Individual Project:

计算Pi

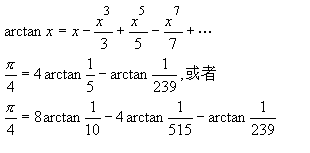
**1.调研：**

可以用来近似Pi的方法非常多，经过调研得到以下方法：

1. 级数：

C:\Users\v-haidin\Desktop\1353306851_8018.png

1. Machin 公式



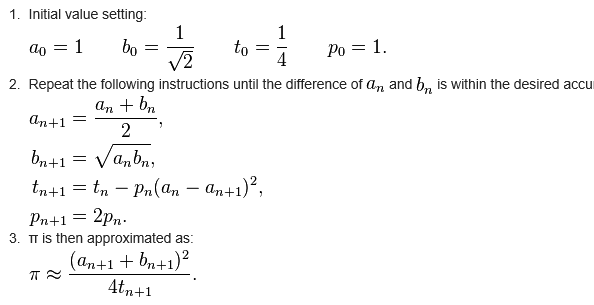
1. Ramanujan 公式

C:\Users\v-haidin\Desktop\3554907526d7a18c2c48124688827272.png

1. Chudnovsky 公式

C:\Users\v-haidin\Desktop\1d310b709e2ad6a5e863d0f8ab0a8b46.png

1. Gauss-Legendre 公式



相比较而言1,2公式较为简单，只涉及到大数的四则运算，而且可以把公式改写为迭代的形式。另外三个方法均涉及到根号的运算，虽然效率要高于1,2 但估计pi的同时还要对Sqrt(2)进行估计，显然较为麻烦。因此选择1,2作为候选方案。

**2. 实施思路：**

公式1和公式2相比而言，两者均可以写成迭代的形式，而且可以很简单地确定需要迭代多少次才能得到预期的精度。此外公式1可以巧妙的写成类似“递归”的形式，而公式2较1而言计算更为简单，只涉及到大数的除法。而且公式2种的减数和被减数可以通过多线程并行运算。

计算Pi的难点在于大数的表示与计算。一个较为常用的方法是将一个大数表示为以一个数（10000，1000000等）为底的数组。这种做法使得聊个大数的乘法的复杂度为O(NxN)，而且可以通过FFT算法把复杂度将为O(NxlogN) 。但对于候选方案：两个乘数中只有一个数为大数，此时简单的乘法已经足够简单了。

**3. unit test**

利用Mathematica得到正确答案（N[Pi, 100001]），将计算结果写到文件中，两个文件进行比较，由于存在最后一位有舍入的区别，最后一位相差为1也视为正确。

**4. 实施与实验结果**

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **N** | **时间s** | **测试通过** |
| **公式1** | **1000** | **0.012** | **是** |
| **10000** | **0.532** | **是** |
| **30000** | **4.911** | **是** |
| **100000** | **52.613** | **是** |
|  | | | |
| **公式2** | **1000** | **0.068** | **是** |
| **10000** | **1.519** | **是** |
| **30000** | **12.331** | **是** |
| **100000** | **135.537** | **是** |

**两者比较，复杂度相同，公式1略快一点。**