Căutare cu adâncime iterativă

Pornind de la programul rezolvat în L006 (căutare limitată în adâncime), implementați în continuare o căutare cu adâncime iterativă. Păstrați în același program și celelalte căutări (lățime, adâncime, cost uniform, căutare limitată în adâncime) pentru a putea compara rutele găsite.

Instrucțiuni legate de căutarea cu adâncime iterativă:

1. Programul se bazează pe căutarea cu adâncime limitată. Este apelată o metodă cu adâncime limitată cu limita 1, 2, 3 etc până ori se ajunge la soluție, ori se golește complet lista de *noduri*.
2. Firește, această căutare nu trebuie să afișeze că nu are soluție pentru fiecare limită ce este prea mică.

Se afișează la final ce limită a fost necesară pentru a se găsi soluția.

Soluție

Iată una din soluțiile primite la care am făcut mici modificări.

Soluție

Observații: programul conține căutările in lățime, adâncime, cost uniform, căutare în adâncime limitată și noul program, căutarea cu adâncime iterativă. Ca de obicei, fiecare căutare se găsește într-o metodă separată.

```
#include <iostream>
#include <string>
#define N 20
#define nyooom 100
using namespace std;
string nume[N] = { "Arad",
        "Bucuresti",
        "Craiova",
        "Severin",
        "Eforie",
        "Fagaras",
        "Giurgiu",
        "Harsova",
        "Iasi",
        "Lugoj",
        "Mehadia",
        "Neamt",
        "Oradea",
        "Pitesti",
        "Vilcea",
```

```
        "Sibiu",
        "Timisoara",
        "Urziceni",
        "Vaslui",
        "Zerind" };
float a[N][N], m[N][N];

void cautareCostUniform(int start, int stop)
{
        int viz[N], noduri[N], parinte[N], speed;
        float t[N];
        int nrNoduri = 0;
        for (int i = 0; i < N; i++)
                viz[i] = 0;     //marcam orasele ca nevizitate
        for (int i = 0; i < N; i++)
                t[i] = 0;
        noduri[nrNoduri++] = start;    //punem orasul de pornire in lista
oraselor de vizitat
        viz[start] = 1;//marcam orasul de start ca vizitat
        bool gasit = false;    //cand avem stop pe prima pozitie in noduri
gasit devine adevarat
        int k = 1;      //contor pentru a afisa pasii din popularea lui
"noduri"
        bool afisPasi = false; //daca e fals nu mai afisam pasii
intermediari
        float cost[N]; //cost[oras] va contine distanta de la start la
oras in km
        cost[start] = 0;        //start e la 0 km de start
        while ((!gasit) && (nrNoduri > 0))
        {
                if (afisPasi)
                {
                        std::cout << "Pasul " << k++ << " ";
                        for (int i = 0; i < nrNoduri; i++)
                                std::cout << endl;
                }

                int curent = noduri[0];         //primul element din noduri
                for (int i = 0; i < nrNoduri - 1; i++)//eliminam elementul
dupa prima pozitie din noduri

                        noduri[i] = noduri[i + 1];
                nrNoduri--;
                if (curent == stop)
                        gasit = true;
                else
                        for (int i = 0; i < N; i++)
                                if (a[curent][i] != 0) //orasul i este
conectat de orasul curent

                {
                        float costNou = a[curent][i] +

                                cost[curent];   //calculam intai costul

                        if ((viz[i] == 0) || (costNou < cost[i]))
                        {
```

```
                                if (viz[i] == 1)        //daca a fost
vizitat anterior, ar trebui sa eliminam vechea aparitie a lui i din
noduri

                                {
                                        int j = 0;
                                        while ((j < nrNoduri) && (noduri[j]

                                                        !=
                                                        i))

                                                j++;    //j reprezinta
pozitia pe care se gaseste i in noduri

                                        if (j < nrNoduri)       //adica j e
inca in noduri(fiindca e posibil sa fi fost scos anterior)
                                        {
                                                for (int q = j; q <
nrNoduri - 1; q++)

                                                        noduri[q] = noduri[q
