Greedy si A* folosind aproximările date de cost uniform

Urmatorul proiect de la curs se refera la implementarea unui program care sa ne permita sa aplicam algoritmii Greedy si A* pentru gasirea rutei de la un oras catre oricare alt oras, nu doar catre Bucuresti, cum am facut pana acum.
Pentru a putea face acest lucru, avem nevoie de aproximari din orice oras catre oricare alt oras.
Pentru a obtine astfel de valori h, se va folosi cautarea cu cost uniform pentru a se gasi cel mai scurt drum de la orice oras catre orice alt oras. Numarul de km dat de rutele de la cost uniform va reprezenta chiar valoarea h.

In acest sens, aveti de ales:
1. Puteti calcula la inceput , inainte de a intra in metoda A*, o matrice h intre oricare doua orase si pe aceasta o utilizati apoi in cadrul algoritmilor Greedy si A*.
2. Puteti modifica metoda de cost uniform astfel incat sa va intoarca costul drumului intre doua orase (in loc de void, cum e acum) si, de cate ori aveti nevoie de o aproximare intre doua noduri n1 si n2, apelati metoda de cost uniform cu argumentele care va trebuie.
3. Varianta cea mai putin eleganta - puteti face doua programe:
3.1 Unul care sa calculeze distantele cu cost uniform intre oricare doua orase si sa le salveze intr-un fisier.
3.2 Al doilea pentru Greedy si A* care sa incarce distantele din fisierul salvat de programul de la 3.1 intr-o matrice h si sa le foloseasca apoi unde are nevoie de aproximarile h.

Acestea sunt directiile la care m-am gandit eu, desigur, voi puteti incerca si altele, daca vreti.

! Subliniez ca sa fiu sigur ca s-a inteles: nu se mai folosesc valorile h anterioare, cele care au fost introduse din partea dreapta a hartii.

Soluție

Soluție

Observații: modificarea principală este făcută la cost uniform, unde se întoarce costul rutei.

```
#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], h[20][20], i, j;
float timpGreedy = 0, timpA = 0, costGreedy, costA;

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";
}

int cautareUniform(int start, int stop) {
	int viz[20], noduri[20], nrNod = 0, gasit = 0, sume[20];
	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]];
		for (i = 0; i < nrNod - 1; i++)
		{
			noduri[i] = noduri[i + 1];
		}
		nrNod--;
		if (nod == stop)
		{
			return sume[stop];
		}
		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;
				}
			}
			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;
					}
				}
			}
		}
	}
}

void cautareGreedy(int start, int stop, float& timpGreddy, float&
costGreedy) {
	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;
                                }
                        }
                        for (i = 0; i < nrNod - 1; i++)
                        {
                                for (j = i + 1; j < nrNod; j++)
                                {
                                        if (h[i][stop] > h[j][stop])
                                        {
                                                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;
                }
                timpGreedy = timpGreedy +
(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;
        costGreedy = sume[stop];
        cout << "Timpul drumului este: ";
        conversieTimp(timpGreedy);
}

void cautareA(int start, int stop, float& timpA, float& costA) {
        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;
                                }
                        }
                        for (i = 0; i < nrNod - 1; i++)
                        {
                                for (j = i + 1; j < nrNod; j++)
                                {
                                        if (distante[noduri[i]] +
h[i][stop] > distante[noduri[j]] + h[j][stop])
                                        {
                                                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;
                }
                timpA = timpA + (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;
        costA = sume[stop];
        cout << "Timpul drumului este: ";
        conversieTimp(timpA);

}

int main() {

        int start = 3, stop = 12;

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

        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];
                        }
                }
        }

        int i1, j1;
        for (i1 = 0; i1 < 20; i1++)
        {
                for (j1 = 0; j1 < 20; j1++)
                {
                        if (h[i1][j1] == 0)
                        {
                                h[i1][j1] = cautareUniform(i1, j1);
                                h[j1][i1] = h[i1][j1];
                        }
                }
        }
```

```
        cout << "Cautare greedy" << endl;
        cout << "-----------------------------------------" << endl;
        cautareGreedy(start, stop, timpGreedy, costGreedy);
        cout << endl;
        cout << "-----------------------------------------" << endl;
        cout << "Cautare A*" << endl;
        cout << "-----------------------------------------" << endl;
        cautareA(start, stop, timpA, costA);
        cout << endl;
        cout << "-----------------------------------------" << endl;
        cout << "Rezultate" << endl;
        cout << "-----------------------------------------" << endl;
        cout << "Cautare greedy: Drum de " << costGreedy << " km parcurs
in ";
        conversieTimp(timpGreedy);
        cout << endl;
        cout << "Cautare A*: Drum de " << costA << " km parcurs in ";
        conversieTimp(timpA);

        _getch();
}
```

Programul rulat

```
Pasul 7: Bucuresti [568] Lugoj [111] Zerind [193] Pitesti [435] C
Cautati drum de la Timisoara la Bucuresti.
Timisoara Arad Sibiu Fagaras Bucuresti
Costul drumului este: 568 km
Timpul drumului este: 5 h 40 min 47 sec
-----------------------------------------
Cautare A*
-----------------------------------------
Pasul 1: Timisoara [0]
Pasul 2: Arad [118] Lugoj [111]
Pasul 3: Lugoj [111] Zerind [193] Sibiu [258]
Pasul 4: Zerind [193] Mehadia [181] Sibiu [258]
Pasul 5: Oradea [264] Mehadia [181] Sibiu [258]
Pasul 6: Mehadia [181] Sibiu [258]
Pasul 7: Sibiu [258] Drobeta [256]
Pasul 8: Drobeta [256] Fagaras [357] Ramnicu Valcea [338]
Pasul 9: Craiova [376] Fagaras [357] Ramnicu Valcea [338]
Pasul 10: Fagaras [357] Pitesti [514] Ramnicu Valcea [338]
Pasul 11: Pitesti [514] Bucuresti [568] Ramnicu Valcea [338]
Pasul 12: Bucuresti [568] Ramnicu Valcea [338]
Cautati drum de la Timisoara la Bucuresti.
Timisoara Arad Sibiu Fagaras Bucuresti
Costul drumului este: 568 km
Timpul drumului este: 5 h 40 min 47 sec
-----------------------------------------
Rezultate
-----------------------------------------
Cautare greedy: Drum de 568 km parcurs in 5 h 40 min 47 sec
Cautare A*: Drum de 568 km parcurs in 5 h 40 min 47 sec
```