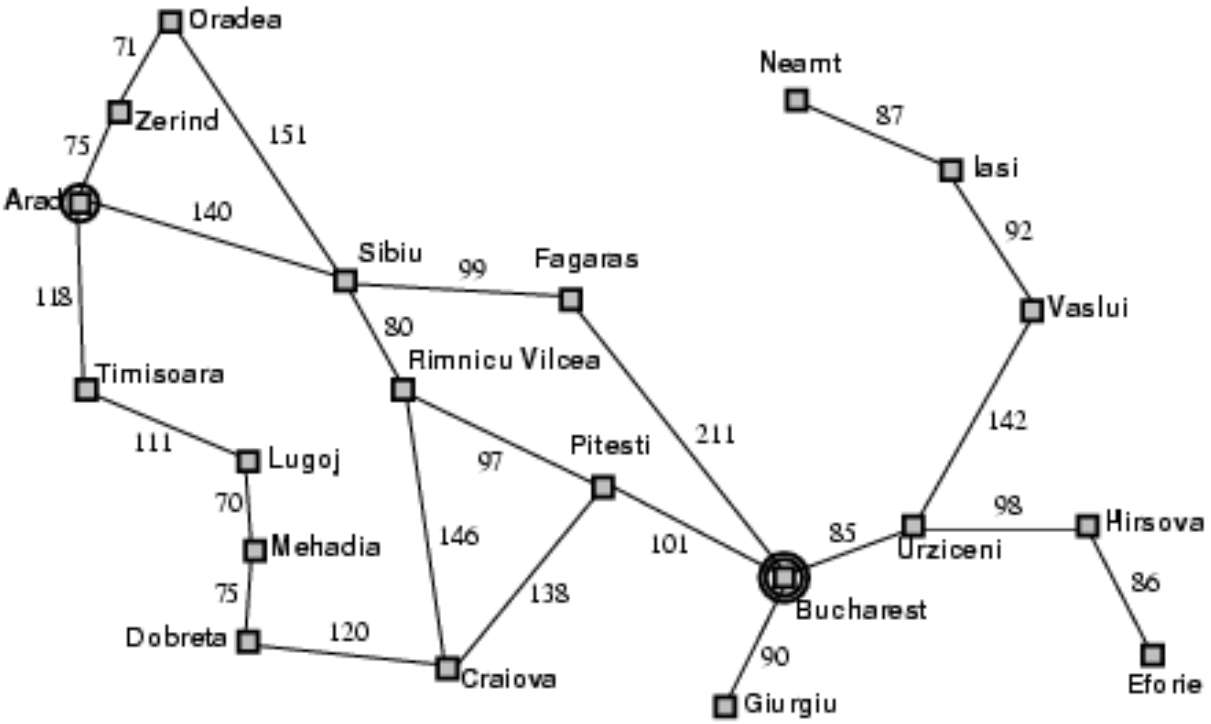
Algoritmii de în lățime, adâncime și cost uniform care cuprind costurile în km și timpii rutelor în ore/minute în funcție de trafic

Fie harta de mai jos:



Știind ca ne deplasam in medie cu 80 km/h si folsind numarul de km intre orase, scrieti un program care sa calculeze timpul de deplasare pe ruta gasita pentru fiecare din cautarile in latime, adancime si cost uniform.

Soluții

Iată câteva rezolvări primite, funcționale.

Soluția 1

Observații: timpii sunt calculați la colectarea soluției, nu în cadrul while-ului în care e populat *noduri*. Durata este afișată inclusiv în secunde.

```
#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","Eforie","Vas
lui","Iasi","Neamt" };

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

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 << " secunde";
			}
		}
		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];
	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]];

		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];
	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]];
		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];
            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]];
		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);
}


int main() {

	int start = 3, stop = 16;

	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[i][j] = trafic[j][i];
			}
		}
	}

	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 << "Rezultate" << endl;
	cout << "-----------------------------------------------" << endl;
	cout << "Cautarea in latime: Drum de " << costLatime << " km " << "parcurs in
";
	conversieTimp(timpLatime);
	cout << endl;
	cout << "Cautarea in adancime: Drum de " << costAdancime << " km " <<
"parcurs in ";
	conversieTimp(timpAdancime);
	cout << endl;
```

```
	cout << "Cautarea cu cost uniform: Drum de " << costUniform << " km " << 
"parcurs in ";
	conversieTimp(timpUniform);

	_getch();
	return 0;
}
```

Programul rulat

```
Pasul 6: Drobeta [256] Sibiu [258] Oradea [264]
Pasul 7: Sibiu [258] Oradea [264] Craiova [376]
Pasul 8: Oradea [264] Ramnicu Valcea [338] Fagaras [357] Craiova [376]
Pasul 9: Ramnicu Valcea [338] Fagaras [357] Craiova [376]
Pasul 10: Fagaras [357] Craiova [376] Pitesti [435]
Pasul 11: Craiova [376] Pitesti [435] Bucuresti [568]
Pasul 12: Pitesti [435] Bucuresti [568]
Pasul 13: Bucuresti [536] Bucuresti [536]
Pasul 14: Bucuresti [536] Urziceni [621] Giurgiu [626]
Pasul 15: Urziceni [621] Giurgiu [626]
Pasul 16: Giurgiu [626] Hrisova [719] Vaslui [763]
Pasul 17: Hrisova [719] Vaslui [763]
Pasul 18: Vaslui [763] Eforie [805]
Pasul 19: Eforie [805] Iasi [855]
Cautati drum de la Timisoara la Eforie.
Timisoara Arad Sibiu Ramnicu Valcea Pitesti Bucuresti Urziceni Hrisova Eforie
Costul drumului este: 805 km
Timpul drumului este: 8 h 3 min
-----------------------------------------
Rezultate
-----------------------------------------
Cautarea in latime: Drum de 837 km parcurs in 8 h 22 min 11 secunde
Cautarea in adancime: Drum de 884 km parcurs in 8 h 50 min 23 secunde
Cautarea cu cost uniform: Drum de 805 km parcurs in 8 h 3 min
```

Soluția 2

Observații: vitezele se calculează pe măsură ce se obține arborele, și respectiv costurile în km. Simplu și eficient.

```
#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","Si
biu","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)
		{
			cout << "Pasul " << k++ << " ";
			for (int i = 0; i < nrNoduri; i++)
				cout << nume[noduri[i]] << " " << cost[noduri[i]] << " ";
			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
				{
					int 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;
}
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;
        cout << endl << endl << "Cautare in adancime 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";
        cout << 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[4][7] = 300;
	m[5][15] = 900;
	m[7][17] = 900;
	m[8][11] = 700;
	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;
	cout << "Introduceti nr celor doua orase separate printr-un spatiu: ";
	cin >> start >> stop;
	latime(start, stop);
	adancime(start, stop);
	cautareCostUniform(start, stop);
}
```

Programul rulat

```
Cautare in latime de la Arad la Bucuresti
Arad Sibiu Fagaras Bucuresti
450 km in 4 ore si 30 minute

Cautare in adancime de la Arad la Bucuresti
Arad Timisoara Lugoj Mehadia Severin Craiova Pitesti Bucuresti
733 km in 7 ore si 19 minute

Solutia folosind costul uniform este urmatoarea:
Arad Sibiu Vilcea Pitesti Bucuresti
418 km in 4 ore si 31 minute
```