CSE 101 Homework 3
Greedy Algorithms
Due Tuesday, December 3
100 points total = 10 %
AND CSE 101 Homework 4
Efficient versions of Algorithms
Due Thursday, December 5
100 points total = 10 %

For each of the following problems, two greedy strategies are described. One gives optimal solutions, the other does not. **Do parts a,b,c for Homework 3. Do part d. for Homework 4.**

a. For each problem, decide which greedy strategy produces optimal solutions.

b. Give a counter-example for the incorrect stragtegy (5 points)

c. Give a proof of correctness for the correct strategy (15 points).

d. For the correct strategy, give an efficient version of the algorithm. Specify which data structures you use, and any preprocessing or restructuring. Give a time analysis for your efficient version. (20 points.)

Largest Independent Set for a forest The problem is to find the largest independent set for the special case when the input graph is a tree, or set of disconnected trees. (Edges are between nodes and their parents and children.) Remember, an independent set S of a graph is a set of nodes that does not contain both of the endpoints of any edge, i.e. for any edge $\{x, y\}$ either $x \notin S$ or $y \notin S$. So here, we must have a set of nodes of the tree S so that we cannot have both a node and its parent in the set.

Candidate Greedy strategy 1: Find any leaf x in one of the trees, i.e., any node with no children. Add x to S. Delete x and x's parent from the forest. Repeat.

Candidate Greedy strategy 2: Find any root x, i.e., any node with no parent. Add x to S. Delete x and x's children from the forest. Repeat.

Room assignment You are given a list of classes C and a list of classrooms R. Each class c has a positive enrollment $E(c)$ and each room r has a positive integer capacity $S(r)$. You want to assign each class a room minimizing the total sizes of assigned classes. However, the capacity of the room assigned to a class must be at least the enrollment of the class. You cannot assign two classes to the same room. (Note: It may be impossible to assign all classes to larger rooms. In that case, the correct algorithm may not find a solution.)

Greedy strategy A: Repeat until all classes are assigned, or a class cannot be included in any unassigned room. Assign the largest unassigned class to the smallest unassigned room larger than or equal to its enrollment.

Greedy Strategy B: Repeat until all classes are assigned or a class cannot be included in any unassigned room: Assign the largest unassigned room to the largest unassigned class smaller than or equal to its capacity.

Covering a spectrum You want to create a scientific laboratory capable of monitoring any frequency in the electromagnetic spectrum between L and H. You have a list of possible monitoring technologies, T_i, $i = 1, ..n$, each with an interval $[l_i, h_i]$ of frequencies that it can be used to monitor. You want to pick as few as possible technologies that together cover the interval $[L, H]$.

Candidate strategy 1: Pick the technology that covers the most bandwidth, ie. the one that maximizes $h_i - l_i$. Then pick the one that covers the most bandwidth not already covered. Repeat until the interval is covered.

Candidate strategy 2: Consider all technologies that cover L, i.e., with $l_i \leq L$. Pick the one with highest h_i. Reset $L = h_i$ and repeat.

(Note: It may be impossible to cover the spectrum with the technologies. In that case, the correct algorithm may not find a solution.)

Weighted minimum average start time You are given a list of n jobs $j_1, ..j_n$, each with a time $t(j)$ to perform the job, and a weight $w(j)$. You must order the jobs as $j_{\sigma(1)}, , ...j_{\sigma(n)}$. to be done in that order. Given such an order, the start time s_i of job j_i is the sum of times taken by the earlier jobs : $s_i = \sum_{j|\sigma(j)<\sigma(i)} t_j$ in such a way as to minimize the weighted average start time: $\sum_i w_i s_i$.

Candidate greedy strategy 1: Schedule the job with the highest weight, breaking ties by the smaller time.

Candidate greedy strategy 2: For each job, compute the ratio $r(j) = t(j)/w(j)$ and schedule the jobs in order by this ratio. are scheduled in overlapping times.

Earliest Flight You are devising a flight scheduler for a travel agency. The scheduler will get a list of available flights, and the customer's origin and destination. For each flight, it is given the cities and times of departure and arrival. The scheduler should output a list of flights that will take the customer from her origin to her destination that arrives as early as possible, subject to giving her at least 15 minutes for each connection.

Formally, the problem specification is:

- Instance: A set of n cities, a set of m flights f, each having an origin O_f which is a city, a destination D_f which is a city, a time of departure d_f and a time of arrival a_f with $d_f < a_f$. We are also given two cities s and t (the customer's origin and destination).
- Solution Space: A list of flights $f_1, ..f_k$.
- Constraints: $O_{f_1} = s, D_{f_k} = t, D(f_i) = O(f_{i+1})$ for $1 \leq i \leq k-1$, and $a_i + 15 < d_{i+1}$ for each $1 \leq i \leq k-1$. (The sequence of flights must be a path from s to t, and each flight must arrive at least 15 minutes before the next departs.)
- Objective: Minimize a_k. (Arrival time at the customers destination)

Candidate Strategy 1: Sort the flights by arrival time. For each city, keep track of an earliest possible arrival time. Initialize s's arrival time to $-\inf$, and others to ?. For each flight, if the earliest arrival time for the city is marked and at most the flight's departing time - 15 minutes, then mark the destination city with the flight's arrival time.

Candidate Strategy 2: For each flight O_f, put an edge from O_f to D_f of weight $a_f - d_f$. Find the shortest path in the resulting graph from s to t.