Probleme de concurs

Prof. Vișinescu Violeta Colegiul Național "I. L. Caragiale" Ploiești

Problema 1: Binar Ionel a învățat recent la Informatică reprezentarea numerelor în baza 2. Pentru a aprofunda cunoștiințele profesorul său a inventat următoarea problemă: Dintr-un fișier text se citește un șir de valori de 1 0 și -1. Valoarea -1 are semnificația de terminare a unui număr, iar valorile de 0 și 1 reprezintă cifrele în baza 2 a câte unui număr natural. Să se determine cele mai mari 3 valori codificate în șir impreună cu numărul lor de apariții.

Date de intrare: Fișierul binar.in cu următoarea structură:

-Pe prima linie numărul N de valori din șir.

-Pe linia a 2, N valori de 1 , 0 sau -1.

Date de ieșire: Fișierul binar.out cu următoarea structură:

-Pe primele 3 linii perechi de valori MAX, NR cu semnificația din enunț și valorile maxime în ordine descrescătoare.

Restricții și precizări:

-10<=N<=2000.

-Înaintea fiecărei valori de -1 se găsește cel puțin o valoare de 0 sau 1.

-Numerele codificate astfel sunt mai mici decât 32000 în baza 10.

-Se poate ca la stânga unui număr codificat să fie doar valori de 0.

-În șir sunt codificate cel puțin 3 valori distincte.

-Șirul de valori se încheie cu o valoare de -1.

Exemplu:

binar.in	binar.out	Explicație
19 1 0 -1 1 -1 1 0 -1 1 1 -1 1 0 1 -1 1 0 1 -1	5 2 3 1 2 2	Cel mai mare număr codificat este 101 în baza 2, adică 5 in baza 10 și apare de 2 ori. De 2 ori. Următorul este 11, adică 3 în baza 10, care apare o data. Cel de-al treilea este 10 adică 2 în baza 10, care apare de 2 ori.

Timp maxim de execuţie: 1 secundă/test.

Memorie totală disponibilă 2 MB, din care 2 MB pentru stivă

Dimensiunea maximă a sursei: 5 KB

(Concurs Empowersoft 2017 Violeta Vișinescu)

Descriere: Se efectueaza succesiv următorii pași:

-Se parcurge șirul de intrare și se formează un vector cu numerele obținute în baza 10.

-Se sortează descrescător vectorul de numere.

-Se construiesc doi vectori auxiliari: unul cu elementele distincte ale vectorului de numere și al doilea cu frecvențele lor de apariție.

-Se tipăresc alternativ primele 3 perechi de valori din vectorii auxiliari.

Programul C++ poate fi următorul:

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream fin("binar.in");
ofstream fout("binar.out");
int n, x[1001],i,j,cifra,numar,k;
int y[1001],aparitii[1001],h,aux;
int main()
{
   fin>>n;numar=0;k=0;
   for(i=1;i<=n;i++)
   { fin>>cifra;
      if (cifra==-1)
      { k++;x[k]=numar;
         numar=0;
      }
      else numar=numar*2+cifra;
   }
  //for (i=1;i<=k;i++) cout<<x[i]<<" ";
  for (i=1;i<=k-1;i++)
  for (j=i+1;j<=k;j++)
```

