

**课 程 实 验 报 告**

**课程名称： 并行编程原理与实践**

**专业班级： CS1602**

**学 号： U201614545**

**姓 名： 谭胜克**

**指导教师： 金海**

**报告日期： 2019/7/22**

**计算机科学与技术学院**

**目录**

[1 串行算法复杂度分析 1](#_Toc15155349)

[1.1任务描述 1](#_Toc15155350)

[1.2实验方案 1](#_Toc15155351)

[1.3实验结果与分析 1](#_Toc15155352)

[2 并行算法实现的正确性 3](#_Toc15155353)

[2.1实验目的与要求 3](#_Toc15155354)

[2.2 实验方案 3](#_Toc15155355)

[2.3 实验结果与分析 3](#_Toc15155356)

[3 大数场景分析 4](#_Toc15155357)

[3.1实验目的与要求 4](#_Toc15155358)

[3.2实验方案 4](#_Toc15155359)

[3.3实验结果与分析 4](#_Toc15155360)

[4 并行优化方案设计 6](#_Toc15155361)

[4.1任务描述 6](#_Toc15155362)

[4.2实验方案 6](#_Toc15155363)

# 1 串行算法复杂度分析

## 1.1任务描述

分析串行算法的复杂度，并设计测试案例予以证明。

## 1.2实验方案

对斐波纳切串行算法进行修改，增设全局变量count用以统计主要核心算法中循环执行的次数。

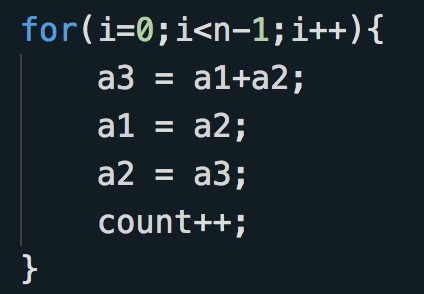


图1.1 修改串行算法

分析串行斐波纳切的计算方式，我采用的是循环方式一项一项计算。对于数列中的第n项，其循环执行的次数应该为n-1次。所以如果需要输出k项。则循环将会执行的计算公式为：

total = ；

设定测试样例为：

k=10，35，50；

根据公式计算得：

* 当k=10时，total = = 45；
* 当k=35时，total = + 34 = 561 + 43 = 595；
* 当k=50时，total = = 1225；

## 1.3实验结果与分析

对上述给出的测试样例（k=10，35，50），分别在实验平台上进行测试验证。测试结果中，num\_size为测试规模，即k；total\_times为循环执行次数。具体结果见下方给出的各项截图。



图1.2 k=10验证图

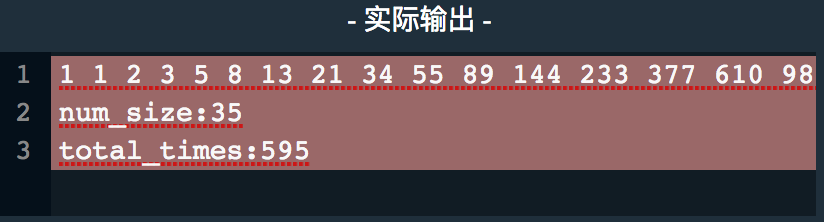


图1.3 k=35验证图

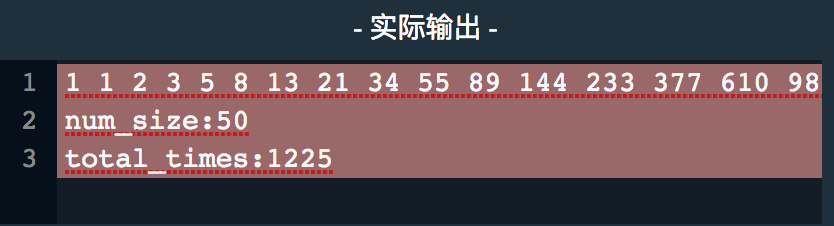


图1.4 k=50验证图

如图1.2，1.3，1.4所示，在k=10，35，50的情况下，循环执行次数分别为45，595，1225。与实验方案中理论计算得出的结果完全吻合，证实了之前的推导。

