CS/ECE 374 ✧ Fall 2018

Homework 9

Due Wednesday, November 28, 2018 at 10am

Groups of up to three people can submit joint solutions. Each problem should be submitted by exactly one person, and the beginning of the homework should clearly state the Gradescope names and email addresses of each group member. In addition, whoever submits the homework must tell Gradescope who their other group members are.

The following unnumbered problems are not for submission or grading. No solutions for them will be provided but you can discuss them on Piazza.

- Let $R_1, R_2, \ldots, R_n$ be a set of red intervals each of which is specified by its two end points. Let $B_1, B_2, \ldots, B_m$ be a set of blue intervals each of which is also specified by its two end points. You wish to find the smallest number of blue intervals that *cover* the red intervals. A blue interval B_j covers a red interval R_i if they contain the same point p on the line. All intervals are close in the sense that the end points are contained in the interval. Describe a greedy algorithm for this problem and prove its correctness.

- 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 $y_1 < y_2 < \ldots < y_m$.

  - 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.
  - 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?

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.

   - 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.

- 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.
- **Not to submit but encouraged to solve:** 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.

2. Suppose you have just purchased a new type of hybrid car that uses fuel extremely efficiently, but can only travel 100 miles on a single battery. The car's fuel is stored in a single-use battery, which must be replaced after at most 100 miles. The actual fuel is virtually free, but the batteries are expensive and can only be installed by licensed battery-replacement technicians. Thus, even if you decide to replace your battery early, you must still pay full price for the new battery to be installed. Moreover, because these batteries are in high demand, no one can afford to own more than one battery at a time. Suppose you are trying to get from San Francisco to New York City on the new InterContinental Super-Highway, which runs in a direct line between these two cities. There are several fueling stations along the way; each station charges a different price for installing a new battery. Before you start your trip, you carefully print the Wikipedia page listing the locations and prices of every fueling station on the ICSH.

   Given this information, how do you decide the best places to stop for fuel? More formally, suppose you are given two arrays $D[1..n]$ and $C[1..n]$, where $D[i]$ is the distance from the start of the highway to the ith station, and $C[i]$ is the cost to replace your battery at the ith station. Assume that your trip starts and ends at fueling stations (so $D[1] = 0$ and $D[n]$ is the total length of your trip), and that your car starts with an empty battery (so you must install a new battery at station 1).

   - Describe and analyze a greedy algorithm to find the minimum number of refueling stops needed to complete your trip. Don't forget to prove that your algorithm is correct.
   - But what you really want to minimize is the total cost of travel. Show that your greedy algorithm in the preceding part does not produce an optimal solution when extended to this setting.
   - **Not to submit but encouraged to solve:** Describe an efficient algorithm to compute the locations of the fuel stations you should stop at to minimize the total cost of travel.

3. Define a TM generator (or enumerator) G as a TM with a single work tape, but also a special write-only output tape that starts with # at the left most cell, and the write head one cell to the

right of that. Assume for simplicity that the input alphabet is binary. From time to time, depending on its computation on the work tape, the generator may write a character on the output tape and move the output head to the right. It is said to generate a word w if at some point in time, $\#w\#$ is on the output tape. The language generated by G, denoted by $L(G)$, is the set of all strings that G ever generates.

- A generator G is well-behaved if the strings it generates are lexicographically ordered. Prove that a language L is recursive if and only if there is a well-behaved generator G such that $L = L(G)$. Note that you need to show both directions separately. *Hint:* Treat the case of finite and infinite languages separately.
- Prove that a language L is recursively enumerable if and only if $L = L(G)$ for some TM generator G. Note that you need to show both directions separately. *Hint:* Use the idea of dovetailing.

In proving the statement it suffices to show a construction. We describe one case. Suppose you want to show that L is recursively enumerable implies that there is a generator G such that $L = L(G)$. In order to prove this you need to come up with an algorithm that uses, as its input, a description/code $\langle M \rangle$ of a TM M and outputs the description/code $\langle G \rangle$ of generator TM G such that $L(G) = L(M)$. You do not have to actually prove the correctness of your algorithm as long as the description is clear.

4. **Not to submit but strongly encourage to solve:** Consider the language

$$L_{\text{OH}} = \{\langle M \rangle \mid M \text{ halts on at least one input string}\}.$$

Note that for $\langle M \rangle \in L_{\text{OH}}$, it is not necessary for M to *accept* any string; it is sufficient for it to *halt* on (and possibly rejects) some string. Prove that L_{OH} is undecidable.