Week 9: PYQs

25 June 2024 14:31

1. May 23

In the Longest Common Subsequence problem we are given two sequences $a_1, a_2 \ldots a_m$ and $b_1, b_2, \ldots b_n$. To get the length of Longest Common Subsequence at $LCS[m][n]$, the recursion formula is given as follows to fill matrix $LCS[i][j]$ where $0 \leq i \leq m$ and $0 \leq j \leq n$.

$$LCS[i,j] = \begin{cases} 0, & if\ i = 0\ or\ j = 0 \\ Expression1, & if\ a_i = b_j \\ Expression2, & if\ a_i \neq b_j \end{cases}$$

Which of the following represent the correct expression for $Expression1$ and $Expression2$?

Options :

[redacted] **Expression1:** $LCS[i-1, j-1]$
Expression2: $max(LCS[i-1, j], LCS[i, j-1])$

[redacted] **Expression1:** $1 + LCS[i-1, j-1]$
Expression2: $min(LCS[i-1, j], LCS[i, j-1])$

[redacted] **Expression1:** $1 + LCS[i-1, j-1]$
Expression2: $1 + max(LCS[i-1, j], LCS[i, j-1])$

[redacted] **Expression1:** $1 + LCS[i-1, j-1]$
Expression2: $max(LCS[i-1, j], LCS[i, j-1])$

2. May 23

Question Label : Short Answer Question

Let M_1, M_2, M_3, M_4 be 4 matrices of dimensions $10 \times 100, 100 \times 20, 20 \times 5, 5 \times 80$ respectively.

What is the minimum number of scalar multiplications required to multiply M_1, M_2, M_3, M_4 using basic matrix multiplication ?

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

Text Areas : PlainText

Possible Answers :

[redacted]

3. Jan 24

The **Longest Increasing Subsequence** problem is defined as below.

Given a list `L` of size `n` non-negative integers, determine the Longest Increasing Subsequence(LIS) i.e., the longest possible subsequence in which the elements of the subsequence are sorted in increasing order.

Consider the following function `LIS` which takes list `L` as input and returns the length of the Longest Increasing Subsequence.

```
1  def LIS(L):
2      n = len(L)
3
4      Lis = [1]*n  #initialize with all 1's
5
6      for i in range(1, n):
7          for j in range(0, i):
8              if L[i] > L[j]:
9                  Lis[i] = ____ # Check here
10
11     return max(Lis)
```

Based on the above data, answer the given subquestions.

In the given code, what expression should be placed at the place of `____` so that it return the correct output?

Options :

- [redacted] `max(Lis[i], Lis[j]+1)`
- [redacted] `max(Lis[i], Lis[j])`
- [redacted] `max(Lis[i], Lis[j+1]+1)`
- [redacted] `max(Lis[i], Lis[j-1]+1)`

4. Jan 24

Question Label : Multiple Choice Question

What is the time complexity of function `LIS()` ?

Options :

- [redacted] $O(n)$
- [redacted] $O(n \log n)$
- [redacted] $O(\log n)$
- [redacted] $O(n^2)$

5. Jan 23

Your final End term exams are going to be over and you are catching up on Netflix. You have a schedule of interesting live shows during the next day. You hate to start or stop watching a show midway, so your aim is to watch as many complete shows as possible during the day.

Suppose there are n such shows $S_1, S_2, \ldots, S_n$ available during the coming day. The shows are ordered by starting time, so for each $i \in 1, 2, \ldots, n-1$, S_i starts before S_{i+1}. However, show S_i may not end before S_{i+1} starts, so for each $i \in 1, 2, \ldots, n-1$, $Next[i]$ is the smallest $j > i$ such that S_j starts after S_i finishes if such a j exists, otherwise `-1`.

Given the sequence $S_1, S_2, \ldots, S_n$ and the values $Next[i]$ for each $i \in 1, 2, \ldots, n$, your aim is to compute the maximum number of complete shows that can be watched.

Based on the above data, answer the given subquestions.

Question Label : Multiple Choice Question

Consider the following dynamic programming approch:

Let $Watch[i]$ denote the maximum number of complete shows that can be watched among $S_i, S_{i+1}, \ldots, S_n$. Which of the following is a correct inductive formulation of $Watch[i]$ for $i \in n-1, n-2, \ldots, 2, 1$? Consider initially $Watch[n] = 1$.

Options :

[redacted]

$$Watch[i] = \begin{cases} Watch[i+1], & if\ \ Next[i] = -1 \\ max(Watch[Next[i]], Watch[i+1]), & if\ \ Next[i] \neq -1 \end{cases}$$

[redacted]

$$Watch[i] = \begin{cases} Watch[i+1], & if\ \ Next[i] = -1 \\ max(Watch[Next[i]], 1 + Watch[i+1]), & if\ \ Next[i] \neq -1 \end{cases}$$

[redacted]

$$Watch[i] = \begin{cases} Watch[i+1], & if\ \ Next[i] = -1 \\ max(1 + Watch[Next[i]], Watch[i+1]), & if\ \ Next[i] \neq -1 \end{cases}$$

[redacted]

$$Watch[i] = \begin{cases} 1 + Watch[i+1], & if\ \ Next[i] = -1 \\ max(Watch[Next[i]], Watch[i+1]), & if\ \ Next[i] \neq -1 \end{cases}$$

[redacted] $Watch[i] = \begin{cases} max(Watch[Next[i]], Watch[i+1]), & if\ Next[i] \neq -1 \end{cases}$

6. Jan 23

How much time will the given dynamic programming approach take to compute the answer? Assume you have direct access to the *Next* list as well, and you don't have to worry about computing it on your own.

