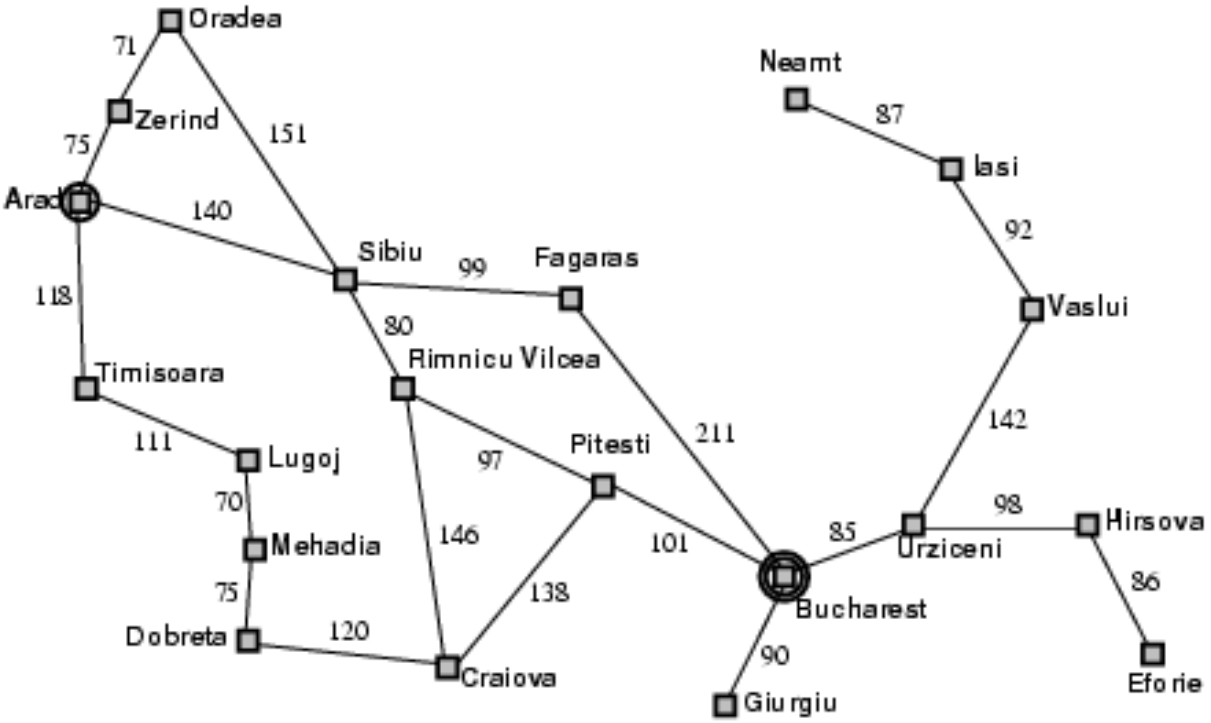
Algoritmul de căutare în lățime - rezolvare

Fie harta de mai jos:



1. Aplicați algoritmul de căutare in lățime pentru a găsi ruta intre doua orase oarecare. Algoritmul este descris la finalul acestui laborator.

Algoritm de cautare in latime
Toate orasele sunt nevizitate.
Adaugam in lista *noduri* orasul de plecare.
Marcam orasul de plecare ca vizitat.
Cat timp solutie negasita si *noduri* ≠ ∅ *executa*
    nod = scoate_din_fata(*noduri*) //stocam primul element din *noduri* in variabila *nod*
    Eliminam primul element din *noduri*
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        Solutia este gasita //facem variabila booleana gasit adevarata
    Altfel
        Adaugam la final in *noduri* orasele nevizitate care sunt conectate de *nod*
        Orasele adaugate sunt marcate ca vizitate
        Se retine pentru oricare din orasele adaugate nodul *parinte* ca fiind *nod*
Sfarsit cat timp
Stocam solutia parcurgand orasele de la destinatie catre start utilizand *parintii* retinuti.

Programe primite ca soluții

În continuare, pun soluțiile primite pentru a le avea cu toții ca exemplu. Nu toate sunt perfecte, însă și din greșeli putem învăța. Ordinea soluțiilor este aleatorie.
Vă amintesc că acest program a fost deja rezolvat la orele de laborator.

