# DIJKSTRA'S ALGORTIHM

DATA STRUCTURES AND ALGORITHMS PROJECT

STUDENT: VERONICA DAN

APRIL, 2014

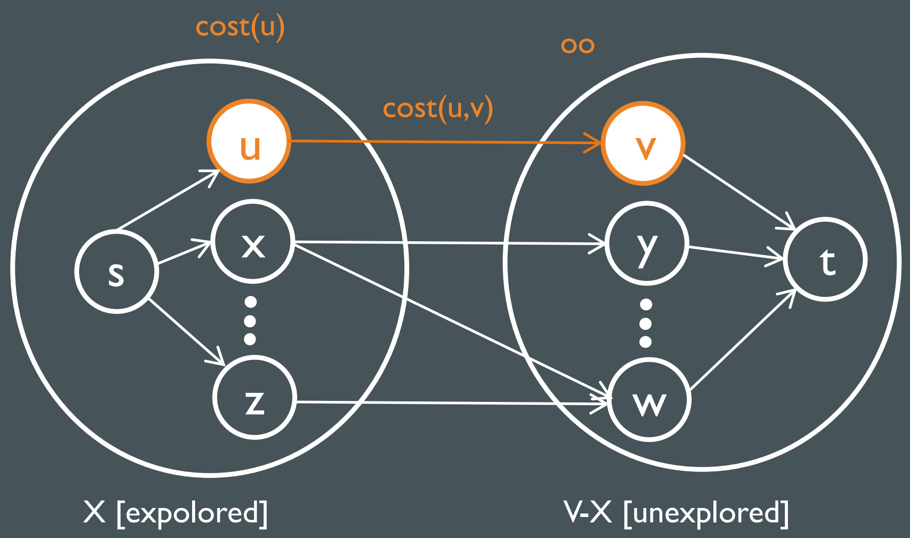
COORDINATOR: CRISTIAN MIHAESCU, PHD.

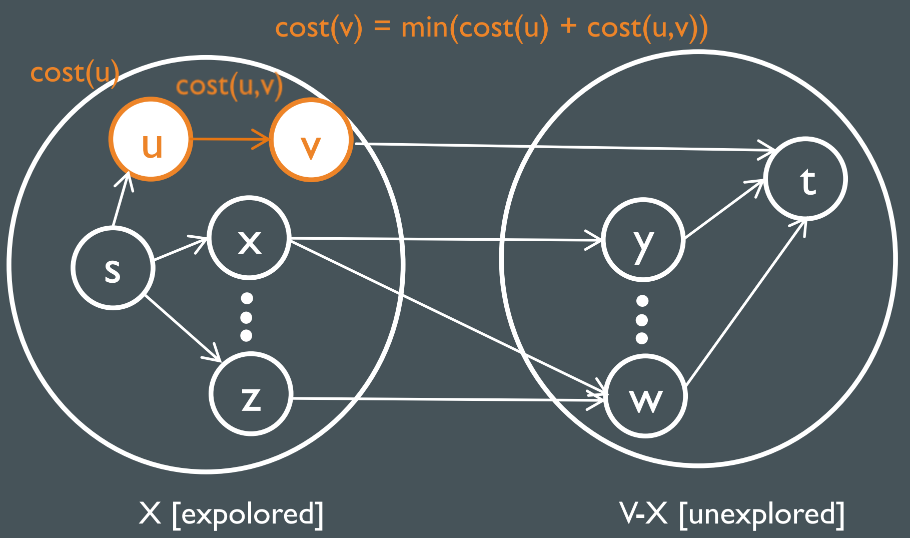
DCTI – IT COMPANIES SEMINARY

Dijkstra's algorithm (named after Edsger Wybe Dijkstra) is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge costs.