Options :

[redacted] $O(n^2)$

[redacted] $O(n^3)$

[redacted] $O(n\ log\ n)$

[redacted] $O(n)$

7. Sep 23

Question Label : Multiple Choice Question

The **longest common substring** of two strings is a contiguous longest string that is a substring in both strings.

Suppose you are given two strings S_1 and S_2:

$S_1 = a_0, a_1, \ldots, a_{n-1}$

$S_2 = b_0, b_1, \ldots, b_{m-1}$

Your task is to find out the length of the longest common substring in S_1 and S_2

Consider the following initialization of a two-dimensional array DP of size $n+1, m+1$.

DP	0	1	2	..	j	..	$n-1$	n
0								0
1								0
2								0
..								0
i								0
..								0
..								0
$m-1$								0
m	0	0	0	0	0	0	0	0

Consider that we start at the bottom right ($DP[n-1][m-1]$) and fill DP array row by row or column by column and want to get the length of the longest common substring for string S_1 and S_2 as $max(DP)$ (maximum value in DP array).

Which of the following inductive structures is correct to fill array DP?

column by column and want to get the length of the longest common substring for string S_1 and S_2 as $max(DP)$ (maximum value in DP array).

Which of the following inductive structures is correct to fill array DP?

Options :

[redacted]

$$DP[i,j] = \begin{cases} 1 + max(DP[i+1,j], DP[i,j+1]), & if\ a_i = b_j \\ 0, & if\ a_i \neq b_j \end{cases}$$

[redacted]

$$DP[i,j] = \begin{cases} DP[i+1,j+1], & if\ a_i = b_j \\ 0, & if\ a_i \neq b_j \end{cases}$$

[redacted]

$$DP[i,j] = \begin{cases} 1 + DP[i+1,j+1], & if\ a_i = b_j \\ 0, & if\ a_i \neq b_j \end{cases}$$

[redacted]

$$DP[i,j] = \begin{cases} 1 + DP[i+1,j+1], & if\ a_i = b_j \\ 1 + max(DP[i+1,j], DP[i,j+1]), & if\ a_i \neq b_j \end{cases}$$

8. Sep 22

Question Label : Multiple Choice Question

An algorithm to find the length of the longest strictly increasing sequence of numbers in list $A_{0...n-1}$ is given below.

1. $n = length(A)$
2. $Initialize\ list\ L_{0...n-1} = 0$
3. $L_0 = 1$
4. $For\ all\ i,\ start\ from\ index\ 1\ to\ n-1:$
5. **Inductive structure**
6. $return\ max(L)$

Note: L_j is the length of the longest strictly increasing sequences ending at A_j, where $0 \leq j \leq n-1$

Which of the following is the correct **inductive structure** to fill at step 5 to return the correct result?

Options :

[redacted]

$$L_i = \begin{cases} 1 + L_{i+1}, & if\ A_i > A_{i+1} \\ 1, & Otherwise \end{cases}$$

$$L_i = \begin{cases} 1 + L_{i-1}, & if\ A_i > A_{i-1} \\ \end{cases}$$

[redacted] $L_i = \begin{cases} 1 + L_{i-1}, & if\ A_i > A_{i-1} \\ 1, & Otherwise \end{cases}$

[redacted] $L_i = \begin{cases} 1 + L_{i-1}, & if\ A_i < A_{i-1} \\ 1, & Otherwise \end{cases}$

[redacted] $L_i = \begin{cases} 1 + L_{i-1}, & if\ A_i >= A_{i-1} \\ 1, & Otherwise \end{cases}$

9. May 22

Question Label : Short Answer Question

Let $A1, A2, A3, A4$ be 4 matrices with dimensions $(10 \times 5), (5 \times 20), (20 \times 10), (10 \times 15)$ respectively. What is the minimum number of scalar multiplications required to find the product $A1 \times A2 \times A3 \times A4$?

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

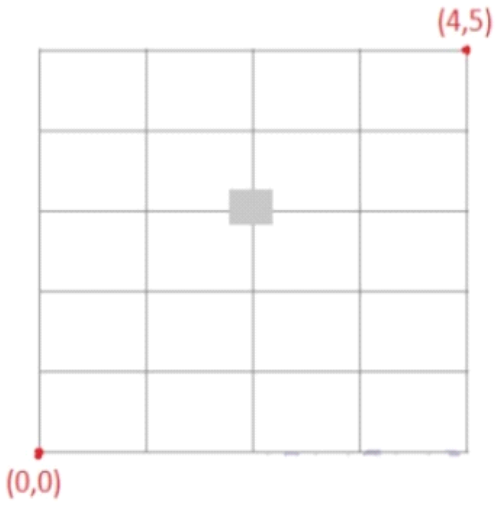
Text Areas : PlainText

Possible Answers :

[redacted]

10. May 22

Consider the following grid.



How many unique paths are available from (0,0) to (4,5)? Condition is that you can only travel one step right or one step up at a time, and the gray box at intersection point (2,3) represents a blockage.

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

Text Areas : PlainText

Possible Answers :