Probleme introductive

Probleme rezolvate

Prof. Vișinescu Radu Colegiul National „I.L. Caragiale" Ploiești

Problema 1 : Considerăm un vector X de elemente distincte. Să se exprime mulţimea formată din aceste elemente printr-o reuniune de intervale disjuncte. Astfel, pentru vectorul : X=10, 3, 8, 11, 5, 4, 16 se va tipări: [3..5]+[8]+[10..11]+[16]. **(O.L.I. Prahova 2001 Clasa IX Radu Vișinescu).**

Rezolvare:Vom folosi două cicluri while, incluse unul în altul. În cadrul primului ciclu, vom determina poziţiile de început ale secvenţelor de numere consecutive. În ciclul interior parcurgem, una câte una, aceste secvenţe. La sfârşitul unei astfel de parcurgeri, dacă secvenţa este formată din cel puţin două numere, se va tipări şi ultimul element al secvenţei. Pseudocodul este următorul:

```
// se sorteaza in ordine crescatoare vectorul X de //lungime N
I:=1;
while (I <= N) do
        { write( '[' , X[I]);
          L:=0;
          while (I<=N-1) and (X[I]+1=X[I+1]) then
                { I:=I+1;
        L:=L+1;
                }
      if (L >0) then write ( '..' , X[I]);
          write( ']+' );
          I:=I+1;
         }
```

Ordinul de complexitate al algoritmului este dat de numărul de incrementări de forma I:=I+1. Aşadar algoritmul este liniar în dimensiunea vectorului de intrare: O(N);

Problema 2: Se citeşte în vectorul X de lungime N de numere întregi o permutare a valorilor {1, 2, …, N}. Să se aranjeze crescător valorile respective efectuându-se un număr minim de interschimbări. Să se afişeze acest număr.

Exemplu N=7 X=7, 4, 3, 2, 1, 5, 6

Se va afisa: NR=4. **(O.L.I. Prahova 2004 Clasa VI Radu Vișinescu)**

Rezolvare:

```
readln(N);
for I:=1 to N do readln(X[I]);
for I:=1 to N do marcaj[I]:=0;
I:=1;NR:=0;
while (I<=N) do
{ while (marcaj[I]=1) do I:=I+1;
if (I<=N) then
 {CAP:=I;marcaj[CAP]:=1;
  I:=X[CAP];
  while (I<>CAP) do
      {marcaj[I]:=1;I:=X[I];NR:=NR+1;}
 }
}
  writeln(NR);
```

Obs: Observăm că fiecare ieşire din ciclul while(I<>CAP) valoarea variabilei I este egală cu valoarea variabilei CAP. În consecinţă numărul total de instrucţiuni executate în acel ciclu este proporţional cu N. Similar, în ciclul while (marcaj[I]=1) se execută N incrementări ale variabilei I. Aşadar ordinul de complexitate al algoritmului este O(N).

Problema 3. Se citește N numar natural. Tipăriți toti divizorii primi ai lui N fără a parcurge numerele mai mici decât N.

Descriere: Pentru a respecta cerința de a nu parcurge toate numerele mai mici ca N alegem metoda folosită la calularea descompunerii în factori primi. Plecăm cu o variabila P de la valoarea 2. Împărțim pe N la P de atâtea ori de câte ori se împarte exact. Dacă am împărțit cel puțin o dată, scriem pe P ca divizor prim. Creștem P cu o unitate și repetăm procedeul. Algoritmul se încheie în momentul în care N devine egal cu 1. Numerele la care s-a împărțit exact vor fi divizori primi căutați. Programul în pseudod este următorul:

```
readln(n);
p:=2;
while (n>1) do
 {exp:=0;
  while (n mod p=0) do
```

```
{exp:=exp+1;n:=n div p;}
 if (exp>=1) then  write(p);
 p:=p+1;}
```

Problema 4 Se citeşte un vector X de lungime N de valori întregi distincte. Să se construiască şi să se afişeze vectorul permutare Y cu valorile {1, 2, …, N}, astfel încât: oricare 1<=I, J<=N, dacă X[I]<X[J] atunci Y[I]<Y[J].