Soluția 1

Observație: numele metodei ar fi trebuit sa fie mai sugestiv, de exemplu „cautareLatime” în loc de „functie”.

```
1. #include <iostream>
2. #include <string>
3. using namespace std;
4. string nume[20] = { "Arad","Zerind","Oradea","Sibiu","Timisoa
   ra","Lugoj","Mehadia","Drobeta","Fagaras","Ramnicu Valcea","P
   itesti","Craiova","Neamt","Iasi","Vaslui","Urziceni","Bucures
   ti","Giugiu","Harsova","Eforie" };
5. int a[20][20];
6. void functie(int c, int b) {
7.     int parinte[20];
8.     int noduri[20];
9.     int vizitat[20];
10.         for (int i = 0; i < 20; i++)
11.             vizitat[i] = 0;
12.         int total_noduri = 0;
13.         noduri[total_noduri++] = c;
14.         vizitat[c] = 1;
15.         int gasit = 0;
16.         while ((gasit == 0) && (total_noduri > 0)) {
17.             int nod = noduri[0];
18.             for (int i = 0; i < total_noduri - 1; i++)
19.                 noduri[i] = noduri[i + 1];
20.             total_noduri--;
21.             if (nod == b)
22.                 gasit = 1;
23.             else
24.                 for (int i = 0; i < 20; i++)
25.                     if ((a[i][nod] == 1) && (vizitat[i] == 0
   )) {
26.                         noduri[total_noduri++] = i;
27.                         vizitat[i] = 1;
28.                         parinte[i] = nod;
29.                     }
30.         }
31.         int solutie[20];
32.         for (int i = 0; i < 20; i++)
33.             solutie[i] = 0;
```

```
34.         int elem_solutie = 0;
35.         int destinatie = b;
36.         while (destinatie != c) {
37.             solutie[elem_solutie] = destinatie;
38.             elem_solutie++;
39.             destinatie = parinte[destinatie];
40.         }
41.         solutie[elem_solutie++] = c;
42.         cout << "Cautare in latime: ruta de la " << nume[c]
    << " pana la " << nume[b]<<endl;
43.         for (int i = (elem_solutie - 1); i >= 0; i--)
44.             cout << nume[solutie[i]] << " ";
45.     }
46.     int main() {
47.         int start = 8, stop = 0;
48.         for (int i = 0; i < 20; i++)
49.             for (int j = 0; j < 20; j++)
50.                 a[i][j] = 0;
51.         a[0][1] = 1;
52.         a[0][4] = 1;
53.         a[0][3] = 1;
54.         a[1][2] = 1;
55.         a[4][5] = 1;
56.         a[3][8] = 1;
57.         a[3][9] = 1;
58.         a[5][6] = 1;
59.         a[2][3] = 1;
60.         a[6][7] = 1;
61.         a[7][11] = 1;
62.         a[9][10] = 1;
63.         a[8][16] = 1;
64.         a[10][11] = 1;
65.         a[16][17] = 1;
66.         a[15][16] = 1;
67.         a[14][15] = 1;
68.         a[13][14] = 1;
69.         a[12][13] = 1;
70.         a[15][18] = 1;
71.         a[18][19] = 1;
72.         a[9][11]=1;
73.         a[10][16]=1;
74.         for (int i = 0; i < 20; i++)
75.             for (int j = 0; j < 20; j++)
76.                 if (a[i][j] == 1)
77.                     a[j][i] = 1;
78.         functie(start, stop);
79.     }
```

