

## CS513: Homework 6 Solutions

1. We will prove this by contradiction. Let there be two different MSTs  $T_1$  and  $T_2$ . Let  $e$  denote the smallest weight edge that is there in exactly one of the two trees. Without loss of generality, let  $e \in T_1$ . Adding  $e$  to  $T_2$  results in a cycle. Any edge removed from this cycle will result in another spanning tree. Moreover, all the edges in this cycle cannot be in  $T_1$  (otherwise  $T_1$  would have a cycle). Therefore, we can remove an edge  $e'$  from the cycle which is not in  $T_1$ . By our assumption,  $weight(e') > weight(e)$ . Thus we obtain a new MST, but with a smaller weight. Contradiction.
2. Firstly, form the complete graph on the  $n$  points, with the edge weights being the distance between the edge endpoints. Then to produce a  $k$ -clustering, we simply run Kruskal's algorithm for MSTs for  $n - k$  steps. It is easy to see that we end up with a forest of  $k$  trees. Each of these trees forms a cluster. The running time of the algorithm is  $O(n^2 \log n)$ .

To prove the correctness of the algorithm, let the output of the algorithm be the clustering  $\mathcal{C} = \{C_1, \dots, C_k\}$ . Note that the minimum spacing in  $\mathcal{C}$  is the weight  $d^*$  of the  $k - 1$ st most expensive edge (i.e. the next edge that would have been added) in the MST. If  $\mathcal{C}' = \{C'_1, \dots, C'_k\}$  is a different clustering, then there exists points  $p$  and  $q$  that are in different clusters in  $\mathcal{C}'$  but in the same cluster in  $\mathcal{C}$ . Clearly,  $d(p, q) \leq d^*$ , so the spacing of  $\mathcal{C}'$  is no greater than that of  $\mathcal{C}$ . The clustering  $\mathcal{C}$  thus identifies a  $k$ -clustering with maximum spacing.

3. A DFS on  $G$  produces a depth-first tree alongwith back edges on  $G$ . For each vertex  $v$ , in addition to its discovery time  $d_v$ , we will maintain an additional variable  $low_v$  defined as

$$low_v = \min \begin{cases} d_v \\ d_w : (u, w) \text{ is a back edge for some descendant } u \text{ of } v \end{cases}$$

Note that  $low_v$  can be efficiently computed while finishing vertex  $v$ . Just check  $low$  values and back edges for the children of  $v$ . After completing the DFS, we need to check two things:

- (a) If the root has atleast two children in the DFS tree, it is an articulation point. This is because if the root were deleted, then there would be no way for the vertices in different subtrees to be connected. (Note that in a DFS tree, back edges do not jump across subtrees.)
- (b) If for any vertex  $v$  and any of its children (say  $s$ ),  $low_s > low_v$ , then  $v$  is an articulation point. This would mean that  $v$  has a child  $s$  such that there is no back edge from  $s$  or any descendant of  $s$  to a proper ancestor of  $v$ . Thus deleting  $v$  will make the graph disconnected.

Therefore, in  $O(E)$  time, we can find out if the graph is biconnected.