1 Longest Common Subsequence Forensics

We are computing the longest common subsequence between two strings of length $S = S_1S_2S_3S_4$ character of the second string are equal? They are different.

Consider the LCS problem from lecture slides. We saw a top-down dynamic programming approach to solve this problem. Now we'd like to see how a bottom-up approach to solve this problem would look like. Which one of the following statements is correct?

- Both of the above.
- No, the best memory complexity is $O(mn)$.
- Yes, we can reduce the memory complexity to $O(m)$.
- No, we can reduce the memory complexity to $O(n)$.

Consider the maximum independent set on trees problem from the lecture slides. We saw a top-down dynamic programming approach to solve this problem. Now we'd like to see how a bottom-up approach to solve this problem would look like. Which one of the following statements is correct?

- Yes, we can reduce the memory complexity to $O(mn)$.
- No, the best memory complexity is $O(mn)$.
- Yes, we can reduce the memory complexity to $O(m)$.
- Yes, we can reduce the memory complexity to $O(n)$.

What is the best run-time for the bottom-up approach to solve the MIS on a tree problem?

- $O(n^6)$
- $O(n^2)$
- $O(n^3)$
- $O(n^4)$

What can be said about $X_i$ when we are filling up the $i$-th row of our dynamic programming table $C$?

- They could be equal or different.
- They are different.
- They are equal.

What is the minimum possible value for $i$ when we are filling up the $i$-th row of our dynamic programming table $C$?

- The entry $Y_{i,j}$ in the $i$-th row and $j$-th column of $C$ gives us the maximum value we can obtain from a knapsack of capacity $j$.
- The array $C$ whose entry $C_{i,j}$ is the length of the longest common subsequence between the prefix of length $i$ from $S_1$ and the prefix of length $j$ from $S_4$.
- The value of $T$.
- The value of $C$.

We are computing the longest common subsequence between two strings of length $S = S_1S_2S_3S_4$. Proposition $C$ gives us the maximum value we can obtain from a knapsack of capacity $j$.

- They could be equal or different.
- They are different.
- They are equal.

When we are filling up the $i$-th row of our dynamic programming table $C$, what do we need to do to access the $j$-th row of $C$?

- We need to access the first row of $C$.
- We need to access the preceding row of $C$.
- We need to access the value in the $i$-th row and ($j-1$)-th row.

Suppose that in some (possibly different) instance of the longest common subsequence problem, we have $C_4 = C_3 = 1$. Does that necessarily mean the 11th character of the first string and the 11th character of the second string are equal?

- Yes
- No
- No
- Yes

We fill a knapsack of capacity $j$. How many items do we put in the knapsack optimally, how many times do we put in the knapsack?

- $O(n)$
- $O(1)$
- $O(n)$
- $O(n)$

Knapsack Forensics

- We fill a knapsack of capacity $j$. How many times do we put in the knapsack optimally, how many times do we put in the knapsack?
- $O(n)$
- $O(1)$
- $O(n)$
- $O(n)$

If we fill a knapsack of capacity $j$, optimally, how many times do we put in the knapsack?

- $O(n)$
- $O(1)$
- $O(n)$
- $O(n)$

Maximum Independent Set on a Tree

Consider the maximum independent set on trees problem from the lecture slides. We saw a top-down dynamic programming approach to solve this problem. How can we reduce the memory complexity to $O(mn)$?

- In order to solve this problem bottom-up we need to order the vertices by decreasing DFS start times.
- No, the best memory complexity is $O(n^2)$.
- Both of the above.
- Any ordering would work.

4 LCS Space Complexity

Consider the LCS problem from lecture slides. We saw a top-down dynamic programming algorithm for it. Given input strings of lengths $n$ and $m$, what is the memory complexity of this algorithm?

- $O(nm)$
- $O(n+m)$
- $O(nm+nm)$
- $O(nm+nm)$

Knapsack Forensics

- Suppose we are trying to solve an instance of the unbounded knapsack problem. We fill the array $C$ whose entry $C_{i,j}$ is the length of the longest common subsequence between the prefix of length $i$ from $S_1$ and the prefix of length $j$ from $S_4$.

2 Knapsack Forensics