CSE321 – Introduction to Algorithm Design

HW-4

Yusuf Arslan
200104004112

Q1-

The time complexity of the '`longestCommonPrefix`' function is O(n * m * log(n)), where n is the number of strings in the input array and m is the length of the longest common prefix.

The function recursively divides the input array into two halves at each step until it reaches a base case where the input array has only one element. For each recursive step, the function compares the characters of the two longest common prefixes found in the left and right halves, which takes O(m) time. Since the input array is divided into halves at each step, the total number of recursive steps is log(n).

Therefore, the total time complexity is O(n * m * log(n)).

Q2-

😟

Q3-

The algorithm uses a single loop to iterate through the input array and another iteration for finding max values which is also has same size with the input array, so the time complexity is directly proportional to the size of the input. In each iteration of the loop, a constant number of operations are performed, so the time complexity is linear.

Therefore, the time complexity of the algorithm is O(2*n) which equals to O(n).

Q4-

a- The time complexity of the dynamic programming solution is O(nm), where n is the number of rows in the map and m is the number of columns in the map. This is because the solution involves looping through each element in the map once and performing a constant amount of work for each element.

   The space complexity is also O(nm), as it involves storing the solutions to the subproblems in a 2D array.

b- The implementation has a time complexity of O(nm) in greedy algorithm solution and a space complexity of O(nm), as it involves looping through each element in the map once and storing the path in an array.

c- Dynamic programming:

a. Time complexity: O(nm)
b. Space complexity: O(nm)
c. Guaranteed to find the global optimal solution
d. May be more efficient for large maps

Greedy algorithm:

e. Time complexity: O(nm)
f. Space complexity: O(nm)
g. May find a good solution, but not guaranteed to find the global optimal solution
h. May be more efficient for large maps

Brute force:

i. Time complexity: O(2^nm)
j. Space complexity: O(nm)
k. Guaranteed to find the global optimal solution
l. May be very inefficient for large maps