**Backtracking în plan**

**Fie următoarea problema: un șoricel se găsește într-un labirint de forma dreptunghiulară , pe care îl vom asimila cu o matrice cu m linii și n coloane. Pereții sunt marcați cu 1 si culoarele cu 0. Se cunosc coordonatele inițiale ale șoricelului: Li, Ci. Să se determine toate posibilitățile pe care le are șoricelul pentru a ieși din labirint. Șoricelul poate avansa pe 4 direcții cate o celulă (sus, dreapta , jos, stânga).**

O astfel de problemă presupune o abordare Backtracking în plan. Traseul șoricelului va fi reținut de un vector cu două câmpuri: coordonatele x si y ale deplasării sale în labirint. Vom defini un tip struct:

**struct pozitie**

**{int x,y;**

**};**

Vectorul d retine drumul parcurs de șoarece : **poziție d[50];**

Pentru generarea unui drum vom defini un subprogram recursiv **void ies(int x,int y)**

care primește ca parametri coordonatele unei componente din matrice. Inițial se apelează

pentru coordonatele inițiale ale șoricelului. O componenta din matrice va putea aparține

drumului daca evident este culoar (a[x][y]=0). O celulă a matricii determinată ca aparținând

drumului se marchează cu 2 (pentru a preveni generarea unor soluții de mai multe ori):

a[x][y]=2;

Se încarcă valoarea corespunzătoare in vectorul d pe nivelul curent:

d[k].x=x;

d[k].y=y;

De fiecare dată când s-a determinat o nouă celulă ca aparținând drumului se determină

dacă s-a ajuns la soluție (condiție care diferă de la o problemă la alta).

În cazul problemei date vom considera că se iese din labirint când se ajunge la (linia 0 , coloana 0)

sau (linia m+1 , coloana n+1). Testul este: if((x<1)||(x>m)||(y<1)||(y>n))

tipar(k);

În caz afirmativ se tipărește (se afișează vectorul drum **d** și/sau matricea labirint a) altfel (dacă

soluția este incompletă) se încearcă parcurgerea, pe rând, a celor 4 celule alăturate. Acest lucru

se realizează prin autoapelul funcției pe cele patru direcții:

ies(x-1,y);

ies(x,y+1);

ies(x+1,y);

ies(x,y-1);

**Observație:** vectorul d este de fapt generat într-o stivă cu două câmpuri.

La revenire din apel se eliberează celula pentru a o putea accesa și în cadrul generării altor

soluții: a[x][y]=0 și se eliberează componenta drumului k=k-1 (practic se coboară în

stivă).

**EXEMPLU**

**Fie următorul labirint: m=6 n=10 Li=4, Ci=3**

**1 1 1 1 0 1 1 1 1 1**

**1 1 1 1 0 1 1 1 1 1**

**1 1 1 1 0 1 1 1 1 1**

**1 1 0 0 0 0 0 0 0 0**

**1 1 1 1 1 1 1 0 1 1**

**1 1 1 1 1 1 1 0 1 1**

**Soluțiile vor fi:**

**soluția 1**

**(4,3) (4,4) (4,5) (3,5) (2,5) (1,5) (0,5)**

**1 1 1 1 2 1 1 1 1 1**

**1 1 1 1 2 1 1 1 1 1**

**1 1 1 1 2 1 1 1 1 1**

**1 1 2 2 2 0 0 0 0 0**

**1 1 1 1 1 1 1 0 1 1**

**1 1 1 1 1 1 1 0 1 1**

**soluția 2**

**(4,3) (4,4) (4,5) (4,6) (4,7) (4,8) (4,9) (4,10) (4,11)**

**1 1 1 1 0 1 1 1 1 1**

**1 1 1 1 0 1 1 1 1 1**

**1 1 1 1 0 1 1 1 1 1**

**1 1 2 2 2 2 2 2 2 2**

**1 1 1 1 1 1 1 0 1 1**

**1 1 1 1 1 1 1 0 1 1**

**soluția 3**

**(4,3) (4,4) (4,5) (4,6) (4,7) (4,8) (5,8) (6,8) (7,8)**

**1 1 1 1 0 1 1 1 1 1**

**1 1 1 1 0 1 1 1 1 1**

**1 1 1 1 0 1 1 1 1 1**

**1 1 2 2 2 2 2 2 0 0**

**1 1 1 1 1 1 1 2 1 1**

**1 1 1 1 1 1 1 2 1 1**

**Programul complet este:**

#include<fstream>

using namespace std;