It works like this: at every step, a node is moved from V-X to X in conformity to Dijkstra's minimum criterion :

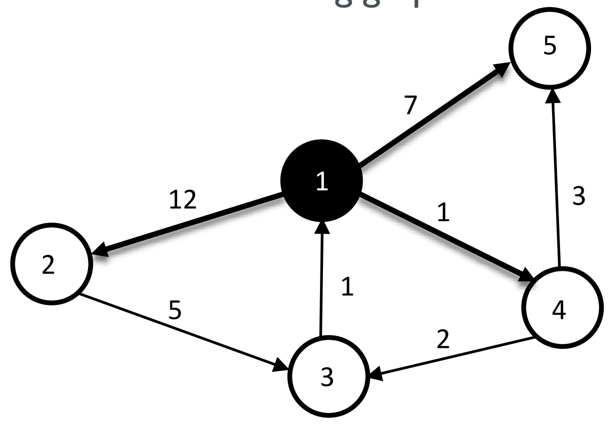
**min( cost(u) + cost(u,v) ) => relax edge (u,v)**





# EXECUTION EXAMPLE

- We consider the following graph:

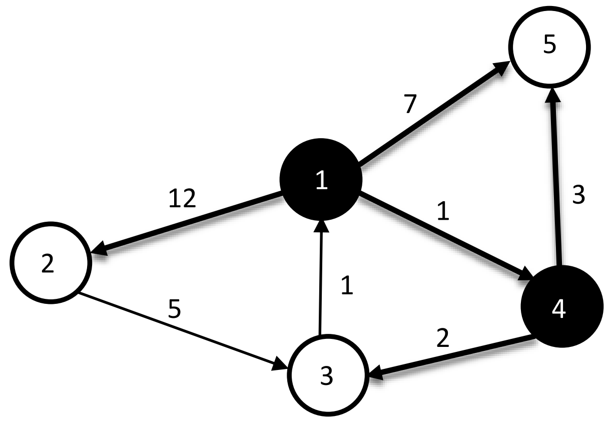


**Step I.**

- We consider node I as source node. From node I we have paths to nodes 2, 4, 5, so we update the distances to this nodes and set node I as visited.
- Next, we select node 4.

| Node | I | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Visited[i] | I | 0 | 0 | 0 | 0 |
| Distances[i] | 0 | I2 | 0 | I | 7 |

**Step II.**

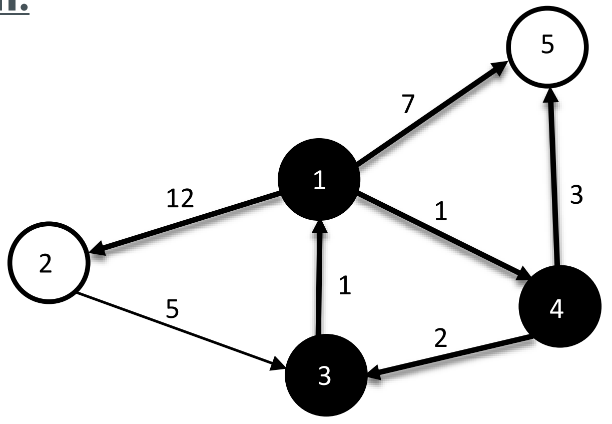


- Node 4 is marked as visited and we notice it has paths to nodes 5 and 3. We can see that for node 7 the path {I, 4, 7} is cheaper (cost= I+3) than the one from node I (cost=7), also now we have a path from the source node to node 3 (cost =I+2) so we update the table conformingly.
- Next, we select node 3.

| Node | I | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Visited[i] | I | 0 | 0 | I | 0 |
| Distances[i] | 0 | I2 | 3 | I | 4 |

## EXECUTION EXAMPLE

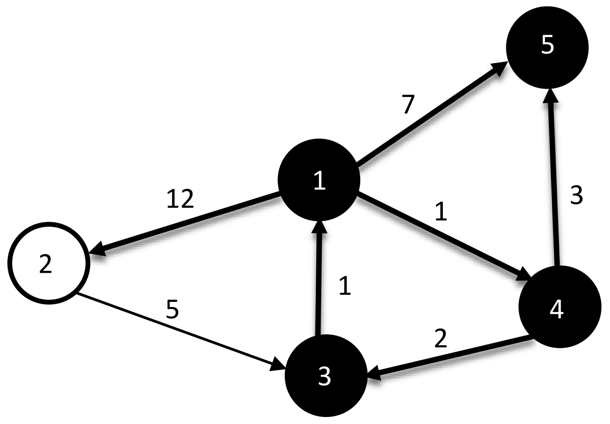
### Step III.



