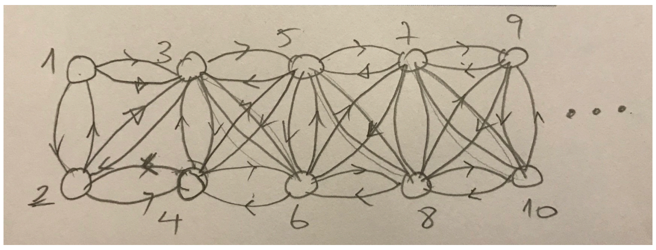
MEF University
Department of Computer Engineering

COMP 303 Analysis of Algorithm
Project 2
Shortest Path in a Graph

In this project, you are to develop a program which finds the shortest path between given source (S) and destination (D) nodes of a Graph by using Dijkstra's shortest path algorithm. In the graph, nodes correspond to cities and weights on the edges correspond to the time to travel between two cities. Given N cities, the graph consists of 2 x N/2 nodes as shown below:



As seen from this graph, there exist a path from node i to node j if | i - j | <= 3, and i and j are not the same. The number of cities, N, the source (S) and destination (D) cities are input the program.

1. (20 points) First describe the Dijkstra's shortest path algorithm. Determine the running time of the algorithm by summing up the running time of each line in the algorithm. Show the steps of the algorithm for N=10, S=1, D=6, when the distance between two cities, $w_{ij} = i + j$ if $| i - j | <= 3$, and i and j are not the same. All other weights are infinity.
2. (30 points) Implement the algorithm in Python by utilizing a min-heap structure. The program reads N (< 20) , S, and D from the user. The weights of edges between nodes i and j are determined by the equation below:

   $$w_{ij} = i + j, \quad \text{if } | i - j | <= 3, \text{ and i and j are not the same.}$$

   The program will output the shortest path together with the cost of this path.

3. (30 points) Run the program for N = 10, 50, 100, 200, 500, 1000 and 2000 with S=1, and D=N. Draw the actual running time and the theoretical running time as graphs. Interpret the results.
4. (20 points) Develop a GUI where the initial graph, the shortest path at each iteration of the algorithm and the final shortest path are shown as red lines. N, S, and D are taken interactively, and the time to travel from S to D is shown on the GUI.