由此，我们也可以知道，串行斐波纳切数列算法的时间复杂度为O(n)。由于并没有进行存储，所以在计算过程中之用到了几个临时变量，所以串行斐波纳切数列算法的空间复杂度为O(1)。

# 2 并行算法实现的正确性

## 2.1实验目的与要求

在“斐波那契数列计算”中实现的多种并行实现中，挑选一种并分析其正确性，并请给出方案确保并行实现的正确性。

## 2.2 实验方案

选择pthread环境下的斐波纳切并行算法作为求证对象，证明该算法执行的正确性。

## 2.3 实验结果与分析

pthread斐波纳切数列算法核心部分如图2.1所示。该部分通过有限次（49次）的循环，完成斐波纳切数列第2项到第50项到计算。所以算法在有限时间内能够求解。

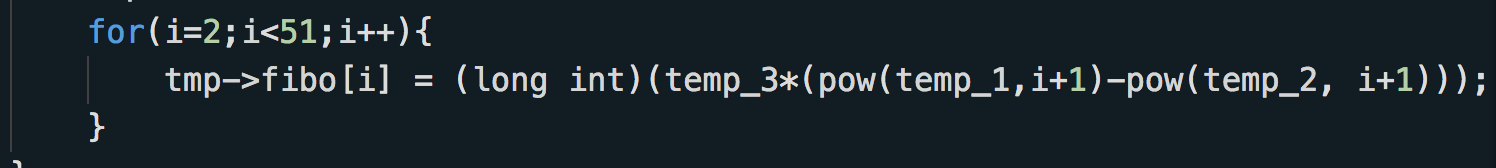


图2.1 设定计时器

对于每一项到计算都是根据一个斐波纳切数列的通项公式（下式）进行计算，从数学依据上确保每一项计算的正确性。

综上可知，该算法在又穷时间内能够正确求解斐波纳切数列。

# 3 大数场景分析

## 3.1实验目的与要求

1. 设计大数计算场景下斐波那契数列的测试用例并分别使用串行和并行的方式实现。
2. 分析并行实现的加速比。
3. 分析并行实现加速比的正确性。

## 3.2实验方案

并行方式选用OpenMP环境下的并行斐波纳切数列计算方法。对于大数规模分别设定为：

n=1000

n=10000

n=100000

设定不同的参数，n=1000，n=10000，n=100000；分别运行串行算法以及并行算法，同时对两种算法进行修改，在开始计算前开始计时，计算完成后结束计时，通过耗时的大小判断算法的执行效率。如图3.1所示，为修改后代码的计时部分。

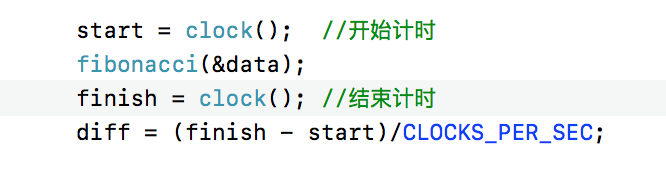


图3.1 设定计时器

## 3.3实验结果与分析

1. 当n=1000时，测试结果如图3.2所示。并行加速比为：26.069；
2. 当n=10000时，测试结果如图3.3所示。并行加速比为：412.927；
3. 当n=100000时，测试结果如图3.4所示。并行加速比为：4559.8076；

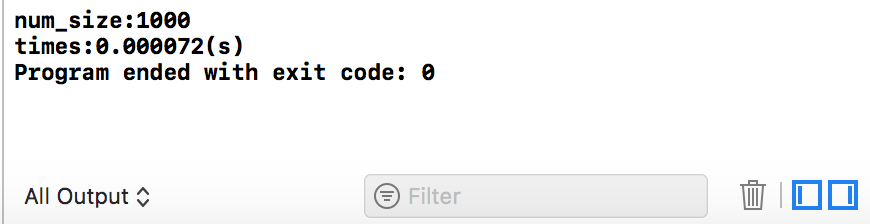
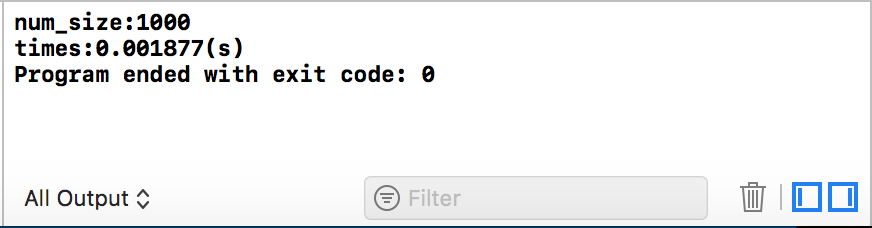


