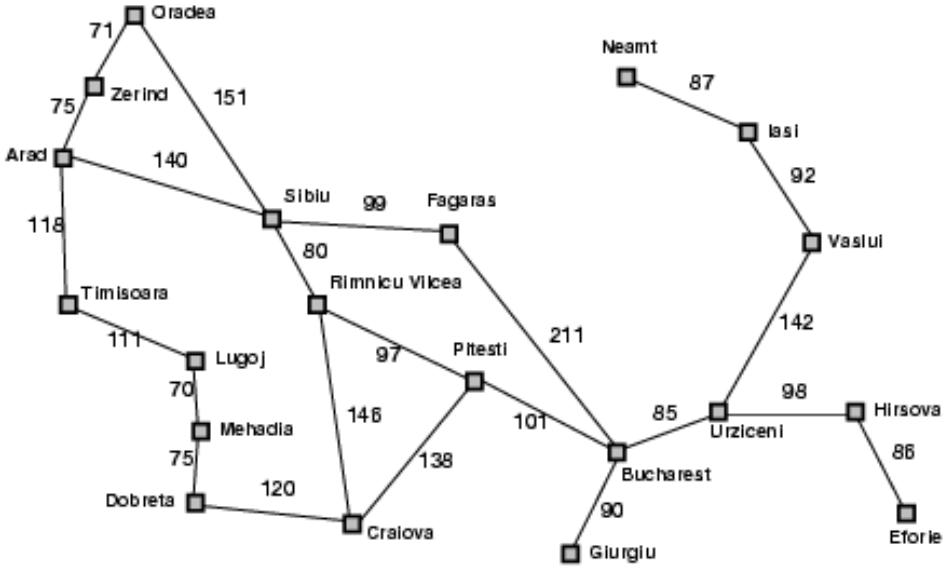
Căutare Greedy



Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Pe lângă informațiile anterioare de la tema L007, avem date cu privire la aproximarea distanței de la fiecare nod la nodul țintă, București, în partea dreaptă a hărții. Folosiți aceste date pentru a aplica o căutare Greedy pentru a găsi ruta de la orice oraș către București.

Instrucțiuni legate de căutarea Greedy:

1. Se introduce încă un tablou care să conțină aproximările (notate în general cu *h*) de la fiecare nod către București.
2. În tabloul *noduri* vom ordona elementele în funcție de aproximarea *h*. Așadar, un nod *i* apare înaintea lui *j* în *noduri* dacă *h*[*noduri*[*i*]] < *h*[*noduri*[*j*]].

Soluție

Iată una din soluțiile primite la care am făcut mici modificări.

Soluție

Observații: programul conține toate căutările de până acum. La Greedy, se adaugă nodurile noi la final și apoi se face o ordonare a intregii liste după h.

```
1.  #include<iostream>
2.  #include<conio.h>
3.  #include<string>
4.
5.  using namespace std;
6.
```

```
7.  string nume[20] = { "Oradea","Zerind","Arad","Timisoara","Lugoj","Mehadia","Drobet
    a","Craiova","Sibiu","Ramnicu Valcea","Fagaras","Pitesti","Bucuresti","Giurgiu","U
    rziceni","Hrisova","Eforie","Vaslui","Iasi","Neamt" };
8.
9.  int distante[20][20], trafic[20][20], h[20], i, j;
10. float timpLatime = 0, timpAdancime = 0, timpUniform = 0, timpAdancimeLim = 0, timp
    AdancimeIter = 0, timpGreedy = 0, costLatime, costAdancime, costUniform, costAdanc
    imeLim, costAdancimeIter, costGreedy;
11. int lim1 = 1;
12.
13. void conversieTimp(float timp) {
14.     if (timp != (int)timp) {
15.         if ((int)timp > 0)
16.         {
17.             cout << (int)timp << " h ";
18.         }
19.         float minute = timp - (int)timp;
20.         minute = minute * 60;
21.         if (minute != (int)minute)
22.         {
23.             if ((int)minute > 0)
24.             {
25.                 cout << (int)minute << " min ";
26.             }
27.             float secunde = minute - (int)minute;
28.             secunde = secunde * 60;
29.             if ((int)secunde > 0)
30.             {
31.                 cout << (int)secunde << " sec";
32.             }
33.         }
34.         else
35.             cout << minute << " min";
36.     }
37.     else
38.         cout << timp << " h";
39. }
40.
