# CS-600-A Homework 8

Zijing Huang 10414952

**C-15.2 Suppose G is a weighted, connected, undirected, simple graph and e is a largest-weight edge in G. Prove or disprove the claim that there is no minimum spanning tree of G that contains e.**

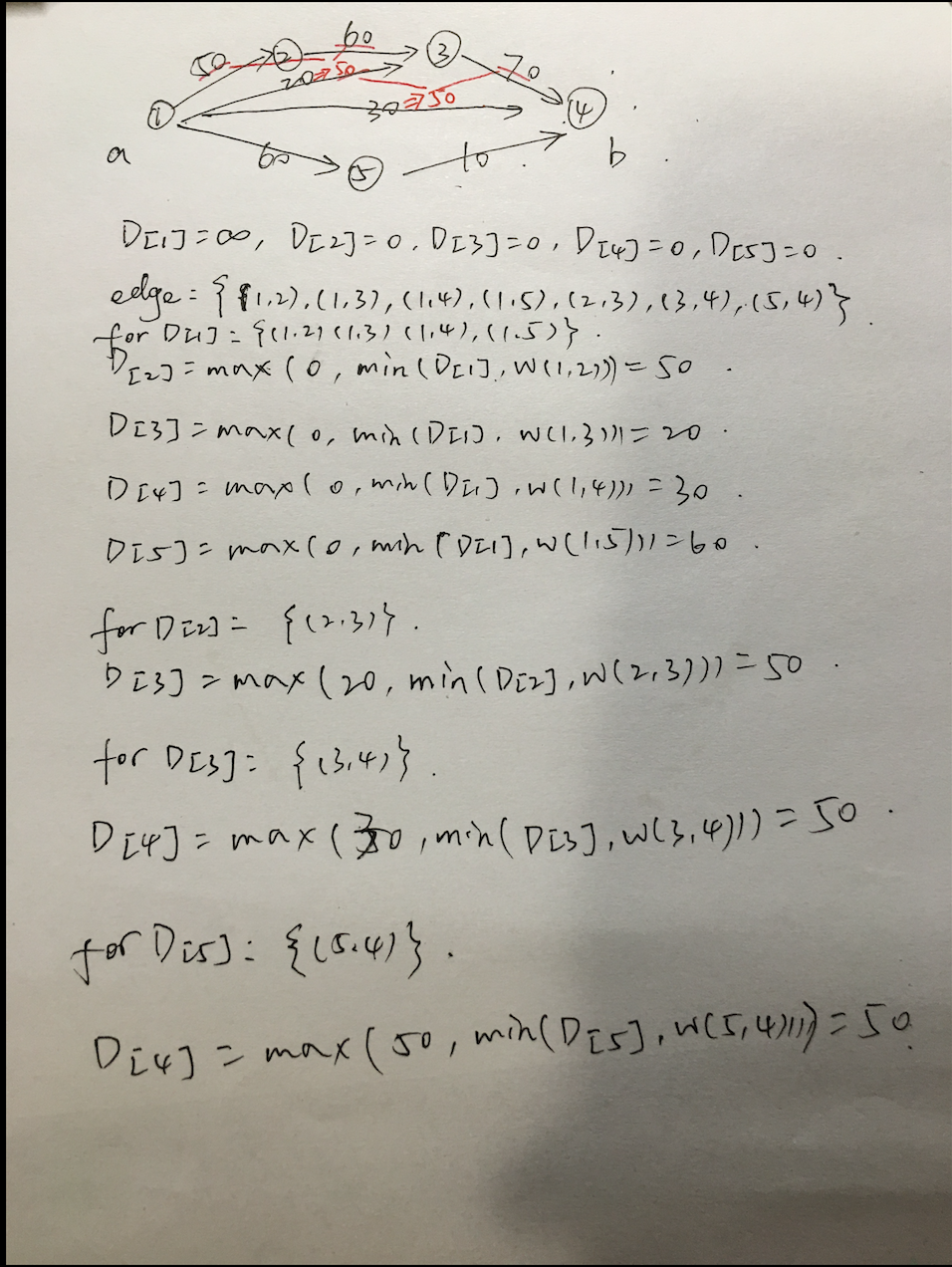
The claim in the question above is false. For example, if the original G is already a tree, which contains the largest-weighted edge, then the minimum spanning tree should contain the largest-weighted edge too. Because a tree such that contains every vertex of a connected graph G is a *spanning tree.*

**A-15.2 Suppose you are given a diagram of a telephone network, which is a graph G whose vertices represent switching centers, and whose edges represent communication lines between two centers. The edges are marked by their bandwidth, that is, the maximum speed, in bits per second, that information can be transmitted along that communication line. The bandwidth of a path in G is the bandwidth of its lowest-bandwidth edge. Give an algorithm that, given a diagram and two switching centers a and b, will output the maximum bandwidth of a path between a and b. What is the running time of your algorithm?**

In this case, we can use an algorithm similar to Dijkstra’s algorithm (Dijkstra’s algorithm is looking for the shortest path between two vertexes, but this algorithm is looking for the path with most weighted).