图3.2 n=1000：串行（左）并行（右）

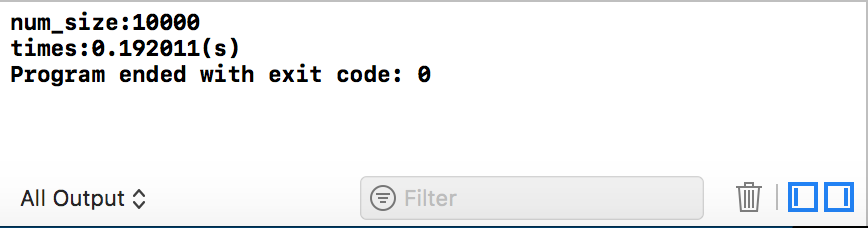


图3.3 n=10000：串行（左）并行（右）

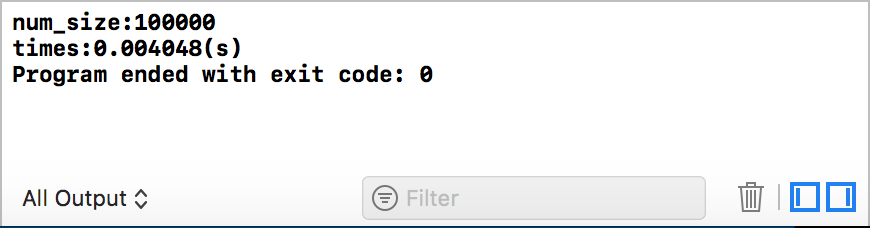
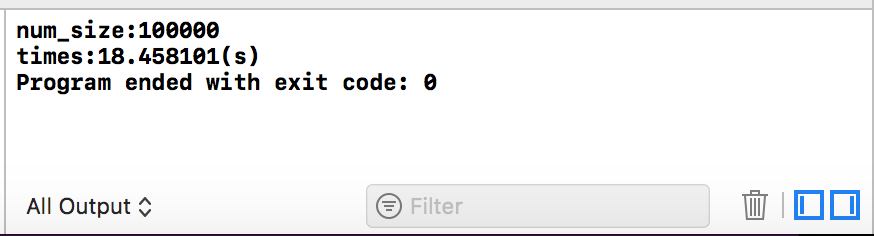


图3.4 n=100000：串行（左）并行（右）

# 4 并行优化方案设计

## 4.1任务描述

1. 设计大数场景并行实现的优化方案。
2. 测试并分析优化方案的加速比与正确性。

## 4.2实验方案

对于OpenMP环境下的并行斐波纳切数列计算方法，再从时间效率上去优化已经十分困难。对于通项公式的计算上，我已经将公共部分提取出来单独完成计算，所以在整个过程中，公共部分只需要完成一次计算。再想对齐优化已经十分困难。

至于并行度上，由于OpenMP环境是由编译器自动完成线程调度的，所以对于并行化的方式无从控制。

对于算法的空间复杂度来说，存在很大的优化空间，当前方案采用的是结构体存储的方式。如图4.1所示，其中的fibo[]数组是存放计算出来的斐波纳切数列。Num存储需要计算的数列的个数。当计算规模n变大时，每一个数据都需要存储，消耗的内存将会线性增长。若是想要优化空间复杂度，则不能够存储这么多项的数列。考虑到使用率的问题，完全可以对一定规模后的数据进行抛弃，不存储。这里假设只存储100项，则可以很大程度上的优化空间复杂度。

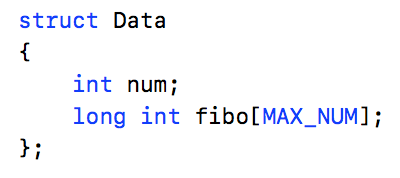


图4.1 数据存储结构