Assignment 2 : Subtask2 – A Maze Simulation

COP290

Souvagya Ranjan Sethi
2019CS10405

Ankit Mehra
2019CS10329

Course Co-ordinator:

Prof. Rijurekha Sen

Problem Statement

We are given a graph containing M nodes, along with a starting and an ending node. Amongst these nodes, N nodes are marked as special nodes. Our objective is to start from the starting node, visit each of these N nodes and then reach the end node. We have to find the shortest path for the same if possible. This problem is commonly referred to as Steiner's TSP (Travelling Salesman Problem) and is a special form of the Travelling Salesman Problem.

Assumptions and constraints

- Each non special node can be visited as many times as you want.
- For the optimal solution to the problem i.e. the shortest path, each special node must be visited exactly once (Necessary but not sufficient).
- Nodes in the graph are connected by edges and each edge can be visited as many times as you want and these edges can be traversed in either direction.
- For the optimal solution, there must be exactly one entry and one exit into each special node (Necessary but not sufficient for Steiner's TSP).

INCORPORTION INTO OUR GAME

We can map the game that we have made into a case of the given problem after drawing the following parallels to the Steiner's TSP.

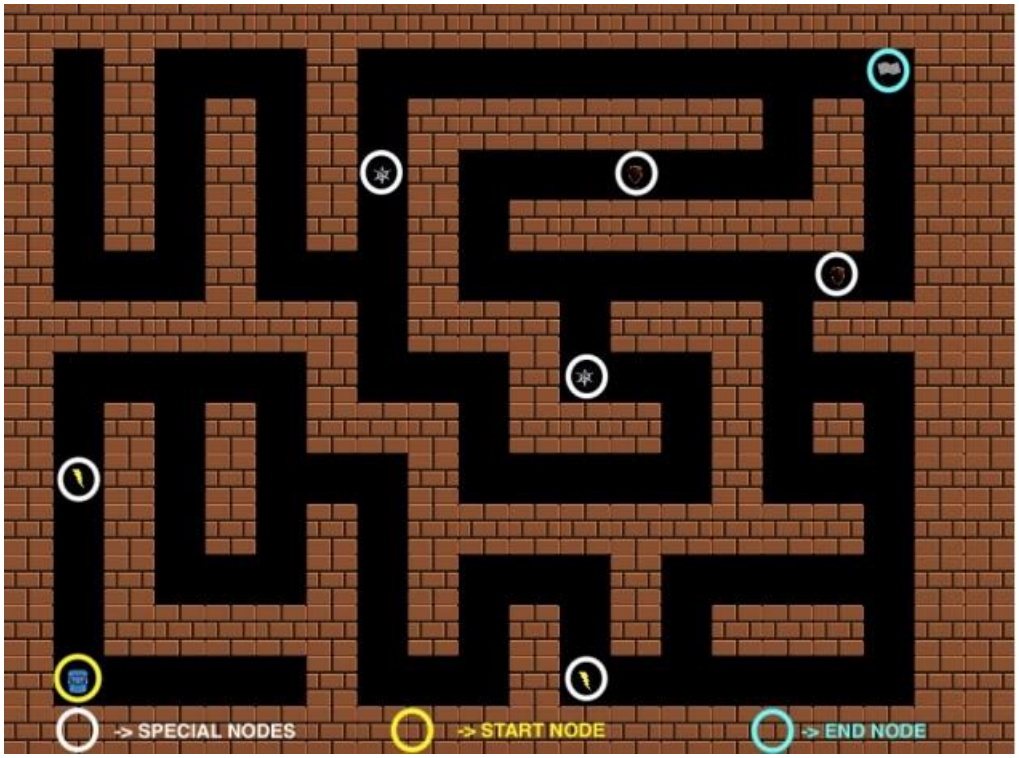
1. Each block in the map can be seen as a node. The total number of nodes (M) in graph is hence the size of the map.
2. Each node in the map has a value assigned to it which is either a 0 or a 1.
3. Each node has a maximum of 4 edges which represents UP, DOWN, LEFT and RIGHT nodes with respect to the current node.
4. An edge exists between 2 nodes if and only if both of the nodes have a zero value assigned to them and they are adjacent to each other in the map.

5. There are 2 flags in the game representing the host and participant base. These can be viewed as the starting and the ending node.

6. There are 6 powerups present in the game. These can be viewed as the N special nodes present in the graph. Collecting these powerups in the map equals visiting the given node in the graph that we have made.

7. The player can only move on the blocks with a value 0 assigned to them which is ensured when edges between different nodes are constrained using (4).

8. All powerups in the game have equal priority for this problem.



ALGORITHM

The algorithm used here is to find the nearest neighbour and then connect all the special nodes (i.e, infinity stones)

Basic assumptions

- The entire graph is visible to us before the simulation along with the coordinates of the special nodes as well as the starting and ending nodes.

- We can traverse through the graph as many times as we want to find out the shortest path

Algorithm

- As we know the coordinates of all special nodes along with the graph. We will first calculate the shortest path amongst all possible pairs of special nodes. We will also calculate the starting and the ending nodes.
- In order to find the shortest path between two nodes, we can use Dijkstra's algorithm.
- After calculating all this, we select any random point(starting point) find the shortest among the distance between the point and the infinity stones.
- Then from the given special point(infinity stone), find the nearest infinity stone.
- Finally from the last infinity stone to the ending point.

TIME COMPLEXITY

Let the total number of nodes be M and the number of special nodes be N. Dijkstra's algorithm has a time complexity of O(E*log(V)) where E is the total number of edges in the graph and V is the total number of vertices. Since, each vertex can have at most 4 edges, the edges cannot be greater than 4V. So, the time complexity from Dijsktra's would be O(M*log(M)). Also, we need to apply this algorithm for each special Node to get its shortest distance from all the other special nodes. So the overall time that we need to spend in this operation is O(N*Mlog(M)). Having calculated this, we use the greedy approach to determine the path, N times, that we need to choose which takes O(N) time, each time. So, the overall complexity is O(N2 + N*Mlog(M)). which is O(N*Mlog(M)) since NM.