You are allowed to discuss with others but are not allowed to use any references other than the course notes and the reference books. Please list your collaborators for each question. You must write your own solutions. See the course outline for the homework policy.

This homework accounts for 8% of your total grade.

For all problems below, you can assume that you are working under the word RAM model.

Due date: October 6th at 11:59pm

Problem 1 (25 Points) - Road Trip

Suppose we are given n gas stations located at position $1, 2, \ldots, n$ along a line. Bob would like to drive a car from the first gas station at position 1 to a destination at position $n+1$. However, Bob's car does not have enough gas to reach his destination directly, and so he must stop at some gas stations to refuel his car. Once Bob stops at a station $i \in \{1, \cdots, n\}$, his car will lose all the remaining gas in the tank before refuelling it. The i^{th} gas station can provide gas to travel a distance of $d_i \geq 1$ (so Bob can always reach the next station). This question asks you to help Bob to find a travelling plan with the minimum number of stops to reach the destination.

More specifically, given an input array $[d_1, \ldots, d_n]$, design a greedy algorithm to find a set of stops $\{i_1, i_2, \ldots, i_t\}$ such that $1 = i_1 < i_2 < \ldots < i_t = n+1$ and $i_{j+1} - i_j \leq d_{i_j}$ for all $1 \leq j \leq t-1$, while minimizing t.

1. Consider the following greedy strategy: travel as far as possible, then stop to refuel. Find a counter example to show this strategy is not optimal.
2. Design and analyze an $O(n)$ time greedy algorithm to solve the problem. You should show proof of correctness and that the algorithm runs in the desired running time.

Problem 2 (25 Points) - Maximum Interval Colouring

In class, we have studied the interval scheduling problem and the interval coloring problem. Here we consider a common generalization of these two problems. We are given n intervals $[s_i, f_i]$ where each s_i and f_i are positive integers with $s_i < f_i$, and a positive integer k. Our goal is to use k colors to color as many intervals as possible, so that each interval receives at most one color (could have no color on an interval) and no two overlapping intervals can receive the same color (two intervals $[s_i, f_i]$ and $[s_j, f_j]$ are overlapping if $[s_i, f_i] \cap [s_j, f_j] \neq \emptyset$).

Notice that the interval scheduling problem is the special case when $k = 1$, and the interval coloring problem is the special case to find the minimum k so that all the intervals can be colored. You can think of this problem as using k rooms (k colors) to schedule as many activities (intervals) without time conflicts as possible.

Design an efficient algorithm to find such a coloring, also provide proof of correctness and the running time of your algorithm. You will get full marks if the time complexity of the algorithm is $O(n \log n)$ and the proofs are correct. To implement the algorithm efficiently, you can use a balanced search tree data structure (such as AVL tree) and assume all the operations (add, delete, search) can be done in $O(\log |T|)$ time where $|T|$ is the size of the search tree.

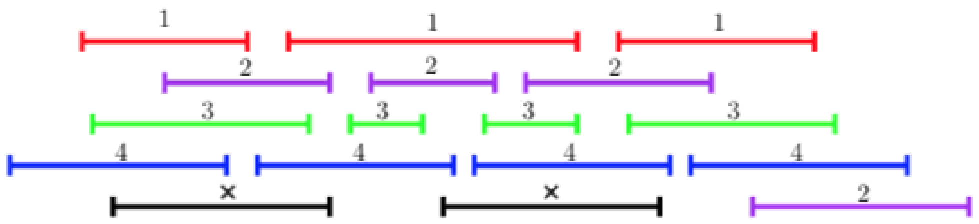

Figure 1: For this example, k=4. We can color all but two of the intervals using four colors as shown in the figure, and this is the maximum possible.

Problem 3 (25 Points) - The photo display problem

You are in charge of a photo contest, and must display all the photos sent in to the contest. You have photos $p_1, p_2, \ldots, p_n$, where photo p_i has width w_i and height h_i (these need not be standard dimensions). **The photos must remain in order.** You need to place them in rows of maximum width W. Photos $p_i, \ldots, p_j$ can go in a row r only if $\sum_{k=i}^{j} w_k \leq W$; the *height* of this row r is defined to be $\max\{h_k : i \leq k \leq j\}$. The goal is to pack the photos into rows to minimize the sum of the heights of the rows.

Give a dynamic programming algorithm to find the minimum sum of row heights. As usual, give a correctness argument and a big-O expression for your runtime.

As part of your answer, explain in words what **subproblem** an entry of your DP table represents (i.e., not *how* it's computed, but what the value it contains is supposed to represent).

Hint. Use n subproblems. A runtime of $O(n^2)$ is possible, but you can get almost full marks for a clear solution with an $O(n^3)$ runtime.

PROBLEM 4 (25 Points) - The power balancing problem

You are in charge of two power stations S_A, S_B, which can handle M_A, M_B maximum watts of electrical load, respectively, before shutting down and a causing total power blackout. Unfortunately, the power demand is currently higher than you can support, so you're going to have to cut off service to some clients.

Certain clients are very important, such as hospitals and telecommunication infrastructure, and you would like to somehow maximize the importance of clients that continue to receive power.

There are a total of n clients attempting to purchase power from your power stations, each of which has a positive integer importance $I_1, I_2, ..., I_n$, and a positive integer demand $D_1, D_2, ..., D_n$ in watts.

Each client can be powered by either S_A or S_B. (You are free to choose which power station is used for each client.)

Your goal is to select two sets of clients $A_1, A_2, ..., A_k$ and $B_1, B_2, ..., B_\ell$ to be powered by stations S_A and S_B, respectively, such that maximum electrical loads of both stations are respected, and the total importance of the powered clients is maximized.

In other words, we want to maximize the total importance $\sum_{j=1}^{k} I_{A_j} + \sum_{j=1}^{\ell} I_{B_j}$ subject to the constraints $\sum_{i=1}^{k} D_{A_i} \leq M_A$ and $\sum_{j=1}^{\ell} D_{B_j} \leq M_B$.

1. Give an efficient dynamic programming algorithm to output the **maximum total importance** that you can obtain. Give a correctness argument, and a big-O expression for your runtime.

   As part of your answer, explain in words what **subproblem** an entry of your DP table represents (i.e., not *how* it's computed, but what the value it contains is supposed to represent).

   Hint: a runtime of $O(n \cdot M_A \cdot M_B)$ is sufficient to receive full marks.

2. Give an algorithm that takes the DP table produced by your algorithm above as input, and prints an optimal assignment of clients to S_1 and S_2.