(O.L.I. Prahova 2002 Clasa IX Radu Vișinescu)

Rezolvare:

```
  readln(n);
  for i:=1 to n do  { readln(x[i]); sel[poz]:=false;}
  k:=0;
  for i:=1 to n do
  { minim=2000000;
    for j:=1 to n do
     if(sel[j]=false)
     if(minim>x[j]){minim:=x[j];poz:=j;}
     k:=k+1;
    y[poz]:=k;
    sel[poz]:=true;}
  for i:=1 to n do
      write(y[i]);
```

Problema 5. Se citeşte un vector X de lungime N de valori întregi distincte. Să se construiască şi să se afişeze un al doilea vector cu denumirea index, de lungime N, ale cărui componente vor fi numerele de la 1 la N. Vectorul index respectă următoarele cerinţe: oricare 1<=I<J<=N avem: X[index[I]]<=X[index[J]].

Exemplu:

Pentru N=7 şi vectorul X=(8, 3, 6, 10, 2, 4, 12) index=(5, 2, 6, 3, 1, 4, 7).

Rezolvare:

```
      readln(n);
  for i:=1 to n do  { readln(x[i]); sel[poz]:=false;}
  k:=0;
```

```
for i:=1 to n do
{ minim=2000000;
  for j:=1 to n do
   if(sel[j]=false)
   if(minim>x[j]){minim:=x[j];poz:=j;}

        k:=k+1;
  index[k]:=poz;
  sel[poz]:=true;}

for i:=1 to n do
     write(index[i]);
```

Observație: Dacă vectorul X este de la început o permutare, atunci dacă se repetă de 2 ori construcția vectorului index, se revine la permutarea inițială. Cu alte cuvinte, operația de determinare a vectorului de indexare are proprietatea de convoluție în spațiul permutărilor de lungime N.

Problema 6. Oraş Se consideră un oraş în care străzile formează un caroiaj alcătuit din celule pătrate de latură egală cu unitatea. Centrul oraşului are coordonatele (0, 0), celelalte intersecţii având coordonate de forma (I, J) cu I, J numere întregi. O persoană plecă din centrul oraşului şi parcurge o secvenţa de străzi şi intersecţii. Drumul persoanei este codificat printr-un şir de lungime L de direcţii cu elemente din mulţimea {E, S, V, N}. Să se determine:

a)Numărul de intersecţii distincte pe care le parcurge (inclusiv cea de coordonate (0, 0)).

b)Lista intersecţiilor în care intră de cel puţin două ori.

c)Una din intersecţiile cele mai întâlnite în drumul său.

Exemplu:

L=16 NEENVSSESENVVNEE

Ieşire:

Numărul de intersecţii: 12

Intersecţiile parcurse de cel puţin două ori:

1 1

1 0

2 0

2 1

Cea mai întâlnită:

1 1 de 3 ori

Restricţii: 1<=L<=100

(O.L.I. Prahova 2005 Clasa VII Radu Vişinescu)

Rezolvare: Se folosesc trei vectori paraleli X, Y, NR cu următoarea semnificaţie: X[J], Y[J]: coordonatele unei intersecţii vizitate, NR[J] : numărul de vizitări respective. Vectorii X, Y, NR se construiesc treptat în funcţie de valorile datelor de intrare.

Ordinul de complexitate al algoritmului este dat de instrucţiunile din ciclul for interior. Deoarece avem relaţia K<=N rezultă ordinul: $O(N^2)$. În pseudocod:

```
readln (L);
k:=0; k:=k+1;
x[k]:=0; y[k]:=0;
nr[k]:=1;
p:=0;q:=0;
for j:=1 to L do
 {read (d);
 if (d='E') then p:=p+1
else if (d='S') then q:=q-1
else if (d='V') then p:=p-1
else if (d='N') then q:=q+1;
 ok:=false; poz:=0;
 for h:=1 to k do
  if (x[h]=p) and (y[h]=q)then {ok:=true; poz:=h;}
 if (ok=false) then
  {k:=k+1; x[k]:=p; y[k]:=q; nr[k]:=1}
  else nr[poz]:=nr[poz]+1;
 }
writeln (k);
for j:=1 to k do
 if (nr[j]>=2)
 then writeln (x[j], y[j]);
max:=-1;
```

```
for j:=1 to k do
 if (max<nr[j]) then
  {max:=nr[j]; poz:=j;}
writeln(x[poz],y[poz]);
```

Problema 7. Mutari: Considerăm o tablă de joc de forma dreptunghiulară având N linii şi M coloane. Piesele jocului se mişcă începând de la o poziţie iniţială aflată pe prima linie până la o poziţie finală aflată pe ultima linie a tablei. Este dată o listă de mutări posibile sub forma unor cvadruple: X1, Y1, X2, Y2 cu semnificaţia că piesa situată în punctul de coordonate (X1, Y1) se poate muta în punctul de coordonate (X2, Y2); în plus de fiecare dată avem relaţia: X1<X2 (piesele avansează spre ultima linie a tablei). Să se determine numărul secvenţelor posibile de mutări pe care le poate efectua o piesă plecând de le un punct situat pe prima linie şi ajungând la un punct situat pe ultima linie a tablei.

