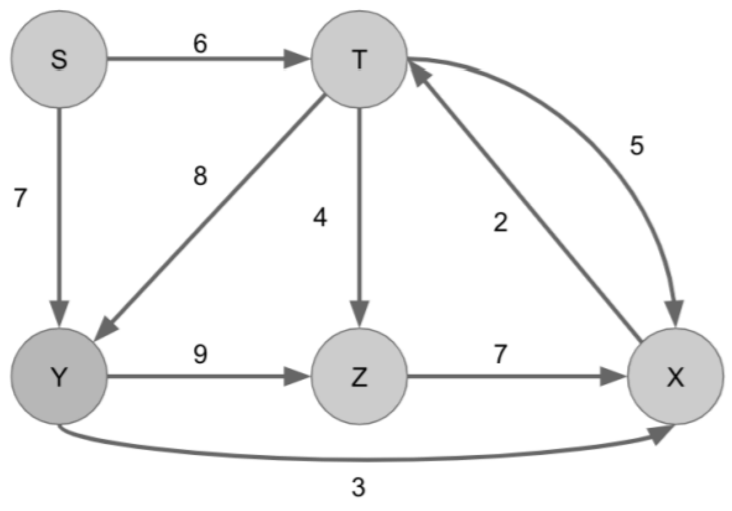
Section 07: Dijkstra's and Graph Modeling

Section Problems

1. Simulating Dijkstra's

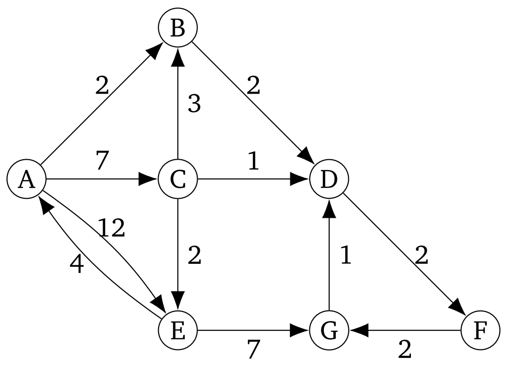
(a) Consider the following graph:



Run Dijkstra's algorithm on this graph starting from vertex s. Use the table below to keep track of each step in the algorithm. Also draw the resulting SPT (shortest path tree) after the algorithm has terminated.

Vertex	Distance	Predecessor	Processed
s			
t			
x			
y			
z			

(b) Here is another graph. What are the final costs and resulting SPT (shortest path tree) if we run Dijkstra's starting on node A?



2. Buggy Dijkstra's

Your best friend tried their hand at writing a more complete Dijkstra's pseudocode. However, you noticed something isn't quite right. Find the bug(s), explain why the behavior isn't correct, and suggest a fix.

```
dijkstraShortestPath(G graph, V start, V end)
    Set known; Map edgeTo, distTo;
    initialize distTo with all nodes mapped to infinity, except start to 0

    while (there are unknown vertices):
        let u be the closest unknown vertex
        known.add(u)
        for each edge (u,v) to unknown v with weight w:
            if (v == end)
                return path // We found the goal node, we're done!
            oldDist = distTo.get(v)
            newDist = distTo.get(u) + w
            if (newDist < oldDist):
                distTo.put(v, w)
                edgeTo.put(v, u)
```

3. Design Problem: DJ Kistra

You've just landed your first big disk jockeying job as "DJ Kistra."

During your show you're playing "Shake It Off," and decide you want to slow things down with "Wildest Dreams." But you know that if you play two songs whose tempos differ by more than 10 beats per minute or if you play only a portion of a song, that the crowd will be very disappointed. Instead you'll need to find a list of songs to play to gradually get you to "Wildest Dreams." Your goal is to transition to "Wildest Dreams" with a playlist of progressively slower songs as quickly as possible (in terms of seconds).

You have a list of all the songs you can play, their speeds in beats per minute, and the length of the songs in seconds.

(a) Describe a graph you could construct to help you solve the problem. At the very least you'll want to mention what the vertices and edges are, and whether the edges are weighted or unweighted and directed or undirected.

(b) Describe an algorithm to construct your graph from the previous part. You may assume your songs are stored in whatever data structure makes this part easiest. Assume you have access to a method `makeEdge(v1, v2, w)` which creates an edge from `v1` to `v2` of weight `w`.

(c) Describe an algorithm you could run on the graph you just constructed to find the list of songs you can play to get to "Wildest Dreams" the fastest without disappointing the crowd.

(d) What is the running time of your plan to find the list of songs? You should include the time it would take to construct your graph and to find the list of songs. Give a simplified big-O running time in terms of whatever variables you need.