动态规划

LIS

$a\{n\}$

- 设计状态: $dp[i]$ 代表数列 $a\{n\}$ 的前 i 个数的最长不下降子序列长度。
- 初始状态: $dp[1] = 1$
- 转移方程: $dp[i] = \max(dp[i], dp[j] + 1)\ (j < i\ \ \&\ \ a[i] \geq a[j])$
- 结果: $dp[n]$

贪心 + 二分

```
int lis(const vector<int>& v)    // 最长不下降子序列
{
    vector<int> d = {v.front()};
    for(size_t i=1;i<v.size();i++)
    {
        if(v[i]>=d.back()) d.push_back(v[i]);
        else *upper_bound(d.begin(), d.end(), v[i]) = v[i];
    }
    return d.size();
}
```

LCS

普通

- 设计状态: $dp[i][j]$ 代表数列 $a\{n\}$ 的前 i 个数以及数列 $b\{n\}$ 的前 j 个数的 LCS 长度。
- 初始状态: $dp[0][0] = 0$
- 转移方程:

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

- 结果: $dp[n][m]$

全排列

$a \to A[1]\ b \to A[2]\ c \to A[3]\ d \to A[4]\ e \to A[5]$

{a}: a b c d e {b}: c b a d e

LCS 长度没有发生变化。P 是 a 与 b 的 LCS, P 一定既是 a 的子序列也是 b 的子序列。P 一定递增。最长的 $P => b$ 的最长上升子序列。