1. In lecture, we saw the number maze problem. I suggest you review your notes about that lecture before continuing with this homework. Suppose we again have an $n \times n$ grid of squares, with each square being labeled with a positive integer. We start with two tokens, each on a *distinct* square. The goal is going to be to swap the positions of the two tokens. In any given turn, you may select any given token and move it up, down, left, or right by *a number of squares equal to the value of the square the other token is currently on.* For example, if the gold token is on a square labeled 3, then you may move the blue token up, down, left, or right by three squares.

   Under no conditions are you ever allowed to move a token off the grid, nor are you allowed to have the two tokens sitting on the same square.

   Your goal is to find the minimum number of moves required to swap the tokens for a given such puzzle, or to correctly report that the puzzle has no solution. Describe how to use a graph to solve this problem. A complete answer includes a description of how to form a graph from an arbitrary puzzle as well as how to find a solution to the puzzle using the graph, or to report that none is possible.

   For example, the following puzzle can be solved with five moves if gold is in the top-left and blue is in the bottom right. Take a moment to think about it before reading the solution that applies to this puzzle, listed below.

   (1,2) (4,4)

G 1 B	2	4	3 B
G 3	4	1	2 G
3	1	2	3 G
2	3	1	2 B

   First, move the gold token down. Because blue is on a two, gold moves down two. Then, move blue up by three (as gold is currently on a three). Gold right, blue left, gold down finishes the five move swap sequence.

We will produce a graph that the nodes are the position of two tokens and the edge means the two tokens can change to the given position respectively.

Then we apply BFS algorith to the graph

2. Fun story: Professor Shindler's ICS 46 class from Fall 2019 had a final on Friday the 13th. In Fall quarters, this problem set gets posted on or around October 31, Halloween. There exists a haunted maze which contains n scare stations, with a designated starting station s and a final station t. To model the haunted maze as a graph, there is a vertex for each scare station and a directed edge from one station to another if it is easy to walk between the two directly (note: because the owners of the haunted house place a restriction on which houses you can visit, the edge between the two scare stations is not bidirectional). Each scare station v has a scare factor of $c(v) > 0$, where the higher $c(v)$ is, the scarier the scare stations are. Thus the graph has costs on the vertices rather than the edges. Shindler is a scaredy-cat, and wants to complete the maze by minimizing the total scare factor of the houses; in other words, he wants to find a path P from s to t such that $\sum_{v \in P} c(v)$ is minimum.

   Suppose you already have Dijkstra's Algorithm implemented for directed graphs. Describe how you can use that implementation to solve this problem. For credit, your answer must involve creating a set of edge weights for the same set of vertices and edges such that a call to Dijkstra's algorithm will produce the desired tree. Do not re-create a "Dijkstra-like" algorithm.

   for Dijtstra's Algorithm, instead of comparing the value of edges, we compare the value of each nodes.

   The rest process of algorithm won't change.

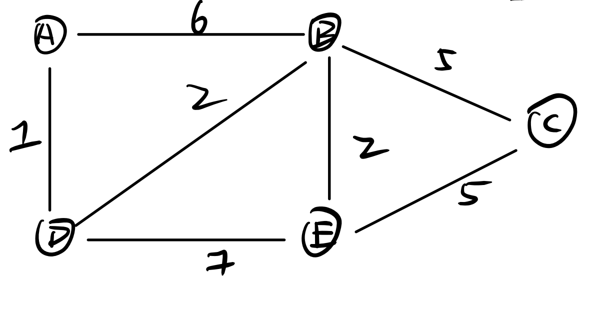
We saw in lecture that Dijkstra's Algorithm can find a single-source **shortest** path tree for a graph with positive edge weights. Suppose I have a graph with positive edge weights and I want a single-source **longest** path tree: that is, for each vertex, I want the longest (simple) path[1] to the other vertices.

I propose the following algorithm: for each edge e, I change the weight w'_e to $1/w_e$. That is, I take the reciprocal of each edge weight. Then I run Dijkstra's Algorithm on this modified graph.

Does this give me the single-source longest path tree? If it does, explain briefly (2-3 sentences) why it does. If it does not, give a counter-example for why it does not. If you are giving a counter-example, you may either draw a graph or describe it. Be sure to demonstrate why this is a counter-example if you do so.

It is not working

lves.



A		
B	$\frac{1}{6}$	A
C	$\frac{1}{6}+\frac{1}{5}$	B
D	$\frac{1}{6}+\frac{1}{2}$	B
E	$\frac{1}{6}+\frac{1}{5}+\frac{1}{5}$	C

For this graph if we wanna find the longest path from A to D it is

ABCED

But the algorithm gives up

ABD