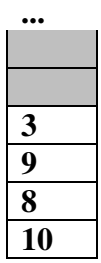
Stiva

Stiva este un tip particular de listă, pentru care atât operația de inserare a unui element în structură, cât şi operația de extragere a unui element se realizează la un singur capăt, denumit vârful stivei.

Singurul element din stivă la care avem acces direct este cel de la vârf.

Operațiile elementare caracteristice stivei sunt:

- crearea unei stive vide;
- inserarea unui element în stivă (operație denumită în literatura de specialitate **PUSH**)
- extragerea unui element din stivă (operație denumită în termeni de specialitate **POP)**
- accesarea elementului de la vârf (operație denumită **TOP**).



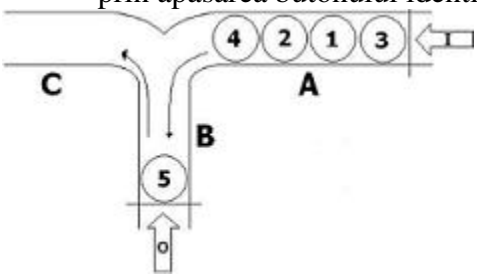
Ca să ne imaginăm mai bine funcționarea unei stive, să ne gândim cum lucrăm cu un teanc de farfurii. Când dorim să punem o farfurie în teanc, o punem deasupra, când dorim să luăm o farfurie din teanc, o luăm tot pe cea de deasupra.

Acest mod de funcționare face ca ultimul element inserat în stivă să fie primul extras. Din acest motiv, *stiva* este definită şi ca o structură de date care funcționează după principiul **LIFO** (**L**ast **I**n **F**irst **O**ut – Ultimul Intrat Primul Ieşit).

bile3

Marele savant Lotocus a inventat dispozitivul din imaginea de mai jos, care functioneaza astfel:

- initial cele n bile, numerotate de la 1 la n, sunt asezate în zona A, într-o ordine oarecare;
- prin apasarea butonului identificat prin litera I prima bila din zona A cade în zona B;
- prin apasarea butonului identificat prin litera O prima bila din zona B urca în zona C.



Cerinta

Deoarece onorabilul savant spera sa-si vânda inventia unei loterii, ajutati-l scriind un program care sa indice ordinea în care trebuie apasate cele doua butoane astfel încât plecând de la o configuratie initiala a bilelor în zona A sa se obtina o anumita configuratie a lor în zona C (ceea ce va dovedi importanta extraordinara a masinariei!).

Date de intrare

Fisierul de intrare bile.in are urmatoarea structura:

- pe prima linie valoarea numarului natural nenul n;
- pe a doua linie n numere naturale nenule, separate prin spatii, reprezentând configuratia initiala a bilelor în zona A, specificate in ordine de la stanga la dreapta;

- pe a treia linie n numere naturale nenule, separate prin spatii, reprezentând configuratia finala la care trebuie sa ajunga bilele în zona C, specificate in ordine de la stanga la dreapta.

Date de iesire

Prima linie a fisierului bile.out va contine o singura linie pe care va fi scris un sir format numai din literele I si O reprezentând ordinea în care trebuie apasate cele doua butoane astfel încât plecând de la configuratia initiala sa se obtina configuratia finala sau mesajul imposibil daca nu se poate realiza acest lucru.

Restrictii

 1 <= n <= 2000

Exemple

bile.in	bile.out	bile.in	bile.out
5 5 4 2 1 3 2 4 1 5 3	IIIOOIOOIO	5 5 4 2 1 3 1 4 2 5 3	imposibil

Idee de rezolvare

Se observa usor ca bilele din zonele A si B sunt organizate sub forma unor stive, notate in program cu sta (cu vârful vfa) si stb (cu vârful vfb).