```
  if (x[i]<x[j]){aux=x[i];x[i]=x[j];x[j]=aux;}
  h=1;
  for (i=1;i<=k-1;i++)
    if(x[i]==x[i+1]) aparitii[h]++;
    else {y[h]=x[i];aparitii[h]++;h++;}
  y[h]=x[k];aparitii[h]++;
  for (i=1;i<=3;i++)
   fout<<y[i]<<" "<<aparitii[i]<<'\n';
  fin.close();
  fout.close();
   return 0;
}
```

Problema 2: Partiție Se consideră mulțimea totală A={1, 2, … , N}. Se notează cu intervalul [X, Y] mulțimea numerelor naturale cuprinse între X și Y inclusiv. În continuare vom numi această mulțime interval de capete X și Y. Fie o partiție a lui A compusă din M astfel de intervale puse "cap la cap": $[I_0, I_1]$, $[I_1+1, I_2]$, $[I_2+1, I_3]$, … $[I_{M-1}+1, I_M]$, cu $I_0=1$ si $I_M=N$.

Cerință: Să se determine partiția compusă din M astfel de intervale puse "cap la cap", similare ca lungimi cu cele inițiale, aranjate în ordine crescătoare a lungimilor lor.

Date de intrare: Fișierul partitie.in cu următoarea structură:

-pe prima linie valorile N și M cu semnificația din enunț, separate printr-un spațiu.

-Pe următoarele M linii, perechi X Y reprezentând capetele unui interval al partiției inițiale.

Date de ieșire: Fișierul partitie.out cu următoarea structură:

-Pe primele M linii câte o pereche X Y reprezentând capetele unui interval al partiției finale.

Restricții și precizări:

-1<=N<=1000000.

-1<=M<=1000.

-N, M, I_0, I_1, …, I_M sunt numere naturale, iar pentru orice P avem inegalitatea $I_P+1<I_{P+1}$.

Exemplu:

partitie.in	partitie.out	Explicație
20 4 1 10	1 2 3 5	Cea mai mică mulțime inițială este [11 12]. Corespunzător acesteia,

11 12 13 15 16 20	6 10 11 20	avem mulțimea finală [1 2]. A doua mulțime ca mărime este [13 15] Corespunzător acesteia avem mulțimea finală [3 5]. Mulțimii [16 20] îi corespunde mulțimea finală [6 10], iar mulțimii [1 10] mulțimea finală [11 20].

Timp maxim de execuţie: 1 secundă/test.

Memorie totală disponibilă 2 MB, din care 2 MB pentru stivă

Dimensiunea maximă a sursei: 5 KB

(O.L.I. Prahova 2018 clasa VI Violeta Vișinescu)

Descriere: Se citesc datele de intrare. Pe lângă doi vectori A și B, ce vor memora capetele celor M intervale, se utilizează un vector auxiliar OK de elemente logice, inițializat cu valori de FALSE. O variabilă K pleacă de la valoarea 1.

În mod repetat, se selectează unul dintre intervalele de lungime minimă, în vectorul auxiliar OK pe respectiva poziție se pune TRUE, iar o variabilă U primește o valoare corespunzătoare astfel încât împreună cu K, să formeze capetele intervalului final corespunzător celui selectat. Se scriu K și U.

Se execută K<-U+1 și se alege următorul interval. Procedeul se repetă de M ori.

Programul C++ poate fi următorul:

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("partitie.in");
ofstream g("partitie.out");
int a[10001],b[10001],ok[10001];
int main()
{ int n,m,i,j,k,poz,minim,u;
   f>>n>>m;
   for(i=1;i<=m;i++) {f>>a[i]>>b[i];ok[i]=0;}
   k=1;
{
      minim=1000000;
      for(j=1;j<=m;j++)
      {
 if (minim>b[j]-a[j]&&ok[j]==0)
         {minim=b[j]-a[j];
            poz=j;}
      }
         ok[poz]=1;
         u=b[poz]-a[poz]+k;
         g<<k<<" "<<u<<endl;
```

for(i=1;i<=m;i++)	k=u+1;} f.close();g.close(); return 0;

Problema 3: Periodic Se citește un număr natural nenul N. Se umple, pe linii, partea de sub diagonala principală, inclusiv, a unei matrici patratice de dimensiune L cu secvențe consecutive de: 1, 2, , … N. Astfel, pentru N=5 si L=7 se obtine triunghiul de valori:

1

2 3

4 5 1

2 3 4 5

1 2 3 4 5

1 2 3 4 5 1

2 3 4 5 1 2 3

Cerință: Fiind dați N și L precum și un indice de coloană C să se calculeze suma valorilor din triunghi aflate pe coloana C.

Date de intrare: fișierul periodic.in cu următoarea structură: pe prima și singura linie valorile N, L, si C cu semnificația din enunț.

Date de ieșire: fișierul periodic.out cu următoarea structură: pe prima și singura linie valoarea S cu semnificația din enunț.

Restricții și precizări:

1<=N<=1000, 1<=L<=10.000.000, 1<=C<=L.

Exemplu:

periodic.in	periodic.out	Explicatii
5 7 2	18	Pe coloana 2 se afla valorile: 3, 5, 3, 2, 2, 3 cu suma 18.

