Review Assignment 2

Problem 1: 3D Printer Schedule

You've been asked to create a daily schedule for the use of a shared 3D printer in a maker space. Only one printer is available, and there is significant demand for its use. The demand comes in the form of *requests*. Each request i has a time t_i that it takes to complete. In addition, some requests are more important than others; each request has an importance level w_i; larger values for w_i indicate a higher importance. We assume that, at the beginning of the day, we have available all of the information for all n requests for that day. A *schedule* assigns an order to the requests; such a schedule determines the *completion time*, C_i for each request i. Our goal is to minimize the weighted sum of the completion times, i.e., to minimize $\sum_{i=0}^{n} w_i \times C_i$.

(a) Give as efficient an algorithm as possible for computing an optimal schedule of a day's requests.

(b) Justify the greedy choice behind this algorithm.

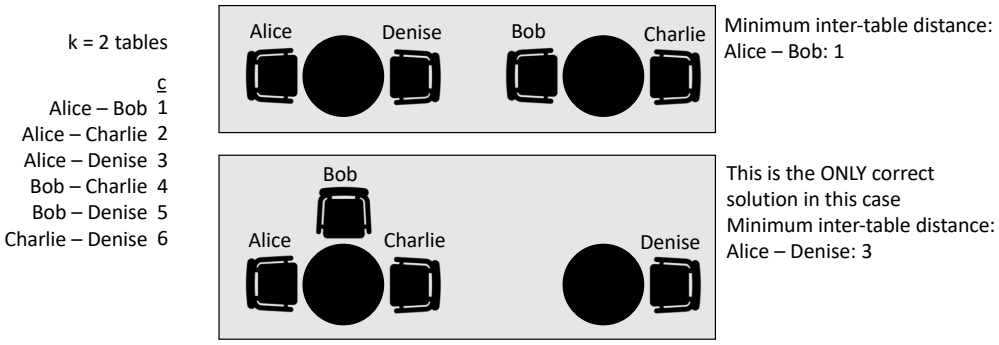
(c) [**5 points**] Give and justify the running time of your algorithm.

Problem 2: Applications of Minimum Spanning Trees

You're planning a dinner party with a group of friends. You want to seat your friends in groups in a way that seats people in a way that discourages them from moving to different tables. We have a function c that, for every pair of guests, g_i and g_j, $c(g_i, g_j)$ gives a measure of the *compatibility* of guests i and j. We assume that lower values of c indicate a higher compatibility. We further assume that compatibility is symmetric (i.e., $c(g_i, g_j) = c(g_j, g_i)$).

You would like to generate seating assignments (people to tables) that clusters together people of high compatibility and minimizes the likelihood that highly compatible people sit at different tables (because they'll get up and walk around or try to switch tables).

You have k tables. You want to generate a seating arrangement (mapping people to tables) that maximizes the minimum c value for any pair of guests seated at different tables. Consider the following example:



To make this problem tractable, each table can have an arbitrary number of seats (it's possible that a table will have only one seat, if there's a guest who's not particularly compatible with all of the other guests). You MUST use all k tables.

a [**15 points**] Design an optimal algorithm for creating the seating chart.

b Give and justify the runtime of your algorithm.

Problem 3: Longest Paths

Given a graph such that all edges have negative weights and source node s and destination node t, devise an algorithm to find the longest path between s and t in $\mathcal{O}(m \log n)$ time where m denotes the number of edges and n denotes the number of nodes in the graph.

Problem 4: Strange Casino Game

You are in a casino that offers the following game: n cards are placed in some order, upside down, on the table. On the side you can't see, each card has a number written on it, that can be positive or negative. If you choose not to play the game, you don't win or lose anything. If you play, you can select a start point s, and an endpoint e such that $e \geq s$. Then, the cards from s to e are turned face up. Let T be the sum of the numbers written on them. If T is positive, you win T dollars, otherwise you pay $-T$ dollars. However, you have cheated and so you know what the numbers on each card are. Provide a linear algorithm that tells you if you should play or not, and, if yes, finds s and e such that T is maximized.

e.g. if the numbers on the cards are:

$$5, 15, -30, 10, -5, 40, 10$$

then the answer should be $s = 4$ and $e = 7$, and you would win 55 dollars.