Lab 1: Implement Dijkstra for an Undirected Graph

20226758 - Jingqi Fan

May 21, 2024

1 Implementation Details

I implement dijsktra algorithm for directed graph and then adjust it to undirected graph. See full code in my another file. Note that both my directed graph and undirected graph porjects have test code *testDirectedGraph.java* and *testUndirectedGraph.java* which need JUnit. You can run my test code separately in two projects to test my algorithm.

2 Adapt to Undirected Graphs

To adapt my Dijkstra's algorithm to work with undirected graphs, I adds reverse edges each time an edge is added. The followings are my Java code where I have modified *addNode(·)* to accommodate the characteristics of undirected graphs. Specifically, each time an edge is added, a reverse edge with the same weight is also added from the end node to the start node, ensuring that each edge in the undirected graph is properly handled.

the change of *addEdge(·)*

```
1  public void addEdge(Node startNode, Node endNode, int weight) {
2          ...
3          adjacencyList.get(startNode).add(new Edge(startNode, endNode, weight));
4          // Adding reverse edges for undirected graph
5          adjacencyList.get(endNode).add(new Edge(endNode, startNode, weight));
6  }
7  public void addEdge(String startNode, String endNode, int weight) {
8          ...
9          adjacencyList.get(nodeStartEdge).add(new Edge(nodeStartEdge, nodeEndEdge,
               weight));
10         // Adding reverse edges for undirected graph
11         adjacencyList.get(endNode).add(new Edge(nodeEndEdge, nodeStartEdge,weight));
12 }
```

3 Complexity Analysis

Theorem 1 *The complexity of my implementation of Dijkstra Algorithm is* $O((n+m)\log(n))$.

Proof The analysis is composed of three steps. Firstly, each node is inserted into the priority queue initially. Insertion into a priority queue (min-heap) takes $O(\log n)$ time. Then, extracting the node with the minimum distance takes $O(\log n)$ time. Each node is extracted exactly once, leading to a total time of $O(n \log n)$.

Secondly, the algorithm checks if a shorter path has been found to a node via another node. The complexity for updating the priority queue is $O(\log n)$ for each operation. And because each edge is checked once for every node it connects, the total complexity is $O(m \log n)$.

Therefore, after combining these factors, we have the comlexity $O((n+m)\log n)$.

□