Analysis of Algorithms Shortest Path in a Graph with Dijkstra

Ertuğrul Yılmaz - 041701030

Introduction

In this project we are supposed to implement the one of the known shortest path algorithm called Dijkstra’s. Project is going to be include the comparison between theoretical running time and actual running time of the algorithm. The graphs which going to be examined will have following edge policy according to the project description. Policy is there can be an edge between node i and j if and only if $|i-j| <=3$ and i and j are not equal then $w_{ij} = i+j$. There will be a simple and useful UI(User Interface) to decide the number of node, starting node and destination node. Also, UI includes the visualization of the algorithm.

Explanation of Dijkstra’s Algorithm

Dijkstra’s algorithm is one of the famous shortest path algorithm. Algorithm takes a graph, starting node and destination node to process. First all nodes has infinity shortest-path estimate(d[v]) except the starting node. Algorithm has two min-heap these are Q and S. Q includes all nodes at the beginning. S is going to include all the nodes which was checked by the algorithm. The idea is that algorithm extracts the node v which has minimum d[v] and inserts this node to S then relaxes all the edges which is leaving node v if it is possible. While doing it, algorithm holds the pi(predecessor) values of all nodes. When the heap Q became empty, it finds the path between starting node and destination node by looking at the pi values.

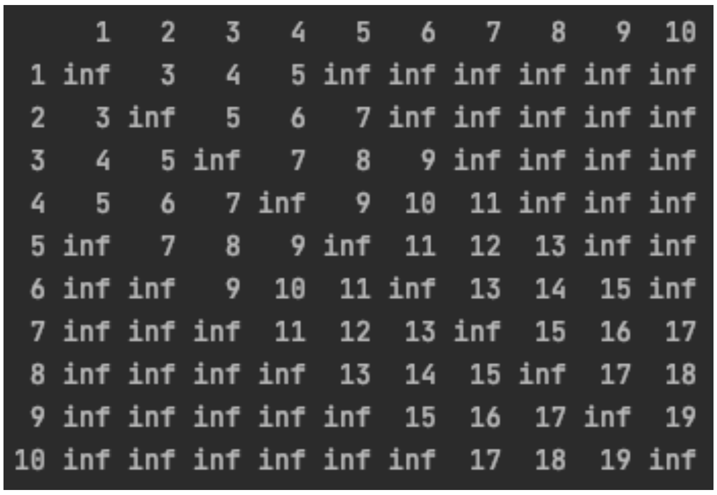
Pseudo Code and Running Time

```
INITIALIZE-SINGLE-SOURCE(V, s)  -> For all vertices O(V)
S←∅
Q ← V[G] ——————————————————————> Min-heap for all vertices O(V)
while Q ≠ ∅ ———————————————————> Check all the vertices O(V)
        do ← EXTRACT-MIN(Q) ————————> Check for the current min vertex O(lgV)
        S ← S ∪ {u}
        for each vertex v ∈ Adj[u]————————> Eventually check all the edges O(E)
                do RELAX(u, v, w)
                Update Q (DECREASE_KEY)—> If it is better path write node O(lgV)
```

Running Time: O(V) + O(V) * O(lgV) + O(E) * O(lgV) = O(ElgV)

Example

When N=10, S=1 and D=6. Shortest path weight estimate in parentheses.



```
    1   2   3   4   5   6   7   8   9  10
 1 inf   3   4   5 inf inf inf inf inf inf
 2   3 inf   5   6   7 inf inf inf inf inf
 3   4   5 inf   7   8   9 inf inf inf inf
 4   5   6   7 inf   9  10  11 inf inf inf
 5 inf   7   8   9 inf  11  12  13 inf inf
 6 inf inf   9  10  11 inf  13  14  15 inf
 7 inf inf inf  11  12  13 inf  15  16  17
 8 inf inf inf inf  13  14  15 inf  17  18
 9 inf inf inf inf inf  15  16  17 inf  19
10 inf inf inf inf inf inf  17  18  19 inf
```

Figure 1: Matrix Representation of N =10.