41. void cautareLatime(int start, int stop, float& timpLatime, float& costLatime) {
42.     int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas = 0, solutie[2
    0], nrSol = 0, final, sume[20];
43.     final = stop;
44.     for (i = 0; i < 20; i++)
45.     {
46.         viz[i] = 0;
47.     }
48.
49.     for (i = 0; i < 20; i++)
50.     {
51.         sume[i] = 0;
52.     }
53.
54.     noduri[nrNod++] = start;
55.     viz[start] = 1;
56.
57.     while ((gasit == 0) && (nrNod > 0))
58.     {
59.         int nod = noduri[0];
60.         int sum = sume[noduri[0]];
61.
62.         cout << "Pasul " << ++pas << ": ";
63.         for (i = 0; i < nrNod; i++)
64.         {
65.             cout << nume[noduri[i]] << " [" << sume[noduri[i]] << "] ";
66.         }
67.         cout << endl;
```

```
68.         for (i = 0; i < nrNod - 1; i++)
69.         {
70.             noduri[i] = noduri[i + 1];
71.         }
72.         nrNod--;
73.         if (nod == stop)
74.         {
75.             gasit = 1;
76.         }
77.         else
78.         {
79.             for (i = 0; i < 20; i++)
80.             {
81.                 if ((distante[nod][i] != 0) && (viz[i] == 0))
82.                 {
83.                     noduri[nrNod++] = i;
84.                     sume[i] = sum + distante[nod][i];
85.                     viz[i] = 1;
86.                     parinte[i] = nod;
87.                 }
88.             }
89.         }
90.     }
91.
92.     float viteza;
93.     while (final != start)
94.     {
95.         float nrMasiniPeKm = (float)trafic[final][parinte[final]] / (float)distant
    e[final][parinte[final]];
96.         if (nrMasiniPeKm >= 10)
97.         {
98.             viteza = 100 * (10 / nrMasiniPeKm);
99.         }
100.                else
101.                {
102.                    viteza = 100;
103.                }
104.                timpLatime = timpLatime + (float)distante[final][parinte[final]] /
    viteza;
105.                solutie[nrSol++] = final;
106.                final = parinte[final];
107.            }
108.            solutie[nrSol++] = start;
109.            cout << "Cautati drum de la " << nume[start] << " la " << nume[stop] <<
     "." << endl;
110.
111.            for (i = nrSol - 1; i >= 0; i--)
112.            {
113.                cout << nume[solutie[i]] << " ";
114.            }
115.            cout << endl;
116.            cout << "Costul drumului este: " << sume[stop] << " km";
117.            cout << endl;
118.            costLatime = sume[stop];
119.            cout << "Timpul drumului este: ";
120.            conversieTimp(timpLatime);
121.        }
122.
123.        void cautareAdancime(int start, int stop, float& timpAdancime, float& costA
    dancime) {
124.            int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas = 0, so
    lutie[20], nrSol = 0, final, sume[20];
125.            final = stop;
126.            for (i = 0; i < 20; i++)
127.            {
128.                viz[i] = 0;
```

```
129.            }
130.
131.            for (i = 0; i < 20; i++)
132.            {
133.                sume[i] = 0;
134.            }
135.
136.            noduri[nrNod++] = start;
137.            viz[start] = 1;
138.
139.            while ((gasit == 0) && (nrNod > 0))
140.            {
141.                int nod = noduri[0];
142.                int sum = sume[noduri[0]];
143.                cout << "Pasul " << ++pas << ": ";
144.                for (i = 0; i < nrNod; i++)
145.                {
146.                    cout << nume[noduri[i]] << " [" << sume[noduri[i]] << "] ";
147.                }
148.                cout << endl;
149.                for (i = 0; i < nrNod - 1; i++)
150.                {
151.                    noduri[i] = noduri[i + 1];
152.                }
153.                nrNod--;
154.                if (nod == stop)
155.                {
156.                    gasit = 1;
157.                }
158.                else
159.                {
160.                    for (i = 0; i < 20; i++)
161.                    {
162.                        if ((distante[nod][i] != 0) && (viz[i] == 0))
163.                        {
164.                            for (j = nrNod - 1; j >= 0; j--)
165.                            {
166.                                noduri[j + 1] = noduri[j];
167.                            }
168.                            noduri[0] = i;
169.                            nrNod++;
170.                            sume[i] = sum + distante[nod][i];
171.                            viz[i] = 1;
172.                            parinte[i] = nod;
173.                        }
174.                    }
175.                }
176.            }
177.