Soluția 2

Observație: programul are și comentarii, probabil este cel mai util pentru a fi înțeles dintre cele primite. Fiecare oraș are la *nume* și poziția din tabloul *nume* pentru a putea fi mai ușor identificat la ce oraș se referă atunci când se fac conexiunile.

```
1.  #include<iostream>
2.  #include<conio.h>
3.  #include<string>
4.  using namespace std;
5.
6.  string nume[20] = { "Arad 0","Bucuresti 1","Craiova 2","Drobeta 3","Euforie 4","Fa
    garas 5","Giurgiu 6","Harsova 7","Iasi 8",
7.  "Lugoj 9","Mehadia 10","Neamt 11","Oradea 12","Pitesti 13","Ramnicu Valcea 14","Si
    biu 15 ","Timisoara 16","Urziceni 17",
8.  "Vaslui 18","Zerind 19" };\
9.
10. int a[20][20];
11. int  n=20;
12.
13. void latime(int start, int stop)
14. {
15.     int  viz[20], noduri[20],
16.         nrnoduri = 0, parinte[20], gasit = 0;
17.
18.     for (int i = 0;i < 20;i++)
19.         viz[i] = 0; // toate orasele sunt nevizitate
20.     noduri[nrnoduri++] = start;//adaugamla lista noduri orasele de plecare
21.     viz[start] = 1; int pas = 0;//marcam orasul de plecare ca vizitat
22.     while ((gasit == 0) && (nrnoduri > 0)) { // cat timpsolutia gasita si noduri=x
    executa
23.
24.         int  nod = noduri[0];// nod=scoate_din_fata(noduri),(scadem primul element
    din noduri in variabila nod)
25.
26.         //cout << nume[nod] << " ";
27.
28.     /*  cout << "Pasul " << ++pas << " ";
29.         for (int i = 0; i < nrnoduri;i++)
30.         cout << nume[noduri[i]] << " ";
31.         cout << endl; */
32.
33.         for (int i = 0;i < nrnoduri - 1;i++)
34.             noduri[i] = noduri[i + 1];//eliminam primul element din noduri
35.         nrnoduri--;
36.
37.         if (nod == stop) // daca problema se aplica in stare(nod) atunci
38.             gasit = 1;  // solutia este gasita,face variabila booleana gasita adv

39.         else
40.             for (int i = 0;i<20;i++)
41.                 if ((a[nod][i]) && (viz[i] == 0))
42.                 {
43.                     noduri[nrnoduri++] = i; // adaugam i pe ultima pozitie
44.                     viz[i] = 1;//marcam orasele vizitate
45.                     parinte[i] = nod;
46.                 }
47.     }
48.
49.     int solutie[20], nrsol = 0, final = stop;\
50.
51.         while (final != start)
52.         {
53.             solutie[nrsol++] = final;
```

```
54.             final = parinte[final];
55.         }
56.     solutie[nrsol++] = start;
57.     cout << "cautare de la " << nume[start] << " la " << nume[stop] << endl;
58.     for (int i = nrsol - 1; i >= 0; i--)
59.         cout << nume[solutie[i]] << " " << endl;
60.     cout << endl;
61. }
62.
63. int main(){
64.
65.
66.
67. for (int i = 0;i < n;i++)
68. {
69.     for (int j = 0;j < n;j++) {
70.         a[i][j] = 0;
71.     }
72. }
73.
74.     a[0][15] = 1;
75.     a[0][16] = 1;
76.     a[0][19] = 1;
77.     a[1][6] = 1;
78.     a[1][13] = 1;
79.     a[1][5] = 1;
80.     a[1][17] = 1;
81.     a[2][3] = 1;
82.     a[2][13] = 1;
83.     a[2][14] = 1;
84.     a[3][10] = 1;
85.     a[4][7] = 1;
86.     a[5][15] = 1;
87.     a[7][17] = 1;
88.     a[8][11] = 1;
89.     a[8][18] = 1;
90.     a[9][10] = 1;
91.     a[9][16] = 1;
92.     a[12][15] = 1;
93.     a[12][19] = 1;
94.     a[13][14] = 1;
95.     a[14][15] = 1;
96.     a[17][18] = 1;
97.
98.
99.     for (int i = 0;i < n;i++)
100.          {
101.              for (int j = 0;j < n;j++)
102.              {
103.                  if (a[i][j] == 1)
104.                      a[j][i] = 1;
105.              }
106.          }
107.
108.          //punct 2
109.
110.      /*  int oras = 7;
111.          cout << "Conexiuni pt " << nume[oras] << " sunt: ";
112.          for (int i = 0;i < 20;i++)
113.              if (a[oras][i] == 1)
114.                  cout<< nume[i]<<" ";
115.          cout << endl;
116.          */
117.
118.          //punct 3
119.          int start = 0, stop = 1;
```

```
120.
121.          latime(start, stop);
122.
123.      }
```

