report

Haopeng Chen 3220103347@zju.edu.cn

December 15, 2024

The idea of Design:

- Design a function LIS to find the strictly increasing subsequence within a given sequence arr. In the function, maintain a max_length array, where the ith position records the longest length of the subsequence ending with $arr[i]$.
- Also, maintain a $last\_location$ array, where the ith position records the position in arr of the previous number in the longest subsequence ending with $arr[i]$.
- The logic of the operation is to start iterating from the first position of the arr array, and in each iteration, compare every element before the current position with the current element. If it is smaller than the current element, compare whether the length is the longest after adding the current element to that subsequence. This process continuously updates the max_length and $last\_location$ arrays.
- After the loop ends, find the position with the maximum max_length to obtain the maximum length of the subsequence, MaxLength, and use the $last\_location$ array to get the complete sorted elements of the subsequence.

Example:

- give a vector $[2, 43, 23, 65, 32, 6, 33, 74, 13, 5, 75, 21]$
- first get the max_length vector $[1, 2, 2, 3, 3, 2, 4, 5, 3, 2, 6, 4]$ and the $last\_location$ vector $[-1, 0, 0, 1, 2, 0, 4, 6, 5, 0, 7, 8]$
- the position of the max length is at 10, so the MaxLength is 6.
- start from postion 10,get number $arr[10]$,which is 75.then $last\_location[10] = 7$,get number $arr[7]$,which is 74,...,after a series of similar operations,finally we get $child\_arr$,which is $[2, 23, 32, 33, 74, 75]$.

Time complexity:

To get max_length,the time we need is:

$$1 + 2 + ... + n - 1 = \frac{n(n-1)}{2}.$$

To get $child\_arr$,the time needed is approximately:

$$O(n)$$

so Time complexity is $O(n^2)$