Recursion

递归

问题：

序列 s 有 n 个成员$[S_1, S_2, \dots, S_n]$，每个成员可以选取$[1, 2, \dots, m]$这 m 种值。

例如当$n = 5$，$m = 3$时，序列 s 有如下排列组合：

$[1, 1, 1, 1, 1]$，$[1, 1, 1, 1, 2]$，$[1, 1, 1, 1, 3]$，$[1, 1, 1, 2, 1]$...

遍历序列 s 的可能排列组合的所有情况。（与本节的 BruteForce 问题一样）

解法：

本节中 BruteForce 存在一个问题，在 BruteForce 函数的代码中，外围 for 循环的数量是固定的：

```
void BruteForce(int s[MAX], int n, int m)
{
    for (int i_0 = 0; i_0 < m; i_0++)
        for (int i_1 = 0; i_1 < m; i_1++)
            for (int i_2 = 0; i_2 < m; i_2++)
                /* ... */
                for (int i_n_1 = 0; i_n_1 < m; i_n_1++) {
                    s[0] = i_0;
                    s[1] = i_1;
                    s[2] = i_2;
                    /* ... */
                    s[n - 1] = i_n_1;
                    BruteForceOutput(s, 0, n);
                }
}
```

在上面的代码中$n = 4$，共 4 层嵌套的 for 循环。若 n 值改变，则 for 循环也必须改变。显然 BruteForce 的代码不能适应动态变化的 n 值，只能满足动态变化的 m 值。

考虑一个下标$prev \in [0, n)$，$prev$从 0 开始，序列 s 中的成员S_{prev}可以取值$i \in [1, m]$，然后$prev = prev + 1$，继续考虑序列 s 中的下一个成员S_{prev+1}。这样直到当 n 个成员都选择了一个值时，即产生序列 s 的一种排列组合。

通过递归可以退回上一个函数栈，从而让每个成员S_{prev}都可以重新选择。

对于成员数量为 n，每个成员有 m 种值的序列 s，遍历所有排列组合的时间复杂度$O(\mathrm{n} \times \mathrm{m})$。