Soluția 3

Observație: programul conține și două soluții comentate ce au o altă sursă de inspirație.

```
1.  #include<stdio.h>
2.  #include<conio.h>
3.  #include<string>
4.  #include <iostream>
5.
6.  using namespace std;
7.
8.  string v[20] = { "Oradea", "Zerind", "Arad", "Sibiu", "Fagaras", "Timisoara", "Lug
    oj", "Mehadia", "Severin", "Craiova", "Vilcea", "Pitesti", "Bucuresti", "Giurgiu",
     "Urziceni", "Hirsova", "Eforie", "Vaslui", "Iasi", "Neamt" };
9.  int a[20][20], c[20];
10.
11. /*/void DFS(int x, int y) {
12. int viz[20] = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 };
13. cout << v[x] << ", ";
14. viz[x] = 1;
15. for (int i = 0;i < y;i++)
16. if (a[x][i] == 1 && viz[i] == 0 && i != y)
17. DFS(i, y);
18. else if (i == y)
19. break;
20. cout << v[y];
21. }/*/
22.
23. /*/void BFS(int x, int y) {
24.     int viz[20] = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 };
25.     int u = 1, p = 1;
26.     bool ok = 0;
27.     cout << v[x] << ", ";
28.     c[1] = x;
29.     viz[x] = 1;
30.     while ((p <= u) && (ok == 0)) {
31.         for (int i = x;i < y;i++)
32.             if (a[c[p]][i] == 1 && viz[i] == 0 && c[p] != y) {
33.                 viz[i] = 1;
34.                 cout << v[i] << ", ";
35.                 c[++u] = i;
36.             }
37.             else if (c[p] == y) {
38.                 ok = 1;
39.                 break;
40.             }
41.             p++;
42.     }
43.     cout << v[y];
44. }/*/
45.
46. void latime(int start, int stop) {
47.     int viz[20], noduri[20], nrnoduri = 0, parinte[20], ok = 0;
48.     noduri[nrnoduri++] = start;
49.     for (int i = 0;i < 20;i++)
```

```
50.         viz[i] = 0;
51.     viz[start] = 1;
52.     int pas = 0;
53.     while ((ok == 0) && (nrnoduri > 0)) {
54.         int nod = noduri[0];
55.         //cout << v[nod] << " ";
56.         cout << "Pasul: " << ++pas << " ";
57.         for (int i = 0;i < nrnoduri;i++)
58.             cout << v[noduri[i]] << " ";
59.         cout << endl;
60.         for (int i = 0;i < nrnoduri;i++)
61.             noduri[i] = noduri[i + 1];  //stergem elem de pe prima pozitie
62.         nrnoduri--;
63.         if (nod == stop) ok = 1;
64.         else
65.             for (int i = 0;i < 20;i++)
66.                 if ((a[nod][i] == 1) && (viz[i] == 0))
67.                 {
68.                     noduri[nrnoduri++] = i;  //adaugam nodul i pe ultima pozitie d
    in noduri
69.                     viz[i] = 1;
70.                     parinte[i] = nod;
71.                 }
72.     }
73.     int solutie[20], nrsol = 0, final = stop;
74.     while (final != start) {
75.         solutie[nrsol++] = final;
76.         final = parinte[final];
77.     }
78.     solutie[nrsol++] = start;
79.     cout << endl << endl << "Cautare in latime de la " << v[start] << " la " << v[
    stop] << endl;
80.     for (int i = nrsol - 1;i >= 0;i--)
81.         cout << v[solutie[i]] << " ";
82.     cout << endl;
83. }
84.
85. int main() {
86.     int z = 5;
87.     //pb1
88.     for (int i = 0;i < 20;i++)
89.         for (int j = 0;j < 20;j++)
90.             a[i][j] = 0;
91.     a[0][1] = 1;
92.     a[0][3] = 1;
93.     a[1][2] = 1;
94.     a[2][3] = 1;
95.     a[2][5] = 1;
96.     a[5][6] = 1;
97.     a[6][7] = 1;
98.     a[7][8] = 1;
99.     a[8][9] = 1;
100.          a[9][11] = 1;
101.          a[11][10] = 1;
102.          a[10][3] = 1;
103.          a[10][11] = 1;
104.          a[11][12] = 1;
105.          a[12][13] = 1;
106.          a[12][14] = 1;
107.          a[14][15] = 1;
108.          a[15][16] = 1;
109.          a[14][17] = 1;
110.          a[17][18] = 1;
111.          a[18][19] = 1;
112.          for (int i = 0;i < 20;i++)
113.              for (int j = 0;j < 20;j++)
```

```
                    if (a[i][j] == 1)
                        a[j][i] = 1;
            //pb2
            int x;
            cout << "Introduceti nr orasului: ";
            cin >> x;
            cout << "Orasul ales este: " << v[x];
            cout << endl << "Orasele conexe sunt: " << endl;
            for (int i = 0;i < 20;i++)
                if (a[x][i] == 1)
                    cout << v[i] << ", ";
            //pb3
            cout << endl << endl << endl;
            cout << "Introduceti nr celor 2 orase: ";
            int b, c;
            cin >> b >> c;
            cout << "Numele celor doua orase sunt: " << v[b] << " si " << v[c] << e
    ndl;
            //BFS(b, c);
            cout << endl << endl;

            //pb4
            cout << "Parcurgere: " << endl;
            //DFS(b, c);

            latime(b, c);
            getch();
        }
```

