

拉丁矩阵问题

一. 问题分析:

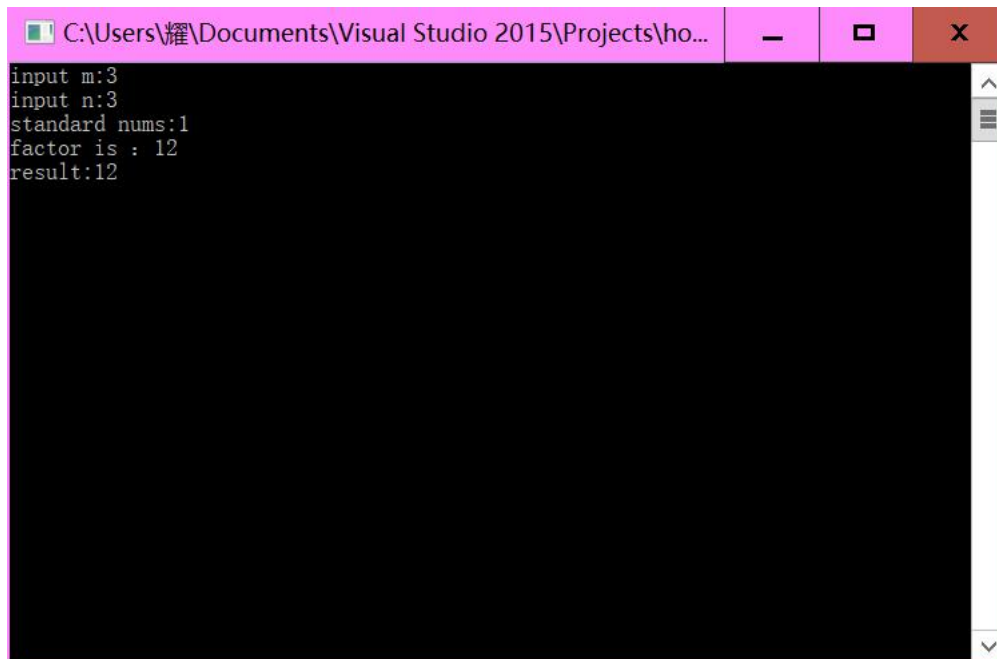
设 n 种宝石编号为 $1, 2, 3 \dots n$, 设宝石矩阵的第一行从左到右排列为 $1, 2, 3 \dots n$, 且第一列从上到下排列为 $1, 2, 3 \dots m$ 的阵列为标准拉丁阵列, 其中宝石矩阵可以看作是标准标准矩阵通过对宝石进行不同的编号, 以及对标准举证的各行进行排列组合得到。当对宝石进行编号时, 有 $n!$ 种编号方案, 当对行进行排列组合时, 有 $\frac{(n-1)!}{(n-m)!}$ 种排列组合, 因此设标准拉丁矩阵的数目为 $L(m, n)$, 总的拉丁矩阵数目为 $R(m, n)$, 则两者的关系是: $R(m, n) = \frac{n!(n-1)!}{(n-m)!} L(m, n)$, 求到标准拉丁矩阵数目就可以求解得到问题的解。

二. 算法设计:

问题的解向量为: 宝石矩阵, **Metrix[m][n]**, 初始化为标准拉丁矩阵。解空间为排列树, 对整个矩阵从左到右, 从上到下的方式进行搜索, 直到抵达最后一个节点, 如果满足要求, 则对其计数结果变量 **result** 加 1, 如果排列的宝石与行和列中的宝石冲突, 则对其进行剪枝。

三. 程序设计:

通过控制台输入两个变量, 宝石种类的数目 n , 排成的行数 m , 输出三个参数, **standard nums** 为标准拉丁矩阵个数, **factor** 为总的拉丁矩阵数目与标准拉丁矩阵的倍数因子, **result** 为总的拉丁矩阵个数。其运行如图下所示:



```
input m:3
input n:3
standard nums:1
factor is : 12
result:12
```

四. 代码

```
// homework6.cpp : 定义控制台应用程序的入口点。
//
```

```
#include "stdafx.h"
#include <vector>
```

```

#include <iostream>
using namespace std;
bool isok(vector<vector<int>> &data, int index, int color)//判断是否冲突
{
    bool isok = true;
    int rows = data.size();
    int cols = data[0].size();
    int x = 1 + int(index / cols);
    int y = int(index%cols);
    for (int i = 0; i < x; i++)
    {
        if (data[i][y] == color) return false;
    }
    for (int i = 0; i < y; i++)
    {
        if (data[x][i] == color) return false;
    }
    return isok;
}

void traceback(vector<vector<int>> &data, int index,int &result)//回溯结构
{
    int rows = data.size();
    int cols = data[0].size();
    int x = 1 + int(index / cols);
    int y = int(index%cols);
    for (int i = 1; i <= cols; i++)
    {
        if (isok(data, index, i))
        {
            data[x][y] = i;
            if (index < ((rows - 1)*((cols - 1))))
            {
                traceback(data, index +
                    1, result);
            }
            else
            {
                result = result + 1;
            }
        }
    }
}

int fac(int a)//计算阶乘

```

```

{
    if (a == 0) return 1;
    else return a*fac(a - 1);
}
int main()
{
    int m, n;
    cout << "input m:";
    cin >> m;
    cout << "input n:";
    cin >> n;
    vector<vector<int>> Metrix;//储存宝石排列信息
    int result=0;
    for (int i = 0; i < m; i++)
    {
        vector<int> tempv;
        for (int j = 0; j < n; j++)
        {
            tempv.push_back(0);
        }
        Metrix.push_back(tempv);

    }
    for (int i = 0; i < m; i++)
    {
        Metrix[i][0] = i;
    }
    for (int j = 0; j < n; j++)
    {
        Metrix[0][j] = j;
    }
    traceback(Metrix, 1, result);
    cout << "standard nums:" << result<<endl;
    int factor = fac(n)*fac(n - 1) / fac(n - m);
    result = result *factor ;
    cout <<"factor is : " <<factor<<endl<< "result:" << result;
    int wait;
    cin >> wait;
    return 0;
}

```