**算法作业1**

**青蛙跳台阶问题**

**姓名：王星洲**

**学号：1652977**

**联系电话：18916111729**

**问题描述：**

一只青蛙一次可以跳上 1 级台阶，也可以跳上 2 级台阶。求该青蛙跳上 1 个 n 级的台阶总共有多少种跳法。

**核心问题：**

本题可以转化为计算斐波那契数列的值，因为3个台阶的方案总数为1个和2个台阶方案数之和，4个台阶的方案总数为2个和3个台阶方案数之和。所以，方案数的变化趋势为：1、2、3、5、8、13、21、34……总体上就是从斐波那契数列第二个数开始之后的值。

计算斐波那契数列的方法有很多，主要有三种：

1.递归思路

即return f(n-2) + f(n-1)

一直计算到f(2)或f(1)为止，时间复杂度为O(n)，但由于是递归算法，所以算法的效率不高，对于90这么庞大的数字已经几乎无法处理。

2.循环方法

即将递归转换为从1~n的循环，时间复杂度为O(n)，不过略优于第一种方法，但效率仍然很差。

3.矩阵的快速幂算法

首先介绍何为快速幂，当我们要计算2^8时，通常要计算7次乘法，然而我们可以通过另一种方式来实现，即先计算2^2，再让结果相乘计算2^4，再让结果相乘计算2^8，将7次计算简化为了3次。实现了|log2(7)|次的优化。

在实际应用时，我们可以将快速幂的过程分情况讨论，对于一个偶数幂n，它的值可表示为n/2与n/2次幂的积。对于一个奇数幂n，它的值可表示为n/2与n/2次幂与底数本身的积。

为何要采用矩阵？

本题的本质是斐波那契数列，而矩阵的乘法可以很便利的解决这个问题。

https://images2017.cnblogs.com/blog/1221690/201802/1221690-20180205111517810-2074558272.png

https://images2017.cnblogs.com/blog/1221690/201802/1221690-20180205111640654-1573809698.png

所以本题中我们只需要让初始矩阵为(2,1)，并让A作为参考，根

(0,0)

据A的幂次进行快速幂运算，得到的结果再与初始矩阵相乘，即可得到最终解。

这样做的时间复杂度为O(log2n)。可以迅速地处理大数情况。

本题我采用第三种方式来解决。

**代码中的注意事项：**

1.对输入数值的合法性判断：

本题中输入数值应为非负整数。其中0对应的结果为0。因此，我们应对输入的合法性进行判断。代码如下：

while (1)

{

//illegal input

if (!(cin >> number))

{

cout << "您的输入非法，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

//A negative number

else if (number < 0)

{

cout << "请输入一个正整数。" << endl;

}

//three special condition

else if (number == 0)

{

cout << "可行的方案有0种。" << endl;

system("pause");

return 0;

}

else if (number == 1)

{

cout << "可行的方案有1种。" << endl;

system("pause");

return 0;

}

else if (number == 2)

{

cout << "可行的方案有2种。" << endl;

system("pause");

return 0;

}

//usual condition

else

break;

}

2.快速幂次数的确定：

由于本题初始情况定为(2,1)，所以，A的幂次应该为阶梯总数-2，具体使用方式如下：

Matrix result = multiply(origin, quickPow(reference, number - 2));

3.使用的变量类型：

由于本题的结果很庞大，int无法容纳如此庞大的数字。因此采用long long类型存储。具体代码如下:

struct Matrix {

//Int can't hold so large number

long long mat[2][2];

};