CPSC 320 2019S2: Assignment 3

This assignment is intended to give you practice in working with greedy algorithms before the midterm exam. It will not be open for submission on Gradescope and will not be graded. We **highly recommend** that you complete all these problems! It is the instructor's opinion that working through these problems will be **more important to your exam preparation** than memorizing textbook readings or working through exams from previous course offerings (and that you should therefore work on these problems first!).

Solutions will be released on the morning of **Sunday, July 21**. But, we recommend you try to solve every problem before you look at the solutions!

1 The Hungry Games

Imagine you're a poor, starving graduate student. You're going to a buffet with some money you earned by TA'ing CPSC 320. The buffet has different food items, and it charges you by the weight of each item you take. To minimize cost, all the foods are in the form of appetizing and nutritious slurries, from which you may dispense any fractional amount in grams you want into compostable paper cups. Oh, the joy.

Your goal is to select the amount (by weight) to take of each food item so that you maximize the amount of calories you consume, without spending more money than you brought with you to the restaurant. For each item, you've calculated the number of calories you get per dollar spent.

1. Design an algorithm to optimally solve this problem. Note that the buffet has a limited quantity of food, and you can take no more than m_i grams of food item i.

2. Give and briefly justify a good asymptotic bound on the running time of your algorithm in terms of n, the number of different food items available in the buffet.

3. Prove that your algorithm is optimal. (We suggest using an exchange argument.)

2 Parking Lot Optimization in Wonderland

A company called Wonderland offers several types of parking permits to its employees, with different durations and prices. The coop student Alice will work in Wonderland for n consecutive days. She wants to figure out the cheapest collection of parking permits that would cover all days she needs to be present at work. Alice can buy as many permits of a given type as she likes.

Let's assume there are k type of permits $1, \ldots, k$: the price of permit type t is p_t dollars and duration D_t. Alice needs to stay at work on n consecutive days $T = [1, 2, 3, \ldots, n-1, n]$. ($T$ is for "time period".) Our goal is to help Alice to compute a collection of permits to buy and when to activate them of the form (t, d), where t is the permit type and d is the starting day for that permit. Of course, this collection of permits has to cover all days in T.

Consider the following greedy approach to the parking problem. Start with all days in T unmarked. Construct a greedy solution as follows. Let i be the first unmarked day in T. For each permit type t starting on day i, check the number of all days in T (including i) that are covered by permit (t, i) and calculate the average cost per day. Pick (t, i) with the smallest average cost per day, add it to the solution and mark all days in T covered by this permit. Repeat until all days in T are covered.

1. Give and briefly explain an example to show that this algorithm does not always produce an optimal solution.

2. Now, assume we have more information about the parking permits offered by Wonderland: the duration of each permit type t is $D_t = 2^t$ and the average cost per day of permits is decreasing – i.e.,

$$p_1/D_1 > p_2/D_2 > \ldots p_k/D_k.$$

   Is the algorithm described above guaranteed to produce an optimal solution? If yes, provide a proof of optimality. If no, give and explain an example in which the algorithm does not produce an optimal solution.

3 Night at the Museum

Suppose that you're in charge of hiring security guards to protect priceless artifacts in a museum exhibit. You are given a line L that represents a long hallway in the show room. You are also given an unordered set $X = \{x_0, x_1, \ldots, x_{n-1}\}$ or real numbers that represent the positions of artifacts in this hallway. Suppose that a single guard can protect all the objects within distance at most d of his or her position, on both sides.

1. Design a greedy algorithm for finding a placement of guards that uses the minimum number of guards to guard all the artifacts with positions in X.

2. Analyze the running time of your algorithm as a function of n, the number of objects that need guarding.

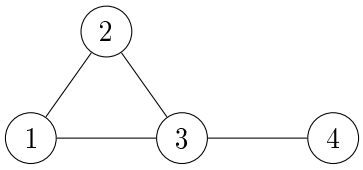
3. Prove that your algorithm is optimal (i.e., employs the minimum possible number of guards). We suggest using a "greedy stays ahead" approach in which you first prove something useful (we won't say what this should be) about the position of the i^{th} guard from the left in the greedy solution compared to the position of the i^{th} guard from the left in the optimal solution, and then using this to show that your greedy algorithm is optimal.

4 Colour Me Confused

We are given a graph $G = (V, E)$. We want to colour each vertex with one of k colours so that two end points of any edge in E receive different colours. This is called a *vertex k-colouring* of the graph.

Recall that the *degree* of a vertex $v \in V$ is the number of edges incident on v. Let d be the *maximum degree* in the graph.

Example:



The maximum degree d in this graph is 3. The graph has a 3-colouring (by using different colours for nodes 1, 2, and 3 and colouring node 4 the same as 1 or 2) and a 4-colouring (by using a different colour for every node), but does not have a 2-colouring.

1. For any value of d, describe a graph with the maximum degree d that can be coloured by two colours.

2. Design a greedy algorithm that will colour vertices of G with at most $d+1$ colors. Your algorithm should consider vertices in an arbitrary order and then greedily choose the colour of each vertex.

3. Explain why your algorithm works correctly (i.e., why it will use at most $d+1$ colours).

4. Now, assume that there is at least one vertex v in V with degree **less than** d. Design a greedy algorithm that will colour vertices of G with at most d colours. You should proceed by first **ordering the vertices** in some way, and then assigning colours using the greedy strategy you developed previously. You should explain why your algorithm uses no more than d colours.