178.            float viteza;
179.            while (final != start)
180.            {
181.                float nrMasiniPeKm = (float)trafic[final][parinte[final]] / (float)
    distante[final][parinte[final]];
182.                if (nrMasiniPeKm >= 10)
183.                {
184.                    viteza = 100 * (10 / nrMasiniPeKm);
185.                }
186.                else
187.                {
188.                    viteza = 100;
189.                }
190.                timpAdancime = timpAdancime + (float)distante[final][parinte[final]
    ] / viteza;
191.                solutie[nrSol++] = final;
192.                final = parinte[final];
```

```
193.            }
194.            solutie[nrSol++] = start;
195.            cout << "Cautati drum de la " << nume[start] << " la " << nume[stop] <<
    "." << endl;
196.
197.            for (i = nrSol - 1; i >= 0; i--)
198.            {
199.                cout << nume[solutie[i]] << " ";
200.            }
201.            cout << endl;
202.            cout << "Costul drumului este: " << sume[stop] << " km";
203.            cout << endl;
204.            costAdancime = sume[stop];
205.            cout << "Timpul drumului este: ";
206.            conversieTimp(timpAdancime);
207.        }
208.
209.        void cautareUniform(int start, int stop, float& timpUniform, float& costUni
    form) {
210.            int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas = 0, so
    lutie[20], nrSol = 0, final, sume[20];
211.            final = stop;
212.            for (i = 0; i < 20; i++)
213.            {
214.                viz[i] = 0;
215.            }
216.
217.            for (i = 0; i < 20; i++)
218.            {
219.                sume[i] = 0;
220.            }
221.
222.            noduri[nrNod++] = start;
223.            viz[start] = 1;
224.
225.            while ((gasit == 0) && (nrNod > 0))
226.            {
227.                int nod = noduri[0];
228.                int sum = sume[noduri[0]];
229.                cout << "Pasul " << ++pas << ": ";
230.                for (i = 0; i < nrNod; i++)
231.                {
232.                    cout << nume[noduri[i]] << " [" << sume[noduri[i]] << "] ";
233.                }
234.                cout << endl;
235.                for (i = 0; i < nrNod - 1; i++)
236.                {
237.                    noduri[i] = noduri[i + 1];
238.                }
239.                nrNod--;
240.                if (nod == stop)
241.                {
242.                    gasit = 1;
243.                }
244.                else
245.                {
246.                    for (i = 0; i < 20; i++)
247.                    {
248.                        if ((distante[nod][i] != 0) && (viz[i] == 0) || ((distante[
    nod][i] != 0) && (viz[i] == 1) && (distante[nod][i] > 0) && (distante[nod][i] + su
    m < sume[i])))
249.                        {
250.                            noduri[nrNod++] = i;
251.                            sume[i] = sum + distante[nod][i];
252.                            viz[i] = 1;
253.                            parinte[i] = nod;
```

```
254.                         }
255.                     }
256.                     for (i = 0; i < nrNod - 1; i++)
257.                     {
258.                         for (j = i + 1; j < nrNod; j++)
259.                         {
260.                             if (sume[noduri[i]] > sume[noduri[j]])
261.                             {
262.                                 int aux = noduri[i];
263.                                 noduri[i] = noduri[j];
264.                                 noduri[j] = aux;
265.                             }
266.                         }
267.                     }
268.                 }
269.             }
270.
271.             float viteza;
272.             while (final != start)
273.             {
274.                 float nrMasiniPeKm = (float)trafic[final][parinte[final]] / (float)
    distante[final][parinte[final]];
275.                 if (nrMasiniPeKm >= 10)
276.                 {
277.                     viteza = 100 * (10 / nrMasiniPeKm);
278.                 }
279.                 else
280.                 {
281.                     viteza = 100;
282.                 }
283.                 timpUniform = timpUniform + (float)distante[final][parinte[final]]
    / viteza;
284.                 solutie[nrSol++] = final;
285.                 final = parinte[final];
286.             }
287.             solutie[nrSol++] = start;
288.             cout << "Cautati drum de la " << nume[start] << " la " << nume[stop] <<
     "." << endl;
289.
