CS102L Data Structures & Algorithms
Lab Week 13

Department of Computer Science

Dhanani School of Science and Engineering

Habib University

Spring 2024

Contents

Introduction

1.1 Instructions

- This lab will contribute 1% towards your final grade.
- The deadline for submission of this lab is at the end of lab time.
- The lab must be submitted online via CANVAS. You are required to submit a .zip file that contains all the *.py* files.
- The .zip file should be named as *Lab_13_aa1234.zip* where *aa1234* will be replaced with your student id.
- **Files that don't follow the appropriate naming convention will not be graded.**
- **Your final grade will comprise of both your submission and your lab performance.**

1.2 Marking scheme

This lab will be marked out of 100.

- 50 Marks are for the completion of the lab.
- 50 Marks are for progress and attendance during the lab.

1.3 Lab Objectives

The objective of this lab is:

- To implement Dijkstra's algorithm in Python.

1.4 Late submission policy

There is no late submission policy for the lab.

1.5 Use of AI

Taking help from any AI-based tools such as ChatGPT is strictly prohibited and will be considered plagiarism.

1.6 Viva

Course staff may call any student for Viva to provide an explanation for their submission.

Prerequisites

2.1 Pytest

Pytest is a popular testing framework in Python known for its simplicity and power in facilitating efficient and scalable testing for applications and software. In order to use Pytest, one first needs to install it and create test functions prefixed with *test_* in a Python file. To install Pytest, you can write the following command on the terminal:

```
pip install pytest
```

After installing Pytest, one can execute all the tests by typing the following command on the terminal:

```
pytest
```

If you want to test any particular file, then you will have to specify the filename as follows:

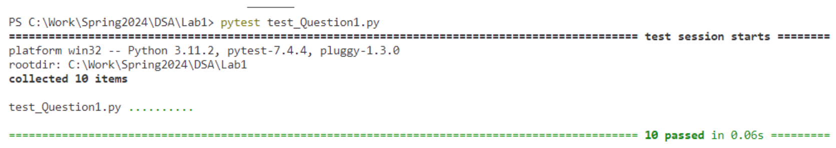
```
pytest <filename>.py
```

In case the above mentioned pytest commands do not work on your system, even though pytest is installed, try using the following commands:

```
py -m pytest
```

```
py -m pytest <filename>.py
```

If all the tests pass successfully, then you should receive a similar output.

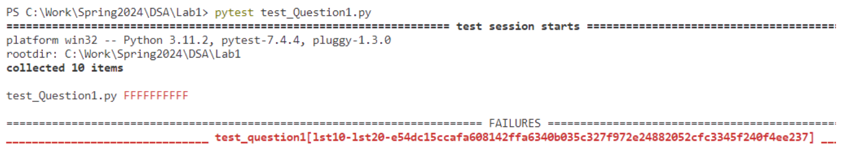


```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
================================================================================================ test session starts ========
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

test_Question1.py ..........

================================================================================================ 10 passed in 0.06s =========
```

However, if all the tests fail, then you should receive a similar output.



```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
================================================================== test session starts ==========================================
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

test_Question1.py FFFFFFFFFF

====================================================================== FAILURES ===================================================
_______________________________ test_question1[lst10-lst20-e54dc15ccafa608142ffa6340b035c327f972e24882052cfc3345f240f4ee237] __
```

Lab Exercises

3.1 Minimum Priority Queue

Complete the function `DeQueue` which takes as arguments a `queue`, a priority queue, and returns the key with the minimum priority.

Note: `EnQueue` and `IsEmpty` have already been implemented for you.

3.1.1 Example

```
>>> queue = []
>>> EnQueue(queue,'A',1)
>>> EnQueue(queue,'B',2)
>>> EnQueue(queue,'C',3)
>>> EnQueue(queue,'D',4)
>>> EnQueue(queue,'E',5)
>>> EnQueue(queue,'F',6)
>>> EnQueue(queue,'G',7)
>>> print(queue)
[('A', 1), ('B', 2), ('C', 3), ('D', 4), ('E', 5), ('F', 6),
    ('G', 7)]
>>>print(DeQueue(queue))
'A'
>>>print(queue)
[('B', 2), ('C', 3), ('D', 4), ('E', 5), ('F', 6), ('G', 7)]
```

This function should be implemented in the file **Question1.py**, accessible within the Lab 13 module on CANVAS.

