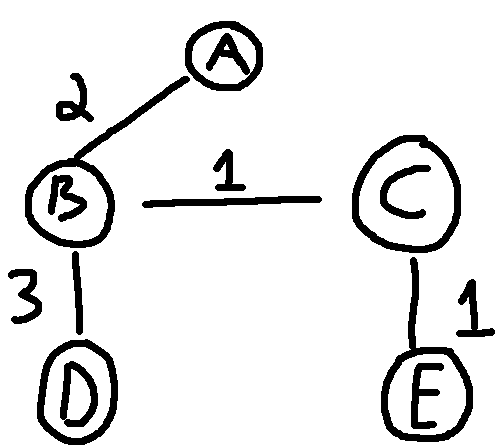
Kevin Sekuj

3/1/2022

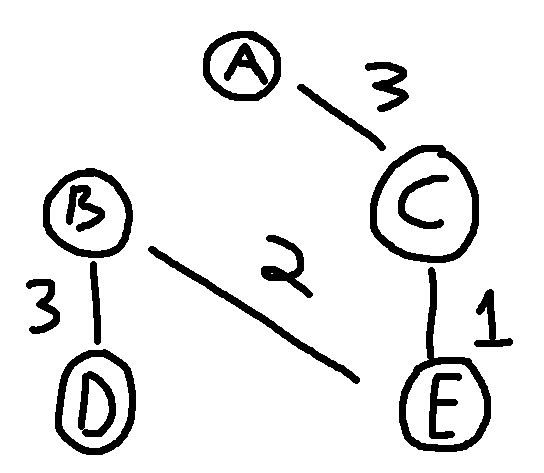
CS325: Analysis of Algorithms

Homework 8 – Graph Algorithms II

1. Draw Minimum Spanning Tree
   1. Draw the minimum spanning tree for the graph



* 1. Draw a spanning tree that is not minimum



1. MST Implementation
   1. MST.py

Both Prim’s and Kruskal’s algorithms are used to construct a minimum spanning tree. The differences are as follows –

**Prim’s algorithm** chooses a random node in the graph and greedily selects neighboring edges that have the minimum weight. It then moves onto neighboring nodes in that node’s adjacency list/matrix and performs the same operation.

**Kruskal’s algorithm** looks for the edge with the minimum length first, by sorting the edges by increasing weight. The algorithm then greedily adds the minimum weighted edges between two trees such that no cycles are created - or in other words, connecting edges of different trees.

1. Apply graph traversal to solve a problem
   1. This algorithm can be solved with a breadth-first search algorithm. We need to compute the shortest path from the given start cell to a given destination cell, and we must avoid barrier cells marked by a “#”.

We initialize a visited set so we don’t visit the same node twice, and conduct our BFS starting from node (x, y) on the input matrix. In every iteration of our BFS, we pop our current coordinates, distance traveled, and current path. We check if we’ve reached the destination cell, and if so, end our algorithm.

Otherwise, we process the current node’s neighbors by putting them on the queue along with distance, incremented by 1 (since we have traveled to that node), and the path (four dimensional – Left, Right, Up, Down) so far.

* 1. Puzzle.py
  2. O(M\*N) – in the worst case, we must traverse all cells of the board, where M/N are the number of rows/columns
  3. Puzzle.py