Soluția 4

Observație: numele metodei ar fi trebuit sa fie mai sugestiv, de exemplu „cautareLatime” în loc de „functie”. Programul afișează și conexiunile orașului 7.

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




using namespace std;


    string nume[20] = { "Arad","Zerind","Oradea","Sibiu","Timisoara","Lugoj","Meha
    dia","Drobeta","Fagaras","R Valcea","Pitesti","Craiova","Neamt","Iasi","Vaslui","U
    rziceni","Bucurestu","Giurgiu","Harsova","Eforie" };
    int a[20][20];

    void functie(int start, int stop) {


        int oras = 7;

        cout << "Conexiuni pentru " << nume[oras];

        for (int i = 0; i < 20; i++)
            if (a[oras][i] == 1)
                cout << nume[i] << " ";

        start = 0;
```

```
26.         stop = 12;
27.
28.         int parinte[20];
29.         int noduri[20];
30.         int vizitat[20];
31.         for (int i = 0; i < 20; i++)
32.             vizitat[i] = 0;
33.         int totalNoduri = 0;
34.         noduri[totalNoduri++] = start;
35.         vizitat[start] = 1;
36.         int gasit = 0;
37.
38.         while ((gasit == 0) && (totalNoduri >0)) {
39.
40.             int nod = noduri[0];
41.             for (int i = 0; i < totalNoduri - 1; i++)
42.                 noduri[i] = noduri[i + 1];
43.             totalNoduri--;
44.             if (nod == stop)
45.                 gasit = 1;
46.             else
47.                 for(int i=0;i<20;i++)
48.                     if ((a[i][nod] == 1) && (vizitat[i] == 0)) {
49.                         noduri[totalNoduri++] = i;
50.                         vizitat[i] = 1;
51.                         parinte[i] = nod;
52.
53.
54.                     }
55.
56.         }
57.
58.         int solutie[20];
59.         int nrSol = 0, final = stop;
60.
61.         while (final != start) {
62.
63.             solutie[nrSol++] = final;
64.
65.     final = parinte[final];
66.         }
67.         solutie[nrSol++] = start;
68.
69.         cout << "Cautare in latime=" << nume[start] << "Pana la " << nume[stop] <<
    endl;
70.
71.         for (int i = (nrSol - 1); i >= 0; i--)
72.             cout << nume[solutie[i]] << " ";
73.         cout << endl;
74.
75.     }
76.     int main() {
77.
78.         int start = 0, stop = 1;
79.         for (int i = 0; i < 20; i++) {
80.             for (int j = 0; j < 20; j++) {
81.                 a[i][j] = 0;
82.             }
83.         }
84.          a[0][1] = 1;
85.          a[0][4] = 1;
86.      a[0][3] = 1;
87.          a[1][2] = 1;
88.          a[4][5] = 1;
89.     a[3][8] = 1;
90.          a[3][9] = 1;
```

```
91.           a[5][6] = 1;
92.           a[2][3] = 1;
93.           a[6][7] = 1;
94.           a[7][11] = 1;
95.           a[9][10] = 1;
96.           a[8][16] = 1;
97.           a[10][11] = 1;
98.           a[16][17] = 1;
99.       a[15][16] = 1;
100.            a[14][15] = 1;
101.                a[13][14] = 1;
102.                a[12][13] = 1;
103.                a[15][18] = 1;
104.                a[18][19] = 1;
105.                a[9][11] = 1;
106.                a[10][16] = 1;
107.               for (int i = 0; i < 20; i++) {
108.                   for (int j = 0; j < 20; j++) {
109.                       if (a[i][j] == 1)
110.                           a[i][j] = 1;
111.                   }
112.               }
113.               functie(start, stop);
114.
115.               return 0;
116.
117.           }
```

Soluția 5

Observație: Pornește de la un proiect generat automat, nu o sa vă funcționeze întocmai în formatul de mai jos. Dar este un cod aerisit, ușor de urmărit.

