# MEF University
Department of Computer Engineering
COMP 303 Analysis of Algorithm
Project 2
Comparison of Two Shortest Path Algorithms

Part1: Shortest Path

In this part of the project, you are to develop a program in Python3 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:
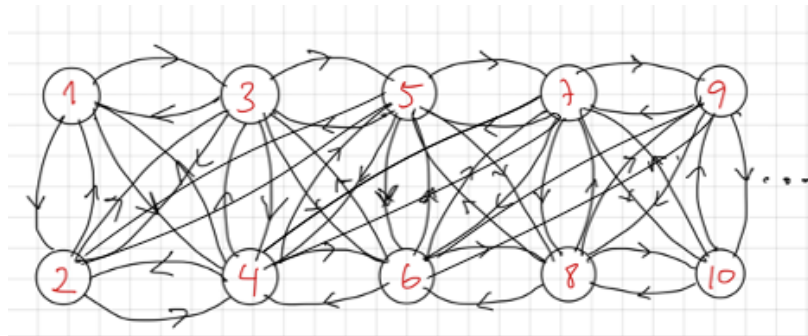
As seen from this graph, there exist an edge from node i to node j if $|i - j| <= 3$, and i and j are not equal. The number of cities, N, the source (S) and destination (D) cities are input the program.

1. (10 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=8, 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. (15 points) Implement the algorithm in Python3 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:

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

In the code, insert comment to the right of every line of the code to explain its purpose and repetition times.

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

3. (15 points) Add counters to every block of repetition in the code to measure the number of repetitions. At the end of the program give the total number of repetitions. Set all costs of repetitions (c_i) to 1. Run the program for N = 10, 50, 100, 200, 500, 1000 and 2000 with S=1, and D=N. Draw and compare the actual (measured) running time, which is total number of repetitions, and the theoretical running time as a graph. Interpret the results.

Part 2: A* Search Algorithm

4. (10 points) First describe the A* search 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=8, when the distance between two cities, $w_{ij} = i + j$ if $| i - j | <= 3$, and i and j are not the same, and the heuristic function between node i and j, h(i,j) is given by h(i,j)=abs (j - i) . All other weights are infinity.

5. (15 points) Implement the A* search algorithm in Python3. The program reads N (< 20) and returns the shortest path between S and D, together with the total cost of the shortest path. The weights of edges between nodes i and j are determined by the equation below:

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

Use the heuristic function h(i,j) = abs (j - i). In addition to this heuristic function, suggest alternative functions which may give rise to the shortest path while visiting a smaller number of nodes.

In the code, insert comment to the right of every line of the code to explain its purpose and repetition times.

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

6. (15 points) Add counters to every block of repetition in the code to measure the number of repetitions. At the end of the program give the total number of repetitions. Set all costs of repetitions (c_i) to 1. Run the program for N = 10, 50, 100, 200, 500, 1000 and 2000. Draw and compare the actual (measured) running time, which is total number of repetitions, and the theoretical running time as a graph. Interpret the results.

Part 3: Comparison of two algorithms (20 points)

Compare these two shortest path algorithms by emphasizing their advantages and disadvantages and by considering their time and storage complexities.

Your report should exhibit all your work including your codes, results, and interpretations. Your project grade will be determined entirely by your report. The programs will be executed to verify your findings in the report.