← (rest[i-1] * 100 + 50) mod n
    dacă rest[i] = 0 atunci
      scrie i perechi de '50'
      ieşire forţată din program
    altfel
      j ← 1
      repetă
        dacă rest[j] = rest[i] atunci
          scrie i-j perechi de '50' urmate de j perechi '00'
          ieşire forţată din program
        altfel
          j ← j + 1
        sfârşit dacă
      până când j = i
    sfârşit dacă
  sfârşit pentru
sfârşit subalgoritm
```