Căutare limitată în adâncime

Pornind de la programul rezolvat în L005 cu costurile în km si trafic, implementați în continuare o căutare limitată în adâncime. Utilizatorul trebuie să poată stabili limita până la care se poate găsi soluția. Păstrați în același program și celelalte căutări (lățime, adâncime, cost uniform) pentru a putea compara rutele găsite.

Soluție

Iată una din soluțiile primite. Programul devine prea lung pentru a concatena mai multe soluții primite, cum am procedat până acum.

Soluție

Observații: programul conține căutările in lățime, adâncime, cost uniform și căutare în adâncime limitată. La finalul codului se găsesc două rulări: una cu limita 4 și una cu limita 5, iar ruta găsită este ușor schimbată. Căutarea în adâncime găsea aceeași soluție cu cea de la limita 5.

```
#include<iostream>
#include<conio.h>
#include<string>

using namespace std;

string nume[20] = {
"Oradea","Zerind","Arad","Timisoara","Lugoj","Mehadia","Drobeta","Craiova
","Sibiu","Ramnicu
Valcea","Fagaras","Pitesti","Bucuresti","Giurgiu","Urziceni","Hrisova","E
forie","Vaslui","Iasi","Neamt" };

int distante[20][20], trafic[20][20], i, j;
float timpLatime = 0, timpAdancime = 0, timpUniform = 0, timpAdancimeLim
= 0, costLatime, costAdancime, costUniform, costAdancimeLim;

void conversieTimp(float timp) {
        if (timp != (int)timp) {
                if ((int)timp > 0)
                {
                        cout << (int)timp << " h ";
                }
                float minute = timp - (int)timp;
                minute = minute * 60;
                if (minute != (int)minute)
                {
                        if ((int)minute > 0)
                        {
```

```
                                cout << (int)minute << " min ";
                        }
                        float secunde = minute - (int)minute;
                        secunde = secunde * 60;
                        if ((int)secunde > 0)
                        {
                                cout << (int)secunde << " sec";
                        }
                }
                else
                        cout << minute << " min";
        }
        else
                cout << timp << " h";
}

void cautareLatime(int start, int stop, float& timpLatime, float&
costLatime) {
        int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas =
0, solutie[20], nrSol = 0, final, sume[20];
        bool afisPasi = false;
        final = stop;
        for (i = 0; i < 20; i++)
        {
                viz[i] = 0;
        }

        for (i = 0; i < 20; i++)
        {
                sume[i] = 0;
        }

        noduri[nrNod++] = start;
        viz[start] = 1;

        while ((gasit == 0) && (nrNod > 0))
        {
                int nod = noduri[0];
                int sum = sume[noduri[0]];
                if (afisPasi)
                {
                        cout << "Pasul " << ++pas << ": ";
                        for (i = 0; i < nrNod; i++)
                        {
                                cout << nume[noduri[i]] << " [" <<
sume[noduri[i]] << "] ";
                        }
                        cout << endl;
                }
                for (i = 0; i < nrNod - 1; i++)
                {
                        noduri[i] = noduri[i + 1];
                }
                nrNod--;
                if (nod == stop)
                {
                        gasit = 1;
```

```
                }
                else
                {
                        for (i = 0; i < 20; i++)
                        {
                                if ((distante[nod][i] != 0) && (viz[i] ==
0))
                                {
                                        noduri[nrNod++] = i;
                                        sume[i] = sum + distante[nod][i];
                                        viz[i] = 1;
                                        parinte[i] = nod;
                                }
                        }
                }
        }

        float viteza;
        while (final != start)
        {
                float nrMasiniPeKm = (float)trafic[final][parinte[final]]
/ (float)distante[final][parinte[final]];
                if (nrMasiniPeKm >= 10)
                {
                        viteza = 100 * (10 / nrMasiniPeKm);
                }
                else
                {
                        viteza = 100;
                }
                timpLatime = timpLatime +
(float)distante[final][parinte[final]] / viteza;
                solutie[nrSol++] = final;
                final = parinte[final];
        }
        solutie[nrSol++] = start;
        cout << "Cautati drum de la " << nume[start] << " la " <<
nume[stop] << "." << endl;

        for (i = nrSol - 1; i >= 0; i--)
        {
                cout << nume[solutie[i]] << " ";
        }
        cout << endl;
        cout << "Costul drumului este: " << sume[stop] << " km";
        cout << endl;
        costLatime = sume[stop];
        cout << "Timpul drumului este: ";
        conversieTimp(timpLatime);
}

void cautareAdancime(int start, int stop, float& timpAdancime, float&
costAdancime) {
        int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas =
0, solutie[20], nrSol = 0, final, sume[20];
        bool afisPasi = false;
        final = stop;
```

```
        for (i = 0; i < 20; i++)
        {
                viz[i] = 0;
        }

        for (i = 0; i < 20; i++)
        {
                sume[i] = 0;
        }

        noduri[nrNod++] = start;
        viz[start] = 1;

        while ((gasit == 0) && (nrNod > 0))
        {
                int nod = noduri[0];
                int sum = sume[noduri[0]];
                if (afisPasi)
                {
                        cout << "Pasul " << ++pas << ": ";
                        for (i = 0; i < nrNod; i++)
                        {
                                cout << nume[noduri[i]] << " [" <<
sume[noduri[i]] << "] ";
                        }
                        cout << endl;
                }
                for (i = 0; i < nrNod - 1; i++)
                {
                        noduri[i] = noduri[i + 1];
                }
                nrNod--;
                if (nod == stop)
                {
                        gasit = 1;
                }
                else
                {
                        for (i = 0; i < 20; i++)
                        {