We can model this diagram as an directed weighted graph. We view the switching center as vertex, the communication lines as edges and the bandwidth as the weight that we assign for each edges. Now what we are looking for is the maximum weight between vertex a and b.

We can maintain an array D[v] representing the weight from a to the current vertex v. And we initialize the D[a] = infinite and D[x] = 0 (x means the rest of the vertex except for a). For every edge of the graph, if d[v] < min(d[u], edge(u,v)), then we give the value of min(d[u], edge(u,v)) to d[v] as the following figure:



The total running time should be the same as Dijkstra’s algorithm, which is O((n+m)\*logn).

**A-15.6 Suppose you have n rooms that you would like to connect in a communication network in one of the dormitories of Flash University. You have modeled the problem using a connected, undirected graph, G, where each of the n vertices in G is a room and each of the m edges in G is a possible connection that you can form by running a cable between the rooms represented by the end vertices of that edge. In this case, however, there are only two kinds of cables that you may possibly use, a 12-foot cable, which costs $10 and is sufficient to connect some pairs of rooms, and a 50-foot cable, which costs $30 and can be used to connect pairs of rooms that are farther apart. Describe an algorithm for finding a minimum-cost spanning tree for G in O(n + m) time.**

We can use the Prim-Jarn ́ık Algorithm. But we don’t use the heap to implement the priority queue, instead we can simply use two doubly linked lists to implement the priority queue, one for the items with $10 and the other for items with $30. Because the running time of insertion and updates of the priority queue, as well as the removeMin(), decrease to O(1) time. The total running time would be O((n+m)\*1).

**R-16.2 Answer the following questions on the flow network N and flow f shown in Figure 16.6a:**

**What are the forward and backward edges of augmenting path π?**

The forward edges: (s, v2), (v2, v3), (v1, v4), (v4, t)

The backward edges: (v1, v3)

**How many augmenting paths are there with respect to flow f? For each such path, list the sequence of vertices of the path and the residual capacity of the path.**

There are six augmenting paths:

(s, v1, v4, t)

(s, v1, v3, v4, t)

(s, v3, v4, t)

(s, v3, v1, v4, t)

(s, v2, v3, v4, t)

(s, v2, v3, v1, v4, t)

**What is the value of a maximum flow in N?**

The maximum flow in N is 15. Because the augmenting paths (s, v1, v4, t) has a residual capacity of 2 and (s, v2, v3, v4, t) has a residual capacity of 3, which add 5 to 4 of 9 capacity from v4->t. Therefore, the maximum flow is 9+3+3 = 15.

**C-16.7 Give an algorithm that determines, in O(n + m) time, whether a graph with n vertices and m edges is bipartite.**

If a graph is bipartite, there are not odd-length cycle in it. (vertex in odd level can only connect to vertex in even level).Therefore we can do a breath-first search on each node. If we found a non-tree edges on the same level, then it is not a bipartite. And then we repeat the same process for the unvisited node, until we visit all the nodes or find out this is not a bipartite.

****

**A-16.2 The city of Irvine, California, allows for residents to own a maximum of three dogs per household without a breeder’s license. Imagine you are running an online pet adoption website for the city, as in the previous exercise, but now for n Irvine residents and m puppies. Describe an efficient algorithm for assigning puppies to residents that provides for the maximum number of puppy adoptions possible while satisfying the constraints that each resident will only adopt puppies that he or she likes and that no resident can adopt more than three puppies.**

As the Reduction to the Maximum Flow Problem showed in page 458 of the textbook, we can build a bipartite H. We can view residents as set X, puppies as set Y and “resident x want to adopt puppy y” as the edge between X and Y. And then we add a new source vertex s and sink vertex t. In addition, we insert a directed edge from s to each vertex in X, and a directed edge from each vertex in Y to t. Finally, we assign to every edge connected to s a capacity of 3 and every edge connected to t a capacity of 1. Finally, we use a maximum flow algorithm on this network. We can get the maximum puppies can be adopted and the which resident should adopt which puppy at the end.