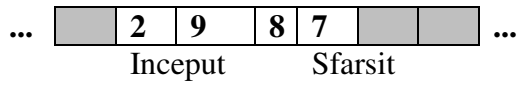
Pentru a rezolva problema parcurgem bilele din configuratia finala, memorata în vectorul out, si încercam sa obtinem la iesire (în zona C) bila curenta, fie din zona A, fie din zona B astfel:

- daca bila curenta este prima din zona B (în vârful stivei stb) atunci o extragem prin apasarea butonului O;
- daca nu, împingem pe rând câte o bila din zona A în zona B folosind butonul I fie pâna când bila curenta din configuratia finala este prima din zona A (în vârful stivei sta), caz în care o extragem apasând mai întâi butonul I si apoi butonul O, fie pâna când nu mai avem nici o bila în zona A, caz în care problema nu are solutie deoarece înseamna ca bila curenta din configuratia finala nu se poate obtine.

Coada

Coada este un tip particular de listă, pentru care operația de inserare a unui element se realizează la un capăt, în timp ce operația de extragere a unui element se realizează la celălalt capăt.

Singurul element din coadă la care avem acces direct este cel de la început.



Operațiile elementare care caracterizează structura de date de tip coadă sunt:

- crearea unei cozi vide;
- inserarea unui element în coadă;
- extragerea unui element din coadă;
- accesarea unui element.

Executarea acestor operații asupra unei cozi presupune cunoașterea începutului cozii (să-l notăm Inceput) şi a sfârşitului acesteia (să-l notăm Sfarsit).

Modul de funcționare a unei cozi este foarte uşor de intuit: toată lumea a „stat la coadă”, măcar o dată. Orice situație în care sunt mai multe cereri de acces la o resursă unică (de exemplu, mai mulți clienți şi o singură vânzătoare; o singură pompă de benzină şi mai multe maşini, un singur pod şi mai multe capre, etc) necesită formarea unei „linii de aşteptare”. Dacă nu apar alte priorități, cererile sunt satisfăcute în ordinea sosirii.

Datorită faptului că întotdeauna este extras („servit”) primul element din coadă, iar inserarea oricărui nou element se face la sfârşit („la coadă”), coada este definită ca o structură de date care funcționează după principiul **FIFO** (**F**irst **I**n **F**irst **O**ut – Primul Intrat Primul Ieşit).

Caroiaj

Se consideră un caroiaj dreptunghiular cu m linii şi n coloane, în care pe anumite poziții sunt plasate obstacole. În poziția inițială (x_0,y_0) se află plasat un mobil. Să se determine, pentru toate pozițiile în care mobilul poate ajunge, distanța minimă de la poziția inițială a mobilului, măsurată în deplasări elementare. Prin deplasare elementară se înțelege deplasarea mobilului cu o poziție stânga, dreapta, sus sau jos.

Soluție

Reprezentăm caroiajul ca o matrice A, în care marcăm obstacolele cu -1, iar pozițiile libere cu -2. La sfârşitul algoritmului în matricea A vom avea:

A[i][j]= -1, dacă poziția (i,j) este obstacol;

A[i][j]= -2, dacă poziția (i,j) nu este accesibilă din (x_0,y_0);

A[i][j]= distanța minimă de la (x_0,y_0) la (i,j), altfel.

Pentru a nu testa la fiecare pas dacă nu am ajuns la o margine a caroiajului, vom borda caroiajul cu obstacole (linia şi coloana 0, respectiv linia n+1 şi coloana m+1 le vom utiliza ca bordură).

