CMPSC 465 Data Structures & Algorithms
Fall 2025 Ke Chen and Yana Safonova

Quiz 10 Solution

Lecture Section: **Monday, Nov 10, 2025**

Student Name: **PSU Email ID:**

1. (2 pts.) What are the key components of formulating and solving a problem using Dynamic Programming? Select all that apply.

(a) Defining a recurrence.

(b) Solving the base case.

(c) Defining a greedy heuristic.

(d) Defining subproblems with optimal substructure.

Answer (a), (b), and (d). A greedy heuristic is not needed for DP since we are computing all subproblems in a bottom-up manner.

2. (2 pts.) The space complexity of the dynamic programming solution for the finding the Longest Increasing Subsequence (LIS) is:

(a) $O(n)$ - can be done in linear space

(b) $O(n^2)$ - need space for all $O(n^2)$ subproblems

(c) $O(1)$ - no space overhead needed, we only incur a cost in time

(d) $O(n \log n)$

Answer (a) $O(n)$ Linear space is sufficient for the dp array. Can be verified by the definition of the subproblems.

3. (2 pts.) Consider the two strings $x := ACTGGACTT$ and $y := AGTCGTTT$. What is the edit distance $d(x,y)$?

(a) 3

(b) 4

(c) 5

(d) 6

Answer (b) 4. 3 substitutions and 1 insertion in the optimal alignment.

4. (2 pts.) In class, you learned that the minimum edit distance problem can be solved in polynomial time using Dynamic Progamming. Suppose there are two strings of equal length n. What is the time complexity of determining the **optimal alignment** of the two sequences (not just the minimum edit distance)?

(a) $O(n^2)$ - same runtime as the edit distance problem

(b) $O(n^3)$ - Need an additional factor of $O(n)$ to compare the best alignment for each subproblem.

(c) $O(n^4)$ - Need an additional factor of $O(n^2)$ to compare the best alignment for each subproblem for both strings.

(d) Cannot be computed in polynomial runtime.

Answer (a) $O(n^2)$. This can be done in the same runtime as the edit distance problem and the optimal alignment can be recovered via traceback pointers.

5. (2 pts.) Consider the sequence $S = (4,6,2,3,8,1,5)$. Recall that we defined the following recurrence for finding the length of the Longest Increasing Subsequence ending at a_j:

$$L(j) = \begin{cases} 1 + \max_{i<j, a_i<a_j}(L(i)) \\ 1, \text{ if no such } i \end{cases}$$

What is $L(4)$ while trying to find the length of the LIS in S? Assume that S is 1-indexed.

(a) 2

(b) 3

(c) 4

(d) 5

Answer (a) 2. The longest subsequence ending in the 4th element 3 is $(2,3)$, which has a length of 2.