CS41 Lab 6

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

The learning goal of lab this week is to practice using the dyamic programming algorithm design process. **Note:** dynamic programming is often subtle for students. Some of you might "get it" right away, while others might need more time to internalize the process. Consider this lab a success if at the end of the lab you feel more comfortable with the DP design process and if you make solid progress on at least one of the problems.

General Hints:

- Focus on the **choice** you might make to construct an optimal solution.
- Initially focus on the first two steps of the dynamic programming process. Don't stress about pseudocode until after you've solved all lab problems.

1. **Backpack heist.** You have been recruited as the expert algorithm designer for a team planning an elaborate heist. As part of the heist, the team has scoped out a very fancy mansion full of nefariously-obtained treasures. However, your team will only be able to sneak out a single backpack of items during the heist.

   You know that the backpack can hold a maximum weight of $W > 0$, and you have already established the list of n items in the mansion $\{1, \ldots, n\}$ each with nonnegative weight $w_i \geq 0$. Your task is to output a subset of items $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i$ is as large as possible, subject to the overall weight limit $\sum_{i \in S} w_i \leq W$.

   Design an analyze a dynamic program to solve this problem. Your algorithm should determine (1) the actual weight of the backpack in the optimal case, and (2) which items to put in the backpack. Your algorithm should run in $O(nW)$ time.

2. **Cassie's Convenience Stores.** Carrie plans to open a chain of convenience stores along Baltimore Pike. Using market research, Carrie identified a series of n locations where she can open stores. For each location, Cassie calculated (again using market research) how much annual profit she is likely to gain by placing a store at this location. She can build as many convenience stores as she wants, as long as they are not too close (otherwise, they will compete with each other for business and lose money).

   In this problem, you will design an algorithm that helps Cassie determine how much annual profit she can make. The input to this problem is an integer K, and two arrays $L[1 \cdots n]$ and $P[1 \cdots n]$. Assume that $0 \leq L[i] \leq N$ for each i[1], and that L is sorted in increasing order. The goal of this problem is to output the maximum possilbe profit by placing convencience stores at locations from $L[1 \cdots n]$ such that the distance between any two locations is at least K.

[1] $L[i]$ represents the the location of store i, where 0 is the westernmost terminus of Baltimore Pike, and N is the easternmost terminus

(a) If Cassie decides to build a convenience store at location $L[k]$, what is the closest location to the east or west that she can build, given her list of locations? Write an algorithm $West[k]$ that returns the index of the closest location k' to k such that $L[k'] < L[k]$. Write a similar algorithm for $East[k]$.

(b) Design a dynamic program that computes the maximum annual profit Cassie can earn by placing her convenience stores.

(c) Modify your dynamic program so it returns the set of locations that maximize Cassie's profit.

3. **Longest Palindrome.** Let Σ be a finite set called an *alphabet*.[2] A *palindrome* is a string which reads the same backwards and forwards. Let s be a string of characters from Σ and let $x \in \Sigma$ be some character. The reversal of s is denoted s^R. Then the strings ss^R (that is, s concatenated with s^R) and sxs^R are both palindromes.

   In the **Longest Palindrome Problem**, you're given a string s of n characters from Σ and must output the length of the longest palindrome that is a substring of s.

   (a) *Briefly* describe a simple $\Theta(n^3)$ algorithm that solves the longest palindrome problem. Why is your algorithm $\Theta(n^3)$?

   (b) Design an algorithm that uses dynamic programming to solve the longest palindrome problem in less than n^3 time.

[2] For example, Σ might be $\{0, 1\}$ or $\{a, b, c, \ldots, z\}$.