CS/ECE 374 Sec A ✧ Spring 2025

Homework 10

Due Wednesday, April 23rd, 2025 at 9am

- You can work in a group of up to **three** students. Read the instructions on the course website for additional details.
- **Submit your solutions electronically on the course Gradescope site as PDF files.** Submit a separate PDF file for each numbered problem. If you plan to typeset your solutions, please use the LaTeX solution template on the course web site. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app (or an actual scanner, not just a phone camera).
- Late submissions will be accepted (for 75% credit) until midnight the day of the deadline.

☞ Some important course policies ☜

- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details including the policy on using AI tools.
- **Avoid the Two Deadly Sins!** Any homework or exam solution that breaks any of the following rules will be given an ***automatic zero***, unless the solution is otherwise perfect. Yes, we really mean it. We're not trying to be scary or petty (Honest!), but we do want to break a few common bad habits that seriously impede mastery of the course material.
  - Always give complete solutions, not just examples.
  - Always declare all your variables, in English. In particular, always describe the specific problem your algorithm is supposed to solve.

See the course web site for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Ed.

---

1. Let $G = (V, E)$ be a directed graph with edge lengths $\ell(e), e \in E$ which may be negative. Let $s \in V$ and without loss of generality assume that s can reach every vertex in V. Recall that Bellman-Ford algorithm can be used to check whether there is a negative length cycle in G, and if not, find the shortest path distances to every vertex v from s. To better understand the details of this we will work through a more refined understanding of various tasks that one can accomplish.

   (a) (3 pts) Suppose G has a negative length cycle. Describe an algorithm to output such a cycle in $O(mn)$ time.

   (b) (2 pts) Call a pair of vertices (u, v) negatively linked if there is a closed walk containing u and v whose total length is negative. Describe an algorithm that given G and u, v decides whether they are negatively linked.

   (c) (5 pts) Even though G may have a negative length cycle there may be valid shortest paths to some vertices. For each vertex v define $d(s, v)$ to be the length of a shortest walk from s to v where we set $d(s, v)$ to be $-\infty$ iff there is a negative length cycle that can reach v (note that s can reach that cycle since we assumed that s reaches all vertices). Otherwise v has a valid shortest path from s and a finite distance. Describe an $O(mn)$ time algorithm that computes the set $\{v \mid d(s, v) = -\infty\}$. *Hint:* Compute meta-graph.

2. Spanning trees have many nice algorithmic properties and are useful in a number of applications.

   (a) (3 pts) 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 strictly 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) (3 pts) Suppose we change the local search algorithm slightly. When considering adding $e = (u, v)$ to T the algorithm replaces the most expensive edge on $p_T(u, v)$ if it is more expensive then e. Prove that this variant of local search algorithm terminates in $O(m)$ swaps.

   (c) (4 pts) Recall that we discussed an algorithm to find the bottleneck distance between s and t in a *directed* graph $G = (V, E)$ with non-negative edge lengths. Undirected graphs have additional structure. We claim the following. Let $G = (V, E)$ be an undirected graph with non-negative edge lengths $\ell(e), e \in E$. Let T be *any* MST of G. Then for *every* pair of vertices (u, v) the unique path $p_T(u, v)$ between u and v in T is a bottleneck shortest path from u to v in G. Thus T is a compact data structure for storing all the bottleneck shortest paths (and hence also distances) in G! Prove this claim.

3. **Not to submit:** 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.

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?