



动态规划

美丽栅栏(二)

排序计数

- 如1, 2, 3, 4的全排列, 共有 $4!$ 种, 求第10个的排列是(从1计起)?
- 先试首位是1, 后面234有 $3! = 6$ 种 < 10 , 说明首位1偏小, 问题转换成求2开头的第 $(10 - 6 = 4)$ 个排列, 而 $3! = 6 \geq 4$, 说明首位恰是2。
- 第二位先试1(1没用过), 后面 $2! = 2$ 个 < 4 , 1偏小, 换成3(2用过了)为第二位, 待求序号也再减去 $2!$, 剩下2了。而此时 $2! \geq 2$, 说明第二位恰好是3。
- 第三位先试1, 但后面 $1! < 2$, 因此改用4, 待求序号减到1。末位则是1了。
- 这样得出, 第10个排列是2-3-4-1。复杂度不超过 n^2

排序计数

本题待求方案的序号为 C

本题就是先假设第1短的木棒作为第一根，看此时的方案数 $P(1)$ 是否 $\geq C$ ，如果否，则应该用第二短的作为第一根， C 减去 $P(1)$ ，再看此时方案数 $P(2)$ 和 C 比如何。如果还 $< C$ ，则应以第三短的作为第一根， C 再减去 $P(2)$

若发现第 i 短的作为第一根时，方案数 $P(i)$ 已经不小于 C ，则确定应该以第 i 短的作为第一根，然后再去确定第二根....

排序计数

本题待求方案的序号为C

本题就是先假设第1短的木棒作为第一根，看此时的方案数P(1)是否 $\geq C$ ，如果否，则应该用第二短的作为第一根，C减去P(1)，再看此时方案数P(2)和C比如何。如果还 $< C$ ，则应以第三短的作为第一根，C再减去P(2)

若发现第i短的作为第一根时，方案数P(i)已经不小于C，则确定应该以第i短的作为第一根，然后再去确定第二根....

试第一根时，假设用的是第k短的：

$$P(n,k) = C[n][k][up] + C[n][k][down];$$

试第i根时(i从1开始算，i及其右边一共n-i+1根)

$$P(n-i+1,k) = C[n-i+1][k][up] + C[n-i+1][k][down];$$

```
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
```

```
const int UP =0;      const int DOWN =1;
```

```
const int MAXN = 25;
```

```
long long C[MAXN][MAXN][2];
```

```
//C[i][k][DOWN] 是S(i)中以第k短的木棒打头的DOWN方案数,
```

```
//C[i][k][UP] 是S(i)中以第k短的木棒打头的UP方案数,第k短指i根中第k短
```

```
void Init(int n) {
```

```
    memset(C,0,sizeof(C));
```

```
    C[1][1][UP] = C[1][1][DOWN] = 1;
```

```
    for( int i = 2 ;i <= n; ++ i )
```

```
        for( int k = 1; k <= i; ++ k ) { //枚举第一根木棒的长度,第k短
```

```
            for( int M = k; M <i ; ++M ) //枚举第二根木棒的长度,比第一根长
```

```
                C[i][k][UP] += C[i-1][M][DOWN];
```

```
            for( int N = 1; N <= k-1; ++N ) //枚举第二根木棒的长度,比第一根短
```

```
                C[i][k][DOWN] += C[i-1][N][UP];
```

```
        }
```

```
    //总方案数是 Sum{ C[n][k][DOWN] + C[n][k][UP] } k = 1.. n;
```

```
} //复杂度 n2
```

$$C[i][k][UP] = \sum C[i-1][M][DOWN]$$

$$M = k \dots i-1$$

$$C[i][k][DOWN] = \sum C[i-1][N][UP]$$

$$N = 1 \dots k-1$$

```

void Print(int n, long long cc) { //n根木棒, 求第cc个排列
    long long skipped = 0; //已经跳过的方案数
    int seq[MAXN]; //最终要输出的答案
    int used[MAXN]; //木棒是否用过
    memset(used, 0, sizeof(used));
    for( int i = 1; i <= n; ++ i ) { //依次确定每一个位置i的木棒
        int k = 0;
        int No = 0; //长度为k的木棒是剩下的木棒里的第No短的, No从1开始算
        for( k = 1; k <= n; ++k ) { //枚举位置i的木棒的长度k
            skipped = 0;
            if( !used[k] ) {
                ++ No; //k是剩下的木棒里的第No短的
                if( i == 1 )
                    skipped = C[n][No][UP] + C[n][No][DOWN];
                else {
                    if( k > seq[i-1] && ( i <= 2 || seq[i-2] > seq[i-1] ) )
                        skipped = C[n-i+1][No][DOWN]; //合法放置
                    else if( k < seq[i-1] && ( i <= 2 || seq[i-2] < seq[i-1] ) )
                        skipped = C[n-i+1][No][UP]; //合法放置
                }
            }
        }
    }
}

```

```
        if( skipped >= cc )
            break;
        else
            cc -= skipped;
    }
}
used[k] = true;
seq[i] = k;
}
for( int i = 1; i <= n; ++i )
    if( i < n )        printf("%d ", seq[i]);
    else                printf("%d", seq[i]);
printf("\n");
}
```

```
int main() {  
    int T,n;  
    long long c;  
    Init(20);  
    scanf("%d",&T);  
    while(T--){  
        scanf("%d %lld",&n,&c);  
        Print(n,c);  
    }  
    return 0;  
}
```