Crețu Laurențiu Vasile
CTI, Grupa 161

Tema 3 - Backtracking

Problema 10

Enunț:

Garfield a aterizat într-un labirint de forma unei matrice cu M linii și N coloane în care zidurile sunt codificate cu 1, iar culoarele cu 0. El se află inițial în puctul L0, C0 și dorește să ajungă în punctul L1, C1 unde se află o cutie cu... lasagna. Care sunt posibilitățile sale știind că zona dreptunghiulară cuprinsă între liniile LZ1 și LZ2 și coloanele CZ1 și CZ2 este controlată de Luca, inamicul său canin, și deci trebuie evitată.

Date de intrare: fișierul date.in conține:

- pe prima linie numerele M și N, în această ordine, cu semnificația din enunț.
- pe următoarele M linii, matricea care memorează configurația labirintului.
- pe penultima linie numerele LZ1, LZ2, CZ1, CZ2, în această ordine, cu semnificația din enunț
- pe ultima linie numerele naturale l0, c0, l1,c1, în această ordine, cu semnificația din enunț

Date de ieșire: fișierul date.out conține soluțiile, separate prin câte o linie goală; o soluție este o matrice în care se codifica cu 0 culoarele prin care nu s-a trecut, cu -1 zidurile și se marchează traseul cu numarul de ordine al pașilor urmați.

Restricții și precizări:

- $N,M \in [2,102]$
- $1 \leq l0,l1 \leq M$
- $1 \leq c0,c1 \leq N$
- $1 \leq LZ1 < CZ2 \leq N$

Exemplu:

date.in	date.out
8 7 1 1 0 1 1 0 1 1 0 0 0 0 0 1 1 1 0 1 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 0 1 1 0 0 0 0 0 1 1 3 4 5 1 1 6 5	-1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 7 -1 -1 -1 0 0 0 8 -1 -1 -1 0 -1 -1 9 10 -1 -1 0 0 0 0 0 -1 -1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 7 -1 -1 -1 0 0 0 8 -1 -1 -1 0 -1 -1 9 12 -1 -1 0 0 0 10 11 -1

	-1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 7 -1 -1 -1 11 10 9 8 -1 -1 -1 12 -1 -1 17 18 -1 -1 13 14 15 16 0 -1 -1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 7 -1 -1 -1 11 10 9 8 -1 -1 -1 12 -1 -1 0 18 -1 -1 13 14 15 16 17 -1 -1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 0 -1 -1 -1 0 0 7 8 -1 -1 -1 0 -1 -1 9 10 -1 -1 0 0 0 0 0 -1 -1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 0 -1 -1 -1 0 0 7 8 -1 -1 -1 0 -1 -1 9 12 -1 -1 0 0 0 10 11 -1 -1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 0 -1 -1 -1 9 8 7 0 -1 -1 -1 10 -1 -1 15 16 -1 -1 11 12 13 14 0 -1 -1 -1 0 -1 -1 0 -1 -1 1 2 0 0 0 -1 -1 -1 3 -1 0 0 -1 -1 -1 4 5 0 0 -1 -1 -1 -1 6 0 -1 -1 -1 9 8 7 0 -1 -1 -1 10 -1 -1 0 16 -1 -1 11 12 13 14 15 -1

Metoda de rezolvare:

Această problemă reprezintă o problemă clasică de Backtracking, și anume *Problema labirintului.* Ideea acestei probleme este de a porni din punctul de coordonate l0 și c0, apoi parcurgem vecinii elementului curent pe direcțiile N(-1,0), E(0,1), S(1,0) și V(0,-1) până când ajungem în punctul de coordonate l1 și c1. La fiecare pas verificăm dacă vecinul de coordonate i și j este sau nu valid și anume, un vecin nu este valid când nu se mai află în matrice (i<0 || i>=m || j<0 || j>=n), reprezintă un zid sau un vecin vizitat deja (matrice[i][j] != 0) și se află în zona controlată de Luca (i>=lz1 && i<=lz2 && j>=cz1 && j<=cz2), în cazul în care vecinul este valid marcăm zona ca și vizitată. În momentul în care am ajuns la destinație afișăm drumul parcurs. La întoarcerea din recursivitate setăm elementele matricii ce au fost vizitate cu valoarea 0. Algoritmul se încheie când toate posibilitățile de parcurgere ale drumului au fost epuizate.

Datele de intrare:

```
int n,m;
int **matrice;
int lz1,lz2,cz1,cz2;
int l0,c0,l1,c1;
int di[] = {-1,0,1,0};
int dj[] = {0,1,0,-1};
int solutie;
```

Citirea:

```
void citire(){
    FILE *fin = fopen("date.in","r");
    fscanf(fin,"%d %d",&m,&n);
    matrice = (int**)malloc(m*sizeof(int*));
    for (int i=0; i<m; i++)
        matrice[i] = (int*)malloc(n*sizeof(int));
    for (int i=0; i<m; i++)
        for (int j=0; j<n; j++)
            fscanf(fin,"%d",&matrice[i][j]);
    fscanf(fin,"%d %d %d %d",&lz1,&lz2,&cz1,&cz2);
    fscanf(fin,"%d %d %d %d",&l0,&c0,&l1,&c1);
    fclose(fin);

    FILE *fout = fopen("date.out","w");
```

```
    fclose(fout);
}
```

Algoritmul de Backtracking:

```
int valid(int i,int j){
    if (i<0 || i>=m || j<0 || j>=n)
        return 0;
    if (matrice[i][j] != 0)
        return 0;
    if (i>=lz1 && i<=lz2 && j>=cz1 && j<=cz2)
        return 0;
    return 1;
}

void bkt(int i_curent,int j_curent,int k){
    if (i_curent==l1 && j_curent==c1)
        afisare();
    else{
        for (int i=0; i<4; i++){
            int i_vecin = i_curent + di[i];
            int j_vecin = j_curent + dj[i];
            if (valid(i_vecin,j_vecin)==1){
                matrice[i_vecin][j_vecin] = k+1;
                bkt(i_vecin,j_vecin,k+1);
                matrice[i_vecin][j_vecin] = 0;
            }
        }
    }
}
```

Afișarea soluțiilor:

```
void afisare(){
    solutie++;
    FILE *fout = fopen("date.out","a");
    fprintf(fout,"Solutia numarul %d:\n",solutie);
    for (int i=0; i<m; i++){
        for (int j=0; j<n; j++){
            if (i==l0 && j==c0)
                fprintf(fout,"1 ");
            else
                if (matrice[i][j]==1)
                    fprintf(fout,"-1 ");
                else
                    fprintf(fout,"%d ",matrice[i][j]);
```

```
        }
        fprintf(fout,"\n");
    }
    fprintf(fout,"\n");
    fclose(fout);
}
```