Report for Programming Problem 3 – Bike Lanes

Team:

Student ID: 2014230130 Name: Ricardo Sintra Tavares

1. Algorithm description

The program starts by reading the input, setting the vertices count (number of POIs), freeing and allocating the necessary memory for each structure, and filling an adjency list with the connections and their respective weight.

After reading the input I calculate all the circuits using the adjency list.

For question 2 I store the maximum calculated circuit size.

For question 3 I build an auxiliary structure of edges to get the lane cost for each circuit, and finally for question number 4 I keep adding that cost.

1.1. **Circuit identification**

The circuits in this problem are graph articulation points that have a size of more than one vertex.

To identify them I used Tarjan's algorithm, performing a DFS traversal and calculating the minimum of each vertex index, adding the vertex to a new structure, thus creating vertex groups that are the circuits.

1.2. **Selection the streets for the bike lanes**

The lanes are paths that connect all vertices in each circuit, to calculate the lanes we need to pick the path that has the lowest total sum of weights.

This description is the minimum spanning tree problem, which can be solved with Kruskal's algorithm. The algorithm starts by ordering the edges in increasing order of weight, and choosing an edge that does not create a cycle by utilizing union-find operations (makeSet, findSet and unionSet). After completing these operations for every edge, it returns the minimum path cost of that group of edges.

2. Data structures

To store the connections, I used an adjency list, which is faster given that for each vertex I iterate its respective connections. This structure is a vector of vector pairs, with the first element being the starting vertex, and that vertex having a vector of pairs with the end vertex and the edge cost.

To store the circuits, I have a vector of vectors, giving me the groups of vertices that are articulation points, this structure is filled during Tarjan's algorithm.

For the lanes, I iterate my articulation point groups that have a total number of elements bigger than 1 (circuits), and filling a vector of pairs of pairs, resulting in a structure that has the weight first and its respective edges. To find the connections and their weight I compare that vertex with the adjency list and retrieve its connected elements.

3. Correctness

My solution gives the correct answer to all questions, utilizing the needed memory only and not iterating the structures unnecessarily, setting the auxiliary structures on previously required loops.

Both algorithms are optimized, using a stack for the minimum vertices on Tarjan's algorithm and minimum find operations on Kruskal's.

Kruskal's algorithm is only called if the number of questions is bigger than 2.

4. Algorithm Analysis

Tarjan's algorithm using a stack to store the minimum vertex has a time complexity of O(V + E), with V being the number of vertices and E the number of edges.

Kruskal's algorithm after sorting and utilizing minimum find operations has O(E * log(V) time complexity.

For the adjency list, the space complexity is O(V + E), worst case O(V) is required for a vertex, and O(E) for storing its connections. Looking up the matrix takes O(V) time.

To build the edges structure for Kruskal, at worse there are 4 nested loops, but the space and time complexity is never too big (never O(V^4)), because the main loop starts with the circuit count, then it will iterate for every edge in the circuit, finding the corresponding connections on the adjency list.

5. References

- Class slides explaining Articulation points and the Minimum Spanning Tree problem.
- Kruskal implementation: https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-using-stl-in-c/
- Kruskal implementation: https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/
- Tarjan implementation: https://www.geeksforgeeks.org/tarjan-algorithm-find-strongly-connected-components/
- Play area to test the algorithms: https://graphonline.ru/en/
- https://www.geeksforgeeks.org/comparison-between-adjacency-list-and-adjacency-matrix-representation-of-graph/