CSE 321

Introduction To Algorithm Desing

Homework #5

Berkan AKIN

171044073

Question #1

In the worst case scenario, where every character in the string is different, all characters will need to be compared and all operations will be performed, resulting in a time complexity of O(n).

In the average case, if we assume that on average, more than half of the characters in the string are the same, then the number of operations will decrease logarithmically. This results in a time complexity of O(log n).

In the best case, where the length of the string is equal to 1, the operations will stop, resulting in a time complexity of O(1).

Case	Time Complexity
Worst Case	O(n)
Average	O(log n)
Best	O(1)

Question #2

Part 1

In the worst case, the time complexity would be O(n), when the length of the prices array is greater than 1. This happens when the array is repeatedly divided in half until the base case is reached. The time complexity in this case is O(n) because the size of the input array is reduced by half in each recursive call.

In the average case, the time complexity would be somewhere between O(1) and O(n), depending on the input array. If the input array is such that it takes more than one recursive call to reach the base case, then the time complexity would be closer to O(n). If the input array is such that the base case is reached in the first recursive call, then the time complexity would be closer to O(1).

Case	Time complexity
Best (n = 1)	O(1)
Worst (n > 1)	O(n)
Average (varies)	O(n)

Part 2

In the best case, the minimum and maximum values are found on the first iteration of the loop, so the time complexity is O(1). In the worst case, the minimum and maximum values are found on the last iteration of the loop, so the time complexity is O(n). In the average case, the time complexity is also O(n), since the loop will iterate through the list of prices an average of n/2 times.

Case	Time Complexity
Best	O(1)
Worst	O(n)
Average	O(n)

Question #3

The worst case occurs when the longest increasing subarray is the entire array, so the function must compare every element to every other element to find the longest subarray. The best case occurs when the longest increasing subarray is the first element of the array, so the function will find the longest subarray in just one iteration. The average case time complexity is also O(n^2), since the time complexity is not affected by the specific input, but rather by the size of the input (in this case, the length of the array).

Case	Time complexity
Worst	O(n^2)
Average	O(n^2)
Best	O(n^2)

Question #4

Part 1

The time complexity of the function dynamic_programing is O(2^(x+y)) This is because at each step of the function, there are two recursive calls made: one for the right and one for the bottom. The function will make these calls until it reaches the base case (when x and y are both 0). The best case scenario for this function occurs when the base case is reached on the first call. In this case, the function will only make one call and the time complexity will be O(1). The worst case scenario occurs when the base case is not reached until x and y are both at their maximum values. In this case, the function will make 2^(x+y) calls and the time complexity will be O(2^(x+y)). The average case time complexity will depend on the specific input values for x and y.

Part 2

The time complexity of this function is O(n), where n is the number of elements in the input map. This is because the function processes each element of the map once. The best-case time complexity is also O(n), which occurs when the function processes all the elements of the map. The worst-case time complexity is also O(n), which occurs when the function processes all the elements of the map. The average-case time complexity is also O(n), which occurs when the function processes all the elements of the map an average number of times.

Case	Time Complexity
Best	O(n)
Worst	O(n)
Average	O(n)