+ 1];   //am eliminat elementul dupa pozitia j din noduri

                                                nrNoduri--;
                                        }
                                }

                                int j = 0;

                                while ((j < nrNoduri) && (costNou >
cost[noduri[j]]))       //cautam in noduri pozitia pana la care costNou e
mai mare decat costul elementului de pe pozitia curenta j

                                        j++;    //j reprezinta pozitia pe
care adaugam orasul i

                                for (int q = nrNoduri; q > j; q--)     //le
deplasam pe toate de la j pana la capat la dreapta cu o pozitie

                                        noduri[q] = noduri[q - 1];
                                noduri[j] = i; //j reperzinta pozitia pe
care trebuie adaugat i

                                nrNoduri++;
                                cost[i] = costNou;
                                parinte[i] = curent;
                                viz[i] = 1;
                                if (((float) m[curent][i] / a[curent][i])
>= 10)

                                        speed = (float) 1000 / ((float)
m[curent][i] / a[curent][i]);
                                else
                                        speed = nyooom;
                                t[i] = t[parinte[i]] + a[curent][i] /
speed;
```

```
                        }       //de la if ((viz[i] == 0) || (costNou < cost[i]))
                }       //de la if (a[curent][i] != 0)

        }       //acolada este de la while
        int solutie[N], temp = stop, i = 0;
        while (temp != start)
        {
                solutie[i++] = temp;
                temp = parinte[temp];
        }

        solutie[i++] = start;
        cout << endl << "Solutia folosind costul uniform este urmatoarea: " << endl;
        for (int j = i - 1; j >= 0; j--)
                cout << nume[solutie[j]] << " ";
        cout << endl << cost[stop] << " km" << " in " << (int) t[stop] << " ore si " <<
                int((t[stop] - (int) t[stop]) *60) << " minute";
        cout << endl;
        cout << endl;
        cout << endl;
}

void latime(int start, int stop)
{
        int viz[20], noduri[20], nrnoduri = 0, parinte[20], ok = 0;
        float t[N], speed;
        float cost[20];
        for (int i = 0; i < N; i++)
                t[i] = 0;
        for (int i = 0; i < 20; i++)
                cost[i] = 0;
        noduri[nrnoduri++] = start;
        for (int i = 0; i < 20; i++)
                viz[i] = 0;
        viz[start] = 1;
        int pas = 0;
        while ((ok == 0) && (nrnoduri > 0))
        {
                int nod = noduri[0];
                for (int i = 0; i < nrnoduri; i++)
                        noduri[i] = noduri[i + 1];      //stergem elem de pe prima pozitie
                nrnoduri--;
                if (nod == stop)

                        ok = 1;
                else
                        for (int i = 0; i < 20; i++)
                                if ((a[nod][i] != 0) && (viz[i] == 0))
                                {
                                        noduri[nrnoduri++] = i;
        //adaugam nodul i pe ultima pozitie din noduri

                                        viz[i] = 1;
```

```
                                        parinte[i] = nod;
                                        cost[i] = cost[nod] + a[nod][i];
                                        if (((float) m[nod][i] / a[nod][i])
>= 10)
                                                speed = (float) 1000 /
((float) m[nod][i] /

                                                        a[nod][i]);

                                        else
                                                speed = nyooom;
                                        t[i] = t[parinte[i]] + a[nod][i] /
speed;
                                }
        }

        int solutie[20], nrsol = 0, final = stop;
        while (final != start)
        {
                solutie[nrsol++] = final;
                final = parinte[final];
        }

        solutie[nrsol++] = start;
        cout << endl << endl << "Cautare in latime de la " << nume[start]
<< " la " <<
                nume[stop] << endl;
        for (int i = nrsol - 1; i >= 0; i--)