```
1.  // AILab2Cpp.cpp : This file contains the 'main' function. Program execution begin
    s and ends there.
2.  //
3.
4.  #include "pch.h"
5.  #include <iostream>
6.  #include <string>
7.  #include <vector>
8.
9.  std::string name[20] = { "Arad", "Zerind", "Oradea", "Neamt", "Timisoara",
10.         "Lugoj", "Mehadia", "Drobeta", "Sibiu", "Craiova", "Pitesti", "Giurgiu", "
    Bucuresti", "Urziceni"
11.         "Vaslui", "Hirsova", "Eforie", "Iasi", "Valcea", "Fagaras" };
12.
13. int a[20][20];
14.
15. void cautareInLatime(int start, int stop)
16. {
17.     int viz[20], parinte[20], noduri[20];
18.
19.     for (int index = 0; index < 20; index++)
20.         viz[index] = 0;
21.
22.     int nrNoduri = 0;
23.
24.     //Primul nod al arborelui este cel de plecare
25.     noduri[nrNoduri++] = start;
26.
```

```
27.     //Marcam orasu de start ca vizitat
28.     viz[start] = 1;
29.
30.     int gasit = 0;
31.
32.     int nod;
33.
34.     int pas = 0;
35.
36.     while ((gasit == 0) && (nrNoduri > 0))
37.     {
38.         nod = noduri[0];
39.
40.         pas++;
41.
42.         std::cout << name[nod] << " ";
43.         std::cout << "Pasul" << " " << pas;
44.         std::cout << std::endl;
45.
46.         for (int index = 0; index < nrNoduri - 1; index++)
47.         {
48.             //Stergem elementu de pe prima pozitie din noduri
49.             noduri[index] = noduri[index + 1];
50.         }
51.
52.         nrNoduri--;
53.
54.         if (nod == stop)
55.             gasit = 1;
56.         else
57.         {
58.             for (int index = 0; index < 20; index++)
59.                 if ((a[nod][index] == 1) && (viz[index] == 0))
60.                 {
61.                     //Adaugam i pe ultima pozitie din lista de noduri
62.                     noduri[nrNoduri++] = index;
63.
64.                     //Marcam index ca vizitat
65.                     viz[index] = 1;
66.
67.                     //Parintele lui index este nod
68.                     parinte[index] = nod;
69.                 }
70.         }
71.     }
72.
73.     int solutie[20];
74.
75.     int nrSol = 0;
76.
77.     int dest = stop;
78.     while (start != dest)
79.     {
80.         solutie[nrSol++] = dest;
81.         dest = parinte[dest];
82.     }
83.
84.     solutie[nrSol++] = start;
85.
86.     for (int index = nrSol - 1; index >= 0; index--)
87.     {
88.         std::cout << name[solutie[index]] << " ";
89.     }
90.
91.     std::cout << std::endl;
92. }
```

