70087 13

Coursework 3

Submitters

sf23 Shihan Fu

8/10 Good. You identified the correct best case in (b) but it still takes longer than you thought.

70087 Algorithms
Assessed Coursework

Shihan Fu

February 28, 2024

4/5

1. Answer to Question 1.

```
procedure COUNT_SORTED($A, B$)
    $N \leftarrow A.length$
    for $i \leftarrow 1$ to $N$ do                                   ✓
        $dp[i] \leftarrow 1$
    end for
    for $i \leftarrow 1$ to $N$ do                              What is Math?
        for $j \leftarrow i-1$ to 0 do
            if $A[i] > A[j]$ then
dp[i]           $sp[i] \leftarrow Math.max(dp[i], dp[j]+1)$        ✓
            end if
        end for
    end for
    return MAX(dp[N])                          max(dp[N]) is just dp[N]    ✗
end procedure
```

2. Answer to Question 2. If my procedure could not use dynamic programming, the time complexity will increase greatly because each subproblems will need to be computed recursively. This may result in an exponential result.

   For the worst case, to get dp[N], we need to compute and compare every dp[i] where i between 1 to N. ✓

   In this case, T(N) = $\Theta(1)$ if N = 0

   T(N) = T(N-1) + T(N-2) + ... + T(0) + Nc + d if N > 1 ✓

   So, T(N-1) = T(N-2) + ... + T(0) + (N-1)c + d if N > 1

   In this way, we can get

   T(N) = 2T(N-1) + c if N > 1

   T(N) = 4T(N-2) + 3c if N > 2 ✓

   ...

   T(N) = $2^i T(N-i)$ + $(2^i - 1)c$ if N > i where $1 \leq i < N$ ✓

   For i = N - 1, we have T(N) = $2^{N-1}T(1)$ + $(2^{N-1} - 1)c$

   T(1) = T(0) + c + d

   So T(N) = $2^{N-1}T(0)$ + $2^{N-1}c$ + $(2^{N-1} - 1)c$ + $2^{N-1}d$

4/5

therefore, T(N) = $(2^N)c$ + $2^{N-1}T(0)$ + $2^{N-1}d$ -c = $\Theta(2^N)$ ✓

For the best case, we assume that the array is in an decreasing order. The result of each comparision will be false and each dp[i] will be 1. ✓

In this case, T(N) = $\Theta(1)$ if N = 0

T(N) = Nc if N > 1

Therefore, T(N) = $\Theta(N)$ ✗

Hence, in total, the upper bound is $O(2^N)$, and the lower bound is $\Omega(N)$. ✗

This is only enough time to check for sequences that end with A[N].
You have to check for sequences ending at any position. In other words you still have to execute the two nested loops, but the condition in line 8 is always false. So, there are no recursive calls, but it still takes Θ(N^2) time.