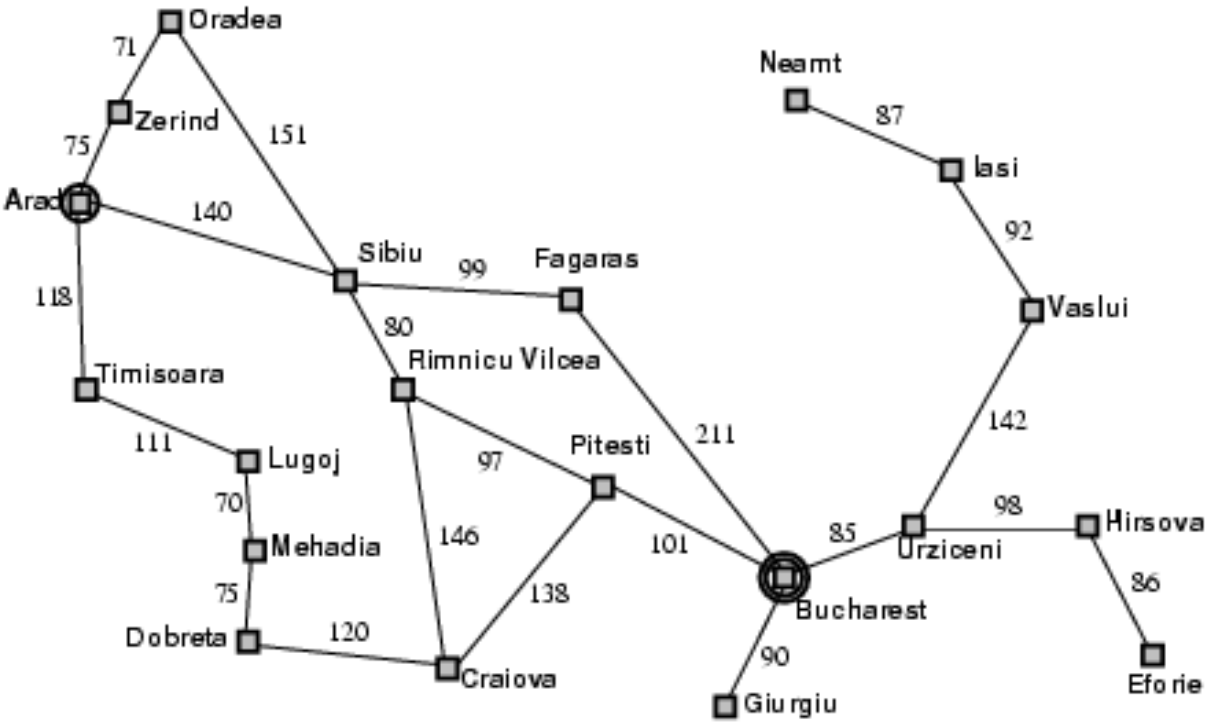
Algoritmii de în lățime, adâncime și cost uniform care cuprind și costurile în km

Fie harta de mai jos:



Folosind modelul de calcul al costului de la radacina la nodul curent din exemplul cautarii cu cost uniform, adaugati costul la cautarea in latime. Faceti acest lucru pentru harta obisnuita de la curs (am atasat si aici fisierul cu harta).

Va rog asadar sa afisati la ruta gasita de la cautarea in latime si costul drumului gasit. Ne va ajuta sa putem compara eficienta algoritmilor.

De altfel, va trebui sa adaugam costurile pentru toti algoritmii de cautare in vederea afisarii numarului de km pentru ruta finala.

Soluție

Adaug un program compilat de mine din mai multe programe astfel încât să cuprindă toate cele 3 căutări lucrate până acum, inclusiv cu costurile calculate.