```
93.
94. void cautareInAdancime(int start, int stop)
95. {
96.
97. }
98.
99. int main()
100.        {
101.            for (int index1 = 0; index1 < 20; index1++)
102.                for (int index2 = 0; index2 < 20; index2++)
103.                    a[index1][index2] = 0;
104.
105.            a[0][1] = 1;
106.            a[1][0] = 1;
107.            a[0][4] = 1;
108.            a[4][0] = 1;
109.            a[0][8] = 1;
110.            a[8][0] = 1;
111.            a[1][2] = 1;
112.            a[2][1] = 1;
113.            a[2][8] = 1;
114.            a[8][2] = 1;
115.            a[4][5] = 1;
116.            a[5][4] = 1;
117.            a[5][6] = 1;
118.            a[6][5] = 1;
119.            a[6][7] = 1;
120.            a[7][6] = 1;
121.            a[7][9] = 1;
122.            a[9][7] = 1;
123.            a[19][8] = 1;
124.            a[8][19] = 1;
125.            a[19][9] = 1;
126.            a[9][19] = 1;
127.            a[19][10] = 1;
128.            a[10][19] = 1;
129.            a[8][18] = 1;
130.            a[18][8] = 1;
131.            a[9][10] = 1;
132.            a[10][9] = 1;
133.            a[12][10] = 1;
134.            a[10][12] = 1;
135.            a[12][18] = 1;
136.            a[18][12] = 1;
137.            a[12][11] = 1;
138.            a[11][12] = 1;
139.            a[12][13] = 1;
140.            a[13][12] = 1;
141.            a[13][15] = 1;
142.            a[15][13] = 1;
143.            a[15][16] = 1;
144.            a[16][15] = 1;
145.            a[14][13] = 1;
146.            a[13][14] = 1;
147.            a[17][14] = 1;
148.            a[14][17] = 1;
149.            a[17][3] = 1;
150.            a[3][17] = 1;
151.
152.            int k;
153.            std::string city;
154.
155.            std::cout << "Read city name"<< std::endl;
156.            std::cin >> city;
157.
158.            for (int index = 0; index < 20; index++)
```

```
159.           {
160.               if (city == name[index])
161.               {
162.                   k = index;
163.                   break;
164.               }
165.
166.           }
167.
168.           for (int index = 0; index < 20; index++)
169.           {
170.               if (a[k][index] == 1)
171.                   std::cout << name[index] << " ";
172.           }
173.
174.           std::cout << std::endl;
175.
176.           //Graf, cautare in latime
177.           cautareInLatime(7, 12);
178.
179.           system("pause");
180.       }
181.
182.       // Run program: Ctrl + F5 or Debug > Start Without Debugging menu
183.       // Debug program: F5 or Debug > Start Debugging menu
184.
185.       // Tips for Getting Started:
186.       //   1. Use the Solution Explorer window to add/manage files
187.       //   2. Use the Team Explorer window to connect to source control
188.       //   3. Use the Output window to see build output and other messages
189.       //   4. Use the Error List window to view errors
190.       //   5. Go to Project > Add New Item to create new code files, or Project >
     Add Existing Item to add existing code files to the project
191.       //   6. In the future, to open this project again, go to File > Open > Proj
    ect and select the .sln file
```

Soluția 6

Observație: numele metodei ar fi trebuit sa fie mai sugestiv, de exemplu „cautareLatime” în loc de „functie”.

```
1.  #include <iostream>
2.  #include <string>
3.  using namespace std;
4.  string nume[20] = { "Arad","Zerind","Oradea","Sibiu","Timisoara","Lugoj","Mehadia"
    ,"Drobeta","Fagaras","Ramnicu Valcea","Pitesti","Craiova","Neamt","Iasi","Vaslui",
    "Urziceni","Bucuresti","Giugiu","Harsova","Eforie" };
5.  int a[20][20];
6.  void latime(int c, int b) {
7.      int parinte[20];
8.      int noduri[20];
9.      int vizitat[20];
10.     for (int i = 0; i < 20; i++)
11.         vizitat[i] = 0;
12.     int total_noduri = 0;
13.     noduri[total_noduri++] = c;
14.     vizitat[c] = 1;
15.     int gasit = 0;
16.     while ((gasit == 0) && (total_noduri > 0)) {
17.         int nod = noduri[0];
18.         for (int i = 0; i < total_noduri - 1; i++)
```

