Dijkstra's Shortest Path Algorithm Implementation & Visualization

Erdal Sidal Dogan, Alp Gokcek
MEF University
December 27, 2019

***Abstract*—Dijkstra's Shortest Path Algorithm is an algorithm which finds the shortest path between two nodes in a graph. It is widely used and adopted for many different applications, such as computer networks, road maps, social platforms etc. In this project we implemented algorithm in Python Language and visualized the shortest path.**

***Index Terms*—Graph, Node, Vertex, Edge**

I. Introduction

Dijkstra's Algorithm calculates the shortest path from one desired node of the graph to every other nodes. There are varios implementations of the algorithm, main difference between them is how they store the verticies of the graph, denoted with Q. They can be stored in regular arrays, linked lists, adjacency lists or tree structures. Running time of the algorithm can be represented with number of edges $|E|$ and number of verticies $|V|$ using big-o notation. Running time mainly depends on these structures, storing Q in arrays it is $\mathcal{O}(|V|^2)$. In this project we utilized the *priority queue* structure while implementing. It is more efficient way to implement comparing to arrays and adjacency lists etc. The *priority queue* implementation requires $\mathcal{O}((|E| + |V|) \log |V|)$ time in the worst case.

Algorithm 1 Using a priority queue [1]

```
1:  function DIJKSTRA(Graph, source)
2:      for each vertex v in Graph do
3:          dist[v] ← INFINITY
4:          prev[v] ← UNDEFINED
5:          add v to Q
6:      end for
7:      dist[source] ← 0
8:      while Q is not empty do
9:          u ← vertex in Q with min dist[u]
10:         remove u from Q
11:         for each neighbor v of u do
12:             alt ← dist[u] + length(u, v)
13:             if alt < dist[v] then
14:                 dist[v] ← alt
15:                 prev[v] ← u
16:             end if
17:         end for-==
18:     end while
19: return dist[], prev[]
20: end function
```

II. The Program

The program consists a simple and intuitive Graphical User Interface. On the top there are text fields where the user can type de number of nodes, start node and the destination node respectively. Below, there is the path from source to destination and time it took to travel between these nodes. Rest of the application window displays the nodes and emphasizes the shortest path between two given nodes in red color edges.

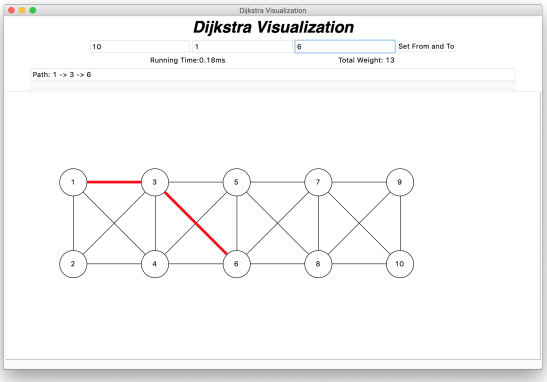
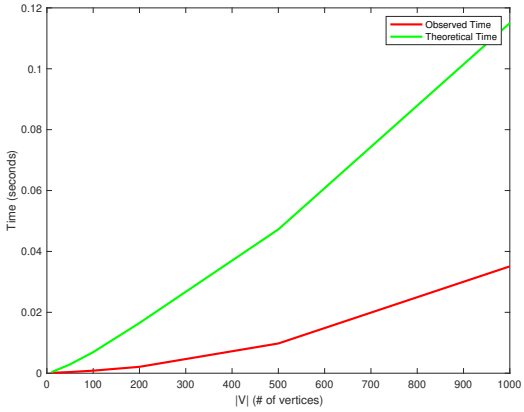


Fig. 1: Main Window of the program for values selected as; N=10, S=1, D=6

III. Time Meausurements



Note that theoretical time (showed in green line) doesn't correspond to any value of time, it is put there to emphasize

the overall similarities such as curvature and trajectory etc. between actual results and calculated results.

REFERENCES

[1] Wikipedia contributors, "Dijkstra's algorithm — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&oldid=931177808, 2019, [Online; accessed 26-December-2019].