Date de intrare: Fişierul de intrare mutari.in are următoarea structură:

- pe prima linie se află numerele N, M, K care indică numărul de linii ale tablei, numărul de coloane ale tablei, respectiv numărul de elemente ale listei de mutări; cele trei numere sunt separate prin câte un spaţiu.
- pe următoarele K linii se află cvadruple de forma X1, Y1, X2, Y2 care indică o mutare posibilă; cele patru numere sunt separate prin spaţii.

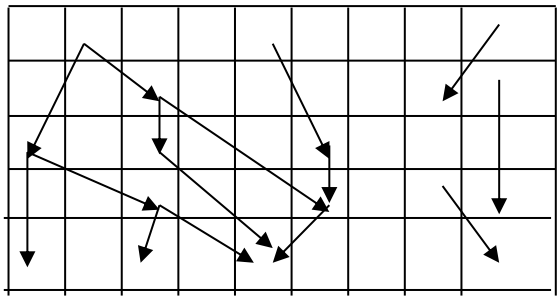
Date de ieşire: Fişierul de ieşire mutari.out trebuie să aibă următoarea structură:

- o singură linie pe care se va afla numărul NR, indicând numărul secvenţelor posibile de mutări.

Exemplu:

mutari.in

```
5 9 15
1 2 3 1
1 2 2 3
2 3 3 3
3 1 5 1
3 1 4 3
4 3 5 3
2 3 4 6
```



3 3 5 5

4 3 5 5

1 5 3 6

3 6 4 6

4 6 5 5

1 9 2 8

2 9 4 9

4 8 5 9

mutari.out

6

Restricţii si precizări:

1<=N, M<=200 1<=K<=1000

(O.L.I. Prahova 2005 clasa X-a Radu Vișinescu)

Rezolvare: Problema poate fi considerată o generalizare a problemei determinării numărului de modalitaţi de a urca o scară cu N trepte, la fiecare pas urcându-se una sau două trepte. Folosim vectorii XP, YP, XF, YF pentru memorarea extremităţilor arcelor (aceste arce reprezintă mutările posibile) şi o matrice auxiliară A cu N linii şi M coloane. Prezentăm rezolvarea în pseudocod:

```
//Se sorteaza vectorii XP,YP,XF,YF crescator dupa valorile din vectorul XP
//Se initializeaza cu 0 matricea A.
    for J:=1 to K do
        if (XP[J]=1) then A[1,YP[J]]:=1;
    for J:=1 to K do
        A[XF[J],YF[J]]:=A[XF[J],YF[J]]+A[XP[J],YP[J]];
    S:=0;
    for J:=1 to M do S:=S+A[N,J];
    write('nr=',s);
```

Algoritmul are ordinul de complexitate O(K*log(K) + N*M}). Datorită existenţei relaţiilor de recurenţă şi a rezolvării lor în modul bottom-up putem spune că problema se încadrează în tehnica programării dinamice.

Alte probleme

Problema 9: Fie un vector X de dimensiune N de numere întregi. Numim platou o secvenţă de valori consecutive şi egale din vector. Determinaţi un platou de lungime maximă. Se cere un algoritm cu ordin de complexitate O(N).

Exemplu: N=10 X=(2, 3, 3, 3, 4, 4, 4, 4, 1,1). Se vor afişa P=4, U=8, VAL=4.

Problema 10: În cadrul expresiilor aritmetice pot apare o serie de paranteze, incluse eventual unele în altele. Codificăm paranteza deschisă cu 0 şi paranteza închisă cu 1. Astfel, şirul de paranteze: (( )) ( ) se codifică prin şirul de valori: 0 0 1 1 0 1. Citindu-se un astfel de şir X de lungime N de valori să se determine dacă corespunde sau nu unui şir de paranteze ce se închid corect. **(O.L.I. Prahova 2003 Clasa VI Radu Vișinescu).**

Problema 12. Construiți un vector X care sa conțină în ordine primele N numere prime. Dacă se citește N=5 vectorul va fi: X=(2, 3, 5, 7, 11).