                                if ((distante[nod][i] != 0) && (viz[i] ==
0))
                                {
                                        for (j = nrNod - 1; j >= 0; j--)
                                        {
                                                noduri[j + 1] = noduri[j];
                                        }
                                        noduri[0] = i;
                                        nrNod++;
                                        sume[i] = sum + distante[nod][i];
                                        viz[i] = 1;
                                        parinte[i] = nod;
                                }
                        }
                }
        }
```

```
        float viteza;
        while (final != start)
        {
                float nrMasiniPeKm = (float)trafic[final][parinte[final]]
/ (float)distante[final][parinte[final]];
                if (nrMasiniPeKm >= 10)
                {
                        viteza = 100 * (10 / nrMasiniPeKm);
                }
                else
                {
                        viteza = 100;
                }
                timpAdancime = timpAdancime +
(float)distante[final][parinte[final]] / viteza;
                solutie[nrSol++] = final;
                final = parinte[final];
        }
        solutie[nrSol++] = start;
        cout << "Cautati drum de la " << nume[start] << " la " <<
nume[stop] << "." << endl;

        for (i = nrSol - 1; i >= 0; i--)
        {
                cout << nume[solutie[i]] << " ";
        }
        cout << endl;
        cout << "Costul drumului este: " << sume[stop] << " km";
        cout << endl;
        costAdancime = sume[stop];
        cout << "Timpul drumului este: ";
        conversieTimp(timpAdancime);
}

void cautareUniform(int start, int stop, float& timpUniform, float&
costUniform) {
        int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas =
0, solutie[20], nrSol = 0, final, sume[20];
        bool afisPasi = false;
        final = stop;
        for (i = 0; i < 20; i++)
        {
                viz[i] = 0;
        }

        for (i = 0; i < 20; i++)
        {
                sume[i] = 0;
        }

        noduri[nrNod++] = start;
        viz[start] = 1;

        while ((gasit == 0) && (nrNod > 0))
        {
                int nod = noduri[0];
                int sum = sume[noduri[0]];
```

```
                if (afisPasi)
                {
                        cout << "Pasul " << ++pas << ": ";
                        for (i = 0; i < nrNod; i++)
                        {
                                cout << nume[noduri[i]] << " [" <<
sume[noduri[i]] << "] ";
                        }
                        cout << endl;
                }
                for (i = 0; i < nrNod - 1; i++)
                {
                        noduri[i] = noduri[i + 1];
                }
                nrNod--;
                if (nod == stop)
                {
                        gasit = 1;
                }
                else
                {
                        for (i = 0; i < 20; i++)
                        {
                                if ((distante[nod][i] != 0) && (viz[i] ==
0) || ((distante[nod][i] != 0) && (viz[i] == 1) && (distante[nod][i] > 0)
&& (distante[nod][i] + sum < sume[i])))
                                {
                                        noduri[nrNod++] = i;
                                        sume[i] = sum + distante[nod][i];
                                        viz[i] = 1;
                                        parinte[i] = nod;
                                }
                        }
                        for (i = 0; i < nrNod - 1; i++)
                        {
                                for (j = i + 1; j < nrNod; j++)
                                {
                                        if (sume[noduri[i]] >
sume[noduri[j]])
                                        {
                                                int aux = noduri[i];
                                                noduri[i] = noduri[j];
                                                noduri[j] = aux;
                                        }
                                }
                        }
                }
        }

        float viteza;
        while (final != start)
        {
                float nrMasiniPeKm = (float)trafic[final][parinte[final]]
/ (float)distante[final][parinte[final]];
                if (nrMasiniPeKm >= 10)
                {
                        viteza = 100 * (10 / nrMasiniPeKm);
```

```
                }
                else
                {
                        viteza = 100;
                }
                timpUniform = timpUniform +
(float)distante[final][parinte[final]] / viteza;
                solutie[nrSol++] = final;
                final = parinte[final];
        }
        solutie[nrSol++] = start;
        cout << "Cautati drum de la " << nume[start] << " la " <<
nume[stop] << "." << endl;

        for (i = nrSol - 1; i >= 0; i--)
        {
                cout << nume[solutie[i]] << " ";
        }
        cout << endl;
        cout << "Costul drumului este: " << sume[stop] << " km";
        cout << endl;
        costUniform = sume[stop];
        cout << "Timpul drumului este: ";
        conversieTimp(timpUniform);
}

void cautareAdancimeLim(int start, int stop, float& timpAdancimeLim,
float& costAdancimeLim, int lim) {
        int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas =
0, solutie[20], nrSol = 0, final, sume[20];
        bool afisPasi = false;
        final = stop;
        for (i = 0; i < 20; i++)
        {
                viz[i] = 0;
        }

        for (i = 0; i < 20; i++)
        {
                sume[i] = 0;
        }

        int adancime[20];
        for (i = 0; i < 20; i++)
        {
                adancime[i] = 0;
        }

        noduri[nrNod++] = start;
        viz[start] = 1;

        int nod;
        while ((gasit == 0) && (nrNod > 0))
        {
                nod = noduri[0];
                int sum = sume[noduri[0]];
                if (afisPasi)
```

```
                {
                        cout << "Pasul " << ++pas << ": ";
                        for (i = 0; i < nrNod; i++)
                        {
                                cout << nume[noduri[i]] << " [" <<
