目录

[**1.使用说明 2**](#_Toc29166415)

[**1.1 实验环境 2**](#_Toc29166416)

[**1.2 使用说明 2**](#_Toc29166417)

[**2设计说明 4**](#_Toc29166418)

[**2.1子程序copy 4**](#_Toc29166419)

[**2.2子程序multiply 5**](#_Toc29166420)

[**2.3子程序add 5**](#_Toc29166421)

[**2.4主程序sum 7**](#_Toc29166422)

[**2.5形式化表示 8**](#_Toc29166423)

[**3.算法说明 8**](#_Toc29166424)

[**3.1 main.ui说明 8**](#_Toc29166425)

[**3.2 statechange.py说明 9**](#_Toc29166426)

[**3.3 main.py函数说明 9**](#_Toc29166427)

[**4.总结 10**](#_Toc29166428)

# 1.使用说明

## 1.1 实验环境

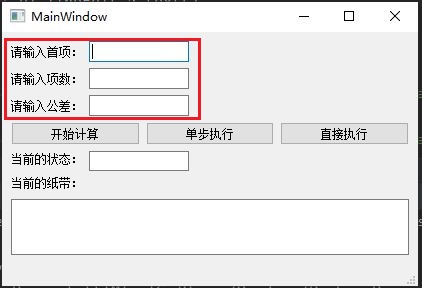
操作系统：Windows10

依赖环境：Python3.6，PyCharm

编程语言：Python

## 1.2 使用说明

运行main.py文件(需要安装PySide2)，显示如下界面：



**图1 进入界面**

在前三个输入框分别输入需要计算的等差数列的首项a、项数n和公差m，点击开始计算，输出当前为“o”，并且生成初始纸带为\*B1aB1nB1mBBB的形式，\*代表图灵机准备读取字符的位置。



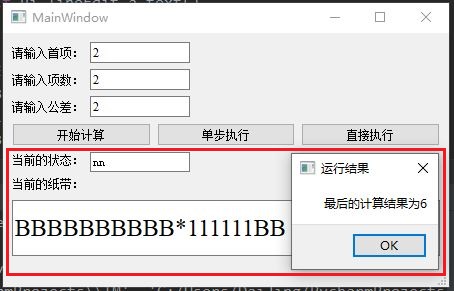
**图2 执行按钮**

点击单步执行，相当于执行一次状态转移函数，当前的状态、纸带和读取位置均发生相应变化。如下图所示，在点击若干次单步执行以后，图灵机的当前状态为“a”，纸带的状态如下。



