Assignment 13
Published on 2020-12-06,
Estimated Time: 30 minutes,
Max.grade 10‰

1 Dijkstra's Algorithm

(Goodrich2011, p.640) defines Dijkstra's algorithm. See also `https://bit.ly/2JSXqMU`. It is an efficient algorithm; it requires $O((m+n)\log_2 n)$ time, if we use priority queues; here m is the number of edges and n is the number of vertices in a graph.
In this exercise you do not need to implement a priority queue; assume that you can always pick the vertex with the smallest distance and add it to the set S of visited vertexes (those having distances already computed).

2 Problem

We start with the graph shown in Figure 1.

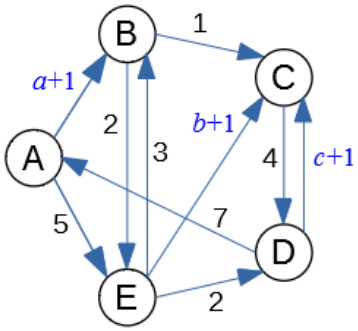

Figure 1: Graph diagram

The following edges (A,B), (E,C), (D,C) have weights $a+1$, $b+1$, and $c+1$ respectively (here a,b,c should be replaced by the digits from your Student ID). Vertex A will be your source vertex. (You can assume that the distance from A to itself is 0; initially all the other distances are infinite, but then Dijkstra's algorithm relaxes them).

(A) Redraw the graph Figure 1, replace the edge weights $a+1$, $b+1$, and $c+1$ by your values of a,b,c.

(B) Run the Dijkstra's algorithm; create a table showing how distances to A,B,C,D,E change as the relaxations are performed. At every iteration highlight which vertex (among those not yet finished) has the minimum distance. Add it to the set S of finished vertices (the set S will have 0 in the very first iteration; after that it will grow by one vertex at a time).

(C) Summarize the result: For each of the 5 vertices tell what is its minimum distance from the source. Also tell what is the shortest path how to get there.