Ideea constă în a calcula distanța minimă până la pozițiile în care se poate ajunge dintr-o singură mutare, apoi la pozițiile la care se poate ajunge din două mutări, etc. Pentru a reține pozițiile în ordinea distanței lor față de poziția inițială, vom folosi o coadă C, în care vom reține pentru fiecare poziție atinsă coordonatele şi distanța minimă de la poziția inițială.

```
#include <fstream.h>
#define DimMax 20
#define DimMaxCoada 400
int dx[4]={-1, 0, 1, 0}, dy[4]={0, 1, 0, -1};
 struct Element
  {int l,c;    //pozitia in caroiaj
   unsigned d; //distanta minima pana la pozitia l,c
  } ;
Element C[DimMaxCoada], x, y;
int A[DimMax][DimMax], n, m, x0, y0, i, j, k, IncC, SfC;
void main()
{ifstream fin("careu.in");
 fin >> n >> m;
fin >> x0 >> y0;
for (i=1;i<=n;i++)
    for (j=1;j<=m;j++) A[i][j] = -2;
```

```
while (fin)
      {fin >> i >> j;   //citesc coordonatele obstacolelor
       A[i][j] = -1; }  //marchez obstacolele cu -1
fin.close();

//BORDARE
for (i=1;i<=n;i++) A[i][0]=A[i][m+1]=-1;
for (i=1;i<=m;i++) A[0][i]=A[n+1][i]=-1;

//INITIALIZARE COADA
x.l=x0; x.c=y0; x.d=0; A[x0][y0]=0; C[IncC]=x;


//algoritmul lui LEE

while (IncC <= SfC)                 //parcurg COADA
      {
       x = C[IncC++];       //EXTRAG UN ELEMENT DIN COADA
      //NE DEPLASAM IN CELE 4 DIRECTII

       for (k=0; k<4; k++)
          { y.l = x.l + dx[k];  y.c = x.c + dy[k];
            if (A[y.l][y.c] == -2)        //y- POZITIE LIBERA
             {
               y.d = x.d + 1; A[y.l][y.c] = y.d;
               C[++SfC]= y; //DUC y IN COADA
             }
          }
      }

//AFISARE SOLUTIE
ofstream fout("careu.out");
for (i=1;i<=n;i++)
   { for (j=1;j<=m;j++)
                fout<<A[i][j]<<' ';
     fout<<'\n';
   }
fout.close();
}
```

De exemplu, pentru fişierul de intrare:

```
  5 5                                   //dimensiunile caroiajului
  3 3                                   //pozitia initiala
  1 2                                   //obstacole
  1 3
  2 4
  4 3
```

4 4
2 1

obținem soluția:

-2 -1 -1 5 4
-1 2 1 -1 3
2 1 0 1 2
3 2 -1 -1 3
4 3 4 5 4

Propleme propuse:
http://campion.edu.ro/arhiva/index.php?page=problem&action=view&id=831

Alee- arhiva campion

Parcul oraşului a fost neglijat mult timp, astfel că acum toate aleile sunt distruse. Prin urmare, anul acesta Primăria şi-a propus să facă reamenajări. Parcul are forma unui pătrat cu latura de n metri şi este înconjurat de un gard care are exact două porți. Proiectanții de la Primărie au realizat o hartă a parcului şi au trasat pe hartă un caroiaj care împarte parcul în nxn zone pătrate cu latura de 1 metru. Astfel harta parcului are aspectul unei matrice pătratice cu n linii şi n coloane. Liniile şi respectiv coloanele sunt numerotate de la 1 la n. Elementele matricei corespund zonelor pătrate de latură 1 metru. O astfel de zonă poate să conțină un copac sau este liberă. Edilii oraşului doresc să paveze cu un număr minim de dale pătrate cu latura de 1 metru zonele libere (fără copaci) ale parcului, astfel încât să se obțină o alee continuă de la o poartă la alta.

Cerință

Scrieți un program care să determine numărul minim de dale necesare pentru construirea unei alei continue de la o poartă la cealaltă.

Date de intrare

Fişierul de intrare alee.in conține pe prima linie două valori naturale n şi m separate printr-un spațiu, reprezentând dimensiunea parcului, respectiv numărul de copaci care se găsesc în parc. Fiecare dintre următoarele m linii conține câte două numere naturale x şi y separate printr-un spațiu, reprezentând pozițiile copacilor în parc (x reprezintă linia, iar y reprezintă coloana zonei în care se află copacul). Ultima linie a fişierului conține patru numere naturale x1 y1 x2 y2, separate prin câte un spațiu, reprezentând pozițiile celor două porți (x1, y1 reprezintă linia şi respectiv coloana zonei ce conține prima poartă, iar x2, y2 reprezintă linia şi respectiv coloana zonei ce conține cea de a doua poartă).

Date de ieşire

Fişierul de ieşire alee.out va conține o singură linie pe care va fi scris un număr natural care reprezintă numărul minim de dale necesare pentru construirea aleii.

Restricții

$1 \leq n \leq 175$
$1 \leq m < n*n$

Aleea este continuă dacă oricare două plăci consecutive au o latură comună.Aleea începe cu zona unde se găseşte prima poartă şi se termină cu zona unde se găseşte cea de a doua poartă. Pozițiile porților sunt distincte şi corespund unor zone libere.Pentru datele de test există întotdeauna

soluție.