**图3 执行过程**

由于在本程序中实验跳转的步数太多，所以可以点击“直接执行”，得到最终计算结果，此时纸带上只剩下计算的结果且指针指向结果的第一个1。

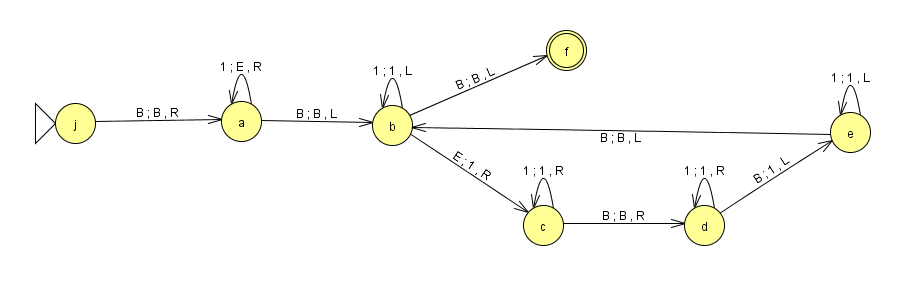


**图4 执行结果**

# 2设计说明

由于在本实验中，根据等差数列的求和的算法，在这里为了使程序简化，将重复多次的计算过程封装为子程序，方便调用和使代码简洁。

## 2.1子程序copy

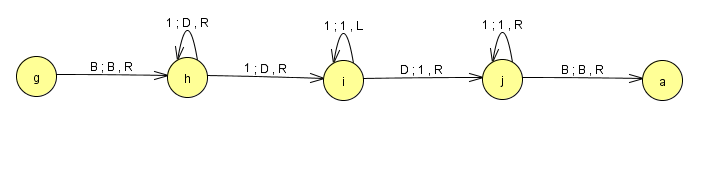


**图5 copy子程序**

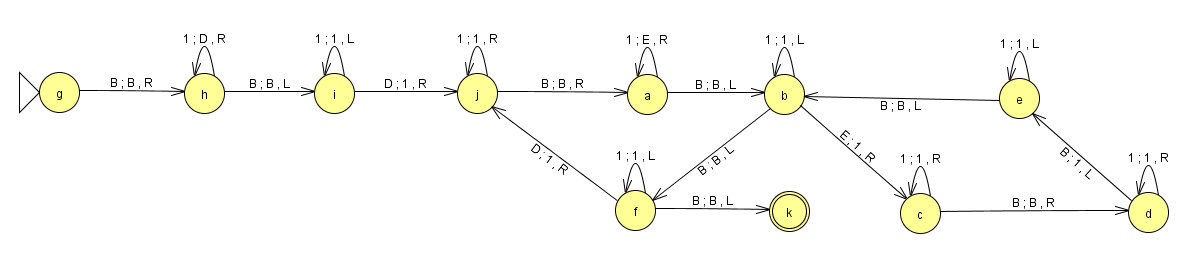
在本次设计中，首先要实现图灵机最基本的复制字符的功能，所以设计子函数copy完成1m的复制功能。经测试，本程序可以实现复制字符串1m的功能。

## 2.2子程序multiply

在完成子程序copy之后，由于在计算中需要调用乘法运算，所以在这里多次调用copy完成乘法multiply的计算。



**图6 multiply子程序的设计**



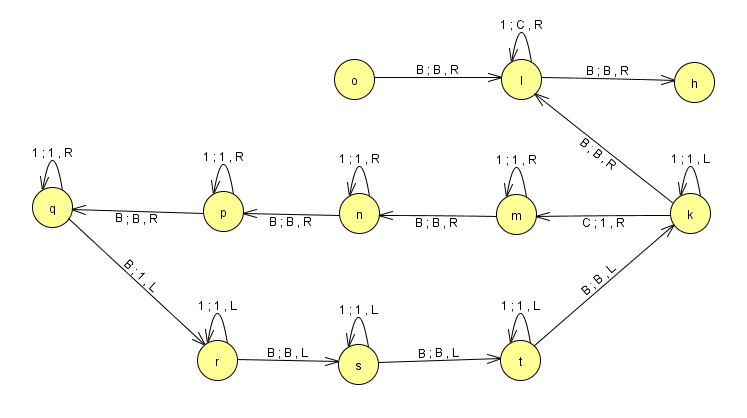
**图7 multiply的完整图**

## 2.3子程序add

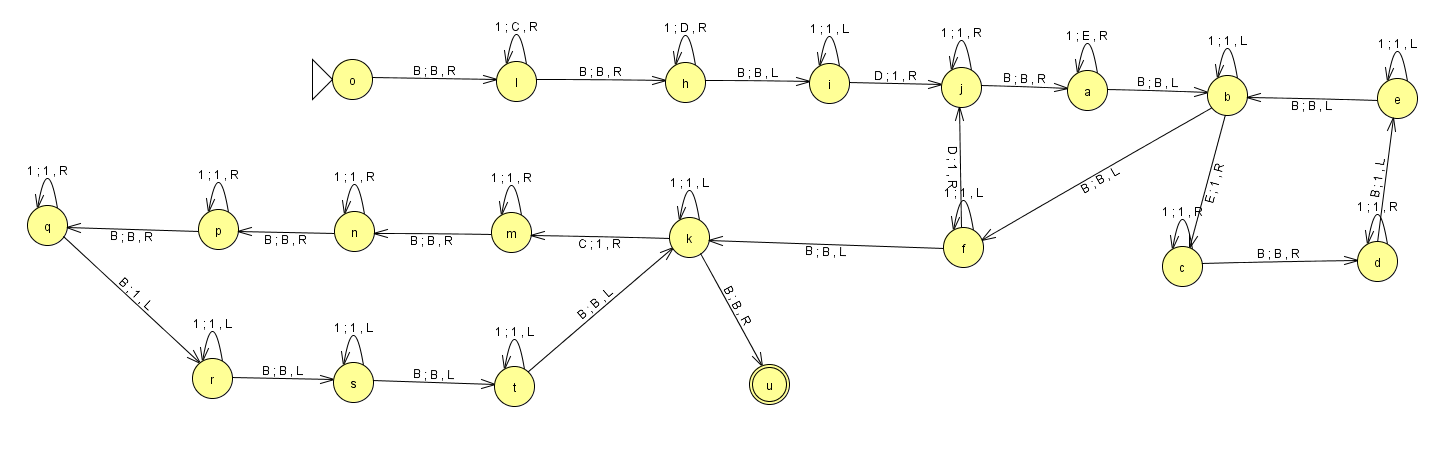
为了图灵机设计的简洁性，在本实验中设计计算等差数列和时从依次向前加到。计算公式和递推公式如下：

