Indian Institute of Technology Roorkee

Department of Computer Science and Engineering

CSN-261: Data Structures Laboratory (Autumn 2019-2020)

Lab Assignment-9 (L9) **Date: October 9, 2019** **Duration: 2 Weeks**

General Instructions:

1. Every Lab Assignment will be performed by the students individually. No group formation is required and the evaluations will be done every week for the students individually.

Submission and Evaluation Instructions:

1. **Submit your** zipped folder (**<filename>.zip** or **<filename>.tar.gz**) through your account in Moodle through the submission link for this Lab Assignment in Moodle course site: https://moodle.iitr.ac.in/course/view.php?id=46.
2. **Hard deadline for Final submission in Moodle: October 23, 2019 (1:00 pm Indian Time).** For any submission after Final Deadline, 20% marks will be deducted (irrespective of it is delayed by a few seconds or a few days). The key to success is starting early. You can always take a break, if you finish early.
3. The submitted zipped folder (**<filename>.zip** or **<filename>.tar.gz**) must contain the following:
   (a) The source code files in a folder
   (b) A report file (**<filename>.DOC** or **<filename>.PDF**) should contain the details like:
      i. Title page with details of the student
      ii. Problem statements
      iii. Algorithms and data structures used in the implementation
      iv. Snapshots of running the codes for each problem
4. The submission by each student will be checked with others' submission to identify any copy case (using such detection software). If we detect that the code submitted by a student is a copy (partially or fully) of other's code, then the total marks obtained by one student will be divided by the total number of students sharing the same code.

Instructions for L9:

1. Objective of this Lab Assignment is to make the students familiar with different data structures while coding the programs in the Python language to solve some real-life problems.
2. The students are expected to have a basic knowledge of data structures and the Python programming language.
3. The student will have to demonstrate and explain the coding done for this Lab Assignment in the next laboratory class to be held on **October 23, 2019** for evaluation.

Problem Statement 1:

Solve rat in a maze problem using A* algorithm in Python 3. Given a maze along with the source and destination, find the shortest path that the rat must traverse to reach the destination.

A Maze is given as N*M binary matrix of blocks where source block and destination block are given as the starting point and the destination point for a rat respectively. A rat starts from source and has to reach the destination. The rat can move only in any directions including diagonally. In a maze matrix, 0 means the block is a dead end or a barrier and 1 means the block can be used in the path from source to destination.

Data structure that must be used: Fibonacci heap (https://pypi.org/project/fibheap/)

Ask for an option for approximation heuristic that shall be used for distance calculation:

1. Manhattan Distance
2. Diagonal Distance
3. Euclidean Distance

Input: A maze in 2D python list (i.e., list of list) in a Pickle format file, “p1.pkl”

Source → (x_s, y_s)

Destination → (x_d, y_d)

Output: Number steps in the found shortest distance from source to destination along with the followed path in format given as:

$$(x_s, y_s) \rightarrow (x_1, y_1) \rightarrow \cdots \rightarrow (x_k, y_k) \rightarrow (x_d, y_d)\,,$$

where, x implies row number and y implies column number with starting index as 0, (x_s, y_s) is the source, (x_d, y_d) is the destination and (x_k, y_k) are all intermediate blocks.

Also, visualize the given maze in a 2D matrix (heatmap) where a block which is a barrier is shown in black color, blocks that can be used in the path in white color and the blocks which are present in the found path are shown in blue color. Also, highlight source block in green color and destination block in red color.

Sample Case:

Input file: p1.pkl

Source: (0, 0)
Destination: (7,7)

Python list representation of p1.pkl:

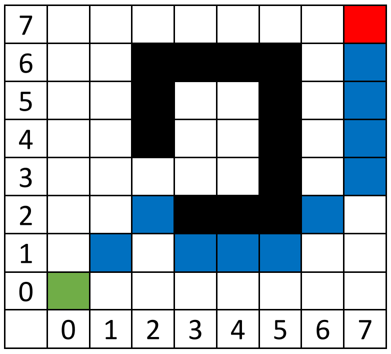
```
[ [  1,  1,  1,  1,  1,  1,  1,  1,  ],
  [  1,  1,  0,  0,  0,  0,  1,  1,  ],
  [  1,  1,  0,  1,  1,  0,  1,  1,  ],
  [  1,  1,  0,  1,  1,  0,  1,  1,  ],
  [  1,  1,  1,  1,  1,  0,  1,  1,  ],
  [  1,  1,  1,  0,  0,  0,  1,  1,  ],
  [  1,  1,  1,  1,  1,  1,  1,  1,  ],
  [  1,  1,  1,  1,  1,  1,  1,  1,  ] ]
```

Output:

Steps: 11 (Using Euclidean Distance)

Path: $(0,0) \rightarrow (1,1) \rightarrow (2,2) \rightarrow (3,1) \rightarrow (4,1) \rightarrow (5,1) \rightarrow (6,2) \rightarrow (7,3) \rightarrow (7,4) \rightarrow (7,5) \rightarrow (7,6) \rightarrow (7,7)$

Visualization:



Problem Statement 2:

Given a Directed Graph identify if the graph is a DAG (Directed Acyclic Graph) or not? If yes, then print the Topological sorting for given DAG. Implement this problem in Python 3.

Data structures that must be used: Set, list, Stack (list based implementation in python)
Note: Use network package in python for this problem

Input: A gpickle format networkx directed graph input file named "p2.gpickle"**.** Each node has a unique label.
Output: Topological sorting for given input graph, if the graph is a DAG else print "the given input is a not a DAG". While printing the Topological sort print the label of the node.

Sample Case:

Input: p2.gpickle

Adjacency Matrix of Input Graph:

V6	1	0	1	0	0	0
V5	1	1	0	0	0	0
V4	0	1	0	0	0	0
V3	0	0	0	1	0	0
V2	0	0	0	0	0	0
V1	0	0	0	0	0	0
	V1	V2	V3	V4	V5	V6

Output: V6, V5, V3, V4, V2, V1