

Lab1:

Graph with 5 vertices and 6 edges:

5 6

0	0	1
0	1	7
1	2	2
2	1	-1
1	3	8
2	3	5

We keep the graph in memory as followed:

```
class DIRECT_GRAPH {  
  
private:  
    int numberOfEdges, numberOfVertices;  
    map <int, vector<int>> edgesIn;  
    map <int, vector<int>> edgesOut;  
    map <pair<int, int>, int> costs;  
    vector <int> vertices;
```

edgesIn: map of edges that enters in a vertice, in our case - edgesIn[0]={0}, edgesIn[1]={0,2}, edgesIn[2]={1}, edgesIn[3]={1,2}

edgesOut: map of edges gets out of a vertice, in our case - edgesOut[0]={0, 1}, edgesIn[1]={2,3}, edgesIn[2]={1,3}

costs: vector that contains the cost for every vertice, in our case - costs[<0,0>]={1}, costs[<0,1>]={7}, costs[<1,2>]={2}, costs[<2,1>]={-1}, costs[<1,3>]={8} , costs[<5,3>]={5}

vertices: list of vertices, from 0 to 5

Lab2:

Graph with 5 vertices and 5 edges

5 5

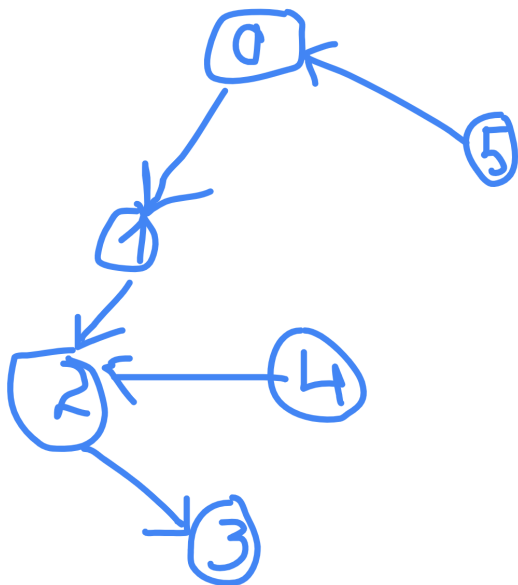
0	1	3
1	2	4
2	3	5
4	2	2
5	0	2

I follow the given documentation:

q - queue with vertices you have to visit

p - vector, where at a given position x you have the predecessor of the vertex x on the shortest path from x to the source

l - vector, contains the shortest path's length from source to the other vertices



source = 3,
target = 0,

	0	1	2	3	4	5
l	3	2	1	0	-	-
p	1	2	3	-	-	-