1 Classical Problems

1.1 最长公共子序列

令 dp[i][j] 表示以第一个串的第 i 位与第二个串的第 j 位结尾的最长公共子序列长度。

状态的转移有三种：

$$dp[i][j] = \max \begin{cases} dp[i-1,j] \\ dp[i,j-1] \\ dp[i-1,j-1]+1, a[i]=b[j]. \end{cases}$$

1.2 最长公共上升子序列

1.2.1 三维 DP

令 dp[i,j,k] 表示以第一个串的第 i 位与第二个串的第 j 位结尾，最后一位是 k 的最长公共上升子序列的长度，则有状态转移方程：

$$dp[i,j,k] = \max \begin{cases} dp[i-1,j,k] \\ dp[i,j-1,k] \\ dp[i-1,j-1,k'], k<k'. \end{cases}$$

1.2.2 降维优化

dp[i,j] 表示以 a 串的第 i 位结尾，b 串的第 i-j 位结尾的 LCS，且最后元素为 b[j]，则有：

$$dp[i,j] = \max \begin{cases} dp[i-1,j] & \\ dp[i-1,j'], & j'<j, \\ & b[j']<b[j], \\ & a[i]=b[j]. \end{cases}$$

1.3 矩阵乘法

矩阵：一个 $N \times M$ 的矩阵被定义为一个 N 行 M 列的数组，数组中的每个元素都是一个实数。一个向量可以表示为一个 $M \times 1$ 的矩阵。

应用：

矩阵乘向量：

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{21} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

有一个任意的向量 $x(x_0, y_0)$ 旋转$\theta°$，则有：

$$x_1 = x_0 \cdot \cos\theta - y_0 \cdot \sin\theta$$
$$y_1 = x_0 \cdot \sin\theta + y_0 \cdot \cos\theta$$

矩阵乘法：定义矩阵乘法运算规则如下：

$$\begin{aligned} C[i,j] &= \sum_{k=1}^{m} A_{ik} B_{kj} \\ &= A_{i1}B_{1j} + A_{i2}B_{2j} + \cdots + A_{im}B_{mj} \end{aligned}$$

时间复杂度为 $O(n \times m \times k)$.

矩阵乘法与 DP 的关系：矩阵乘法优化递推转移。

例题：斐波那契数列升级版

题解：矩阵快速幂优化递推转移。

（抱歉，Atom 崩了，推导过程没写，具体过程可参照题解）