CSE 331 Homework 5

1. largestCommonSubstring function use divide and conquer algorithm. In divide and conquer algorithms, array divided into subarrays and finding largest common substring operation firstly applied on to these subarrays. Then the operation applied on subarrays results. For example, let assume that array = [“programmable”, “programmatic”, “programming”, “programmings”]. Firstly, “programmable” and “programmatic” strings are compared, and the result is “programma”. Secondly, “programming” and “programmings” strings are compared, and the result is “programming”. Lastly, “programma” and “programmings” strings are compared and the largest common substring is “programm”.

   The worst-case time complexity of the largestCommonSubstring function is O(mlogn) where n is the number of strings and m is the length of the largest string in the array. compareStrings method has O(m) worst-case time complexity because if all characters of two strings are matched, loop of the compareStrings method runs m times. So, T(n) = 2T(n/2) + O(m). We can inference T(n) = O(mlogn) with this formula.

2.

   a. findMaxProfitDAC function use divide and conquer algorithm with divideAndConqueer function. divideAndConqueer function takes array, start index and end index as input. If the start is equal to or greater than end, returns a garbage point. Otherwise, it calls divideAndConqueer function for right and left part of the array. In addition to the max profit in right and left part, there may be another day pair which low price day is on the left and high price day is on the right part of the array. After that, these three profits must be compared with each other and the points which provide max profit must be returned.

      The worst-case time complexity of the findMaxProfitDAC function depends on the time complexity of the divideAndConqueer function. In divideAndConqueer function, finding max or min takes O(n) in worst-case. Besides the finding max or min, divideAndConqueer function calls itself recursively with first and second half of the array. So, T(n) = 2T(n/2) + O(n). We can inference that, worst-time time complexity of the divideAndConqueer is O(nlogn).

b. Working principle of findMaxProfitNormal function is that iteration starts from end to the beginning of the array. If the current element is bigger than maxPrice, it is set as maxPrice and the maxPriceIndexPrev element is set as current element index. If the maxPrice - current element is bigger than max profit, maxPrice - current element is set as max profit, minPriceIndex is set as current element index, and maxPriceIndexCur is set as maxPriceIndexPrev. If all elements are visited, prints the minPriceIndex and maxPriceIndexCur.

   The time complexity of the findMaxProfitNormal function is O(n). Because no matter what, for loop runs n times, and there is no operation in for loop other than constant time operations.

c. The time complexity of not based on divide and conquer approach for this problem is better than the algorithm which based on divide and conquer algorithm. Because not based on divide and conquer algorithm takes linear time, whereas based on divide and conquer algorithm takes loglinear time.

3. findLICS function uses dynamic programming algorithm to find the longest increasing consecutive number sequence in an array. In this function, an array named LIS is created, with the same size as the array and it fills with 1s. A variable named LISMax is set as 1. After that, all elements of the array visited starting from last element to the beginning. If the element is less than right element, LIS of the right element is added to the LIS of the current element. If the LIS of the current element is bigger than the LISMax, LISMax is updated.

   The time complexity of the findLICS function is O(n). Because it has only one for loop and all operations in it takes constant time.

4.

a. findMaxPoints function uses dynamic programming to find the path which contains max points. Firstly, an array named MPP is created, with the same size with array. First row and first column of the MPP array is filled with the data of first row and first column of the array. Rest of the data of the MPP array is filled with the data of the same position of the array + maximum of the upper or right element of the MPP array. In the end, last element of the last row of the MPP array is the max points which can be collected from the array.

Time complexity of the findMaxPoints function is O(n*m), where n is the number of rows and m is the number of columns. The other for loops does not affect the time complexity of the findMaxPoints because they take less time than nested for loop. There is no complex operation in nested for loop. So, the time complexity of the function only depends on nested for loop.

b. findMaxPointsGreedy function uses greedy algorithm to find the path which contains max points. The path starts with the first element of the first row and continues with the maximum of the right and bottom elements until the end of the array.

The time complexity of the findMaxPointsGreedy is O(n). Because there is only one for loop and there is no early exit from it. Also, there is no complex operation inside the for loop.

c. In both brute-force solution and dynamic programming algorithm, all possible solutions are considered and best of them is chosen. On the other hand, greedy algorithm, does not care all possible solutions, it choses best of the given options. So, brute-force and dynamic programming algorithms correctness is the same and better than greedy algorithm.

The algorithm which has the best time complexity is greedy algorithm in these algorithms. Although it does not always give the correct result, greedy algorithm takes linear time. The second the best time complexity is time complexity of the dynamic programming algorithm. Because it gives correct result and still has less complexity than brute force (O(n*m) < O(2^(n*m)).