Soluția

Observație: lățime, adâncime și cost uniform cu costuri calculate

```
#include<iostream>
#include<string>
#define N 20

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" };

int a[N][N];


void cautareCostUniform(int start, int stop){
    int viz[N], noduri[N], parinte[N];
    int nrNoduri = 0;
    for (int i = 0; i < N; i++)
        viz[i] = 0;   //marcam orasele ca nevizitate
    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
    int 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)
        {//afisam componenta tabloului noduri doar daca afisPasi este adevarat
            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 de la start la curent
```

```
                if ((viz[i] == 0) || (costNou < cost[i]))//(nu a fost vizitat) ori (a fost si in acest caz avem pentru el un cost calculat anterior, adica cost[i], ce se compara cu costul nou)
                {
                    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--;
                        }
                    }
                    //cautam pozitia pe care trebuie sa introducem j in "noduri"
                    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

                    //adaugam orasul i pe pozita j
                    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;
                }//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]] << " " << cost[solutie[j]] << "  ";
    cout << endl;

    cout << endl;
}

void latime(int start, int stop)
{
    int viz[20], noduri[20], nrnoduri = 0, parinte[20], ok = 0, cost[20];
    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;
```

```
103.            int pas = 0;
104.            while ((ok == 0) && (nrnoduri > 0))
105.            {
106.                int nod = noduri[0];
107.                for (int i = 0; i < nrnoduri; i++)
108.                    noduri[i] = noduri[i + 1]; //stergem elem de pe prima pozitie
109.                nrnoduri--;
110.                if (nod == stop)
111.                    ok = 1;
112.                else
113.                    for (int i = 0; i < 20; i++)
114.                        if ((a[nod][i]!=0) && (viz[i] == 0))
115.                        {
116.                            noduri[nrnoduri++] = i; //adaugam nodul i pe ultima poz
    itie din noduri
117.                            viz[i] = 1;
118.                            parinte[i] = nod;
119.                            cost[i]=cost[nod] + a[nod][i];
120.                        }
121.            }
122.            int solutie[20], nrsol = 0, final = stop;
123.            while (final != start)
124.            {
125.                solutie[nrsol++] = final;
126.                final = parinte[final];
127.            }
128.            solutie[nrsol++] = start;
129.            cout << endl << endl << "Cautare in latime de la " << nume[start] << "
    la " << nume[stop] << endl;
130.            for (int i = nrsol - 1; i >= 0; i--)
131.                cout << nume[solutie[i]] << " " << cost[solutie[i]] <<"  ";
132.            cout<<endl;
133.
134.        }
135.        void adancime(int start, int stop)
136.        {
137.            int noduri[20], nod, viz[20], nrnoduri = 0, parinte[20], ok = 0, cost[2
    0];
138.            for (int i = 0; i < 20; i++)
139.                viz[i] = 0;
140.            for (int i = 0; i < 20; i++)
141.                cost[i] = 0;
142.            noduri[0] = start;
143.            nrnoduri++;
144.            viz[start] = 1;
145.            while ((ok== 0) && (nrnoduri > 0))
146.            {
147.                nod = noduri[0];
148.                for (int i = 0; i < 20; i++)
149.                noduri[i] = noduri[i + 1];
150.                nrnoduri--;
151.                if (nod == stop)
152.                    ok=1;
153.                else
154.                {
155.                    for (int i = 0; i < 20; i++)
156.                        if ((a[nod][i] != 0) && (viz[i]==0))
157.                        {
158.                            for (int j = nrnoduri; j > 0; j--)
159.                                noduri[j] = noduri[j - 1];
160.                            noduri[0] = i;
161.                            nrnoduri++;
162.                            viz[i] = 1;
163.                            parinte[i] = nod;
164.                            cost[i]=cost[parinte[i]]+a[nod][i];
165.                        }
```

```
166.                 }
167.             }
168.             int solutie[20], nrsol = 0, final = stop;
169.             while (final != start)
170.             {
171.                 solutie[nrsol++] = final;
172.                 final = parinte[final];
173.             }
174.             solutie[nrsol++] = start;
175.             cout << endl << endl << "Cautare in adancime de la " << nume[start] <<
    " la " << nume[stop] << endl;
176.             for (int i = nrsol - 1; i >= 0; i--)
177.                 cout << nume[solutie[i]] << " " << cost[solutie[i]]<<"  ";
178.             cout<<endl;
179.         }
180.
181.
182.
183.         int main(){
184.
185.
186.
187.         for (int i = 0;i < N;i++)
188.             for (int j = 0;j < N;j++)
189.                 a[i][j] = 0;
190.
191.             a[0][19]=75;
192.             a[0][15]=140;
193.             a[0][16]=118;
194.             a[1][6]=90;
195.             a[1][13]=101;
196.             a[1][5]=211;
197.             a[1][17]=85;
198.             a[2][13]=138;
199.             a[2][14]=146;
200.             a[2][3]=120;
201.             a[3][10]=75;
202.             a[4][7]=80;
203.             a[5][15]=99;
204.             a[7][17]=98;
205.             a[8][11]=87;
206.             a[8][18]=92;
207.             a[9][10]=70;
208.             a[9][16]=111;
209.             a[12][19]=71;
210.             a[12][15]=151;
211.             a[13][14]=97;
212.             a[14][15]=80;
213.             a[17][18]=142;
214.
215.             for (int i = 0;i < N;i++)
216.             {
217.                 for (int j = 0;j < N;j++)
218.                 {
219.                     if (a[i][j] != 0)
220.                         a[j][i] = a[i][j];
221.                 }
222.             }
223.
224.
225.             int start = 0, stop = 1;
226.             latime(start, stop);
227.             adancime(start, stop);
228.             cautareCostUniform(start, stop);
229.         }
```

Programul rulat

```
Cautare in latime de la Arad la Bucuresti
Arad 0  Sibiu 140  Fagaras 239  Bucuresti 450


Cautare in adancime de la Arad la Bucuresti
Arad 0  Timisoara 118  Lugoj 229  Mehadia 299  Severin 374  Craiova 494  Pitesti 632  Bucuresti 733

Solutia folosind costul uniform este urmatoarea:
Arad 0  Sibiu 140  Vilcea 220  Pitesti 317  Bucuresti 418


--------------------------------
Process exited after 0.2457 seconds with return value 0
Press any key to continue . . .
```