ShortestCycle

Benjamin Katz

February 2022

1 The Algorithm

My code implements many java classes co-authored by Robert Sedgewick, Kevin Wayne, and Nate Liu. The code takes a list of edges, copies all those edges except for the edge of interest, and constructs a non directed graph. Still in the constructor, the code implements Dijkstra's algorithm to construct a tree of shortest paths to every vertex from one of the vertices of the edge of interest. While in doIt(), the code looks for for the path from one vertex of interest, to the other vertex of interest. Since the original edge of interest was deleted the shortest path between the two is, in actuality, the shortest cycle once the original edge is readded. The doIt() adds the original vertex and returns the shortest cycle to the client.

2 Proving Correctness

Theorem: The path returned by my implementation of the doIt() method is always a shortest cycle containing a given edge.

- The shortest cycle must contain the edge itself within the cycle. There is no way to reduce the total cycle weight such that it does not include the edge of interest
- An alternate path between the two vertices of interest + the edge of interest creates a cycle
- The shortest alternate path between the two vertexes of interest would therefore create the shortest cycle
- As proven by Sedgewick(Algorithms p. 652) Dijkstra's algorithm gives a shortest path between two vertexes in a directed weighted graph. My implementation uses an undirected weighted graph, undirected graphs are functionally equivalent to directed graphs where every edge runs in both directions.
- Sedgewick's proof is therefore true in cases of undirected weighted graphs as well.

- This implementation always returns a shortest cycle
- QED

3 Proving Time Complexity

Theorem: The cost of finding the shortest cycle is $\theta E log V$

- Construction of the shortest path tree starting at a given vertex to every other vertex is $\theta E log V$(Sedgewick 654)
- Finding the path to any given vertex on the tree in the worst case is θE.
- In the worst case where the tree is a single path containing every edge, the code must iterate through every edge one time.
- The cost of adding the original edge of interest back is constant
- Since the time complexity of the construction of the shortest path tree dominates the time complexity of the searching finding the path($\theta E log V > \theta E + C$) the time complexity remains $\theta E log V$
- QED