                cout << nume[solutie[i]] << " ";
        cout << endl << cost[stop] << " km" << " in " << (int) t[stop] <<
" ore si " <<
                int((t[stop] - (int) t[stop]) *60) << " minute";
}

void adancime(int start, int stop)
{
        int noduri[20], nod, viz[20], nrnoduri = 0, parinte[20], ok = 0;
        float speed, t[N];
        for (int i = 0; i < N; i++)
                t[i] = 0;
        float cost[20];
        for (int i = 0; i < 20; i++)
                viz[i] = 0;
        for (int i = 0; i < 20; i++)
                cost[i] = 0;
        noduri[0] = start;
        nrnoduri++;
        viz[start] = 1;
        while ((ok == 0) && (nrnoduri > 0))
        {
                nod = noduri[0];
                for (int i = 0; i < 20; i++)
                        noduri[i] = noduri[i + 1];
                nrnoduri--;
                if (nod == stop)
                        ok = 1;
                else
```

```
                {
                        for (int i = 0; i < 20; i++)
                                if ((a[nod][i] != 0) && (viz[i] == 0))
                                {
                                        for (int j = nrnoduri; j > 0; j--)
                                                noduri[j] = noduri[j - 1];
                                        noduri[0] = i;

                                        nrnoduri++;
                                        viz[i] = 1;
                                        parinte[i] = nod;
                                        cost[i] = cost[nod] + a[nod][i];
                                        if (((float) m[nod][i] / a[nod][i])
>= 10)
                                                speed = (float) 1000 /
((float) m[nod][i] /

                                                                a[nod][i]);

                                        else
                                                speed = nyooom;
                                        t[i] = t[parinte[i]] + a[nod][i] /
speed;
                                }
                }
        }

        int solutie[20], nrsol = 0, final = stop;
        while (final != start)
        {
                solutie[nrsol++] = final;
                final = parinte[final];
        }

        solutie[nrsol++] = start;
        std::cout << endl << endl << "Cautare in adancime de la " <<
nume[start] << " la " << nume[stop] << endl;
        for (int i = nrsol - 1; i >= 0; i--)
                std::cout << nume[solutie[i]] << " ";
        std::cout << endl << cost[stop] << " km" << " in " << (int)
t[stop] << " ore si " <<
                int((t[stop] - (int) t[stop]) *60) << " minute";
        std::cout << endl;
}

int adancimeLim(int start, int stop, int lim)
{
        int noduri[20], nod, viz[20], nrnoduri = 0, parinte[20], ok = 0,
adancime[N];
        float speed, t[N];
        adancime[start] = 0;
        int nrsol = 0;
        for (int i = 0; i < N; i++)
                t[i] = 0;
        float cost[20];
        for (int i = 0; i < 20; i++)
                viz[i] = 0;
```

```
        for (int i = 0; i < 20; i++)
                cost[i] = 0;
        noduri[0] = start;
        nrnoduri++;
        viz[start] = 1;
        while ((ok == 0) && (nrnoduri > 0))
        {
                nod = noduri[0];
                for (int i = 0; i < 20; i++)
                {
                        noduri[i] = noduri[i + 1];
                }

                nrnoduri--;
                if (nod == stop)
                        ok = 1;
                else
                {
                        for (int i = 0; i < 20; i++)
                        {
                                if ((a[nod][i] != 0) && (viz[i] == 0) &&
(adancime[nod] < lim))
                                {
                                        for (int j = nrnoduri; j > 0; j--)
                                        {
                                                noduri[j] = noduri[j - 1];
                                        }

                                        noduri[0] = i;

                                        nrnoduri++;
                                        viz[i] = 1;
                                        parinte[i] = nod;
                                        cost[i] = cost[nod] + a[nod][i];

                                        adancime[i] = adancime[nod] + 1;

