Question#2

Algorithm idea:

Assume I have an input of dictionary of all Edges and its corresponding cost which is only 5 or 10. The idea is using Kruskal’s algorithm. First I will sort the dictionary by looping though all edges and put all cost =5 in one set and 10 in another set. Next, with these two sets, I can go through the array which has all cost 5 edges first and then go through the 10’s one. While going through sets, before adding edge into my answerlist, I will need to check whether or not this edges will make a circle in the graph (according to the lecture slide). If yes, don’t add this edge into the answerlist. In order to check the circle exists or not, I need another dictionary of all vertex and their corresponding root (highest level parent) and keep update their root if the vertex will be in the same path. With this idea, in the end I will get a set representing MST that contain edges that can connect all vertices in a minimum cost.

Algorithm detail:

#Assume I have a input of dictionary such that Edges[(edge1,edge2)]=cost

Cost5 = Set()

Cost10 = Set()

Answer = []

# initialize all vertices ’s root is themselves

For vertex in Graph:

Root[vertice]=vertex

For item in Edges:

If Edges[item] == 5:

Cost5.add(item) # add to cost5 set if cost equal to 5

Else:

Cost10.add(item) # 10 add to cost10 set

For item in Cost5:

If (add item not forming a circle in graph) in other word (Root[item[0]!=Root[item[1]]]):

Answer.append(item)

Root[item[1]]=Root[item[0]]/ Root[item[0]]=Root[item[1]

For item in Cost10:

If (add item not forming a circle in graph) in other word (Root[item[0]!=Root[item[1]]]):

Answer.append(item)

Root[item[1]]=Root[item[0]]/ Root[item[0]]=Root[item[1]

#Note that the length of 2 sets together is m which is the total number of edges.

return Answer #minimum spanning Tree

Proof of Correctness idea:

Proof by Contradiction (In lecture note of Correctness of Kruskal’s Algorithm). My original Output Tree (T) generate by Kruskal’s Algorithm. Assume Output Tree (T’) has a cycle. Cost of T’ should be larger than original’s cost because it takes an addition step to explore a node that explored before. Therefore, the algorithm above with an input graph which contains all edges of positive cost output a minimum spanning tree.

Proof detail:

Because all cost of edges are 5 or 10 which are all positive. Because cost only have two option, so the sorting can only take O(n), but spilt them to two sets. Since the cost of edges are positive, T’(Tree has a circle)=T + cost(e). Therefore, total cost of T’ will always greater than the total cost of T. As a result, my algorithm above will produce a tree with minimum cost.

Runtime Analysis:

Initialize all root of vertices: O(n) times since n vertex of Graph

Sorting step of Kruskal’s Algorithm (generate two sets (Cost5 and Cost10)): O(m) times

Kruskal’s Algorithm go through all Cost5 and Cost10 to generate Answer: O(m) times

Overall: O(n)+O(m)+O(m)=O(n+m)