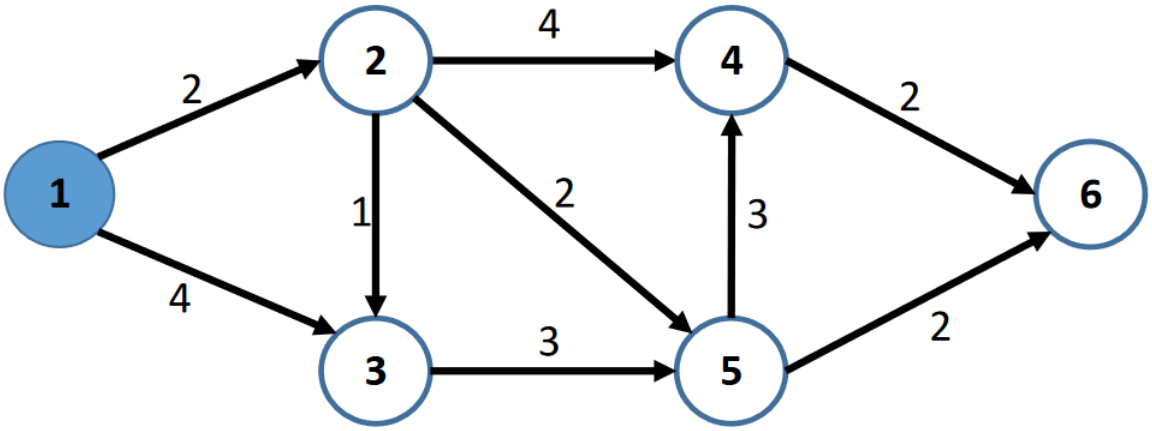
Grafuri - Drumuri minime

Analitic:

Exercitiu: Completati urmatorul tabel pe masura ce aplicati pasii din algoritmul de mai sus pentru diverse grafuri (cel utilizat in laboratorul pecedent, cel de la curs, alte grafuri).



v	vizitat	dist[s,v]	previous
1	F	0	
2	F	∞	
3	F	∞	
4	F	∞	
5	F	∞	
6	F	∞	
…	…	…	

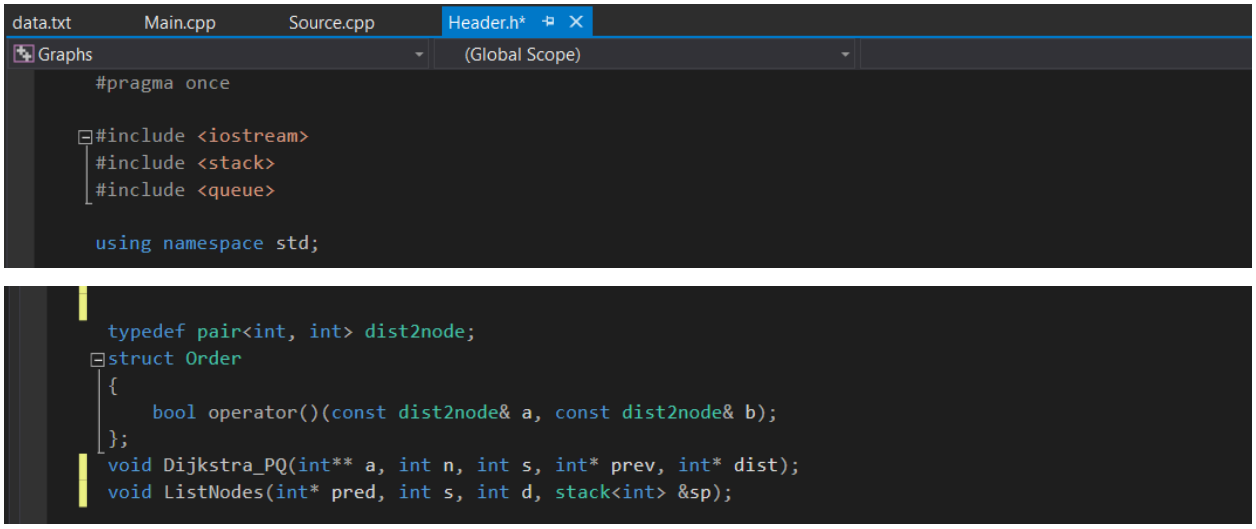
Algoritmul Dijkstra

```
G=(V,E)
Q – set de noduri nevizitate
S – nodul sursa
for each nod v in V
      dist[S,v] := ∞
      adauga v in Q
      prev[v] := undefined
end-for
dist[S,S] := 0
while Q ≠ empty do
      u := nodul din Q cu dist[u] minima
      scoate u din Q
      for each vecin v al lui u din Q //v - vecin nevizitat
            alt :=  dist[S,u] + cost[u,v]
            if alt < dist[S,v] then
                  dist[S,v] := alt
                  prev[v] := u
            end-if
      end-for
end-while
```