Timp maxim de execuţie: 1 secundă/test.

Memorie totală disponibilă 4 MB, din care 2 MB pentru stivă

Dimensiunea maximă a sursei: 5 KB

(Concurs Empowersoft 2018 Violeta Vișinescu)

Descriere: Fie elementul a[i][j]. Pe primele i-1 linii se află i*(i-1)/2 valori. Împreună cu cele C valori de pe linia I, fac in total i*(i-1)/2+C valori. Așadar valoarea a[i][j] poate fi calculata în funcție de N astfel:

NR=(i*(i-1)/2+C)%N;

If (NR==0)NR=N; Adica în O(1).

Se calculează în acest mod toate valorile de pe coloana C și se insumează. Algoritml are ordin de complexitate O(L). Programul C++ poate fi următorul:

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("periodic2.in");
ofstream g("periodic2.out");
long long n, L, C;
int main()
{ long long s,i,nr;
   f>>n>>L>>C;
s=0;
   for(i=C;i<=L;i++)
   {
      nr=(i*(i-1)/2+C)%n;
      s+=nr;
   }
   g<<s<<endl;
   return 0;
}
```

Problema 4: Cerc Considerăm N poziții aflate pe un cerc. Se pleacă din poziția 1 și la fiecare pas, se merge pe cerc un număr de P poziții în sensul acelor de ceasornic sau, dacă valoarea lui P este negativă, în sens invers acelor de ceasornic. Pe poziția la care se ajunge se inserează valoarea V. Dacă în prealabil mai există altă valoare pe acea poziție, aceasta se pierde. În final se vor tipări numai valorile aflate pe cerc în ordinea pozițiilor, de la 1 la N.

Date de intrare: Fișierul cerc.in cu următoarea structură: pe prima linie numerele N și NR reprezentând numărul de poziții pe cerc și respectiv numărul de inserări. Pe următoarele NR linii perechi de forma P V cu semnificația din enunț.

Date de ieșire: Fișierul cerc.out cu următoarea structură: Pe prima și singura linie valorile rămase pe cerc în urma inserărilor efectuate, în ordinea pozițiilor de la 1 la N.

Restricții și precizări:

1<=N<=1000;

1<=NR<=5000;

-2*N<=P<=2*N, P diferit de 0;

1<=V<=50000.

La sfârșitul fișierului de ieșire există un caracter spațiu.

Exemplu:

cerc.in	cerc.out	Explicație
10 4 2 100 -5 5 4 1 1 15	1 15 5	Se merge pe cerc la poziția 3 și se inserează valoarea 100. Mai apoi se merge invers pe cerc până la poziția 8 și se inserează valoarea 5. Se merge apoi până pe poziția 2 și se inserează valoarea 1. În final se merge pe poziția 3 și valoarea 100 se inlocuiește cu valoarea 15. Valorile rămase pe cerc sunt în ordine: 1 15 și 5.

Timp maxim de execuţie: 1 secundă/test.

Memorie totală disponibilă 2 MB, din care 2 MB pentru stivă

Dimensiunea maximă a sursei: 5 KB

(O.L.I. Prahova 2019 clasa VI Violeta Vișinescu)

Descriere: Se citesc N și NR. Poziția curentă se setează la valoarea 1. Se repetă de NR ori citirea unei perechi Pozitie Valoare, se parcurg circular numărul de poziții corespunzător și se inserează valoarea citită pe poziția curentă.

În final se tipăresc valorile nenule din vector. Programul C++ poate fi următorul:

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("cerc.in");
ofstream g("cerc.out");
int x[1001], poz,nr,p,v, n;
int main()
{
   f>>n>>nr;poz=1;
   for (int i=1;i<=nr;i++)
   {
      f>>p>>v;
      if (p>0)
 else //p<0
      {for(int j=1;j<=-p;j++)
       {poz--;if (poz<1)poz=n;}
        x[poz]=v;}
   }  for(int i=1;i<=n;i++)
      if(x[i]!=0) g<<x[i]<<" ";
   f.close();
   g.close();
   return 0;
}
```

{for(int j=1;j<=p;j++) {poz++;if (poz>n)poz=1;} x[poz]=v;}	