# CSCI 570 - Spring 2021 - HW 3

Due Sunday March 07 (by 4:00 AM)

For all divide-and-conquer algorithms follow these steps:

1. Describe the steps of your algorithm in plain English.
2. Write a recurrence equation for the runtime complexity.
3. Solve the equation by the master theorem.

For all dynamic programming algorithms follow these steps:

1. Define (in plain English) subproblems to be solved.
2. Write the recurrence relation for subproblems.
3. Write pseudo-code to compute the optimal value
4. Compute its runtime complexity in terms of the input size.

## 1

Solve the following recurrences by giving tight $\Theta$-notation bounds in terms of $n$ for sufficiently large $n$. Assume that $T(\cdot)$ represents the running time of an algorithm, i.e. $T(n)$ is positive and non-decreasing function of $n$ and for small constants $c$ independent of $n$, $T(c)$ is also a constant independent of $n$. Note that some of these recurrences might be a little challenging to think about at first. Each question has 4 points. For each question, you need to explain how the Master Theorem is applied (2 points) and state your answer (2 points).

- (a) $T(n) = 4T(n/2) + n^2 \log n$.
- (b) $T(n) = 8T(n/6) + n \log n$.
- (c) $T(n) = \sqrt{6006}T(n/2) + n^{\sqrt{6006}}$.
- (d) $T(n) = 10T(n/2) + 2^n$.
- (e) $T(n) = 2T(\sqrt{n}) + log_2 n$.

- $T(n) = T(n/2) - n + 10$
- $T(n) = 2^n T(n/2) + n$
- $T(n) = 2T(n/4) + n^{0.51}$
- $T(n) = 0.5T(n/2) + 1/n$
- $T(n) = 16T(n/4) + n!$

## 2

Consider an array $A$ of $n$ numbers with the assurance that $n > 2$, $A_1 \geq A_2$ and $A_n \geq A_{n-1}$. An index $i$ is said to be a local minimum of the array $A$ if it satisfies $1 < i < n$, $A_{i-1} \geq A_i$ and $A_{i+1} \geq A_i$.

(a) Prove that there always exists a local minimum for $A$.

(b) Design an algorithm to compute a local minimum of $A$.

Your algorithm is allowed to make at most $O(logn)$ pairwise comparisons between elements of $A$.

## 3

There are $n$ cities where for each $i < j$, there is a road from city $i$ to city $j$ which takes $T_{i,j}$ time to travel. Two travelers Marco and Polo want to pass through all the cities, in the shortest time possible. In the other words, we want to divide the cities into two sets and assign one set to Marco and assign the other one to Polo such that the sum of the travel time by them is minimized. You can assume they start their travel from the city with smallest index from their set, and they travel cities in the ascending order of index (from smallest index to largest). The time complexity of your algorithm should be $O(n^2)$. Prove your algorithm finds the best answer.

## 4

Erica is an undergraduate student at USC, and she is preparing for a very important exam. There are $n$ days left for her to review all the lectures. To make sure she can finish all the materials, in every two consecutive days she must go through at least $k$ lectures. For example, if $k = 5$ and she learned 2 lectures yesterday, then she must learn at least 3 today. Also, Erica's attention and stamina for each day is limited, so for $i$'th day if Erica learns more than $a_i$ lectures, she will be exhausted that day.

You are asked to help her to make a plan. Design an **Dynamic Programming** algorithm that output the lectures she should learn each day (lets say $b_i$), so that she can finish the lectures and being as less exhausted as possible(Specifically, minimize the sum of $max(0, b_i - a_i)$). Explain your algorithm and analyze it's complexity.

hint: $k$ is $O(n)$.

## 5

Due to the pandemic, You decide to stay at home and play a new board game alone. The game consists an array $a$ of $n$ positive integers and a chessman. To begin with, you should put your character in an arbitrary position. In each steps, you gain $a_i$ points, then move your chessman at least $a_i$ positions to the right (that is, $i' >= i + a_i$). The game ends when your chessman moves out of the array.
Design an algorithm that cost at most $O(n)$ time to find the maximum points you can get. Explain your algorithm and analyze its complexity.

## 6

Joseph recently received some strings as his birthday gift from Chris. He is interested in the similarity between those strings. He thinks that two strings $a$ and $b$ are considered J-similar to each other in one of these two cases:

1. $a$ is equal to $b$.
2. he can cut $a$ into two substrings $a_1, a_2$ of the same length, and cut $b$ in the same way, then one of following is correct:
   (a) $a_1$ is J-similar to $b_1$, and $a_2$ is J-similar to $b_2$.
   (b) $a_2$ is J-similar to $b_1$, and $a_1$ is J-similar to $b_2$.

Caution: the second case is not applied to strings of odd length.
He ask you to help him sort this out. Please prove that only strings having the same length can be J-similar to each other, then design an algorithm to determine if two strings are J-similar within $O(nlogn)$ time (where $n$ is the length of strings).

## 7

Chris recently received an array $p$ as his birthday gift from Joseph, whose elements are either 0 or 1. He wants to use it to generate an infinite long super-array. Here is his strategy: each time, he inverts his array by bits, changing all 0 to 1 and all 1 to 0 to get another array, then concatenate the original array and the inverted array together. For example, if the original array is [0, 1, 1, 0], then the inverted array will be [1, 0, 0, 1] and the new array will be [0, 1, 1, 0, 1, 0, 0, 1]. He wonders what the array will look like after he repeat this many many times.
He ask you to help him sort this out. Given the original array $p$ of length $n$ and two indices $a, b$ ($n \ll a \ll b$, $\ll$ means much less than) Design an algorithm to

calculate the sum of elements between $a$ and $b$ of the generated infinite array $\hat{p}$, specifically, $\sum_{a \leq i \leq b} \hat{p}_i$. He also wants you to do it real fast, so make sure your algorithm runs less than $O(b)$ time. Explain your algorithm and analyze its complexity.