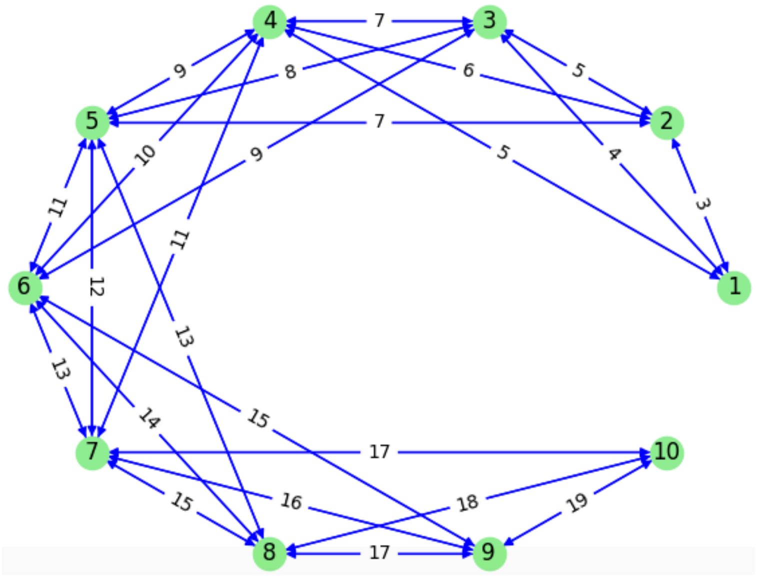


Figure 2: Graph Representation of N =10.

S=<>, Q=<1(0), 2(∞), 3(∞), 4(∞), 5(∞), 6(∞), 7(∞), 8(∞), 9(∞), 10(∞)>

S=<1(0)>, Q=<2(3), 3(4), 4(5), 5(∞), 6(∞), 7(∞), 8(∞), 9(∞), 10(∞)>

S=<1(0), 2(3)>, Q=<3(4), 4(5), 5(10), 6(∞), 7(∞), 8(∞), 9(∞), 10(∞)>

S=<1(0), 2(3), 3(4)>, Q=<4(5), 5(10), 6(13), 7(∞), 8(∞), 9(∞), 10(∞)>

S=<1(0), 2(3), 3(4), 4(5)>, Q=<5(10), 6(13), 7(16), 8(∞), 9(∞), 10(∞)>

S=<1(0), 2(3), 3(4), 4(5), 5(10)>, Q=<6(13), 7(16), 8(23), 9(∞), 10(∞)>

S=<1(0), 2(3), 3(4), 4(5), 5(10), 6(13)>, Q=<7(16), 8(23), 9(28), 10(∞)>

S=<1(0), 2(3), 3(4), 4(5), 5(10), 6(13), 7(16)>, Q=<8(23), 9(28), 10(33)>

S=<1(0), 2(3), 3(4), 4(5), 5(10), 6(13), 7(16), 8(23)>, Q=<9(28), 10(33)>

S=<1(0), 2(3), 3(4), 4(5), 5(10), 6(13), 7(16), 8(23), 9(28)>, Q=<10(33)>

S=<1(0), 2(3), 3(4), 4(5), 5(10), 6(13), 7(16), 8(23), 9(28), 10(33)>, Q=<>

δ(1,6) = 13 = 1 -> 3 -> 6

Code

The definitions of the functions and classes which is used for this project is below:

Main.py

Includes the main functions and UI.

- takeNumberN(): Takes the number of city from an input form.
- runDijkstra(): Runs the Dijkstra algorithm when button pressed.
- getGraphSpecs(): Gets the graph specs before plot.
- printGraph(): Print the graph matrix to console.
- fillGraph(): Fills the graph object according to given graph matrix.
- relax(): Relaxes an edge if it is possible.
- dijkstra(): Runs dijkstras algorithm for given starting node and destination on graph.
- writeToGraph(): Writes graph matrix properties to graph object.

justDijkstraComparison.py

Includes the function and plotting for different number of nodes for a graph and return the running times as a plot.

