知识精炼（二）

主讲人：邓哲也

HDU 2294 Pendant

你有 k 种珍珠，想用它们组成一个长度在1～n之间的一条首饰并且满足 k 种珍珠都至少出现一次。

问有多少种不同的首饰。

$n \leqslant 10^9$, $k \leqslant 30$

样例：

n = 3, k = 2

答案：8（ab, ba, aab, aba, abb, baa, bab, bba）

HDU 2294 Pendant

用 f[i][j] 表示现在拼出了长度为 i 的序列，用了 j 种珍珠。

决策是第 i 个珍珠是否使用新的种类：

如果是旧的，那么有 j 种可能： j * f[i-1][j]

如果是新的，那么有 k-j+1 种可能：(k-j+1)*f[i-1][j-1]

因此我们得到了：

f[i][j] =

```
    j * f[i - 1][j] + (k - j + 1) * f[i - 1][j - 1];
```

HDU 2294 Pendant

答案就是 f[k][k] + f[k+1][k] + … + f[n][k]

由于 n 特别大，我们只能考虑矩阵乘法优化。

显然 f[i] 都是由 f[i - 1] 转移过来的。

那么我们可以把 f[i] 看作是一个长度为 k + 1 的向量。

[f[i][0], f[i][1], …, f[i][k]]

然后就可以构造转移矩阵 A。

这样 f[i] = f[i-1] * A

HDU 2294 Pendant

$f[n] = f[0] * A^n$

$f[n - 1] = f[0] * A^{n-1}$

…

因此我们需要计算出 $f[0] * (I + A + A^2 + \cdots + A^n)$

这是一个经典问题，二分 + 矩阵快速幂即可解决。

HDU 2294 Pendant

```
struct matrix{
      int data[35][35];
}a;

matrix mul(matrix a, matrix b){
    matrix c;
    memset(c.data, 0, sizeof(c.data));
    for (int i = 1;i <= n;i ++)
        for (int j = 1;j <= n;j ++)
            for(int k = 1;k <= n;k ++)
                c.data[i][j] = (c.data[i][j] + 1LL *
a.data[i][k] * b.data[k][j]) % m;
    return c;
}
```

HDU 2294 Pendant

```
matrix add(matrix a, matrix b){
    for (int i = 1;i <= n;i ++)
        for (int j = 1;j <= n;j ++)
            a.data[i][j] = (a.data[i][j] +
b.data[i][j]) % m;
    return a;
}
```

HDU 2294 Pendant

```
matrix quickpow(matrix a, int k){
    matrix c;
    memset(c.data, 0, sizeof(c.data));
    for (int i = 1;i <= n;i ++)
        c.data[i][i] = 1;
    while(k){
        if (k & 1) c = mul(c, a);
        k >>= 1;
        a = mul(a, a);
    }
    return c;
}
```

HDU 2294 Pendant

```
matrix sum(matrix a, int k){
    if (k == 1) return a;
    matrix c;
    memset(c.data, 0, sizeof(c.data));
    for (int i = 1;i <= n;i ++)
        c.data[i][i] = 1;
    c = add(c, quickpow(a, k >> 1);
    c = mul(c, sum(a, k >> 1));
    if (k & 1) c = add(c, quickpow(a, k));
    return c;
}
```

HDU 2294 Pendant

```
matrix a;

for(int i = 1;i <= k;i ++){
      a.data[i - 1][i] = k - i + 1;
      a.data[i][i] = 1;
}
a = sum(a, n);
```

a.data[0][k] 即为答案。
时间复杂度 $O(k^3 \log n)$

下节课再见