- Node 3 is marked as visited and we see it has a path to node 1. This does not influence any of the distances that were previously computed so we select node 5.

| Node | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Visited[i] | 1 | 0 | 1 | 1 | 0 |
| Distances[i] | 0 | 12 | 3 | 1 | 4 |

### Step IV.

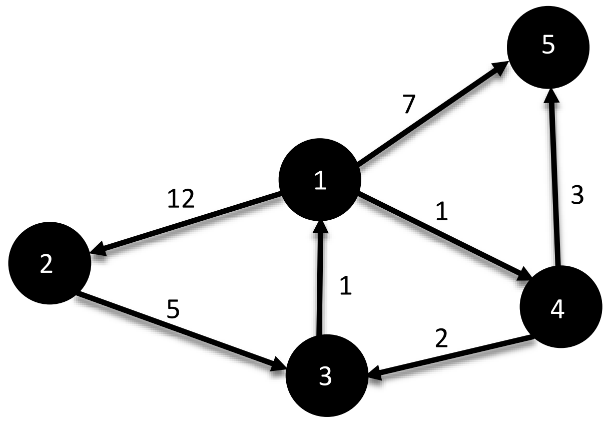


- Node 5 is marked as visited and since it does not have any path to any node, the distances will not be modified.
- Next, we select node 2.

| Node | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Visited[i] | 1 | 0 | 1 | 1 | 1 |
| Distances[i] | 0 | 12 | 3 | 1 | 4 |

## EXECUTION EXAMPLE

**Step V.**



- Node 2 is marked as visited and we see that it has a path to node 3, which does not modify any of the distances obtained so far. Since there are no more unvisited nodes, the algorithm stops.

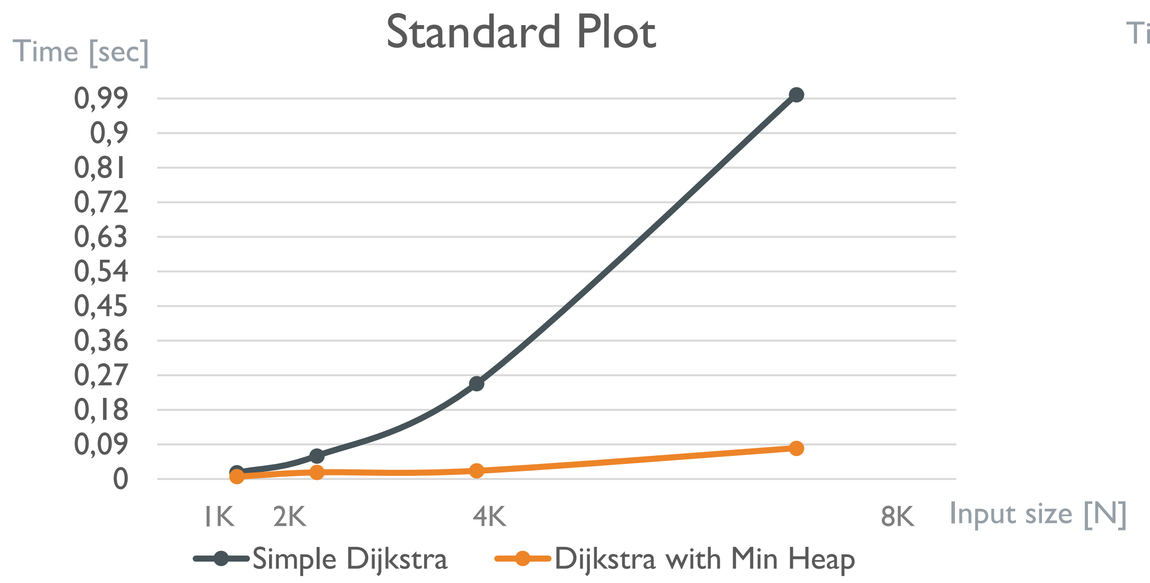
| Node | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Visited[i] | 1 | 1 | 1 | 1 | 1 |
| Distances[i] | 0 | 12 | 3 | 1 | 4 |