- dijkstra(): Runs dijkstra algorithm with writing it to graph object.
- dijkstraMultipleDestinations(): Calls dijkstra function for given N number.
- main(): Run dijkstra for N=10, 50, 100, 200, 500, 1000, 2000 and plots the running times.

UI

User interface has two windows called openningWindow and mainWindows.

- openningWindow: User can enter the desired number city or look the comparison of the running times of different N number.

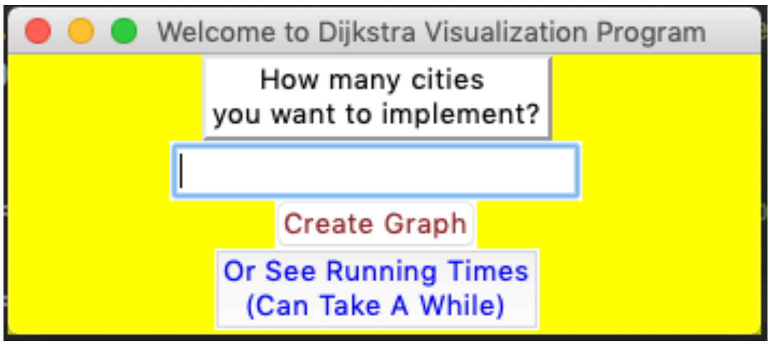
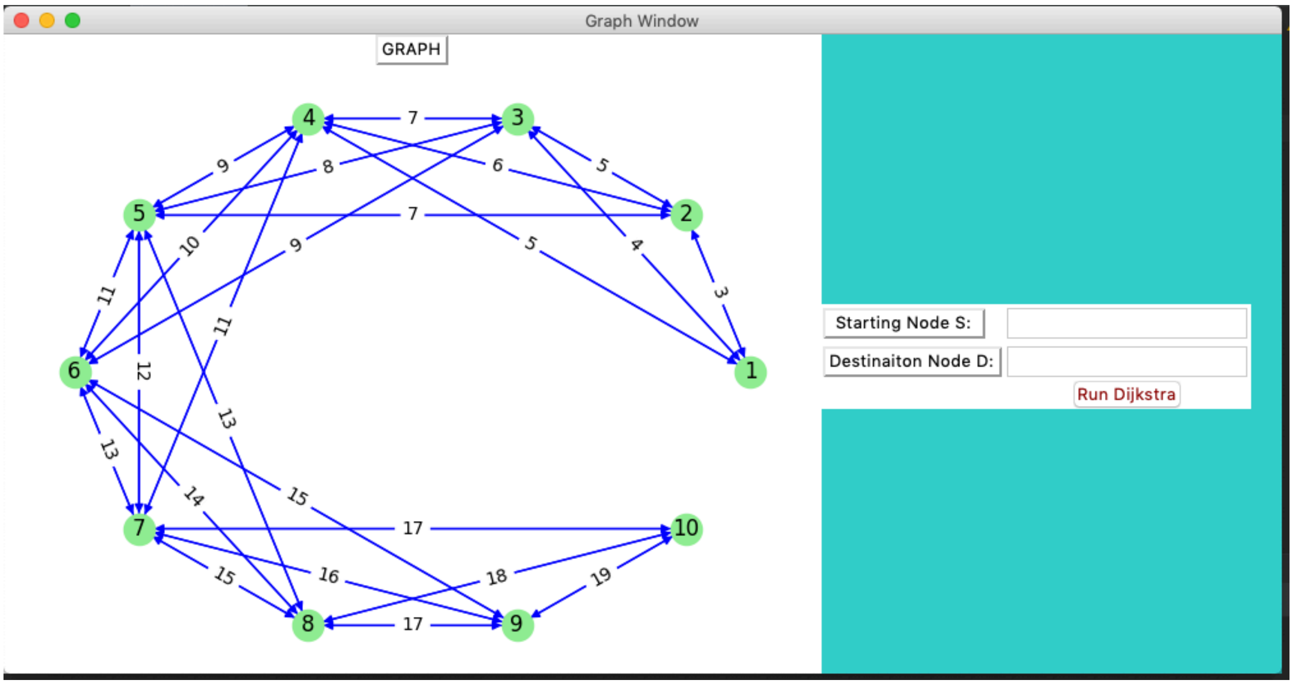