                                        if (((float) m[nod][i] / a[nod][i])
>= 10)
                                                speed = (float) 1000 /
((float) m[nod][i] / a[nod][i]);

                                        else
                                                speed = nyooom;
                                        t[i] = t[parinte[i]] + a[nod][i] /
speed;

                                }
                        }
                }
        }

        if (ok == 1)
        {
                int solutie[20], nrsol = 0, final = stop;
                while (final != start)
                {
```

```
                        solutie[nrsol++] = final;
                        final = parinte[final];
                }

                solutie[nrsol++] = start;

                for (int j = nrsol - 1; j >= 0; j--)
                {
                        cout << nume[solutie[j]] << " " <<
adancime[solutie[j]] << endl;
                }

                cout << endl << cost[stop] << " km" << " in " << (int)
t[stop] << " ore si " <<
                        int((t[stop] - (int) t[stop]) *60) << " minute" <<
endl;
        }

        return ok;

        cout << endl;

        cout << endl;

        /*

        std::cout << endl << endl << "Cautare in adancime de la " <<
nume[start] << " la " << nume[stop] << endl;
        for (int i = nrsol - 1; i >= 0; i--)
                std::cout << nume[solutie[i]] << " ";
        std::cout << endl << cost[stop] << " km" << " in " << (int)t[stop]
<< " ore si "<< int((t[stop] - (int)t[stop]) *60) << " minute";
        std::cout << endl;

        */
        cout << endl;

}

void AdIt(int start, int stop)
{
        int limit = 0;

        bool ok = false;
        do {
                if (!ok && limit < 20)
                        limit++;
                ok = adancimeLim(start, stop, limit);
        } while (!ok);

        cout << "Limita necesara este " << limit << endl;
}

int main()
{
        for (int i = 0; i < N; i++)
                for (int j = 0; j < N; j++)
```

```
                a[i][j] = 0;

        a[0][19] = 75;
        a[0][15] = 140;
        a[0][16] = 118;
        a[1][6] = 90;
        a[1][13] = 101;
        a[1][5] = 211;
        a[1][17] = 85;
        a[2][13] = 138;
        a[2][14] = 146;
        a[2][3] = 120;
        a[3][10] = 75;
        a[4][7] = 80;
        a[5][15] = 99;
        a[7][17] = 98;
        a[8][11] = 87;
        a[8][18] = 92;
        a[9][10] = 70;
        a[9][16] = 111;
        a[12][19] = 71;
        a[12][15] = 151;
        a[13][14] = 97;
        a[14][15] = 80;
        a[17][18] = 142;

        m[0][19] = 800;
        m[0][15] = 1200;
        m[0][16] = 1100;
        m[1][6] = 400;
        m[1][17] = 700;
        m[1][13] = 1000;

        m[1][5] = 1200;
        m[2][14] = 1600;
        m[2][3] = 900;
        m[3][10] = 600;
        m[5][15] = 900;
        m[7][17] = 900;
        m[8][14] = 1200;
        m[9][10] = 400;
        m[9][16] = 1000;
        m[12][15] = 2500;
        m[12][19] = 700;
        m[13][14] = 1300;
        m[14][15] = 800;
        m[17][18] = 1400;
        for (int i = 0; i < N; i++)
                for (int j = 0; j < N; j++)
                        if (a[i][j] != 0)
                                a[j][i] = a[i][j];

        for (int i = 0; i < N; i++)
                for (int j = 0; j < N; j++)
                        if (m[i][j] != 0)
                                m[j][i] = m[i][j];
```

```
        int start, stop, lim = 5, limit = 0;
        cout << "Introduceti nr celor doua orase separate printr-un
spatiu: ";
        cin >> start >> stop;
        latime(start, stop);
        adancime(start, stop);
        cautareCostUniform(start, stop);

        cout << "Cautare in adancime cu limita " << lim << ":" << endl;
        int ok = adancimeLim(start, stop, lim);
        if (ok == 0)
                cout << "Cautarea limitata cu limita " << lim << " nu a
gasit solutia.";
        cout << endl;
        cout << "Adancimea iterativa:" << endl;
        AdIt(start, stop);

}
```

Program rulat

```
Cautare in adancime cu limita 5:
Arad 0
Sibiu 1
Vilcea 2
Pitesti 3
Bucuresti 4

418 km in 4 ore si 30 minute

Adancimea iterativa:
Arad 0
Sibiu 1
Fagaras 2
Bucuresti 3

450 km in 4 ore si 30 minute
Limita necesara este 3
```