290.             for (i = nrSol - 1; i >= 0; i--)
291.             {
292.                 cout << nume[solutie[i]] << " ";
293.             }
294.             cout << endl;
295.             cout << "Costul drumului este: " << sume[stop] << " km";
296.             cout << endl;
297.             costUniform = sume[stop];
298.             cout << "Timpul drumului este: ";
299.             conversieTimp(timpUniform);
300.         }
301.
302.         void cautareAdancimeLim(int start, int stop, float& timpAdancimeLim, float&
     costAdancimeLim, int lim) {
303.             int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas = 0, so
    lutie[20], nrSol = 0, final, sume[20];
304.             final = stop;
305.             for (i = 0; i < 20; i++)
306.             {
307.                 viz[i] = 0;
308.             }
309.
310.             for (i = 0; i < 20; i++)
311.             {
312.                 sume[i] = 0;
313.             }
314.
```

```
315.            int adancime[20];
316.            for (i = 0; i < 20; i++)
317.            {
318.                adancime[i] = 0;
319.            }
320.
321.            noduri[nrNod++] = start;
322.            viz[start] = 1;
323.
324.            int nod;
325.            while ((gasit == 0) && (nrNod > 0))
326.            {
327.                nod = noduri[0];
328.                int sum = sume[noduri[0]];
329.                cout << "Pasul " << ++pas << ": ";
330.                for (i = 0; i < nrNod; i++)
331.                {
332.                    cout << nume[noduri[i]] << " [" << sume[noduri[i]] << "] ";
333.                }
334.                cout << endl;
335.                for (i = 0; i < nrNod - 1; i++)
336.                {
337.                    noduri[i] = noduri[i + 1];
338.                }
339.                nrNod--;
340.                if (nod == stop)
341.                {
342.                    gasit = 1;
343.                }
344.                else
345.                {
346.                    for (i = 0; i < 20; i++)
347.                    {
348.                        if ((distante[nod][i] != 0) && (viz[i] == 0) && (adancime[n
    od] + 1 <= lim))
349.                        {
350.                            parinte[i] = nod;
351.                            adancime[i] = adancime[parinte[i]] + 1;
352.                            for (j = nrNod - 1; j >= 0; j--)
353.                            {
354.                                noduri[j + 1] = noduri[j];
355.                            }
356.                            noduri[0] = i;
357.                            nrNod++;
358.                            sume[i] = sum + distante[nod][i];
359.                            viz[i] = 1;
360.
361.                        }
362.                    }
363.                }
364.            }
365.
366.            if (nod != stop)
367.            {
368.                cout << "Nu exista solutie.";
369.            }
370.            else
371.            {
372.                float viteza;
373.                while (final != start)
374.                {
375.                    float nrMasiniPeKm = (float)trafic[final][parinte[final]] / (fl
    oat)distante[final][parinte[final]];
376.                    if (nrMasiniPeKm >= 10)
377.                    {
378.                        viteza = 100 * (10 / nrMasiniPeKm);
```

```
379.                    }
380.                    else
381.                    {
382.                        viteza = 100;
383.                    }
384.                    timpAdancimeLim = timpAdancimeLim + (float)distante[final][pari
    nte[final]] / viteza;
385.                    solutie[nrSol++] = final;
386.                    final = parinte[final];
387.                }
388.                solutie[nrSol++] = start;
389.                cout << "Cautati drum de la " << nume[start] << " la " << nume[stop
    ] << "." << endl;
390.
391.                for (i = nrSol - 1; i >= 0; i--)
392.                {
393.                    cout << nume[solutie[i]] << " ";
394.                }
395.                cout << endl;
396.                cout << "Costul drumului este: " << sume[stop] << " km";
397.                cout << endl;
398.                costAdancimeLim = sume[stop];
399.                cout << "Timpul drumului este: ";
400.                conversieTimp(timpAdancimeLim);
401.            }
402.        }
403.
404.        void cautareAdancimeIter(int start, int stop, float& timpAdancimeIter, floa
    t& costAdancimeIter) {
405.            int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas = 0, so
    lutie[20], nrSol = 0, final, sume[20];
406.            final = stop;
407.            for (i = 0; i < 20; i++)
408.            {
409.                viz[i] = 0;
410.            }
411.
