# Individual Project I (Pi) Report

Xizhou (v-xizzhu)

2014/9/9

花费了一天半的时间调研，了解了SuperPi、PiFast、y-cruncher三个计算Pi的项目，主要参考资料

1. B. Haible and T. Papanikolaou, Fast multiprecision evaluation of series of rational numbers, report TI-97-7.binsplit, TH Darmstadt.
2. FFT based multiplication of large numbers

<http://numbers.computation.free.fr/Constants/Algorithms/fft.html>

1. y-cruncher - Language and Algorithms

<http://www.numberworld.org/y-cruncher/algorithms.html>

1. Binary splitting method

<http://numbers.computation.free.fr/Constants/PiProgram/pifast.html>

1. x64 (amd64) Intrinsics List

<http://msdn.microsoft.com/en-us/library/hh977022.aspx>

最终确定使用Chudnovsky Formula：

http://www.numberworld.org/formulas/pi/pi_chudnovsky.png

利用Binary Splitting method将上式求解时间复杂度降到，其中N为所需位数（与求和级数项数正比），M(N)为乘法复杂度，乘法使用Karatsuba Multiplication，复杂度为

花了半天确定接口和编写单元测试，Unit Test包括

1. 高精度整数加法
2. 高精度整数减法
3. 高精度整数乘法
4. 高精度整数进制转换（实际操作时未使用）
5. 平方根倒数、倒数（这部分的Unit test在实现高精度浮点数后编写的）

花了两天时间实现，一天时间优化，实现时，高精度整数使用进制，即一个uint64存放一位数，优点是 进位加法、借位减法、带余除法（未使用）、高低位分别存放的乘法 均有指令集优化方案（x64 (amd64) Intrinsics List，上述参考资料5）

还实现了高精度浮点数，用于平方根倒数、倒数与最终求Pi的相关操作，优点是可以根据所需精度截断数字，平方根倒数、倒数均采用牛顿迭代法。

乘法原本使用朴素方法，经过性能分析后发现乘法占大约95%的时间，决定优化，尝试了FFT，考虑到精度问题，FFT内部采用double，输入输出使用32位整数，结果误差不能让人接受，FFT卷积的误差可能出现在任何输出的认为一位数上，如果高位差了1最后结果就会令人尴尬，根据参考资料2，16位整数应该也不能控制误差在理想范围，8位应该可以，但是还是放弃了。最终选择Karatsuba Multiplication，二分支的乘法，复杂度略低于朴素乘法，而且方便并行。

最终结果Pi的高精度浮点数，通过每次乘以输出19位10进制数，复杂度，严重影响了性能。

最后验证了计算结果无误，在公司的电脑上100K位大约0.4s，1M位大约19s，其中计算1M位Pi时，大约7秒在计算Pi，12秒在转成10进制并输出的过程中。