子程序add主要完成上面第二式的加法计算。根据等差数列的计算公式：

子程序add的设计思路在的基础上先加上a，再调用(t-1)次copy子程序，得到的结果。设计的图灵机模型如下图所示：



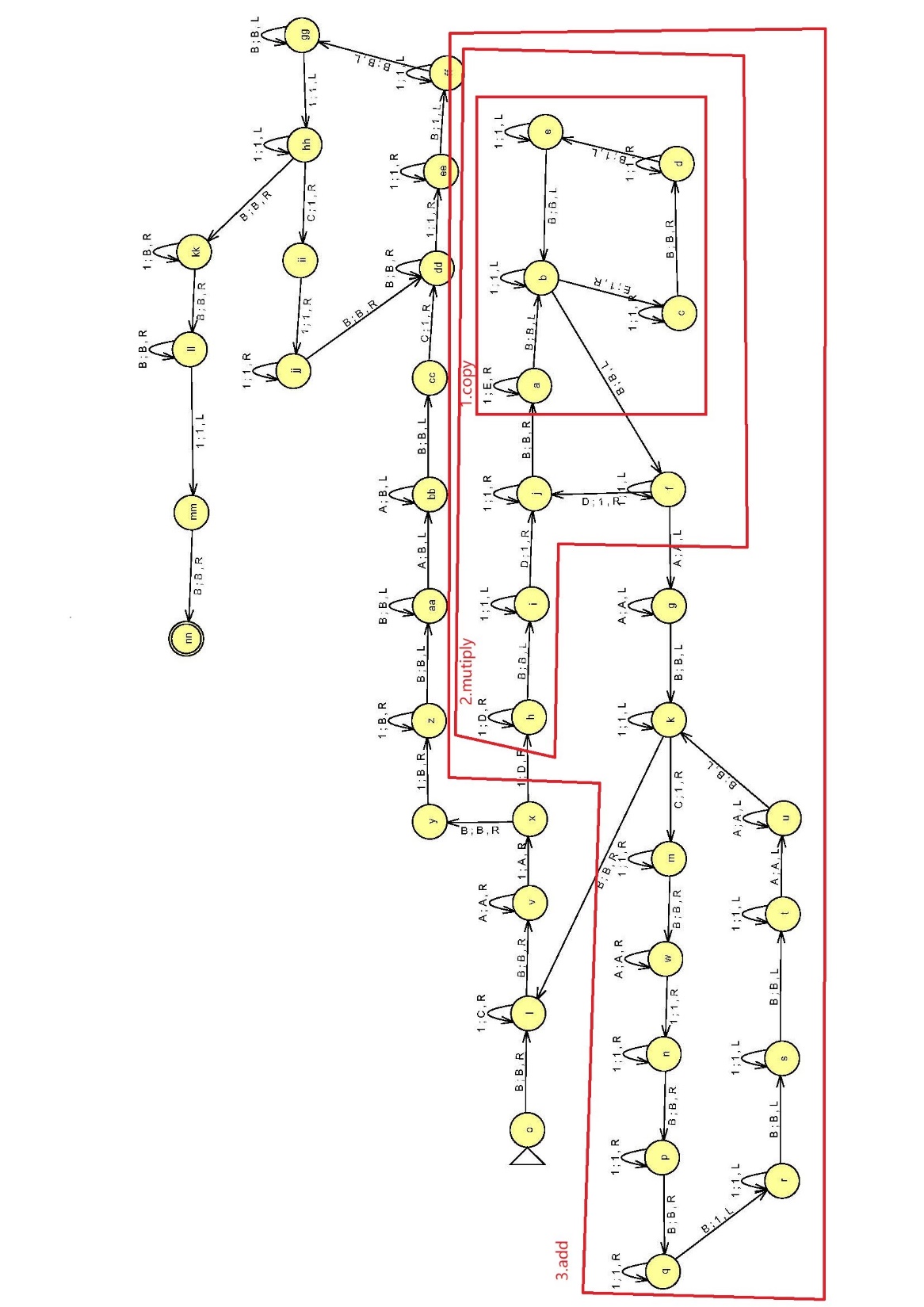
**图8 add子程序的设计**



**图9 add子程序的完整图**

## 2.4主程序sum

在完成上面计算过程后，每调用一次add完成求和过程中的一次加法；通过多次循环嵌套，完成整个等差数列。整个图灵机的设计和子程序之间的嵌套关系见下图。



**图10 图灵机**

## 2.5形式化表示

完成上述设计过程后，将整个图灵机形式化表示为：

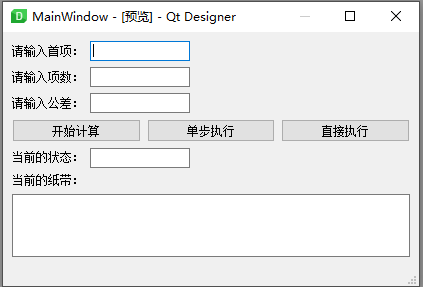
* 共40个状态
* 为转移函数，由于转移函数太多且已经给出图灵机的状态转移图，故不将其一一列出

# 3.算法说明

在本程序设计中，主要用Python语言写成，主要包括main.py和statechange.py的脚本文件，以及由QtDesigner设计的图形界面main.ui文件。

## 3.1 main.ui说明

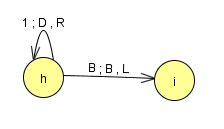
此文件为图形界面的生成文件，提供图灵机输入、输出的图形界面表示。具体如下图所示：



**图11 ui界面设计**

## 3.2 statechange.py说明

statechange.py主要是图灵机的状态转移函数。由于设计的图灵机是一个标准的图灵机，所以只需要根据状态转移图对应编写即可。利用两个if…elif…循环完成状态转移下面以状态h为例，说明状态转移函数。



**图12 状态“h”图示**

对应h的两个状态转移函数为：

对应的代码段为

|  |  |
| --- | --- |