412.            for (i = 0; i < 20; i++)
413.            {
414.                sume[i] = 0;
415.            }
416.
417.            int adancime[20];
418.            for (i = 0; i < 20; i++)
419.            {
420.                adancime[i] = 0;
421.            }
422.
423.            noduri[nrNod++] = start;
424.            viz[start] = 1;
425.
426.            int nod;
427.            while ((gasit == 0) && (nrNod > 0))
428.            {
429.                nod = noduri[0];
430.                int sum = sume[noduri[0]];
431.                cout << "Pasul " << ++pas << ": ";
432.                for (i = 0; i < nrNod; i++)
433.                {
434.                    cout << nume[noduri[i]] << " [" << sume[noduri[i]] << "] ";
435.                }
436.                cout << endl;
437.                for (i = 0; i < nrNod - 1; i++)
438.                {
439.                    noduri[i] = noduri[i + 1];
440.                }
```

```
441.                nrNod--;
442.                if (nod == stop)
443.                {
444.                    gasit = 1;
445.                }
446.                else
447.                {
448.                    for (i = 0; i < 20; i++)
449.                    {
450.                        if ((distante[nod][i] != 0) && (viz[i] == 0) && (adancime[n
    od] + 1 <= lim1))
451.                        {
452.                            parinte[i] = nod;
453.                            adancime[i] = adancime[parinte[i]] + 1;
454.                            for (j = nrNod - 1; j >= 0; j--)
455.                            {
456.                                noduri[j + 1] = noduri[j];
457.                            }
458.                            noduri[0] = i;
459.                            nrNod++;
460.                            sume[i] = sum + distante[nod][i];
461.                            viz[i] = 1;
462.
463.                        }
464.                    }
465.                }
466.            }
467.
468.            if (nod != stop)
469.            {
470.                cout << "Nu exista solutie cu adancimea " << lim1 << endl;
471.                lim1++;
472.                cautareAdancimeIter(start, stop, timpAdancimeIter, costAdancimeIter
    );
473.            }
474.            else
475.            {
476.                float viteza;
477.                while (final != start)
478.                {
479.                    float nrMasiniPeKm = (float)trafic[final][parinte[final]] / (fl
    oat)distante[final][parinte[final]];
480.                    if (nrMasiniPeKm >= 10)
481.                    {
482.                        viteza = 100 * (10 / nrMasiniPeKm);
483.                    }
484.                    else
485.                    {
486.                        viteza = 100;
487.                    }
488.                    timpAdancimeIter = timpAdancimeIter + (float)distante[final][pa
    rinte[final]] / viteza;
489.                    solutie[nrSol++] = final;
490.                    final = parinte[final];
491.                }
492.                solutie[nrSol++] = start;
493.                cout << "Cautati drum de la " << nume[start] << " la " << nume[stop
    ] << "." << endl;
494.
495.                for (i = nrSol - 1; i >= 0; i--)
496.                {
497.                    cout << nume[solutie[i]] << " ";
498.                }
499.                cout << endl;
500.                cout << "Adancimea cea mai mica la care se gaseste o solutie este:
    " << lim1;
```

```
501.                cout << endl;
502.                cout << "Costul drumului este: " << sume[stop] << " km";
503.                cout << endl;
504.                costAdancimeIter = sume[stop];
505.                cout << "Timpul drumului este: ";
506.                conversieTimp(timpAdancimeIter);
507.            }
508.        }
509.
510.        void cautareGreedy(int start, int stop, float& timpGreddy, float& costGreed
    y) {
511.            int viz[20], noduri[20], nrNod = 0, parinte[20], gasit = 0, pas = 0, so
    lutie[20], nrSol = 0, final, sume[20];
512.            final = stop;
513.            for (i = 0; i < 20; i++)
514.            {
515.                viz[i] = 0;
516.            }
517.
518.            for (i = 0; i < 20; i++)
519.            {
520.                sume[i] = 0;
521.            }
522.
523.            noduri[nrNod++] = start;
524.            viz[start] = 1;
525.
526.            while ((gasit == 0) && (nrNod > 0))
527.            {
528.                int nod = noduri[0];
529.                int sum = sume[noduri[0]];
530.