sume[noduri[i]] << "] ";
                        }
                        cout << endl;
                }
                for (i = 0; i < nrNod - 1; i++)
                {
                        noduri[i] = noduri[i + 1];
                }
                nrNod--;
                if (nod == stop)
                {
                        gasit = 1;
                }
                else
                {
                        for (i = 0; i < 20; i++)
                        {
                                if ((distante[nod][i] != 0) && (viz[i] ==
0) && (adancime[nod] + 1 <= lim))
                                {
                                        parinte[i] = nod;
                                        adancime[i] = adancime[parinte[i]]
+ 1;
                                        for (j = nrNod - 1; j >= 0; j--)
                                        {
                                                noduri[j + 1] = noduri[j];
                                        }
                                        noduri[0] = i;
                                        nrNod++;
                                        sume[i] = sum + distante[nod][i];
                                        viz[i] = 1;

                                }
                        }
                }
        }


        if (nod != stop)
        {
                cout << "Nu exista solutie.";
        }
        else
        {
                float viteza;
                while (final != start)
                {
                        float nrMasiniPeKm =
(float)trafic[final][parinte[final]] /
(float)distante[final][parinte[final]];
                        if (nrMasiniPeKm >= 10)
                        {
```

```
                                viteza = 100 * (10 / nrMasiniPeKm);
                        }
                        else
                        {
                                viteza = 100;
                        }
                        timpAdancimeLim = timpAdancimeLim +
(float)distante[final][parinte[final]] / viteza;
                        solutie[nrSol++] = final;
                        final = parinte[final];
                }
                solutie[nrSol++] = start;
                cout << "Cautati drum de la " << nume[start] << " la " <<
nume[stop] << "." << endl;

                for (i = nrSol - 1; i >= 0; i--)
                {
                        cout << nume[solutie[i]] << " ";
                }
                cout << endl;
                cout << "Costul drumului este: " << sume[stop] << " km";
                cout << endl;
                costAdancimeLim = sume[stop];
                cout << "Timpul drumului este: ";
                conversieTimp(timpAdancimeLim);
        }
}


int main() {

        int start = 2, stop = 7, lim = 8;

        for (i = 0; i < 20; i++)
        {
                for (j = 0; j < 20; j++)
                {
                        distante[i][j] = 0;
                        trafic[i][j] = 0;
                }
        }

        distante[0][1] = 71;
        distante[0][8] = 151;
        distante[1][2] = 75;
        distante[2][3] = 118;
        distante[2][8] = 140;
        distante[3][4] = 111;
        distante[4][5] = 70;
        distante[5][6] = 75;
        distante[6][7] = 120;
        distante[7][9] = 146;
        distante[7][11] = 138;
        distante[8][9] = 80;
        distante[8][10] = 99;
        distante[9][11] = 97;
        distante[10][12] = 211;
```

```
	distante[11][12] = 101;
	distante[12][13] = 90;
	distante[12][14] = 85;
	distante[14][15] = 98;
	distante[14][17] = 142;
	distante[15][16] = 86;
	distante[17][18] = 92;
	distante[18][19] = 87;

	trafic[0][1] = 700;
	trafic[0][8] = 2500;
	trafic[1][2] = 800;
	trafic[2][3] = 1100;
	trafic[2][8] = 1200;
	trafic[3][4] = 1000;
	trafic[4][5] = 400;
	trafic[5][6] = 600;
	trafic[6][7] = 900;
	trafic[7][9] = 1300;
	trafic[7][11] = 1600;
	trafic[8][9] = 800;
	trafic[8][10] = 900;
	trafic[9][11] = 1300;
	trafic[10][12] = 1200;
	trafic[11][12] = 1000;
	trafic[12][13] = 400;
	trafic[12][14] = 700;
	trafic[14][15] = 900;
	trafic[14][17] = 1400;
	trafic[15][16] = 300;
	trafic[17][18] = 1200;
	trafic[18][19] = 700;

	for (i = 0; i < 20; i++)
	{
		for (j = 0; j < 20; j++)
		{
			if (distante[i][j] != 0)
			{
				distante[j][i] = distante[i][j];
				trafic[j][i] = trafic[i][j];
			}
		}
	}

	cout << "Cautare in latime:" << endl;
	cout << "-----------------------------------------------" << endl;
	cautareLatime(start, stop, timpLatime, costLatime);
	cout << endl;
	cout << "-----------------------------------------------" << endl;
	cout << "Cautare in adancime" << endl;
	cout << "-----------------------------------------------" << endl;
	cautareAdancime(start, stop, timpAdancime, costAdancime);
	cout << endl;
	cout << "-----------------------------------------------" << endl;
	cout << "Cautare cu cost uniform" << endl;
	cout << "-----------------------------------------------" << endl;
```

```
        cautareUniform(start, stop, timpUniform, costUniform);
        cout << endl;
        cout << "-----------------------------------------------" << endl;
        cout << "Cautare in adancime cu limita " << lim << endl;
        cout << "-----------------------------------------------" << endl;
        cautareAdancimeLim(start, stop, timpAdancimeLim, costAdancimeLim,
lim);
        cout << endl;
        cout << "-----------------------------------------------" << endl;
        cout << "Rezultate" << endl;
        cout << "-----------------------------------------------" << endl;
        cout << "Cautare in latime: Drum de " << costLatime << " km " <<
"parcurs in ";
        conversieTimp(timpLatime);
        cout << endl;
        cout << "Cautare in adancime: Drum de " << costAdancime << " km "
<< "parcurs in ";
        conversieTimp(timpAdancime);
        cout << endl;
        cout << "Cautare cu cost uniform: Drum de " << costUniform << " km
" << "parcurs in ";
        conversieTimp(timpUniform);
        cout << endl;
        if (timpAdancimeLim == 0)
        {
                cout << "Cautare in adancime cu limita " << lim << ": Nu
exista solutie";
        }
        else
        {
                cout << "Cautare in adancime cu limita " << lim << ": Drum
de " << costAdancimeLim << " km " << "parcurs in ";
                conversieTimp(timpAdancimeLim);
        }

        _getch();
        return 0;
}
```

