

1. 动态规划: 300. 最长递增子序列

解题思路:

10 9 2 5 3 7 10 18

状态表示: $dp[i]$: 表示以 i 位置结尾最长子序列

状态转移方程:

状态: i 本身, i 之前

$dp[j] + 1$ 表示 i 之前最长递增子序列

Diagram illustrating the state transition for the Longest Increasing Subsequence problem. The array elements are 10, 9, 2, 5, 3, 7, 10, 18. The state $dp[i]$ represents the length of the longest increasing subsequence ending at index i . The state transition equation is $dp[i] = \max(dp[j] + 1)$ for all $j < i$ such that $arr[j] < arr[i]$. The diagram shows the array and the state transition logic, with $dp[j] + 1$ representing the length of the longest increasing subsequence ending at j plus the current element at i .

代码:

```

1  class Solution {
2  public:
3      int lengthOfLIS(vector<int>& nums) {
4          int n = nums.size();
5          int max1 = 1;
6          vector<int> dp(n, 1);
7          for (int i = 1; i < n; i++)
8          {
9              for (int j = 0; j < i; j++)
10             {
11                 if (nums[i] > nums[j])
12                 {
13                     dp[i] = max(dp[i], dp[j] + 1);
14                 }
15             }
16             max1 = max(max1, dp[i]);
17         }
18         return max1;
19     }
20 };

```

2. 动态规划: 376. 摆动序列

代码:

```
1 class Solution {
2 public:
3     int wiggleMaxLength(vector<int>& nums) {
4         int n = nums.size();
5         vector<int> f(n,1); // up
6         auto g = f;        //down
7         int ret = 1;
8         for(int i = 1;i < n;i++)
9         {
10             for(int j = 0;j < i;j++)
11             {
12                 if(nums[j] > nums[i])
13                 {
14                     g[i] = max(g[i],f[j] + 1);
15                 }
16                 if (nums[j] < nums[i])
17                 {
18                     f[i] = max(f[i],g[j] + 1);
19                 }
20             }
21             ret = max(max(ret,f[i]),g[i]);
22         }
23         return ret;
24     }
25 }
26 }
27 };
```