531.                cout << "Pasul " << ++pas << ": ";
532.                for (i = 0; i < nrNod; i++)
533.                {
534.                    cout << nume[noduri[i]] << " [" << sume[noduri[i]] << "] ";
535.                }
536.                cout << endl;
537.                for (i = 0; i < nrNod - 1; i++)
538.                {
539.                    noduri[i] = noduri[i + 1];
540.                }
541.                nrNod--;
542.                if (nod == stop)
543.                {
544.                    gasit = 1;
545.                }
546.                else
547.                {
548.                    for (i = 0; i < 20; i++)
549.                    {
550.                        if ((distante[nod][i] != 0) && (viz[i] == 0))
551.                        {
552.                            noduri[nrNod++] = i;
553.                            sume[i] = sum + distante[nod][i];
554.                            viz[i] = 1;
555.                            parinte[i] = nod;
556.                        }
557.                    }
558.                    for (i = 0; i < nrNod - 1; i++)
559.                    {
560.                        for (j = i + 1; j < nrNod; j++)
561.                        {
562.                            if (h[noduri[i]] > h[noduri[j]])
563.                            {
564.                                int aux = noduri[i];
```

```
565.                                noduri[i] = noduri[j];
566.                                noduri[j] = aux;
567.                            }
568.                        }
569.                    }
570.                }
571.            }
572.
573.            float viteza;
574.            while (final != start)
575.            {
576.                float nrMasiniPeKm = (float)trafic[final][parinte[final]] / (float)
    distante[final][parinte[final]];
577.                if (nrMasiniPeKm >= 10)
578.                {
579.                    viteza = 100 * (10 / nrMasiniPeKm);
580.                }
581.                else
582.                {
583.                    viteza = 100;
584.                }
585.                timpGreedy = timpGreedy + (float)distante[final][parinte[final]] /
    viteza;
586.                solutie[nrSol++] = final;
587.                final = parinte[final];
588.            }
589.            solutie[nrSol++] = start;
590.            cout << "Cautati drum de la " << nume[start] << " la " << nume[stop] <<
     "." << endl;
591.
592.            for (i = nrSol - 1; i >= 0; i--)
593.            {
594.                cout << nume[solutie[i]] << " ";
595.            }
596.            cout << endl;
597.            cout << "Costul drumului este: " << sume[stop] << " km";
598.            cout << endl;
599.            costGreedy = sume[stop];
600.            cout << "Timpul drumului este: ";
601.            conversieTimp(timpGreedy);
602.        }
603.
604.        int main() {
605.
606.            int start = 3, stop = 12, lim = 5;
607.
608.            for (i = 0; i < 20; i++)
609.            {
610.                for (j = 0; j < 20; j++)
611.                {
612.                    distante[i][j] = 0;
613.                    trafic[i][j] = 0;
614.                }
615.            }
616.
617.            distante[0][1] = 71;
618.            distante[0][8] = 151;
619.            distante[1][2] = 75;
620.            distante[2][3] = 118;
621.            distante[2][8] = 140;
622.            distante[3][4] = 111;
623.            distante[4][5] = 70;
624.            distante[5][6] = 75;
625.            distante[6][7] = 120;
626.            distante[7][9] = 146;
627.            distante[7][11] = 138;
```

```
628.            distante[8][9] = 80;
629.            distante[8][10] = 99;
630.            distante[9][11] = 97;
631.            distante[10][12] = 211;
632.            distante[11][12] = 101;
633.            distante[12][13] = 90;
634.            distante[12][14] = 85;
635.            distante[14][15] = 98;
636.            distante[14][17] = 142;
637.            distante[15][16] = 86;
638.            distante[17][18] = 92;
639.            distante[18][19] = 87;
640.
641.            trafic[0][1] = 700;
642.            trafic[0][8] = 2500;
643.            trafic[1][2] = 800;
644.            trafic[2][3] = 1100;
645.            trafic[2][8] = 1200;
646.            trafic[3][4] = 1000;
647.            trafic[4][5] = 400;
648.            trafic[5][6] = 600;
649.            trafic[6][7] = 900;
650.            trafic[7][9] = 1300;
651.            trafic[7][11] = 1600;
652.            trafic[8][9] = 800;
653.            trafic[8][10] = 900;
654.            trafic[9][11] = 1300;
655.            trafic[10][12] = 1200;
656.            trafic[11][12] = 1000;
657.            trafic[12][13] = 400;
658.            trafic[12][14] = 700;
659.            trafic[14][15] = 900;
660.            trafic[14][17] = 1400;
661.            trafic[15][16] = 300;
662.            trafic[17][18] = 1200;
663.            trafic[18][19] = 700;
664.