Welcome to Dijkstra Visualization Program
How many cities you want to implement?
Create Graph
Or See Running Times (Can Take A While)

Figure 3: Opening window of the program.

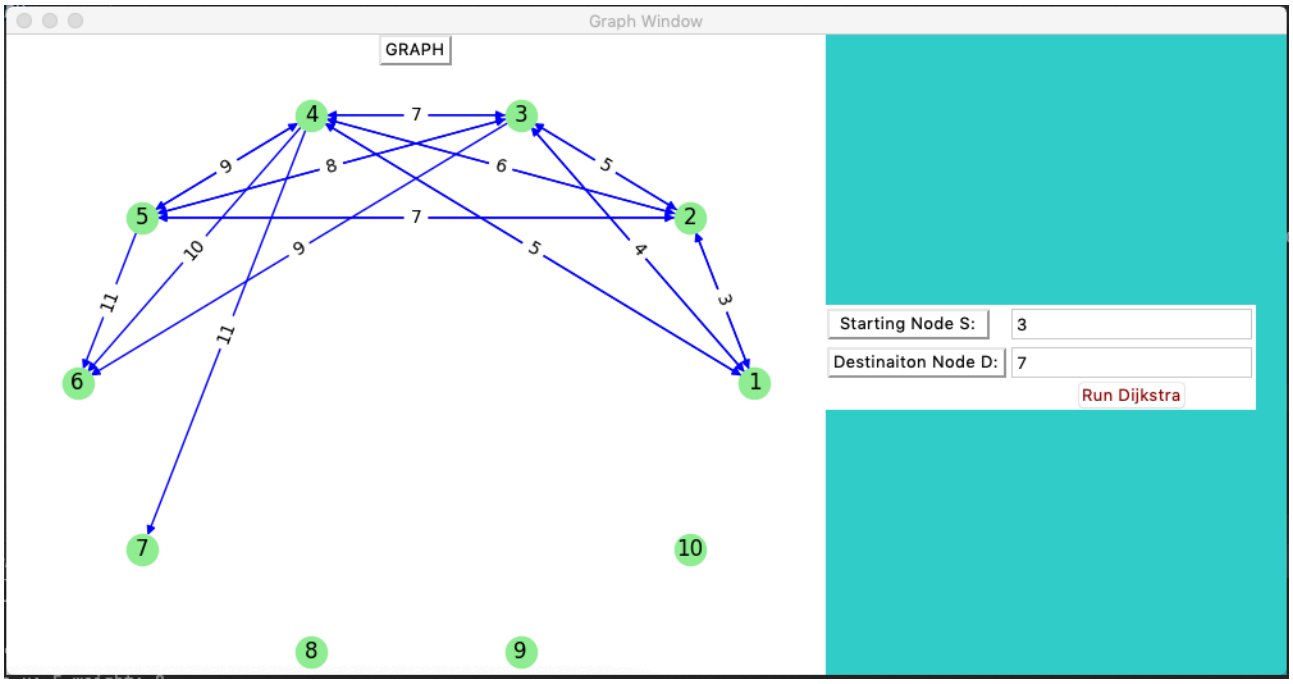
- mainWindow: User enters the desired starting node and destination node. When Run Dijkstra is pressed there will pop-up plot that show the iterations and final path for given parameters.