```
            noduri[i] = noduri[i + 1];
        total_noduri--;
        if (nod == b)
            gasit = 1;
        else
            for (int i = 0; i < 20; i++)
                if ((a[i][nod] == 1) && (vizitat[i] == 0)) {
                    noduri[total_noduri++] = i;
                    vizitat[i] = 1;
                    parinte[i] = nod;
                }
    }
    int solutie[20];
    for (int i = 0; i < 20; i++)
        solutie[i] = 0;
    int elem_solutie = 0;
    int destinatie = b;
    while (destinatie != c) {
        solutie[elem_solutie] = destinatie;
        elem_solutie++;
        destinatie = parinte[destinatie];
    }
    solutie[elem_solutie++] = c;
    cout << "Cautare in latime: ruta de la " << nume[c] << " pana la " << nume[b]<
<endl;
    for (int i = (elem_solutie - 1); i >= 0; i--)
        cout << nume[solutie[i]] << " ";
}
int main() {
    int start = 8, stop = 0;
    for (int i = 0; i < 20; i++)
        for (int j = 0; j < 20; j++)
            a[i][j] = 0;
    a[0][1] = 1;
    a[0][4] = 1;
    a[0][3] = 1;
    a[1][2] = 1;
    a[4][5] = 1;
    a[3][8] = 1;
    a[3][9] = 1;
    a[5][6] = 1;
    a[2][3] = 1;
    a[6][7] = 1;
    a[7][11] = 1;
    a[9][10] = 1;
    a[8][16] = 1;
    a[10][11] = 1;
    a[16][17] = 1;
    a[15][16] = 1;
    a[14][15] = 1;
    a[13][14] = 1;
    a[12][13] = 1;
    a[15][18] = 1;
    a[18][19] = 1;
    a[9][11]=1;
    a[10][16]=1;
    for (int i = 0; i < 20; i++)
        for (int j = 0; j < 20; j++)
            if (a[i][j] == 1)
                a[j][i] = 1;
    latime(start, stop);
}
```

Soluția 7

Observație: numele metodei ar fi trebuit sa fie mai sugestiv, de exemplu „cautareLatime” în loc de „functie”.

```
#include<iostream>
#include<string>
using namespace std;

string nume[20]={"arad","zenid","oradea","sibiu","timisoara","lugoj","mehadia","dr
obeta","fagaras","VL","pitesti","craiova","NT","iasi","vaslui","urziceni","bucures
ti","giurgiu","harsova","eforie"};
int a[20][20];

void functie(int c,int b){
        int parinte[20];
        int noduri[20];
        int vizitat[20];
        for(int i=0;i<20;i++)
            vizitat[i]=0;
        int total_noduri=0;
        noduri[total_noduri++]=c;
        vizitat[c]=1;
        int gasit=0;
        while((gasit==0)&&(total_noduri>0)){
            int nod=noduri[0];
            for(int i=0;i<total_noduri-1;i++)
                noduri[i]=noduri[i+1];
            total_noduri--;
            if(nod==b)
                gasit=1;
            else
                for(int i=0;i<20;i++)
            if((a[i][nod]==1)&&(vizitat[i]==0)){
                noduri[total_noduri++]=i;
                vizitat[i]=1;
                parinte[i]=nod;
            }
        }

        int solutie[20];
        for(int i=0;i<20;i++)
            solutie[i]=0;
        int elem_solutie=0;
        int destinatie=b;
        while(destinatie!=c){
            solutie[elem_solutie]=destinatie;
            elem_solutie++;
            destinatie=parinte[destinatie];
        }
         solutie[elem_solutie++]=c;
         cout<<"Cautare in latime: ruta de la "<<nume[c]<<"pana la "<<nume[b]<<end
l;
         for(int i=(elem_solutie-1);i>=0;i--)
            cout<<nume[solutie[i]]<<"";
}
int main(){
         int start=8;
         int stop=8;
         for(int i=0;i<20;i++)
            for(int j=0;j<20;j++)
                        a[i][j]=0;

         a[0][1]=1;
         a[0][4]=1;
```

```
58.           a[0][3]=1;
59.           a[1][2]=1;
60.           a[4][5]=1;
61.           a[3][8]=1;
62.           a[3][9]=1;
63.           a[5][6]=1;
64.           a[2][3]=1;
65.           a[6][7]=1;
66.           a[7][11]=1;
67.           a[9][10]=1;
68.           a[8][16]=1;
69.           a[10][11]=1;
70.           a[16][17]=1;
71.           a[15][16]=1;
72.           a[14][15]=1;
73.           a[13][14]=1;
74.           a[12][13]=1;
75.           a[15][18]=1;
76.           a[18][19]=1;
77.           a[9][11]=1;
78.           a[10][16]=1;
79.
80.          for(int i=0;i<20;i++)
81.              for(int j=0;j<20;j++)
82.              if(a[i][j]==1)
83.                    a[i][j]=1;
84.          functie(start,stop);
85.
86.
87. }
```