Dijkstra's Shortest Path Algorithm Implementation & Visualization

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

***Abstract*—Dijkstra's Shortest Path Algorithm [1] 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

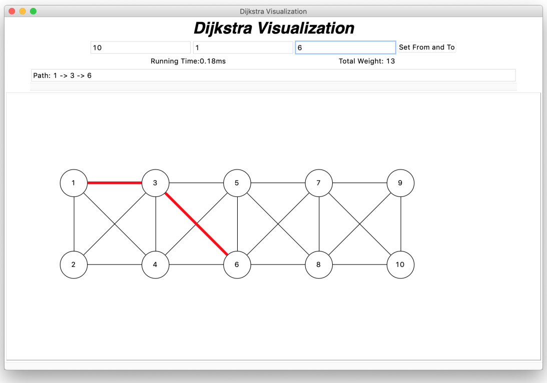
Dijkstra's Algorithm calculates the shortest path from one desired node of the graph to every other nodes. There are varios implementations of the algorithm, main difference between them is how they store the verticies of the graph, denoted with Q. They can be stored in regular arrays, linked lists, adjacency lists or tree structures.[2] 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 [3] — Cost

```
 1: function DIJKSTRA(Graph, source)
 2:     for each vertex v in Graph do                     v + 1
 3:         $dist[v] \leftarrow$ INFINITY                    v
 4:         $prev[v] \leftarrow$ UNDEFINED                   v
 5:         add $v$ to $Q$                                   v
 6:     end for
 7:     $dist[source] \leftarrow 0$                          1
 8:     while $Q$ is not empty do                         v + 1
 9:         $u \leftarrow$ vertex in $Q$ with min $dist[u]$   v
10:         remove $u$ from $Q$                              v
11:         for each neighbor $v$ of $u$ do       $\sum_{i=0}^{v}(v-i+2)$
12:             $alt \leftarrow dist[u] + length(u, v)$   $\sum_{i=0}^{v}(v-i+1)$
13:             if alt < dist[v] then             $\sum_{i=0}^{v}(v-i+2)$
14:                 $dist[v] \leftarrow alt$          $\sum_{i=0}^{v}(v-i+2)$
15:                 $prev[v] \leftarrow u$            $\sum_{i=0}^{v}(v-i+2)$
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.


Dijkstra Visualization

10 | 1 | 6 | Set From and To

Running Time:0.18ms   Total Weight: 13

Path: 1 -> 3 -> 6

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

III. Time Measurements

After the implementation, we measured the time it takes to calculate shortest path from 1 to N in a graph with N number of nodes. We observed a correlation between theory and the practice. Theoritical results are calculated by the $\mathcal{O}((|E| + |V|)\log|V|)$ which stated in *Introduction* section previously. Time it takes to run the program depends on the computer it ran on. During our demonstration we used the same machine with same configuration in order to get most accurate results. For different machines the expectation is to get ratios that are closer to ratio of our results. Results can be seen in detail from the figure below;

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.

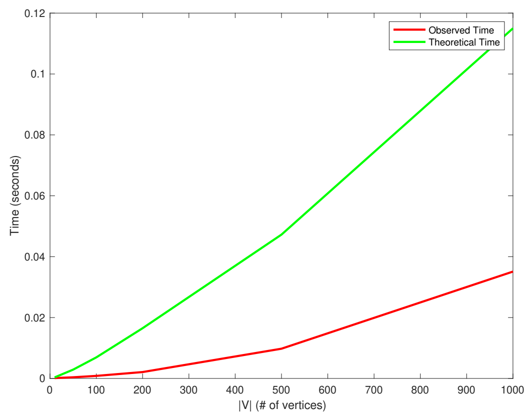


Fig. 2: $|V|$ vs. time (in seconds)

IV. Conclusion & Discussion

In this project we implemented *Dijkstra's Algorithm* using Python. We developed an GUI in order to visualize the graph and the shortest path. After the implementation, we compared our results with the theoritical results. This project helped us to develop a deeper insight about the graphs and the shortest path algorithms.

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: https://doi.org/10.1007/BF01386390

[2] G. Oberhauser and R. Simha, "Fast data structures for shortest path routing: a comparative evaluation," in *Proceedings IEEE International Conference on Communications ICC '95*, vol. 3, June 1995, pp. 1597–1601 vol.3.

[3] 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].