Exemple

alee.in	alee.out	Explicații
8 6 2 7 3 3 4 6 5 4 7 3 7 5 1 1 8 8	15	O modalitate de a construi aleea cu număr minim de dale este: OOO----- --OO--x- --xO---- ---OOx-- ---xO--- ----OO-- --x-xOO- ------OO (cu X am marcat copacii, cu - zonele libere, iar cu O dalele aleii).

2. RJ- arhiva campion
http://campion.edu.ro/arhiva/index.php?page=problem&action=view&id=898

În ultima ecranizare a celebrei piese shakespeariene Romeo şi Julieta trăiesc într-un oraş modern, comunică prin e-mail şi chiar învaţă să programeze. Într-o secvenţă tulburătoare sunt prezentate frământările interioare ale celor doi eroi încercând fără succes să scrie un program care să determine un punct optim de întâlnire.

Ei au analizat harta oraşului şi au reprezentat-o sub forma unei matrice cu n linii şi m coloane, în matrice fiind marcate cu spaţiu zonele prin care se poate trece (străzi lipsite de pericole) şi cu X zonele prin care nu se poate trece. De asemenea, în matrice au marcat cu R locul în care se află locuinţa lui Romeo, iar cu J locul în care se află locuinţa Julietei.

Ei se pot deplasa numai prin zonele care sunt marcate cu spaţiu, din poziţia curentă în oricare dintre cele 8 poziţii învecinate (pe orizontală, verticală sau diagonale).

Cum lui Romeo nu îi place să aştepte şi nici să se lase aşteptat n-ar fi tocmai bine, ei au hotărât că trebuie să aleagă un punct de întâlnire în care atât Romeo, cât şi Julieta să poată ajunge în acelaşi timp, plecând de acasă. Fiindcă la întâlniri amândoi vin într-un suflet, ei estimează timpul necesar pentru a ajunge la întâlnire prin numărul de elemente din matrice care constituie drumul cel mai scurt de acasă până la punctul de întâlnire. Şi cum probabil există mai multe puncte de întâlnire posibile, ei vor să îl aleagă pe cel în care timpul necesar pentru a ajunge la punctul de întâlnire este minim.

Cerinţă

Scrieţi un program care să determine o poziţie pe hartă la care Romeo şi Julieta pot să ajungă în acelaşi timp. Dacă există mai multe soluţii, programul trebuie să determine o soluţie pentru care timpul este minim.

Date de intrare

Fişierul de intrare rj.in conține:

- pe prima linie numerele naturale N M, care reprezintă numărul de linii şi respectiv de coloane ale matricei, separate prin spațiu;
- pe fiecare dintre următoarele N linii se află M caractere (care pot fi doar R, J, X sau spațiu) reprezentând matricea.

Date de ieşire

Fişierul de ieşire rj.out va conține o singură linie pe care sunt scrise trei numere naturale separate prin câte un spațiu tmin x y, având semnificația:

- x y reprezintă punctul de întâlnire (x – numărul liniei, y – numărul coloanei);
- tmin este timpul minim în care Romeo (respectiv Julieta) ajunge la punctul de întâlnire.

Restricții şi precizări

1 < N, M < 101

Liniile şi coloanele matricei sunt numerotate începând cu 1.

Pentru datele de test există întotdeauna soluție.

Exemple

rj.in	rj.out	*Explicație*
5 8 XXR XXX X X X J X X X XX XXX XXXX	4 4 4	Traseul lui Romeo poate fi: (1,3), (2,4), (3,4), (4,4) Deci timpul necesar lui Romeo pentru a ajunge de acasă la punctul de întâlnire este 4. Traseul Julietei poate fi: (3,1), (4,2), (4,3), (4,5). Timpul necesar Julietei pentru a ajunge de acasă la punctul de întâlnire este de asemenea 4. În plus 4, este punctul cel mai apropiat de ei cu această proprietate.

Bibliografie

E. Cerchez, M. Serban – „Programarea în limbajul C/C++”. Volumul I; Editura Polirom