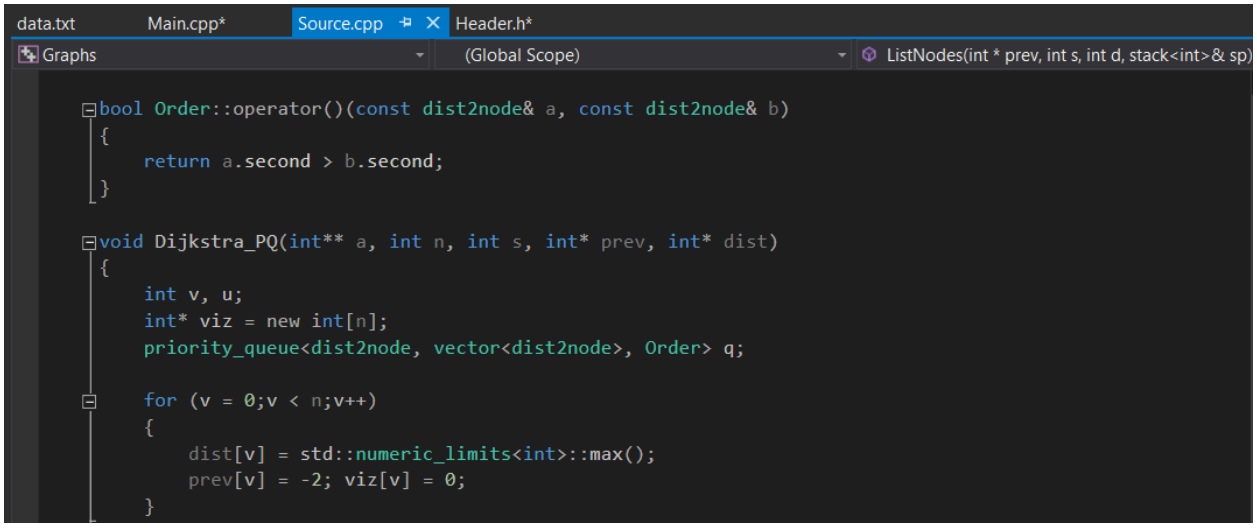
Pentru listarea nodurilor care compun drumul veti folosi:

```
ST - stiva
D  – nodul destinatie
push(ST, D)
u := D
while u ≠ S
      push(ST, prev[u])
      u := prev[u]
end-while
```

Header.h



Functii.cpp



Main.cpp

```
    int* prev;
    int* dist;
    int s, d;
    stack<int> sp;
    prev = new int[n];
    dist = new int[n];
    // ...
    Dijkstra_PQ(a, n, 0, prev, dist);
    cout << endl << "i prev[i] dist[i]\n";
    for (int i = 0;i < n;i++)
    {
        cout << i + 1 << "\t" << prev[i] + 1 << "\t" << dist[i] << endl;
    }
    cout << endl;
    s = 0;
    d = 5;
    ListNodes(prev, s, d, sp);

    cout << "\n Dijkstra's alg: path(" << s + 1 << "; " << d + 1 << "): ";
    while (!sp.empty())
    {
        cout << sp.top()+1 << " ";
        sp.pop();
    }
```

```
C:\WINDOWS\system32\cmd.exe

 Matricea de adiacenta:
0       2       4       0       0       0
0       0       1       4       2       0
0       0       0       0       3       0
0       0       0       0       0       2
0       0       0       3       0       2
0       0       0       0       0       0


i prev[i] dist[i]
1       -1      0
2       1       2
3       2       3
4       2       6
5       2       4
6       5       6


 Dijkstra's alg: path(1; 6): 1 2 5 6

Press any key to continue . . .
```

Algoritmul A*

Algoritmul A*

```
G=(V,E)
S - nodul sursa
D - nodul destinatie
closedSet := {} //set de noduri evaluate
openSet := {S} //set de noduri descoperite, dar neevaluate
inca
for each v in V
     prev[v] := undefined
     g[v] := ∞ //dist[S,v]
     f[v] := ∞ //cost[S,D] prin v, adica f(v)=g(v)+h(v)
end-for
g[S] := 0
f[S] := h[S]
while openSet ≠ empty
     current := nodul v din openSet cu f[v] minim
     if current = D then
          return reconstruct_path(prev, current)
     end-if
     scoate current din openSet
     adauga current la closedSet
     for each vecin v al lui current not in closedSet
          if v not in openSet
               adauga v la openSet
          end-if
          alt_g := g[current] + cost[current,v]
          if alt_g < g[v] then
               prev[v] := current
               g[v] := alt_g
               f[v] := g[v] + h[v]
          end-if
     end-for
end-while
return failure
```

Creati un fisier cu distantele rutiere

Aplicati algoritmul A*