3.1.2 Testing

To test your function, type one of the following commands in the terminal:

```
pytest test_Question1.py
```

```
py -m pytest test_Question1.py
```

3.2 Shortest Path between two nodes using Dijkstra

Complete the function `GetShortestPath` which takes as arguments a `graph`, a starting node `source`, an ending node `destination`, and computes the shortest path between the source and destination by implementing Dijkstra's algorithm. The function returns a list of tuples, each representing a segment of the shortest path from the source node to the destination node in the graph. Each tuple contains three elements: the starting node of the segment, the ending node of the segment, and the weight or distance associated with traversing that segment. If the shortest path does not exist, then the function returns -1.

Algorithm 1 GetShortestPath(graph, source, destination)

```
Initialize empty list $output$
Initialize empty priority $queue$
Initialize empty dictionaries $dist$ and $prev$
for each node $v$ in $graph$ do
    $prev[v] \leftarrow$ None
    if $v$ is source then
        $dist[v] \leftarrow 0$
        EnQueue($queue, v, 0$)
    else
        $dist[v] \leftarrow \infty$
        EnQueue($queue, v, \infty$)
    end if
end for
while $queue$ is not empty do
    Find node $v$ with minimum distance in $queue$
    Process neighbors of $v$
    Update distances and previous nodes if shorter path found
end while
Reconstruct the shortest path if exists
Return reversed $output$ list if path found, otherwise return $-1$
```

3.2.1 Example

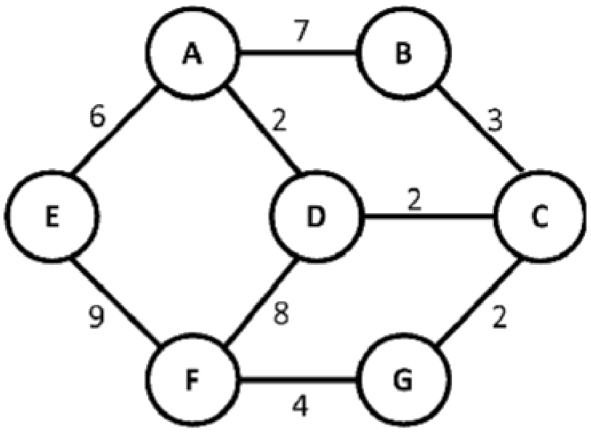


Figure 3.1: Sample Graph

```
>>> graph = {'A': [('D', 2), ('E', 6), ('B', 7)], 'B': [('C'
   , 3), ('A', 7)], 'C': [('B', 3), ('D', 2), ('G', 2)], 'D'
   : [('A', 2), ('C', 2), ('F', 8)], 'E': [('A', 6), ('F',
   9)], 'F': [('D', 8), ('E', 9), ('G', 4)], 'G': [('C', 2),
    ('F', 4)]}
>>> print(GetShortestPath(graph,"A","G"))
[('A', 'D', 2), ('D', 'C', 2), ('C', 'G', 2)]
```

This function should be implemented in the file **Question2.py**, accessible within the Lab 13 module on CANVAS.

3.2.2 Testing

To test your function, type one of the following commands in the terminal:

```
pytest test_Question2.py
```

```
py -m pytest test_Question2.py
```

3.3 Shortest Path in a Grid

Complete the function `GetShortestPathGrid` that takes as arguments: a 2D matrix representing a `grid` with obstacles (indicated by -1), a starting coordinate `source`, and an ending coordinate `destination`, and computes the shortest path between the source and destination by using Dijkstra's algorithm. The function first creates an Adjacency List Representation of the Grid from the 2D matrix. The function returns a list of tuples, each representing a segment of the shortest path from the source node to the destination node in the graph. Each tuple contains three elements: the starting node of the segment, the ending node of the segment, and the weight or distance associated with traversing that segment. If the shortest path does not exist, then the function returns -1.

Note: In the grid, we can only move left, right, up, and down. No diagonal movements are allowed. In addition, each node is a tuple of the form (i,j) representing the grid coordinates and each edge has a weight of 1.

