# Practice Problem Set 10

## . Problem 1

In class we saw the dynamic programming solution for Longest Palindrome Substring, which produces the table $L[i,j]$ where $L[i,j]$ is defined as 1 if the substring $s[i,j]$ is a palindrome, and 0 otherwise. The table dimensions are $n \times n$. Using the results of a DP table, describe an algorithm to output **all** palindrome substrings of maximum length, and justify its runtime of $\Theta(n^2)$. *Note: You must provide a general algorithm. The table below is just an example DP table for $L[i,j]$ where $n = 9$*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | | | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 4 | | | | 1 | 1 | 0 | 0 | 0 | 0 |
| 5 | | | | | 1 | 0 | 0 | 0 | 0 |
| 6 | | | | | | 1 | 0 | 1 | 0 |
| 7 | | | | | | | 1 | 0 | 1 |
| 8 | | | | | | | | 1 | 0 |
| 9 | | | | | | | | | 1 |

## Problem 2

In class we studied the Longest Palindrome Subsequence. Write a procedure (with pseudo-code) that takes as input the completed DP table and the initial strings, and outputs the longest palindrome Subsequence.

## Problem 3

A set of $n$ items is such that each item has a specific weight, $w_i$, for $1 \leq i \leq n$. We would like to find a subset of those items that has total weight $T$ (if there is one). Describe a brute-force algorithm for this problem and determine its runtime. Describe a recursive solution (with pseudo-code) for this problem. Next, provide a dynamic programming solution to this problem and explain the runtime. Describe how to use the DP table to output the elements whose total weight is $T$.

## Problem 4

Suppose each of the items in Problem 3 (above) has an associated *value*, $v_i$, and also a weight $w_i$. Provide a dynamic programming solution that finds a subset of the items with maximum value, such that the total weight is at most $T$ and explain the runtime. Can you also output the specific elements which correspond to the maximum value?

## Problem 5

Suppose you are going on a long bike ride with your e-bike. You must ride exactly n miles. You will need to stop to replace your battery along the way. The bike trail is lined with battery pick-up stations, where you can **trade your battery for a new charged battery**. There is one battery station at every mile marker. However, the batteries at each station vary. For example, suppose that at station 1 they have batteries that last 3 miles, whereas station 2 may have batteries that last 7 miles. The goal is to complete the bike ride using the **minimum number of stops to replace your battery**. You cannot run out of power! If you pick up a battery pack worth 4 miles, then you cannot bike more than 4 miles!

The input to the problem is an array b[0...n], where b[i] stores the battery life of the batteries at station $i$. The output is the minimum number of stops required. Assume you start the trail at mile 0, and that you receive the battery at station 0. Provide a dynamic programming solution to this problem, and determine the runtime of your algorithm.

**Problem 6:**

Let $C[1, 2, \ldots, n]$ be an array of coin denominations. The goal is to determine number **number of ways** of making a total of exactly $T$ cents. For example, if $C = [1, 3, 5]$ and $T = 6$, there are four ways of making 6 cents: $(1,1,1,1,1,1), (3,1,1,1), (3,3), (5,1)$.

Provide a DP solution that returns the number of ways of making exactly $T$ cents, using coins of denominations selected from $C[]$. Note that a coin may be used more than once. You **must** use the following DP table: $W[0, \ldots, n, 0, \ldots, T]$ where entry $W[i, j]$ represents the number of ways of making $j$ cents using coins of denominations selected from $C[0, 1, \ldots, i]$.

**Problem 7:**

Consider a hiking trail of length $n$, which consists of exactly $n$ rest stops, indexed by $i = 1, 2, \ldots, n$. Let $A[1, \ldots, n]$ be a sequence of $n$ numbers, which represents the altitude of each of the rest stops. Suppose at each rest stop, there is a certain amount of prize money. Let $P[1, \ldots, n]$ be a sequence that represents the prize money, where $P[i]$ is the prize money at rest stop $i$. The hiker would like to start the trail at $i = 1$ and finish at $i = n$, and may **choose** at which stops to rest. If the hiker's rest stop is **higher in altitude** than the **previous** rest stop, then the hiker is awarded the prize money at that stop. By default, the hiker will also receive the prize money from the first rest stop they select. An example is given below. In this example, the best choice is to stop at altitudes $2, 4, 9, 12$ and be awarded $15 + 12 + 25 + 30 = 82$ dollars.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A[] | 5 | 2 | 4 | 11 | 9 | 7 | 12 | 1 | 6 | 3 | 8 | 10 |
| Profit: $ | 20 | 15 | 12 | 8 | 25 | 10 | 30 | 10 | 15 | 8 | 20 | 12 |

Design a DP solution that finds the maximum prize money awarded to the hiker. Your solution must use the DP table $H[i]$ which is defined as the maximum total prize possible on the trail from stop 1 to stop $i$, where we specifically stop at rest stop $i$. Write the pseudo-code for the procedure that outputs the list of stops which correspond to the maximum prize.