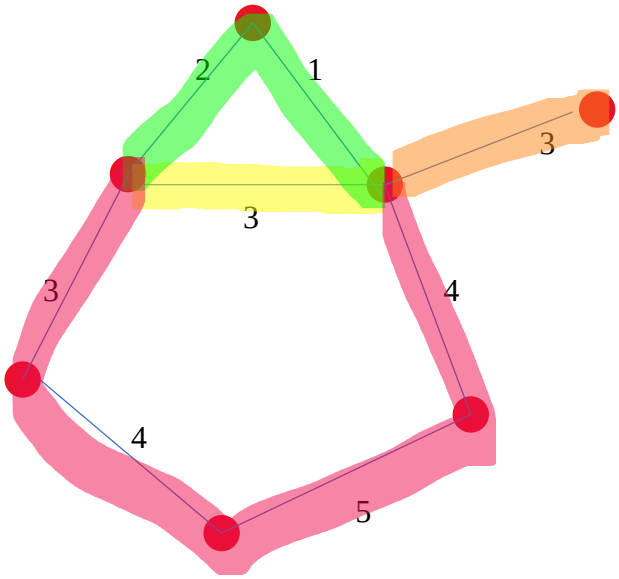
Adam Frenkel Shortest Cycle Assignment

Before describing my algorithm, I will show that the shortest cycle that an edge is a part of will always be that edge, plus the shortest path (not including that edge) between the two vertices of that edge. First of all, the reason why that edge has to be part of it is because otherwise it wouldn't be the shortest cycle **containing** that edge. To prove the point that the rest of the cycle is simply the shortest path between that edge's vertices I will do a proof by contradiction, but first I will use a diagram to make this idea clear:



In this diagram the edge of interest (EOI) is colored yellow, by definition it must be part of the cycle containing itself. Now, to create a cycle containing the EOI it's clear that the orange path won't work because it doesn't connect the two vertices, so it's evident that in order to make a cycle with the EOI, one must connect it's two vertices with another path. Additionally, the **shortest** cycle containing the EOI will be the shortest path that connects those two vertices. In the diagram, the red path will certainly create a longer cycle than the green path, and considering the green path is the shortest path between the two vertices it must be part of the shortest cycle. Thus, in this graph the shortest cycle containing the EOI will be the yellow and green edges. Now I will prove this is true for all graphs by contradiction.

Theorem: the shortest cycle containing the EOI, not including the EOI itself, is the shortest path between the 2 vertices of the EOI.

Assume: the shortest cycle containing the EOI, not including the EOI itself, is **not** the shortest path between the 2 vertices of the EOI:

Proof:

- Given an EOI of length "yellow" in a graph and that the EOI is part of multiple cycles.
- By the assumption, there's a path of length "red" between the 2 vertices that is longer than another path of length "green" between the 2 vertices.

i.e., red > green (1)

- By the assumption, the path of length red plus the EOI would be the shortest cycle.

i.e., red + yellow = the length of the shortest cycle. (2)

- The EOI plus the green path also forms a cycle (as they form two different paths connecting these two vertices).

i.e., green + yellow = the length of a cycle that's not the shortest cycle. (3)

- Putting equation's (2) and (3) together we get,

red + yellow < green + yellow

- Subtract yellow from both sides and get:

red < green

- This contradicts equation (1).

Thus, by contradiction, the shortest cycle containing the EOI, not including the EOI itself, is the shortest path between the 2 vertices of the EOI. □

My Algorithm: Create a graph **without** the EOI, but with its vertices. Then, perform Dijkstra[1] on one of the two vertices of the EOI. Then, use that to find the shortest path between that vertex and the other vertex of the EOI. This will give a list of the edges that form the shortest path between the 2 vertices of the EOI. Next, add the EOI to that list. Now, we have found the shortest path between the 2 vertices of the EOI and added the EOI to that, by the proof above, this will give the shortest cycle containing the EOI.

This algorithm takes $O(E \times \log(V))$ time. This is because running Dijkstra takes $O(E \times \log(V))$, per Sedgewick. The other the most significant action is finding the shortest path to one of the vertices and that merely takes $O(E)$ time, because at worst the longest path will be E, in which case you'll have to iterate over every edge once to find the shortest path. $O(E) + O(E \times \log(V))$ equals $O(E \times \log(V))$. $O(E \times \log(V)) < O((E + V) \times \log(V))$, thus my algorithm meets the performance requirement.

[1] Credit to Robert Sedgewick, Kevin Wayne and Nate Liu for their implantation of Dijkstra, EdgeWeightedGraph, and IndexMinPq.