# CS/ECE 374 Sec A ✧ Spring 2023

# Homework 10

Due Wednesday, April 19, 2023 at 10am

---

1. Spanning trees have many nice algorithmic properties and are useful in a number of applications. For those interested, see the connection to abstract structures called matroids.

   (a) Consider the following "local-search" algorithm for MST. It starts with an arbitrary spanning tree $T$ of $G$. Suppose $e = (u, v)$ is an edge in $G$ that is not in $T$. It checks if it can add $e$ to $T$ and remove an edge $e'$ on the unique path $p_T(u, v)$ from $u$ to $v$ in $T$ such that tree $T' = T - e' + e$ is cheaper than $T$. If $T'$ is cheaper then it replaces $T$ by $T'$ and repeats. Assuming all edge weights are integers one can see that the algorithm will terminate with a "local-optimum" $T$ which means it cannot be improved further by these single-edge "swaps". Assuming all edge weghts are distinct prove that a local-optimum tree is an MST. Note that you are not concerned with the running time here.

   (b) Let $G = (V, E)$ be an edge-weighted undirected graph. We are interested in computing a minimum spanning tree $T$ of $G$ to find a cheapest subgraph that ensures connectivity. However, some of the nodes in $G$ are unrealiable and may fail. If a node fails it can disconnect the tree $T$ unless it is a leaf. Thus, you want to find a cheapest spanning tree in $G$ in which all the unreliable nodes (which is a given subset $U \subset V$) are leaves. Describe an efficient for this problem. Note that your algorithm should also check wither a feasible spanning tree satisfying the given constraint exists in $G$. Justify the correctness of your algorithm.

2. You are given two sets of intervals one colored red and the other blue. Let $R = \{I_1, \ldots, I_n\}$ be the set of red intervals and let $B = \{J_1, \ldots, J_m\}$ be the set of blue intervals. Each interval $I$ is specified by two integers, $a(I)$ and $b(I)$, the left and right end points of the interval which are part of the interval (that is $I = [a(I), b(I)]$). A red interval $I_i$ *covers* a blue interval $J_j$ if $I_i$ overlaps with $J_j$. The goal is to choose a smallest set of red intervals from $R$ such that each blue interval is covered by some red interval in the chosen set.

   (a) Consider the following greedy algorithm. Pick a red interval that covers the largest number of blue intervals, add it to the chosen set, remove the covered blue intervals. Repeat until there are no blue intervals left. Give an example to show that this algorithm does not always produce an optimal solution.

   (b) Describe a greedy algorithm that finds an optimal solution. Describe an efficient implementation that runs in $O((n + m) \log(n + m))$ time.

   **Not to submit:** Suppose each red $I_i$ has a non-negative weight $w_i$ and now the goal is to find the minimum weight subset of red intervals that cover the blue intervals. No natural greedy algorithm is known to work for this weighted case. Describe an efficient dynamic programming based algorithm for the weighted problem.

3. **Not to submit:** We saw in lecture that Borouvka's algorithm for MST can be implemented in $O(m \log n)$ time where $m$ is the number of edges and $n$ is the number of nodes. We also saw that Prim's algorithm can be implemented in $O(m + n \log n)$ time. Obtain an algorithm for MST with running time $O(m \log \log n)$ by running Borouvka's algorithm for some number of steps and then switching to Prim's algorithm. This algorithm is better than either of the algorithms when $m = \Theta(n)$. Formalize the algorithm, specify the parameters and argue carefully about the implementation and running time details. No proof of correctness required but your algorithm should be clear.

4. **Not to submit:** Red street in the city Shampoo-Banana can be modeled as a straight line starting at $0$. The street has $n$ houses at locations $x_1, x_2, \ldots, x_n$ on the line. The local cable company wants to install some new fiber optic equipment at several locations such that every house is within distance $r$ from one of the equipment locations. The city has granted permits to install the equipment, but only at some $m$ locations on the street given y locations $y_1, y_2, \ldots, y_m$. For simplicity assume that all the $x$ and $y$ values are distinct. You can also assume that $x_1 < x_2 < \ldots < x_n$ and that $y_1 < y_2 < \ldots < y_m$.

   (a) Describe a greedy algorithm that finds the minimum number of equipment locations that the cable company can build to satisfy the desired constraint that every house is within distance $r$ from one of them. Your algorithm has to detect if a feasible solution does not exist. Prove the correctness of the algorithm. One way to do this by arguing that there is an optimum solution that agrees with the first choice of your greedy algorithm.

   (b) The cable company has realized subsequently that not all locations are equal in terms of the cost of installing equipment. Asssume that $c_j$ is the cost at location $y_j$. Describe a dynamic programming algorithm that minimizes the total cost of installing equipment under the same constraint as before. Do you see why a greedy algorithm may not work for this cost version?

5. **Not to submit:** Let $G = (V, E)$ an undirected edge-weighted graph. The bottleneck weight of a spanning tree $T$ is the weight of the maximum weight edge in $T$. The minimum spanning tree of a graph is also a spanning tree which minimizes the bottleneck weight; try to prove this for yourself. You can compute the MST and hence a minimum bottleneck weight spanning tree in $O(m + n \log n)$ time. Can we do better? It turns out to be yes via a nice clever algorithm that you will work out below.

   - Consider the following algorithm for finding a minimum bottleneck spanning tree. First, given a number $\alpha$ describe a linear-time algorithm that checks whether there is a spanning tree in $G$ with bottleneck weight at most $\alpha$; this is a decision procedure. Using the decision procedure as a black box, use binary search on the sorted edge weights to find a minimum bottleneck weight spanning tree in $O(m \log n)$ time. Your goal here is to understand the algorithm and write down a formal description and justify its running time.

- Now improve the running time of the algorithm to be linear by using the following idea. Instead of sorting the edge weights find the median of the edge weights in linear time using the selection algorithm we discussed previously in the course. Let us call the median edge weight $\beta$. Now use the decision procedure to check whether there is a bottleneck spanning tree of weight at most $\beta$. In either case show how you can effectively recurse on a graph with at most half the edges of the original graph to obtain a linear time algorithm. Write down the algorithm carefully and justify its running time. No proof of correctness necessary but explain how the steps of your algorithm so that it can be properly understood.