# CSCI 570 Homework 3
# Spring 2023

*Due Date: Feb. 25, 2023 at 11:59 P.M.*

1. Suppose you are responsible for organizing and determining the results of an election. An election has a winner if one person gets at least half of the votes. For an election with n voters and k candidates, design an optimal algorithm to decide if the election has a winner and finds that winner. Analyze the time complexity of your algorithm.

2. Alice has learned the sqrt function in her math class last week. She thinks that the method to compute sqrt function can be improved at least for perfect squared numbers. Please help her to find an optimal algorithm to find perfect squared numbers and their squared root.

3. Alice has recently started working on the Los Angeles beautification team. She found a street and is proposing to paint a mural on their walls as her first project. The buildings in this street are consecutive, have the same width but have different heights. She is planning to paint the largest rectangular mural on these walls. The chance of approving her proposal depends on the size of the mural and is higher for the larger murals. Suppose n is the number of buildings in this street and she has the list of heights of buildings. Propose a divide and conquer algorithm to help her find the size of the largest possible mural and analyze the complexity of your algorithm. The picture below shows a part of that street in her proposal and two possible location of the mural:

4. Solve the following recurrence using the master theorem.
   a) $T(n) = 8T(\frac{n}{2}) + n^2 log(n)$
   b) $T(n) = 4T(\frac{n}{2}) + n!$
   c) $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$
   d) $T(2^n) = 4T(2^{n-1}) + 2^{\frac{n}{2}}$

5. Suppose you have a rod of length N, and you want to cut up the rod and sell the pieces in a way that maximizes the total amount of money you get. A piece of length $i$ is worth $p_i$ dollars. Devise a Dynamic Programming

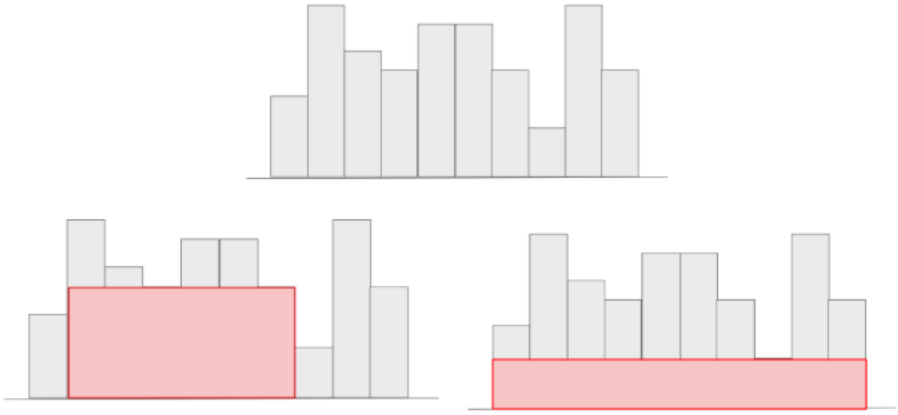

Figure 1: The example in Q3

algorithm to determine the maximum amount of money you can get by cutting the rod strategically and selling the cut pieces.

a. Define (in plain English) subproblems to be solved.

b. Write a recurrence relation for the subproblems

c. Using the recurrence formula in part b, write pseudocode to find the solution.

d. Make sure you specify

   i. base cases and their values

   ii. where the final answer can be found

e. What is the complexity of your solution?

6. Tommy and Bruiny are playing a turn-based game together. This game involves N marbles placed in a row. The marbles are numbered 0 to N-1 from the left to the right. Marble $i$ has a positive value $m_i$. On each player's turn, they can remove either the leftmost marble or the rightmost marble from the row and receive points equal to the sum of the remaining marbles' values in the row. The winner is the one with the higher score when there are no marbles left to remove.

   Tommy always goes first in this game. Both players wish to maximize their score by the end of the game.

Assuming that both players play optimally, devise a Dynamic Programming algorithm to return the difference in Tommy and Bruiny's score once the game has been played for any given input. Your algorithm must run in $O(N^2)$ time.

a. Define (in plain English) subproblems to be solved.

b. Write a recurrence relation for the subproblems

c. Using the recurrence formula in part b, write pseudocode to find the solution.

d. Make sure you specify

   i. base cases and their values

   ii. where the final answer can be found

e. What is the complexity of your solution?

7. The Trojan Band consisting of n band members hurries to lined up in a straight line to start a march. But since band members are not positioned by height the line is looking very messy. The band leader wants to pull out the minimum number of band members that will cause the line to be in a formation (the remaining band members will stay in the line in the same order as they were before). The formation refers to an ordering of band members such that their heights satisfy $r_1 < r_2 < ... < r_i > ... > r_n$, where $1 \leq i \leq n$.

   For example, if the heights (in inches) are given as

$$R = (67, 65, 72, 75, 73, 70, 70, 68)$$

   the minimum number of band members to pull out to make a formation will be 2, resulting in the following formation:

$$(67, 72, 75, 73, 70, 68)$$

   Give an algorithm to find the minimum number of band members to pull out of the line.

   **Note:** you do not need to find the actual formation. You only need to find the minimum number of band members to pull out of the line, but you need to find this minimum number in $O(n^2)$ time.

a. Define (in plain English) subproblems to be solved.

b. Write a recurrence relation for the subproblems

c. Using the recurrence formula in part b, write pseudocode to find the solution.

d. Make sure you specify

   i. base cases and their values

   ii. where the final answer can be found

e. What is the complexity of your solution?

8. From the lecture, you know how to use dynamic programming to solve the 0-1 knapsack problem where each item is unique and only one of each kind is available. Now let us consider knapsack problem where you have in- finitely many items of each kind. Namely, there are n different types of items.

   All the items of the same type $i$ have equal size $w_i$ and value $v_i$. You are offered infinitely many items of each type. Design a dynamic programming algorithm to compute the optimal value you can get from a knapsack with capacity $W$.

   a. Define (in plain English) subproblems to be solved.

   b. Write a recurrence relation for the subproblems

   c. Using the recurrence formula in part b, write pseudocode to find the solution.

   d. Make sure you specify

      i. base cases and their values

      ii. where the final answer can be found

   e. What is the complexity of your solution?

9. Given n balloons, indexed from 0 to $n-1$. Each balloon is painted with a number on it represented by array nums. You are asked to burst all the balloons. If you burst balloon $i$ you will get $nums[left] * nums[i] * nums[right]$ coins. Here left and right are adjacent indices of i. After the burst, the left and right then becomes adjacent. You may assume

$nums[-1] = nums[n] = 1$ and they are not real therefore you can not burst them. Design an dynamic programming algorithm to find the maximum coins you can collect by bursting the balloons wisely.

Here is an example. If you have the *nums* array equals $[3, 1, 5, 8]$. The optimal solution would be 167, where you burst balloons in the order of 1, 5 3 and 8. The left balloons after each step is:

$$[3, 1, 5, 8] \rightarrow [3, 5, 8] \rightarrow [3, 8] \rightarrow [8] \rightarrow []$$

And the coins you get equals:

$$167 = 3 * 1 * 5 + 3 * 5 * 8 + 1 * 3 * 8 + 1 * 8 * 1.$$

a. Define (in plain English) subproblems to be solved.

b. Write a recurrence relation for the subproblems

c. Using the recurrence formula in part b, write pseudocode to find the solution.

d. Make sure you specify

   i. base cases and their values

   ii. where the final answer can be found

e. What is the complexity of your solution?

10. **Online Questions.** Please go to DEN (https://courses.uscden.net/) and take the online portion of your assignment.