Problema 13. Se citește N număr natural. Tipariți primele N perechi de numere prime de forma K, K+2 câte o pereche pe cate o linie a ecranului. Dacă se citește N=3 se va tipari:

3 5

5 7

9 11

Indicație: Citim pe N și inițializăm o variabilă K cu valoarea 0. Variabila K va contoriza numărul de perechi de numere prime găsite. Se parcurge cu variabila I șirul numerelor impare începând cu valoarea 3. Dacă numerele I și I+2 sunt simultan numere prime atunci incrementăm cu o unitate variabila K și scriem valorile I, I+2 pe o linie pe ecran. După acest pas mărim pe I cu 2 unitați și repetam procedeul pentru noua valoare a lui I. Procesul se oprește când am găsit exact N perechi. Putem lucra și cu un ciclu while unde condiția de repetiție este K<N.

Problema 14. Se consideră şirul numerelor naturale dispuse pe linii şi coloane după cum urmează:

1 2 4 7 11

3 5 8 12

6 9 13

10 ...

,,,

Determinaţi primul număr prim situat pe linia L sau pe coloana C.

Date de intrare: Dacă se doreşte numărul prim de pe linie se va tasta cifra 1 urmată de numărul de linie, dacă se doreşte numărul prim de pe o coloană se va tasta cifra 2 urmată de numărul coloanei.

Exemplu:

Dacă se tastează numerele:

1 3

Răspuns : 13

(O.L.I. Prahova 2006 clasa V Radu Vișinescu)

Indicație: Citim valoarea 1 sau 2 într-o variabila nr. Daca nr=1, adică căutăm primul numar prim de pe o linie, se va citi in variabila L numarul linie. Primul element de pe linia L va fi exact 1+2+...+L. Mai observăm că diferențele între două valori consecutive pe linia L sunt de forma: L, L+1,L+2 etc. Parcurgem aceste valori și reținem și tipărim prima valoare prima gasită. Dacă de la început am citit nr=2 căutăm primul numar prim pe o coloană. Citim numărul coloanei în variabila C. Observăm ca valoarea de început de pe coloana C este egal cu suma 1+(1+2+...+(C-1)). Diferențele între două valori succesive pe coloană vor fi: C+1,C+2, C+3 etc. Parcurgem valorile de pe coloană și reținem prima valoare numar prim.

Problema 14: Poarta Se consideră harta universului ca fiind o matrice cu 250 de linii şi 250 de coloane. În fiecare celulă se găseşte o aşa numită poartă stelară, iar în anumite celule se găsesc echipaje ale porţii stelare. La o deplasare, un echipaj se poate deplasa din locul în care se află în oricare alt loc în care se găseşte o a doua poartă, în cazul nostru în orice altă poziţie din matrice. Nu se permite situarea simultană a mai mult de un echipaj într-o celulă. La un moment dat un singur echipaj se poate deplasa de la o poartă stelară la alta.

Dându-se un număr p ($1<p<5000$) de echipaje, pentru fiecare echipaj fiind precizate poziţia iniţială şi poziţia finală, determinaţi numărul minim de deplasări necesare pentru ca toate echipajele să ajungă din poziţia iniţială în cea finală.

Datele de intrare

Se citesc din fisierul text **poarta.in** în următorul format:

– pe prima linie numărul natural p reprezentând numărul echipaje,

– pe următoarele *p* linii câte 4 numere naturale, primele două reprezentând coordonatele poziţiei iniţiale a unui echipaj (linie coloană), următoarele două reprezentând coordonatele poziţiei finale a aceluiaşi echipaj (linie coloană).

Datele de ieşire

Pe prima linie a fişierului text **poarta.out** se scrie un singur număr reprezentând numărul minim de deplasări necesar.

Exemplu:

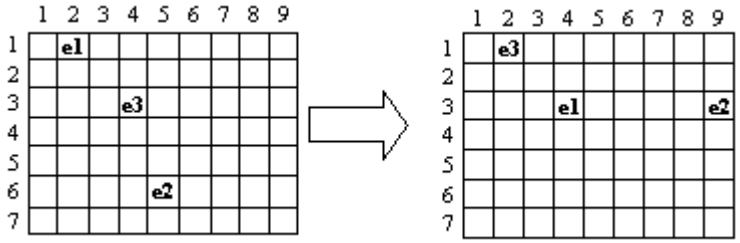
poarta.in

3

1 2 3 4

6 5 3 9

3 4 1 2



poarta.out

4

Observaţii:

– coordonatele poziţiilor iniţiale şi finale ale echipajelor sunt numere naturale din intervalul [1, 250]

– poziţiile iniţiale ale celor *p* echipaje sunt distincte două cîte două;

– poziţiile finale ale celor *p* echipaje sunt distincte două câte două.

Timp maxim de executare: 1 sec/test

(O.J.I. 2002 clasa IX Radu Vișinescu)