Assignment 6 – report

Airi Kokuryo

CSE 13S – Winter 24

Purpose

The purpose of this report is to have a deeper understanding of graphs, stacks, and path findings as well as its steps in implementation. Here, we will use the traveling salesman problem as an example.

Questions

- What benefits do adjacency lists have? What about adjacency matrices?
  The benefits of the adjacency lists are that it may efficiently use the memory for sparse graphs. It requires less space when the number of edges is significantly less than the maximum possible edges, so it's suitable for graphs with fewer connections between vertices. On the other hand, the adjacency matrices have a faster lookup for presence or absence of an edge, and is suitable for dense graphs with many connections between vertices.

- Which one will you use. Why did we chose that (hint: you use both)
  I will use the adjacency lists because the number of edges are significantly less when we only try to visit each place once.

- If we have found a valid path, do we have to keep looking? Why or why not?
  If we find the valid path, there is no need to keep looking. This is because the goal of the tsp problem is to find the shortest path that visits each area exactly once and return to Santa Cruz. Revisiting any other area will only increase the total distance.

- If we find 2 paths with the same weights, which one do we choose?
  The first found path with the same weights will be chosen.

- Is the path that is chosen deterministic? Why or why not?
  The path choice is deterministic, because the graph structure and weights are fixed.

- What type of graph does this assignment use? Describe it as as you can
  The assignment uses a weighted, and the undirected graph. Each edge has an associated weight, representing the distance between connected vertices. The undirected graph means that the edges have no direction, and the weights are both the same from the start to the end.

- What constraints do the edge weights have (think about this one in context of Alissa)? How could we optimize our dfs further using some of the constraints we have?
  The constraints include non-negativity, representing physical distances. The optimization can consider constraints on edge weights, such as avoiding certain paths or areas. Leveraging constraints can help optimize the DFS by pruning paths that violate the constraints early in the search, reducing the search place.