P2241 统计方形

题目背景

1997年普及组第一题

题目描述

有一个 n * m 方格的棋盘，求其方格包含多少正方形、长方形（不包含正方形）。

输入格式

一行，两个正整数 n,m（n <=5000,m <= 5000$）。

输出格式

一行，两个正整数，分别表示方格包含多少正方形、长方形（不包含正方形）。

样例 #1

样例输入 #1

```
2 3
```

样例输出 #1

```
8 10
```

```cpp
#include<iostream>
using namespace std;
long long n,m,rec,sqr;
int main() {
    cin>>n>>m;
    for(int i=0; i<n; i++)//循环，从n-0到n-(n-1)
        for(int j=0; j<m; j++) {//循环，从m-0到m-(m-1)
            if(i==j) sqr+=(n-i)*(m-j);//如果i==j，说明是正方形
            else rec+=(n-i)*(m-j);//如果不等说明是矩形
        }
    cout<<sqr<<" "<<rec<<endl;//输出
    return 0;
}
```

```cpp
#include <iostream>

int main() {
    // 读取输入
    int n, m;
    std::cin >> n >> m;

    // 统计正方形数量
    long long squares = 0;
    for (int i = 1; i <= std::min(n, m); ++i) {
        squares += (n - i + 1) * (m - i + 1);
    }

    // 统计长方形数量（不包含正方形）
    long long rectangles = static_cast<long long>(n) * (n + 1)
/ 2 * m * (m + 1) / 2 - squares;

    // 输出结果
    std::cout << squares << " " << rectangles << std::endl;

    return 0;
}
```

P2089 烤鸡

题目背景

猪猪 Hanke 得到了一只鸡。

题目描述

猪猪 Hanke 特别喜欢吃烤鸡（本是同畜牲，相煎何太急！）Hanke 吃鸡很特别，为什么特别呢？因为他有 10 种配料（芥末、孜然等），每种配料可以放 1 到 3 克，任意烤鸡的美味程度为所有配料质量之和。

现在， Hanke 想要知道，如果给你一个美味程度 n ， 请输出这 10 种配料的所有搭配方案。

输入格式

一个正整数 n，表示美味程度。

输出格式

第一行，方案总数。

第二行至结束，10 个数，表示每种配料所放的质量，按字典序排列。

如果没有符合要求的方法，就只要在第一行输出一个 0。

样例 #1

样例输入 #1

```
11
```

样例输出 #1

```
10
1 1 1 1 1 1 1 1 1 2
1 1 1 1 1 1 1 1 2 1
1 1 1 1 1 1 1 2 1 1
1 1 1 1 1 1 2 1 1 1
1 1 1 1 1 2 1 1 1 1
1 1 1 1 2 1 1 1 1 1
1 1 1 2 1 1 1 1 1 1
1 1 2 1 1 1 1 1 1 1
1 2 1 1 1 1 1 1 1 1
2 1 1 1 1 1 1 1 1 1
```

提示

对于 100% 的数据，n<=5000。

```
#include <iostream>
#include <vector>

void generate_combinations(int n, int current_sum,
std::vector<int>& current_combination, int index,
std::vector<std::vector<int> >& result) {
    if (index == 10) {
        if (current_sum == n) {
            result.push_back(current_combination);
        }
        return;
    }

    for (int i = 1; i <= 3; ++i) {
        if (current_sum + i <= n) {
            current_combination[index] = i;
            generate_combinations(n, current_sum + i,
current_combination, index + 1, result);
            current_combination[index] = 0;
        }
    }
}
```

```cpp
int main() {
    int n;

    std::cin >> n;

    std::vector<std::vector<int> > result;
    std::vector<int> current_combination(10, 0);

    generate_combinations(n, 0, current_combination, 0,
result);

    if (result.empty()) {
        std::cout << "0\n";
    } else {
        std::cout << result.size() << '\n';
        for (const auto& combination : result) {
            for (int i = 0; i < 10; ++i) {
                std::cout << combination[i] << ' ';
            }
            std::cout << '\n';
        }
    }

    return 0;
}
```