665.            for (i = 0; i < 20; i++)
666.            {
667.                for (j = 0; j < 20; j++)
668.                {
669.                    if (distante[i][j] != 0)
670.                    {
671.                        distante[j][i] = distante[i][j];
672.                        trafic[i][j] = trafic[j][i];
673.                    }
674.                }
675.            }
676.
677.            h[0] = 380;
678.            h[1] = 374;
679.            h[2] = 366;
680.            h[3] = 329;
681.            h[4] = 244;
682.            h[5] = 241;
683.            h[6] = 242;
684.            h[7] = 160;
685.            h[8] = 253;
686.            h[9] = 192;
687.            h[10] = 176;
688.            h[11] = 10;
689.            h[12] = 0;
690.            h[13] = 77;
691.            h[14] = 80;
692.            h[15] = 151;
693.            h[16] = 161;
```

```
694.            h[17] = 199;
695.            h[18] = 226;
696.            h[19] = 234;
697.
698.            cout << "Cautare in latime:" << endl;
699.            cout << "-----------------------------------------------" << endl;
700.            cautareLatime(start, stop, timpLatime, costLatime);
701.            cout << endl;
702.            cout << "-----------------------------------------------" << endl;
703.            cout << "Cautare in adancime" << endl;
704.            cout << "-----------------------------------------------" << endl;
705.            cautareAdancime(start, stop, timpAdancime, costAdancime);
706.            cout << endl;
707.            cout << "-----------------------------------------------" << endl;
708.            cout << "Cautare cu cost uniform" << endl;
709.            cout << "-----------------------------------------------" << endl;
710.            cautareUniform(start, stop, timpUniform, costUniform);
711.            cout << endl;
712.            cout << "-----------------------------------------------" << endl;
713.            cout << "Cautare in adancime cu limita" << endl;
714.            cout << "-----------------------------------------------" << endl;
715.            cautareAdancimeLim(start, stop, timpAdancimeLim, costAdancimeLim, lim);
716.            cout << endl;
717.            cout << "-----------------------------------------------" << endl;
718.            cout << "Cautare in adancime iterativa" << endl;
719.            cout << "-----------------------------------------------" << endl;
720.            cautareAdancimeIter(start, stop, timpAdancimeIter, costAdancimeIter);
721.            cout << endl;
722.            cout << "-----------------------------------------------" << endl;
723.            cout << "Cautare greedy" << endl;
724.            cout << "-----------------------------------------------" << endl;
725.            cautareGreedy(start, stop, timpGreedy, costGreedy);
726.            cout << endl;
727.            cout << "-----------------------------------------------" << endl;
728.            cout << "Rezultate" << endl;
729.            cout << "-----------------------------------------------" << endl;
730.            cout << "Cautare in latime: Drum de " << costLatime << " km parcurs in
    ";
731.            conversieTimp(timpLatime);
732.            cout << endl;
733.            cout << "Cautare in adancime: Drum de " << costAdancime << " km parcurs
     in ";
734.            conversieTimp(timpAdancime);
735.            cout << endl;
736.            cout << "Cautare cu cost uniform: Drum de " << costUniform << " km parc
    urs in ";
737.            conversieTimp(timpUniform);
738.            cout << endl;
739.            if (timpAdancimeLim == 0)
740.            {
741.                cout << "Cautare in adancime cu limita " << lim << ": Nu exista sol
    utie";
742.            }
743.            else
744.            {
745.                cout << "Cautare in adancime cu limita " << lim << ": Drum de " <<
    costAdancimeLim << " km parcurs in ";
746.                conversieTimp(timpAdancimeLim);
747.            }
748.            cout << endl;
749.            cout << "Cautare cu adancime iterativa: Drum de " << costAdancimeIter <
    < " km cu o adancime de " << lim1 << " parcurs in ";
750.            conversieTimp(timpAdancimeIter);
751.            cout << endl;
```

```
752.            cout << "Cautare greedy: Drum de " << costGreedy << " km parcurs in ";
753.            conversieTimp(timpGreedy);
754.
