Dynamic Programming for dummies

Stanislav Ostapenko

January 28, 2024

300. longest increasing subsequence

Let's define $L(i)$ as the length of the longest strictly increasing subsequence ending at index i. The recurrence formula for the longest strictly increasing subsequence is given by:

$$L(i) = 1 + \max_{\substack{j<i \\ \text{arr}[j]<\text{arr}[i]}} L(j)$$

This equation states that the length of the longest increasing subsequence ending at index i is 1 plus the maximum length obtained by considering all indices j less than i, where the corresponding element arr[j] is less than arr[i].

```
class Solution {
        private int max(int[] L) {
                int maxLength = Integer.MIN_VALUE;
                for (final int length : L) {
                        maxLength = Math.max(maxLength, length);
                }
                return maxLength;
        }

        public int lengthOfLIS(int[] nums) {
                int n = nums.length;
                int[] L = new int[n];
                // Initialize the array with minimum length 1 for each index
                Arrays.fill(L, 1);

                // Iterate to fill in the values of L(i) using the recurrence relation
                for (int i = 1; i < n; i++) {
                        for (int j = 0; j < i; j++) {
                                if (nums[i] > nums[j] && L[i] < L[j] + 1) {
                                        L[i] = L[j] + 1;
                                }
                        }
                }
                // Find the maximum value in the array L
                return max(L);
        }
}
```