CS141, Programming II Report **Due: 23:59, Wed 10/12, 2022**

Name: Majd Kawak / NetID: mkawa025 / CodeForces ID: mkawa025 / Student ID: 862273310

A. Merge Them!

Submission#: 175467040

in this program, I first declared an array to store all students scores. Then I went on and implemented Merge Sort algorithm where I divide array of scores into two recursively, till I reach 1 element or an array with a size of 1. When merging back from bottom(i.e. 2 arrays into 1) I count the difference between highest score and lowest in that array by subtracting first from last score. Add the number to a total int called candies. This was done recursively inside merge sort so each time 2 arrays gets merge, number of candies are calculated and saved. At the end I'm end up with total amount of candies(displayed using cout) professor Yihan need to get for students. Time complexity is $O(n\ log\ n)$.

B. Voting for the programming contest!

Submission#: 175467070

This program is very similar to part A in term of algorithm, I also implemented Merge Sort here. Firstly I allocated couple of ints, where I calculate number of departments needed which are half of total departments +1 department to win voting. Then I go on and sort departments from least amount of people to largest using Merge Sort Algorithm. Then I iterate through my array of total department up till I reach my number of departments needed to win, divide number of people of each department by 2 and add 1 to win that department. Add all people needed to one variable, at the end I'm end up with total people needed for professor Yihan to persuade. Time complexity is $O(n\ log\ n + n) = O(n\ log\ n)$.

C. Number Candles

Submission#: 175838424

In this program, I start off with allocating couple of arrays to handle best candle prices and combination. Then I move on to sort prices from lowest to highest with no duplicates using vector sort which cost $O(n\ log\ n)$(according to stackoverflow). Then I pass prices along with my total amount of dollars to a function that computes all possible Combination Sum of prices that equal to dollars. I implemented a recursive call to find this that will cost $O(n^2)$. Each array element will call this function recursively n times, with n elements in the array, total time complexity will be $O(n * n) = O(n^2)$. Then I got search for largest combos since I'm looking for larges number(quantity)$O(1)$. Iterate through my prices array and assign highest candle to a different array(minim price highest candle number)$O(1)$. Lastly I sort this array to shape highest number with candles, get the max element in this array and display it $O(n\ log\ n)$. I end it with $O(n\ log\ n) + O(n^2) + O(1) + O(1) + O(n\ log\ n) \leq O(n^3)$. Time complexity is $O(n^3)$.

D. Easy sorting

Submission#: 175467886

In this program, I start off with allocating ints and 2 arrays for my elements. While taking input I copy elements to first and second array at the same time. Then I go on and use Merge Sort to sort one of them. I iterate the sorted array and compare element to unsorted one, if they're the same meaning element has not been swapped. I keep track of all elements not in place and store result in a variable. at the end I divide number of element swapped by 2 because 2 elements not in place meaning they got swapped with each other. Time complexity is $O(n\ log\ n + n) = O(n\ log\ n)$.

References

[1] geeksforgeeks *Merge Sort* https://www.geeksforgeeks.org/merge-sort/

[2] tutorialcup *Combination Sum* https://www.tutorialcup.com/interview/array/combination-sum.html

[3] stackoverflow *What is the time complexity of std::sort() in the C++ standard library?* https://stackoverflow.com/questions/4484900/what-is-the-time-complexity-of-stdsort-in-the-c-standard-library