Homework 3: Greedy

Be sure to read the grading guidelines and style guidelines. Especially to see the suggested format for describing algorithms.

We sometimes describe how long are justifications or proofs are. These lengths are intended to help you estimate how much detail we're expecting, you should not take those estimates as hard length-limitations.

Our solutions for any individual problem will fit in approximately one page or less.

You are allowed (and encouraged!!) to collaborate with each other. Brainstorming is much easier to do in a group than alone! But you must follow the collaboration policy (which includes needing to write your submission on your own).

You will submit to Gradescope; we will have a different box for each problem, so please give yourself extra time to submit.

1. Find a Counterexample [10 points]

An independent set I in a undirected graph $G = (V, E)$ is a subset $I \subseteq V$ of vertices such that no two vertices in I are joined by an edge of E. Consider the following greedy algorithm to try to find a maximum size independent set which is based on the general idea that choosing vertices with small degree to be in I will rule out fewer other vertices:

```
function GreedyIndependentSet($G = (V, E)$)
    $I = \varnothing$
    while $G$ is not empty do
        Choose a vertex of smallest degree in $G$                    ▷ Not counting deleted edges
        Add the vertex $v$ to $I$
        Delete $v$ and all of its neighbors and their incident edges from $G$
                                ▷ None of the neighboring vertices can be included since $v$ is included
    return $I$
```

Prove that this algorithm is incorrect by counterexample. Specifically, give an example of a graph on which this algorithm does not produce a largest size independent set. Show both the independent set that the algorithm finds and a larger independent set.

Note: If you would like to see more discussion of independent sets, see section 1.2 of the text.

2. Dog Sled Safety [25 points]

You are helping to plan a long distance dog sled race. There will be t sleds in this race, which will have a total length of D miles. During this race, any of the sleds can swap out the dogs on its team. To guarantee safety of the dogs, every dog participating in the race needs to undergo a health and equipment check. Your task is to identify where to locate the necessary check points along the race route.

Prior to the race, each of the t teams will submit a plan for when they will change the dogs on their team. We will express the plan for team i as a sequence of mileposts $s_1^{(i)} = 0, s_2^{(i)}, \ldots, s_{r_i}^{(i)}$ where r_i is the number of segments for team i.

When a team changes dogs, the check can occur at the beginning of that segment, the end of the segment, or anywhere within the segment. For example, checkpoints at mile markers 5 and 10 can check dogs for all segments of the following teams:

(a) A team that changes dogs at mile markers 5, and 7

(b) A team that changes dogs at mile marker 7

(c) A team that changes dogs at mile markers 5, 7, and 10

(d) A team that does not change dogs on the team at all

Describe an efficient algorithm that, given the plans for all teams, finds a sequence of mileposts to locate the checkpoints that minimizes the number needed. (It is OK for there to be more than one checkpoint along a single segment for a team. Being inspected at one of these stations is good enough.)

You should be able to compute this using $O(n \log n)$ time where n is the total number of segments of across all of the teams; of course you need to prove your claims.

3. Ice Cream Strategy [25 points]

Molly Moon's is not a sponsor of this problem.

Nathan's favorite ice cream in town is Molly Moon's ice cream. During the winter, when there is no line at the counter, is a great opportunity to order their "ice cream flight", which is a tray filled with a small scoop of every one of their ice cream flavors! There is one (well, at least one) clear drawback from ordering this ice-cream flight. The different flavors melt at different rates! To avoid having a lukewarm liquid mess before we get a chance to try them all, we need to develop a strategy, nay, an ALGORITHM for tackling the frozen confection.

We have done our research and know the rate at which each flavor melts and the amount of time it will take us to eat the scoop of each flavor. The input to the algorithm will be a list of n pairs (m_i, t_i) representing the melt rate (in milliliter per second) and eating time (in seconds) of each of the n ice cream flavors. Our goal is to decide on the order to eat the ice cream to minimize the total amount of melting that occurs.

We will assume that we cannot stop eating an ice cream flavor once we've started. We will also assume that ice cream melts at a constant rate until we finish eating it. So if we finish eating flavor j at time f_j, then the amount melting of flavor j will be $f_j \cdot m_j$. The total melting is therefore given by $\sum_{i=1}^{n} m_i \cdot f_i$.

Design an efficient algorithm that takes as input n pairs consisting of the eating time t_i and melting rate m_i for the i-th ice cream flavor and produces a schedule that minimizes the total melting.

4. Goldilocks and the n Bears [25 points]

You are a caretaker for n bears and have n bowls of porridge to divide between them. You must give each bear exactly one bowl of porridge. (You cannot share the same bowl between two different bears.) The bowls have a variety of sizes $s_1, \ldots, s_n$. Each bear has a hunger level h_i which is the minimum size bowl of porridge that will satisfy them.

Find an efficient algorithm to assign bowls of porridge to bears that will satisfy the largest number of bears and argue that your solution is correct.