Program rulat cu limita 4

```
Cautare in adancime cu limita 4
-----------------------------------------------
Cautati drum de la Arad la Craiova.
Arad Sibiu Ramnicu Valcea Craiova
Costul drumului este: 366 km
Timpul drumului este: 3 h 39 min 35 sec
-----------------------------------------------
Rezultate
-----------------------------------------------
Cautare in latime: Drum de 366 km parcurs in 3 h 39 min 35 sec
Cautare in adancime: Drum de 689 km parcurs in 6 h 53 min 23 sec
Cautare cu cost uniform: Drum de 366 km parcurs in 3 h 39 min 35 sec
Cautare in adancime cu limita 4: Drum de 366 km parcurs in 3 h 39 min 35 sec
```

Program rulat cu limita 5

```
-----------------------------------------------
Cautare in adancime cu limita 5
-----------------------------------------------
Cautati drum de la Arad la Craiova.
Arad Sibiu Fagaras Bucuresti Pitesti Craiova
Costul drumului este: 689 km
Timpul drumului este: 7 h 6 min 36 sec
-----------------------------------------------
Rezultate
-----------------------------------------------
Cautare in latime: Drum de 366 km parcurs in 3 h 39 min 35 sec
Cautare in adancime: Drum de 689 km parcurs in 7 h 6 min 36 sec
Cautare cu cost uniform: Drum de 366 km parcurs in 3 h 39 min 35 sec
Cautare in adancime cu limita 5: Drum de 689 km parcurs in 7 h 6 min 36 sec
```