ofstream fout("labirint.out");

struct pozitie

{int x,y;};

int a[20][20];// matricea labirint

int k,n,m,Li,Ci,nr\_sol;

pozitie d[50];

void afis\_mat()

{fout<<'\n';

for(int i=1;i<=m;i++)

{for(int j=1;j<=n;j++)

fout<<a[i][j]<<" ";

fout<<'\n';}

}

void tipar(int k) //tipareste vectorul drum

{nr\_sol++;

fout<<"solutia "<<nr\_sol<<'\n';

for(int i=1;i<=k;i++)

fout<<"("<<d[i].x<<','<<d[i].y<<") ";

afis\_mat();

fout<<'\n';

}

void ies(int x,int y) //genereaza drumul

{if(a[x][y]==0)

{k++;

a[x][y]=2;

d[k].x=x;

d[k].y=y;

if((x<1)||(x>m)||(y<1)||(y>n))

tipar(k);

else

{

ies(x-1,y);

ies(x,y+1);

ies(x+1,y);

ies(x,y-1);

}

a[x][y]=0; //la revenire din apel demarchez celula pentru a o putea

//accesa si in cadrul altei prelucrari

k--;//eliberez componenta din vectorul drumului

}

}

int main()

{

ifstream fin("labirint.in");

fin>>m>>n;

for(int i=1;i<=m;i++)

for(int j=1;j<=n;j++)

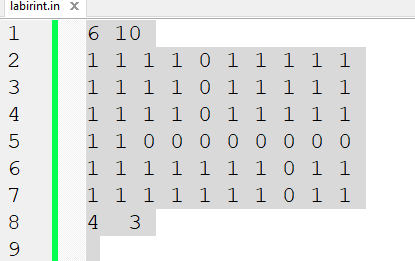
fin>>a[i][j];

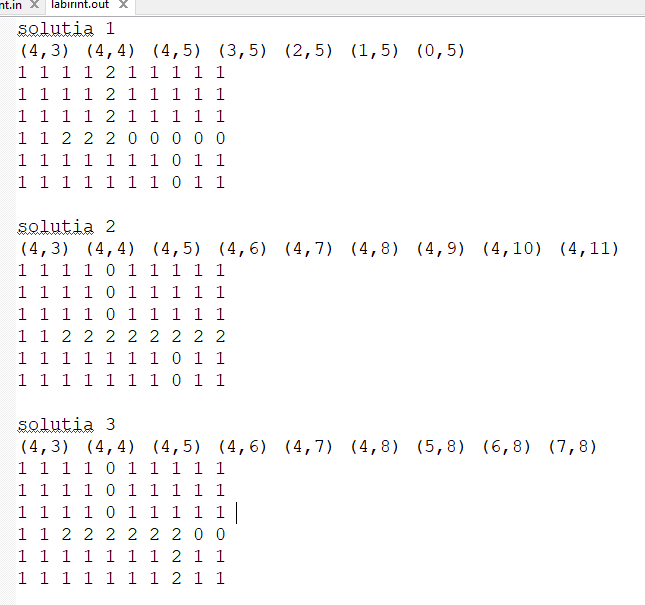
fin>>Li>>Ci; //coordonatele punctului in care se afla inițial șoricelul

ies(Li,Ci);

}

Pentru ex . dat mai sus fișierul de intrare trebuie sa aibă acest conținut:



conținutul fișierului de ieșire este dat mai jos: