INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology
Department of Computing (B.Eng.) in Software Engineering

Module: 5SENG001W Algorithms

Module Leader: Klaus Draeger

Coursework Submission Report

- Date of Submission : April 12, 2022
- Student ID : 2019822
- Student UoW ID : w1761405
- Name : D. W. K. S. Gunathilake
- Tutorial Group : Part Time – Group A

Contents

Question 01:

A short explanation of your choice of data structure and algorithm.

Data Structure

Since all the nodes are not inter-connected graph has used to find all the reachable nodes from the given node. To achieve the shortest path, it should not contain back tracking or path overlapping. So, graph has marked as a directed graph.

There can be multiple paths in a single graph. The all-connected vertices to a selected vertex have stored with an extra element which shows the distance of each connected vertices. Hash table was useful to store vertices with distance (as key).

All the positions are marked as pair of x-axis (*i*) and y-axis (*j*) values. Single position has marked as a vertex class and it has above-mentioned hash map to show the connected vertices, position values, and distance from previous vertex.

There is an 2d array has used to check the selected vertex has visited or not.

To find the shortest path from all the possible path, linked list has used. It helps to link each of every vertex with their two neighbor vertices.

To add all the possible path as inter-connected vertices, first tried with an iterative solution using while loop. Since it makes the code bit complex moved to a recursive approach to find all the inter-connected vertices until unvisited vertices array becomes empty (condition).

Algorithm

As mentioned, drawn graph is a single directed graph. So, all the visited edges define with a constant direction which mean edge cannot contain a negative value. Since this graph is a weighted graph, we have to consider the weight of connected nodes to find the shortest path.

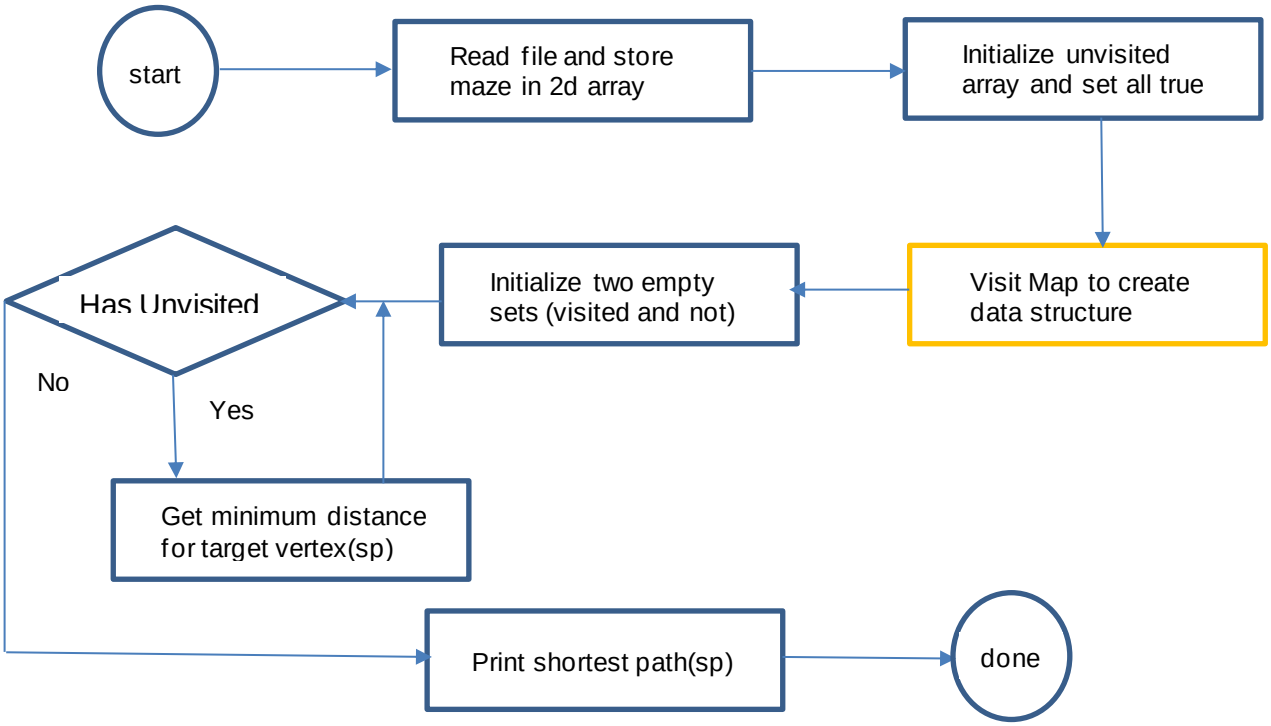
Greedy approach is better for this kind of a graph since we can find multiple solutions. Normally greedy algorithms help us to get the either maximum or minimum benchmark for a solution based on our optimization approach. Algorithm should find the next best path once find a path. It should check with all the possible paths.

The best greedy algorithm find for this situation is Dijkstra's Algorithm. Since I have decided to go with adjacency list as a hash map for represent the neighbor vertices. Using BFS inside a vertex can find the adjacency vertices with minimum head.

Question 02:

A run of your algorithm on a small benchmark example. This should include the supporting information as described in Task 4.

Flow chart: Following will show you how shortest path has decided for sample maze given.



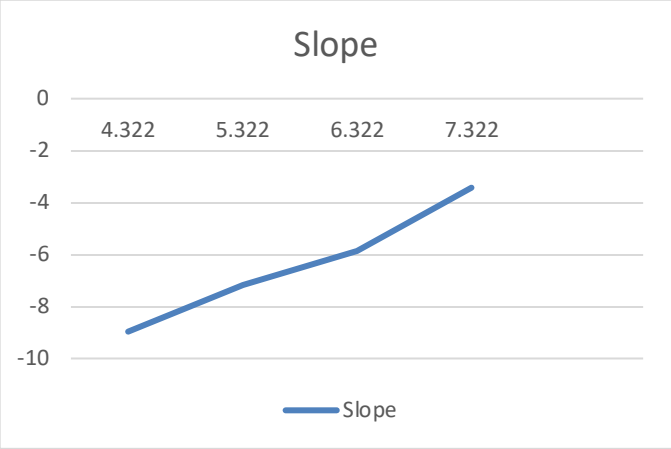

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
1. Start at (10, 1)
2. Move down to (10, 2)
3. Move left to (6, 2)
4. Move down to (6, 10)
5. Move right to (8, 10)
6. Move up to (8, 8)
7. Move right to (9, 8)
8. Move up to (9, 6)
9. Move left to (3, 6)
10. Move up to (3, 1)
11. Move left to (1, 1)
12. Move down to (1, 2)
13. Move right to (4, 2)
14. Move down to (4, 3)
15. Move left to (2, 3)
16. Move down to (2, 5)
17. Done!

Process finished with exit code 0
```

Question 03:

A performance analysis of your algorithmic design and implementation. This can be based either on an empirical study, e.g., doubling hypothesis, or on purely theoretical considerations, as discussed in the lectures and tutorials. It should include a suggested order-of-growth classification (Big-O notation).

I have used the empirical study to analyze the performance of the algorithm. Since we had inputs for large input range doubling hypothesis used to measure b in power law relationship.

Following is the Log-log plot comparison.



This is likely to be a straight line. X-axis contains the Log 2base value for number of inputs and Y-axis contain the log 2base time conceded.

$$\lg Tn \; \alpha \; m \lg n$$

m = (-7.158-(-8.966))/(5.322-4.322) = 1.908
suggested order of growth for this is 2.

Above expression will leads to an equation. Since this is Y= mx + C type equation need to find the C(constant).

$$\lg Tn = 2\lg n + \lg a$$

Tn = a * n **2

0.008 = a * 30 ** 2
a = 8 / 9 * 10**5

$Tn = \frac{8}{9}10^{-5} * n^2$ ← This is the built Hypothesis

So, the estimation time for last puzzle will be.

$Tn = \frac{8}{9}10^{-5} * 2560^2$

$Tn = 58.254\ (s)$ ← Estimated Time

- Suggested order of growth – N^2
- $Tn = \frac{8}{9}10^{-5} * n^2$ (Hypothesis)