# 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]

Recall from class that 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($G = (V, E)$)
    $I \leftarrow \varnothing$
    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.

## 2. Get Your Kicks! [25 points]

US Route 66 was an historic highway connecting Chicago to Los Angeles that was completed in 1926. You have been asked to help plan the logistics of a $100^{\text{th}}$ anniversary commemorative relay race for self-driving electric cars that will run west along Route 66 (actually its closest current approximation). Since individual cars cannot drive along the route continually, each team can split the distance along the route in any way they choose among a team of cars pre-positioned along the route, with splits between relay segments that they specify in advance.

Your job is to help plan the checking of all cars used in the race. While it is easy to check that a team completes the route in accordance with the plan that they have filed at the start (using a digital "baton" given to each team), there is concern that teams may cheat by using cars that do not meet the race specifications. To ensure that every car used in the relay meets the specifications each car must be inspected at some point along its portion of the route.

Your job is to take the plans filed by the teams and use them to set up inspection stations along the route so that you can inspect every car used in the race at some point along its relay segment.

The plan filed for each team consists of a sequence of starting mileposts designating the start of the race and the car changeover mileposts for each leg of the relay. Each such start/changeover is a milepost $s$ with $0 \leq s < D$ where $D$ is the total length of the race. Different teams may have different numbers of starting mileposts. There are $t$ teams and for the $i$-th team for $i = 1$ to $t$ suppose that starting mileposts are $s_1^{(i)} = 0, s_2^{(i)}, \ldots, s_{r_i}^{(i)}$ where $r_i$ is the number of relay segments for team $i$.

Describe an efficient algorithm that, given the plans for all teams, finds a sequence of mileposts to put inspection stations that minimizes the number of stations that you have to set up between Chicago and Los Angeles. (It is OK for there to be more than one inspection station along a single relay 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 relay segments of all of the teams; of course you need to prove your claims.

## 3. The Waiting is the Hardest Part [25 points]

You are in the business of renting a high end video-conferencing studio for those who are very tired of Zoom and want much better production values. Each reservation request you receive for your studio is for a fixed duration $t_i$ in hours and involves $p_i$ people participating. You happen to know that no person participates in more than one reservation request.

You want to schedule all of these requests at your studio, which can only handle one request at a time, in the best way possible according to the following criterion: Each of the people participating wants to have their respective conference over as soon as possible so you want to schedule them to minimize the total person-hours that participants need to wait until their video conference is finished. If you think of the start of your schedule as time 0 and have scheduled request $i$ to finish at time $f_i$, the total person-hours of waiting for that request would be $p_i f_i$.

Design an efficient algorithm that takes as input $n$ pairs consisting of the duration $t_i$ and number of participants $p_i$ for the $i$-th request and produces a schedule that minimizes the total person-hours of waiting summed over all requests.

## 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.