Carsten Hood
HW 12
CSCE 221-200

- -

1. Consider the situation described in problem R-13.17 in the text (p. 656). Draw the graph representing this problem and show your answer on this graph. Describe which graph algorithm you would use to solve this problem, and its computational complexity. (30 points)

An MST is desired. Use Kruskal's algorithm which takes O((n+m)logn). This partitions every node into a separate "cloud" and repeatedly merges clouds with the shortest distances between them. A total of n-1 edges, in this case 7, are added total. Maintaining the priority queue structure after each edge is added with the removeMinElement() operation takes O(logn) using a heap. This occurs once for each of the m edges in Q. The complexity simplifies to O(mlogn).

- -

2. Show that if all the weights in a connected weighted graph G are distinct, then there is exactly one minimum spanning tree for G. (35 points)

An algorithm to calculate a MST relies on a priority queue to build a MST. When some edges share a weight, different MST's are possible because choosing a minimum element to extract from a priority queue when there are two or more minimum elements is arbitrary. However, when individual edges are distinct, these algorithms have only one option with each step: they must choose the next smallest edge, of which is there is only one. Therefore, the resulting MST will be the only MST, because MST-building algorithms could take no other course.

- -

3. NASA wants to link n stations spread over the country using communication channels. Each pair of stations has a different bandwidth available, which is known a priori. This can be modeled by a weighted graph G=(V,E), where V is the set of stations and E is the set of channels between the stations and the weight w(e) of an edge is the bandwidth of the corresponding edge. NASA wants to select (n-1) channels in such a way that all the stations are linked by the channels and the total bandwidth (defined as the sum of the individual bandwidths of the channels) is maximum. Give an efficient algorithm for this problem that uses the graph G=(V,E) and determine its worst-case time complexity. (35 points.)

```
Algorithm Kruskal(G):
    P = partition of vertices of G; each vertex forms a separate set
    Q = priority queue storing the edges, with key equal to their weights
    T = initially-empty output tree
    while Q is not empty do
        (u,v) = Q.removeMaxElement()
        if P.find(u) != P.find(v) then
            T.add(u,v)
            P.union(u,v)
    return T
```

Use Kruskal’s algorithm, but use a max-heap instead of a min-heap. Partition every node into a separate cloud. Repeatedly merge clouds with the largest distance between them. This will result in a maximum spanning tree. The worst case complexity is O((n+m)logn), the time of Kruskal’s algorithm. Maintaining the priority queue structure with the removeMaxElement() operation, using a heap, takes O(logn). This occurs O(m) times, for each of the m edges in Q.