| elif state == "h":  if tape[point] == "1":  tape[point] = "D"  state = "h"  point += 1  elif tape[point] == "B":  tape[point] = "B"  state = "i"  point -= 1 | 判断当前在状态“h” 1.读取字符“1” 2.写入字符“D” 3.跳转到状态“h”  右移 1.读取字符“B”  2.写入字符“B”  3.跳转到状态“i”  左移 |

可以看到，在上面所举的例子中，状态转移的过程与图灵机的作用完全相同，主要分为3个步骤：1.读取字符；2.写入字符；3.状态跳转、指针移动。这样的设计大大降低了编程的难度。

在整个计算过程中，由状态“o”开始，由状态转移函数决定转移状态；状态达到“nn”状态结束运行。

## 3.3 main.py函数说明

主程序主要使用Turing类实现的计算等差数列和的计算，现将其主要属性的方法介绍如下：

|  |  |
| --- | --- |
| Turing | |
| 属性 | 说明 |
| a | 等差数列的首项 |
| n | 等差数列的项数 |
| m | 等差数列的公差 |
| tape | 图灵机的纸带 |
| state | 图灵机的状态 |
| point | 图灵机读取纸带上的位置 |
| 方法 | 说明 |
| start | 获取参数并初始化图灵机 |
| step\_one | 单步计算，完成一次状态转移 |
| step\_all | 完成全部计算，输出结果 |

# 4.总结

本次实验设计了一个可以完成等差数列求和的图灵机的设计，其中使用了1条纸带，设计了40个状态，完成了从输入到输出的计算任务。界面显示了图灵机在计算过程中的状态转移、纸带的变化以及图灵机读取位置的变化。本本实验采用了1进制，因此纸带会比较长，而且状态数比较多，状态迁移也比较复杂。另外在图灵机的设计过程中，采用了子程序的设计思想，由难到易，完成了图灵机的设计。