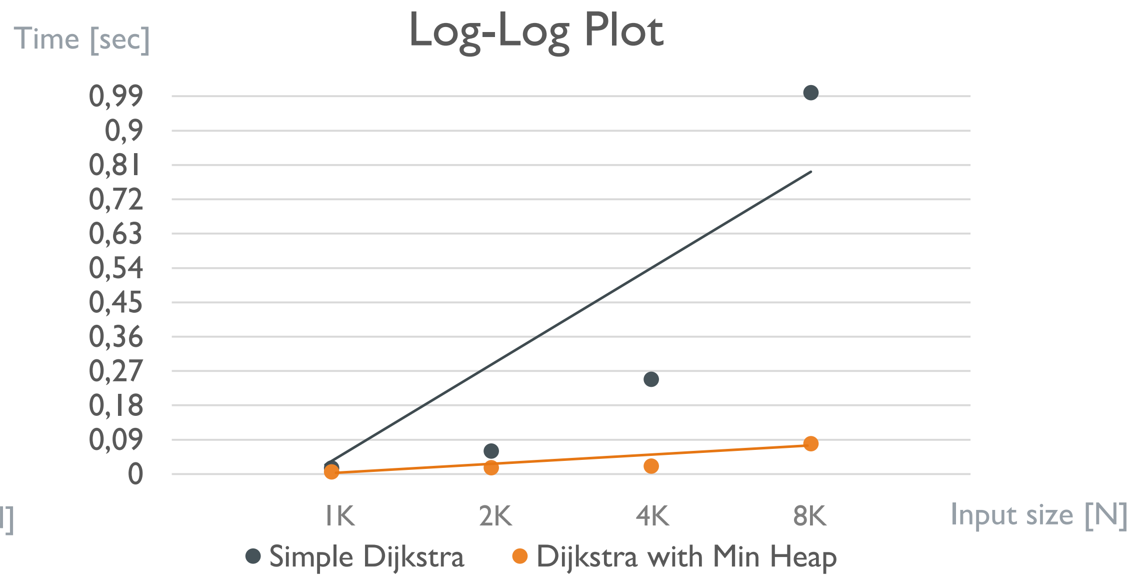
## PSEUDOCODE

```
function Dijkstra (Graph, source):   //using Priority Queue
    distance [source] := 0                      // Initializations
    for each vertex v in Graph:
        if v ≠ source
            distance [v] := infinity            // Unknown distance from
                                                   source to v
            predecessor [v] := undefined        // Predecessor of v
        end if
        PQ.add_with_priority (v, distance[v])
    end for
    while PQ is not empty:                      // The main loop
        u := PQ.extract_min()                   // Remove and return best vertex
        for each neighbor v of u:               // where v has not
                                                   yet been removed from PQ
            alt = distance[u] + length(u, v)
            if distance[u] + length(u, v) < distance[v]
            // Relax the edge (u, v)
                distance[v] := distance[u] + length(u, v)
                predecessor[v] := u
                PQ.decrease_priority (v, distance[u] + length(u, v))
            end if
        end for
    end while
    return distance[]
```

## RESULTS

- In the project, the algorithm is implemented both with a priority queue and without.
- For testing, a brute force method was implemented using the depth first search traversal of the graph to find all the possible paths from source to all the other nodes and then extract the minimum. From the comparison it resulted that the two implementations of the algorithm were succesfull and returned correct results.
- The data analysis results are the following:





## REFERENCES

- Sedgewick, R. and Wayne, K., *Algorithms 4th Edition*, Princeton University
- Burdescu, D. and Mihaescu, C., *Algorthims and Data Structures,* Academica Greifswald
- www.Wikipedia.com

## CODE

https://github.com/mihaescu/ADS/tree/master/Graphs/Distances/Dijkstra%20by%20Dan%20Veronica