考虑N皇后问题

例：程序N\_Queen\_test1，该程序的执行效率不高

效率高的N皇后问题：位运算

参考<http://www.matrix67.com/blog/archives/266>

例：程序N\_Queen\_test2

// Copyright 2017.刘珅珅

// author：刘珅珅

// 回溯算法-N皇后问题

#include <algorithm>

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

int N = 0;

long sum = 0;

long upper\_column = 1;

// 查看数字的二进制表示

template<typename T> void BinaryRecursion(T n)

{

T a = n % 2;

n = n >> 1;

if (n != 0)

BinaryRecursion(n);

cout << a;

}

template<typename T> void BinaryCount(T n, int count)

{

for (int i = count - 1; i >= 0; --i)

{

cout << ((n >> i) & 1);

}

cout << endl;

}

// 位运算的程序按行寻找可以放皇后的地方

// row代表某一行在纵列限制下不能放置皇后的列

// left\_diagonal代表某一行受左对角线限制下不能放置皇后的列

// right\_diagonal代表某一行受右对角线限制下不能放置皇后的列

// 以6x6的棋盘为例：

// row=101010代表在某一行的第1,3,5列不能放置皇后

// left\_diagonal=100100代表在某一行的第1,4列不能放置皇后

// right\_diagonal=000111代表在某一行的第4,5,6列不能放置皇后

// 参考http://www.matrix67.com/blog/archives/266

// 中的讲解，第一个插图代表第4行的情况，前3行都已放置了皇后

// row为红色线条，left\_diagonal为蓝色线条，由图中可以看出，蓝色线条

// 与第4行相交于第1,4列，所以left\_diagonal=100100，right\_diagonal为绿色

// 与第4行相交于第4,5,6列，right\_diagonal=000111

// 由pos = upper\_column & ~(row | left\_diagonal | right\_diagonal)可以计算出

// pos=010000，表示第2列可以放置皇后，取出最右边可用列：p=010000

// 从第二个插图可以看出，在第4行第2列放置皇后后，第5行的情况如下

// row=row+p, left\_diagonal=(left\_diagonal+p)<<1,right\_diagonal=(right\_diagonal+p)>>1

// row的变化代表下一行的p中1的列不可以放置皇后了

// left\_diagonal+p左移1位是因为left\_diagonal的最高位为第4行第1列的限制，它

// 是由前3行的皇后放置决定的，不会作用到第5行，应该由第4行的皇后放置

// 来确定第5行第1列的皇后限制，所以要左移1位，同理，right\_diagonal+p右移1位也是如此

void Test(long row, long left\_diagonal, long right\_diagonal)

{

if (row != upper\_column)

{

// pos代表某一行所有可以放置皇后的列

long pos = upper\_column & ~(row | left\_diagonal | right\_diagonal);

while (pos)

{

// 取出最右边的为1的bit

long p = pos & (~pos + 1);

// 从pos中剔除掉最右的可用列

// 为下一次获取最右可用列做准备

pos -= p;

Test(row + p, (left\_diagonal + p) << 1, (right\_diagonal + p)>> 1);

}

} else {

// row的所有位都为，代表所有行都成功放置了皇后

++sum;

}

}

int main(int argc, char\* argv[])

{

N = 6;

// 生成有N个1组成的二进制数

// N个皇后只需N为存储，有皇后的列对应的位为1

upper\_column = (upper\_column << N) - 1;

//BinaryRecursion(upper\_column);

//cout << endl;

Test(0, 0, 0);

cout << N << " Queen answer is " << sum << endl;

//ofstream write;

//write.open("result.txt", ios::out);

//for (int i = 0; i < result.size(); ++i)

//{

// write << "solution " << i + 1 << endl;

// for (int j = 0; j < result[i].size(); ++j)

// {

// write << result[i][j].c\_str() << endl;

// }

// write << endl;

//}

return 0;

}

1 1

2 0

3 0

4 2

5 10

6 4

7 40

8 92

9 352

10 724

11 2680

12 14200

13 73712

14 365596

15 2279184

16 14772512

17 95815104

18 666090624

19 4968057848

20 39029188884

21 314666222712

22 2691008701644

23 24233937684440

24 227514171973736

25 2207893435808352