755.            _getch();
756.            return 0;
757.        }
```

Program rulat

```
Cautare greedy
-----------------------------------------
Pasul 1: Timisoara [0]
Pasul 2: Lugoj [111] Arad [118]
Pasul 3: Mehadia [181] Arad [118]
Pasul 4: Drobeta [256] Arad [118]
Pasul 5: Craiova [376] Arad [118]
Pasul 6: Pitesti [514] Ramnicu Valcea [522] Arad [118]
Pasul 7: Bucuresti [615] Ramnicu Valcea [522] Arad [118]
Cautati drum de la Timisoara la Bucuresti.
Timisoara Lugoj Mehadia Drobeta Craiova Pitesti Bucuresti
Costul drumului este: 615 km
Timpul drumului este: 6 h 9 min
-----------------------------------------
Rezultate
-----------------------------------------
Cautare in latime: Drum de 568 km parcurs in 5 h 40 min 47 sec
Cautare in adancime: Drum de 615 km parcurs in 6 h 9 min
Cautare cu cost uniform: Drum de 536 km parcurs in 5 h 21 min 35 sec
Cautare in adancime cu limita 5: Drum de 568 km parcurs in 5 h 40 min 47 sec
Cautare cu adancime iterativa: Drum de 568 km cu o adancime de 4 parcurs in 5 h 40 min 47 sec
Cautare greedy: Drum de 615 km parcurs in 6 h 9 min
```

Soluția 2

Observație: Alternativ, pun doar metoda în care se face ordonarea elementelor din noduri după *h* pe măsură ce sunt adăugate, adică se identifică pe ce poziție *j* trebuie să se pună elementul nou *i*.

```
1.  void cautare_greedy(int start, int stop) {
2.
3.      int viz[20], parinte[20];
4.      int noduri[20];//contine lista de noduri aflate in asteptare
5.      int nrNoduri = 0;//numarul de elemente din noduri
6.      int pas = 0; int h[N];
7.
8.
9.      for (int i = 0; i < 20; i++)
10.      viz[i] = 0;//toate orasele sunt nevizitate
11.     noduri[nrNoduri++] = start; //adaugam la lista noduri orasele de plecare
12.     viz[start] = 1;//marcam orasul de start ca vizitat
13.     int gasit = 0;//solutia nu este gasita deocamdata
14.
15.     while ((gasit == 0) && (nrNoduri > 0)) {
16.      pas++;
17.      int nod = noduri[0]; //stocam pr elem din noduri in var nod
18.
19.      cout << nume[nod] << " ";
20.      cout << "Pass" << pas << " ";
21.      cout << endl;
22.
23.      for (int i = 0; i < nrNoduri - 1; i++)
24.          noduri[i] = noduri[i + 1];//stergem elementul de pe prima pozitie din nod
    uri
```

```
25.      nrNoduri--;
26.      if (nod == stop)
27.          gasit = 1;
28.      else
29.      {
30.          for (int i = 0; i<20; i++)
31.              if ((a[nod][i] !=0) && (viz[i] == 0))
32.              {
33.
34.                  int j = 0;
35.                  while ((j < nrNoduri) && (h[i]<h[noduri[j]]))
36.                      j++;
37.                      for (int q=nrNoduri; q>j; q--)
38.                          noduri[q] = noduri[q -1];
39.                      noduri[j] = i;
40.
41.                      nrNoduri++;
42.                  viz[i] = 1;
43.                  parinte[i] = nod;
44.
45.              }
46.
47.      }
48.
49.     }
50.     int nrSol = 0, dest = stop, solutie[20];
51.     cout << "Cautarea_greedy";
52.     cout << endl;
53.     cout << "Plecam din " << nume[start] << " si dorim sa ajungem in " << nume[des
    t] << " prin " << endl;
54.     while (dest != start) {
55.      solutie[nrSol++] = dest;
56.      dest = parinte[dest];
57.     }
58.     solutie[nrSol++] = start;
59.
60.     for (int i = nrSol - 1; i >= 0; i--)
61.      cout << nume[solutie[i]] << " ";
62. }
```