Starting Node S:

Destinaiton Node D:

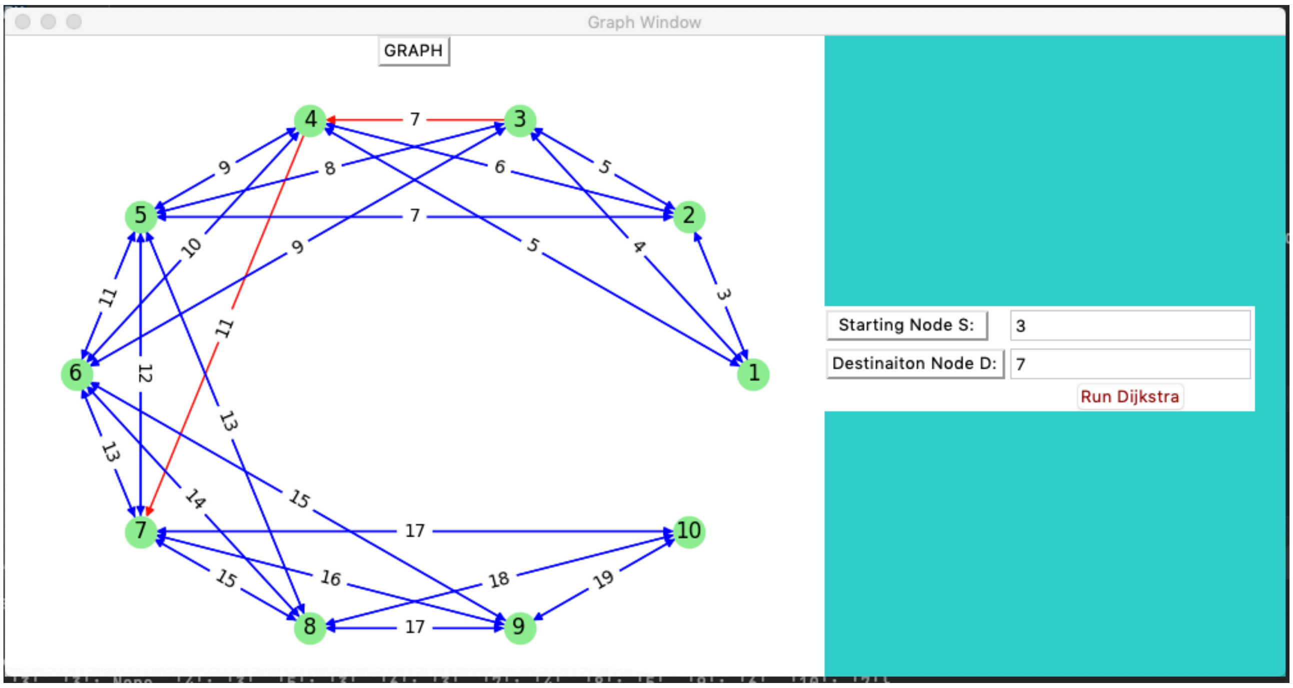
Figure 4: In main window after graph is created.



Starting Node S: 3

Destinaiton Node D: 7

Figure 5: Dijkstra iterations on GUI.



Starting Node S: 3

Destinaiton Node D: 7

Figure 6: End of iterations path can be seen with red edges.

Conclusion

In conclusion, Dijkstra has simple logic but it is very useful. Can be said that project has consists of three main parts. First is theoretical and implementation of Dijkstra. Second is usage of nextworkx library. Third is the creation of the user interface and combining of the libraries networkx, mathplotlib and tkinter. Anybody can get information about Dijkstra algorithm by looking at the dynamic visual presentation.

References

- COMP303 Analysis of Algorithms Lecture Notes
- NetworkX Developers, “Tutorial”, NetworkX, Aug. 22, 2020. Accessed on: Jan. 22, 2021. [Online]. Available: https://networkx.org/documentation/stable/tutorial.html
- tutorialspoint, “Python - GUI Programming (Tkinter)”, 2021. Accessed on: Jan. 22, 2021. [Online]. Available: https://www.tutorialspoint.com/python/python_gui_programming.htm