CSE 101 Fall 2018
Homework 5
Due: Tuesday, November 6, 2018 at 11:59pm

---

INSTRUCTIONS

This homework assignment may be completed in groups of size 1-4. The solutions must be **typed** (using a computer.) Figures and graphs can be hand drawn. For algorithm descriptions we require a high-level English description AND an implementation description. If you find it necessary to also include a pseudo code to help understand your description then feel free to include it.

Please refer to the course page for requirements in writing up answers for algorithm questions.

---

1. **Quests that maximize gold (25 points)** All the remaining questions concern variations of the following problem.

   In a role-playing video game, there are various quests your character can undertake. When your character completes a quest, the character gains experience points (xp). Your character starts with 0 xp, and the game ends when your character gets $M$ or more $xp$. For each of $n$ quests, $q_1, ..q_n$, you have the following information: $u_i$ : the amount of xp your character must have or exceed to unlock the quest. $x_i$ : the amount of xp your character gains on completion of the quest, and $g_i$: the amount of gold your character gains on the quest. Your character can perform each quest any number of times, given the above. You want to find the sequence of quests that maximizes the gold your character gains on completion of all quests. (Your character never loses gold, or has to spend gold.)

   **Part 1 : 5 points** Below is a greedy strategy for the max gold sequence of quests problem. Give a counter-example where it fails to produce the optimal solution.
   Candidate Strategy A : Always pick the unlocked quest that that gives you the most gold.

   **Part 2: 3 pts** Illustrate the above strategy on the following set of quests:

   | | | | |
   |---|---|---|---|
   | Collect Rattails: | unlock=0, | xp=1, | gold=1; |
   | ExploreCemetary: | unlock=0, | xp=1, | gold=2; |
   | ClimbTower: | unlock=1, | xp=1, | gold=3; |
   | BrewPotion: | unlock=2, | xp=1, | gold=2; |
   | FightOgre: | unlock=3, | xp=1, | gold=2; |
   | ExploreDungeon: | unlock=4, | xp=1, | gold=2; |
   | DefeatDragon: | unlock=5, | xp=1, | gold=6, |
   | $M = 10$ | | | |

   **Part 3: 12 points** Prove that, if every quest gives one xp, i.e., $x_i = 1$ for every $i$, then Candidate Strategy A finds the optimal sequence of quests.

   **Part 4: 5 points** Describe an efficient algorithm that carries out Strategy A. Your description should specify which data structures you use, and any pre-processing steps. (Assume that xp = 1 for all quests.) Give a time analysis, in terms of $n$ and $M$.

2. **Oxen pairing (25 points)** Consider the following problem: We have $n$ oxen, $Ox_1, ..Ox_n$, each with a strength rating $S_i$. We need to pair the oxen up into teams to pull a plow; if $Ox_i$ and $Ox_j$ are in a team, we must have $S_i + S_j \geq P$, where $P$ is the weight of a plow. Each ox can only be in at most one team. Each team has exactly two oxen. We want to maximize the number of teams.

   Here are two greedy strategies for this problem. One always gives a maximal number of teams, the other does not. First, say which one is maximal, and which strategy is not. (3 point) Then, give an example where the non-optimal strategy fails to find the maximum. (5 points) Third, prove that the other strategy does always find the maximum (12 points). Finally, describe how to implement the maximal strategy efficiently and give a time analysis (5 points.)

**Candidate Greedy Strategy I:** Take the strongest and weakest oxen. If together they meet the strength requirement, make them a team. Recursively find the most teams among the remaining oxen.

Otherwise, delete the weakest ox. Recursively find the most teams among the remaining oxen.

**Candidate Greedy Strategy II:** Take the weakest two oxen, If together they meet the strength requirement, make them a team. Recursively find the most teams among the remaining oxen.

Otherwise, delete the weakest ox. Recursively find the most teams among the remaining oxen.

**3. Cell Tower Placement (25 points)** You have a cell-phone company, and are setting up cell phone towers along a long straight highway. You have a list of your customers locations, at points $x_1 \leq x_2 \leq ..x_n$ along the highway. You need to put up base stations at points along the highway, and each base station has an identical range of $R$ miles. All of your customers need to be in range of some base station. Given this, you want to put up as few base stations as possible. Describe a greedy strategy that finds the minimal number of towers required to serve all customers (5 points). Prove that your strategy is optimal (13 points). Then describe an efficient implementation of your strategy and give a time analysis (7 points).

**4.Grade maximization (25 points)** We are taking a class with $k$ projects. We have $H$ hours to divide among the projects, and will spend an integer amount of time on each project. For every project $i$, we are given an array $G_i[0..H]$ so that, if we spend $h$ hours on project $i$, our grade for that project will be $G_i[h]$. Naturally, $G_i$ is increasing with $h$, and spending no time on an assignment gets 0 points, $0 = G_i[0] \leq G_i[1] \leq G_i[2] \leq ...G_i[H]$. Furthermore, for this version, each $G_i$ has *diminishing returns*: there is at least as much benefit for adding an hour the less time you are already putting into the project. In other words, $G_i[1] - G_i[0] \geq G_i[2] - G_i[1] \geq G_i[3] - G_i[2]... \geq G_i[H] - G_i[H-1]$. We need to allocate $H$ hours among projects $1...n$, i.e., find non-negative integers $h_1, ..h_k$ with $\sum h_i = H$ in order to maximize $\sum_i G_i[h_i]$.Below are 2 Greedy Strategies.Which Strategy is optimal?How would you implement it?Give the time complexity and Proof of Correctness.

Candidate Greedy Strategy 1: For each hour, spend the hour on the assignment where it improves the grade the most. In other words, if we are currently spending $h_i$ hours on each assignment $i$, until the sum of the $h_i = H$, increment the $h_i$ which maximizes $G_i[h_i + 1] - G_i[h_i]$.

Candidate Greedy Strategy 2: For each hour, spend the hour on the assignment whose current grade is smallest. In other words, if we are currently spending $h_i$ hours on each assignment $i$, until the sum of the $h_i = H$, increment the $h_i$ which minimizes $G_i[h_i]$.