P1618 三连击

题目描述

将 1, 2,\ldots, 9 共 9 个数分成三组，分别组成三个三位数，且使这三个三位数的比例是 A:B:C，试求出所有满足条件的三个三位数，若无解，输出 `No!!!`。

输入格式

三个数，A,B,C。

输出格式

若干行，每行 3 个数字。按照每行第一个数字升序排列。

样例 #1

样例输入 #1

```
1 2 3
```

样例输出 #1

```
192 384 576
219 438 657
273 546 819
327 654 981
```

提示

保证 A<B<C.

```
#include <iostream>
using namespace std;
int a[10], b1, b2, b3, tag, k1, k2, k3, ans;
int main() {
    cin >> k1 >> k2 >> k3;
    for (int b = 1; b <= 1000 / k3; ++b) {
        b1 = b * k1;
        b2 = b * k2;
        b3 = b * k3;

        if ((b2 > 999) || (b3 > 999)) {
            break;
        }
        for (int i = 1; i <= 3; i++) {
            a[b1 % 10]++;
            b1 /= 10;
            a[b2 % 10]++;
            b2 /= 10;
            a[b3 % 10]++;
            b3 /= 10;
```

```
        }

        for (int c = 1; c <= 9; c++) {
            if (a[c] != 1) {
                tag = 1;
                break;
            }
        }
        for (int c = 1; c <= 9; c++) {
            a[c] = 0;
        }
        if (!tag) {
            cout << b * k1 << " " << b * k2 << " " << b * k3
<< endl;
            ans++;
        }
        else{
            tag = 0;
        }
    }
    if (!ans) {
        cout << "No!!!";     //判断无解情况
    }
    return 0;
}
```

循环枚举

循环枚举是一种暴力枚举的具体实现方式，它通过使用循环结构来遍历所有可能的情况。这种方法适用于问题的解空间可以通过一定范围的参数值进行穷举的情况。

以下是循环枚举的一般步骤和特点：

1. **确定参数范围：** 首先确定问题中需要枚举的参数范围，这可以是数组的索引范围、数字的取值范围等。
2. **使用循环结构：** 利用循环结构遍历参数范围内的所有可能取值。
3. **处理每一种情况：** 在循环中处理每一种情况，执行相应的操作或验证。
4. **找到满足条件的解：** 如果找到满足问题条件的解，则结束循环，认为找到了解决方案。

5. **时间复杂度与参数范围相关：** 循环枚举的时间复杂度与参数范围相关，通常为 O(n) 或 O(n^2)，其中 n 是参数的范围。

子集枚举

子集枚举是一种特殊的枚举方法，用于生成一个集合的所有可能子集。这种方法通常涉及到二进制的思想，其中集合中的每个元素可以被选择或者不选择，对应于二进制中的 1 或 0。

以下是子集枚举的一般步骤和特点：

1. **使用二进制表示：** 将集合中的每个元素与二进制位相对应。如果某个元素被选择，对应的二进制位为 1，否则为 0。
2. **利用位运算生成子集：** 利用位运算（如位与、位或）来生成所有可能的子集。
3. **遍历所有可能的子集：** 遍历从 0 到 2^n - 1 的二进制数，其中 n 是集合中元素的个数，对每个二进制数生成相应的子集。
4. **处理每个子集：** 对于生成的每个子集，执行相应的操作。
5. **时间复杂度为 O(2^n)：** 由于需要考虑所有可能的子集，时间复杂度通常为 O(2^n)，其中 n 是集合中元素的个数。

排列枚举

排列枚举是一种枚举方法，用于生成一个集合的所有可能排列。不同于子集枚举，排列枚举关注元素之间的顺序，因此每个元素在排列中只能出现一次。

以下是排列枚举的一般步骤和特点：

1. **使用递归或循环：** 可以使用递归或循环的方式生成所有可能的排列。
2. **标记使用过的元素：** 在生成排列的过程中，需要标记哪些元素已经被使用过，以确保每个元素在排列中只出现一次。
3. **遍历所有可能的排列：** 通过递归或循环遍历所有可能的排列，对每个排列执行相应的操作。
4. **时间复杂度为 O(n!)：** 由于需要考虑所有可能的排列，时间复杂度通常为 O(n!)，其中 n 是集合中元素的个数。