Algorithm 2 GetShortestPathGrid(grid, source, destination)

```
Initialize a dictionary $G$ to store the graph
Enumerate through the grid and add the nodes to $G$
Enumerate through the grid and add the edges to $G$
GetShortestPath($graph, source, destination$)
```

3.3.1 Example

For example, if we have a grid as follows:

1	1	1
−1	1	1
1	−1	1

then the corresponding graph is as shown in Figure 3.2

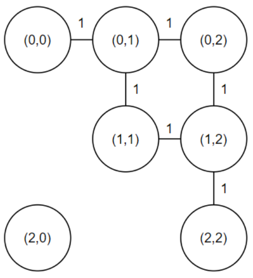


Figure 3.2: Graph corresponding to a 2D Grid with obstacles

```
>>> grid =[[1, 1, 1], [-1, 1, 1], [1, -1, 1]]
>>> source = (0, 0)
>>> destination = (2,2)
>>> print(GetShortestPathGrid(grid, source, destination))
[((0, 0), (0, 1), 1), ((0, 1), (0, 2), 1), ((0, 2), (1, 2),
   1), ((1, 2), (2, 2), 1)]
```

This function should be implemented in the file **Question3.py**, accessible within the Lab 13 module on CANVAS.

3.3.2 Testing

To test your function, type one of the following commands in the terminal:

```
pytest test_Question3.py
```

```
py -m pytest test_Question3.py
```

3.4 Shortest Distance between Cities of Pakistan

Complete the function `GetShortestDistanceBetweenCities` that takes as arguments: a starting city `source`, and an ending city `destination`, and computes the shortest path between the source and destination by using Dijkstra's algorithm. The function first creates an Adjacency List Representation of the Graph by reading the csv file `connections.csv`. The function returns a list of tuples, each representing a segment of the shortest path from the source node to the destination node in the graph. Each tuple contains three elements: the starting node of the segment, the ending node of the segment, and the weight or distance associated with traversing that segment. If the shortest path does not exist, then the function returns -1.

Algorithm 3 GetShortestDistanceBetweenCities(source, destination)

```
Read the CSV file 'connections.csv'.
Convert the CSV data into a list.
Initialize empty dictionary for the adjacency list..
Create an adjacency list representation of the graph using the csv data.
GetShortestPath(graph,source,destination).
```

3.4.1 Reading CSV files in Python

Please follow the following tutorial on how to read and process csv files in Python.

3.4.2 Example

	Islamabad	Taxila	Murree	Nathiagali	Abbottabad	Mansehra	Balakot	Kaghan	Naran	Chilas	Bisham	Gilgit	Hunza	Khunjerab Pass	Malam Jabba	Skardu	Muzaffarabad
Islamabad	0	46	49	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Taxila	46	0	-1	-1	74	-1	-1	-1	-1	-1	-1	-1	-1	-1	254	-1	-1
Murree	49	-1	0	36	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	68
Nathiagali	-1	-1	36	0	34	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Abbottabad	-1	74	-1	34	0	23	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Mansehra	-1	-1	-1	-1	23	0	37	-1	-1	-1	113	-1	-1	-1	-1	-1	-1
Balakot	-1	-1	-1	-1	-1	37	0	59	-1	-1	136	-1	-1	-1	-1	-1	-1
Kaghan	-1	-1	-1	-1	-1	-1	59	0	22	-1	-1	-1	-1	-1	-1	-1	-1
Naran	-1	-1	-1	-1	-1	-1	-1	22	0	113	-1	-1	-1	-1	-1	-1	-1
Chilas	-1	-1	-1	-1	-1	-1	-1	-1	113	0	199	133	306	-1	-1	-1	-1
Bisham	-1	-1	-1	-1	-1	113	136	-1	-1	199	0	-1	-1	-1	61	-1	-1
Gilgit	-1	-1	-1	-1	-1	-1	-1	-1	-1	133	-1	0	186	-1	-1	208	-1
Hunza	-1	-1	-1	-1	-1	-1	-1	-1	-1	306	-1	186	0	151	-1	381	-1
Khunjerab Pass	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	151	0	-1	-1	-1
Malam Jabba	-1	254	-1	-1	-1	-1	-1	-1	-1	-1	61	-1	-1	-1	0	-1	-1
Skardu	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	208	381	-1	-1	0	484
Muzaffarabad	-1	-1	68	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	484	0

Table 3.1: connections.csv

```
>>> print(GetShortestDistanceBetweenCities("Islamabad",'
    Nathiagali'))
>>> [('Islamabad', 'Murree', 49), ('Murree', 'Nathiagali',
    36)]
```

This function should be implemented in the file **Question3.py**, accessible within the Lab 13 module on CANVAS.

3.4.3 Testing

To test your function, type one of the following commands in the terminal:

```
pytest test_Question4.py
```

```
py -m pytest test_Question4.py
```