Project 1: Performance Measurement

author：耿飙、邓墨琳、崔瑜翔

Date:2016-10-10

I.Introduction

To write algorithms that can compute X^N for some positive integer N.

We can do this in three ways. Not only can we use the method with N-1 multiplications, but also we can use a recursive algorithm or an iterative algorithm to make it.

During writing these algorithms, we should also take the time and space complexity into consideration. To do so, we pay a lot of attention to the amount of calculation and the space that data storage needs.

II.Algorithm Specification

Algorithm 1 is to use multiplications for N-1 times. It simply uses the "for" cycle to finish repeated multiplications.

```
INPUT      X, N
        result =1
FOR (repeat for N-1 times ) {
        result =   result *X
}
OUTPUT     result
```

Algorithm 2 uses iterative algorithm with the help of binary operation. It divides the calculation into to two equal parts in every loop. It efficiently puts the result of former calculation into the coming calculation and avoids repetition of the same multiplication operation.

It works in the following way: if N is even, $X^N = X^{N/2} \times X^{N/2}$; and if N is odd, $X^N = X^{(N-1)/2} \times X^{(N-1)/2} \times X$.

To decide when and how to separate the calculation, we use the bitwise operation. Using the bitwise operation can manipulate the data directly in binary system. It saves the space for data storage and increased the calculating speed.

```
INPUT     X, N
        result =1
        t=X
WHILE (exponent hasn't been counted down to the end){
         IF (binary form of "N" has "1" in the end) {
                  result = result * t
         }
         using bitwise operation to delete the last number of the N in
binary form
         t = t × X
}
OUTPUT     result
```

Algorithm 3 uses the recursive algorithm. It divides the calculation into smallest pieces , two parts at one time. And every time the small pieces finish their calculations. They give their results to the upper calculation until all the results come back to the beginning one.

```
INPUT     X, N
IF ( N=0)
      OUTPUT    1
ELSE IF (N is odd)
      OUTPUT    the result of same function for (X&N/2) multiply X
ELSE
      OUTPUT the result of same function for (X&N/2)
```

III.Testing Results

	N	1000	5000	10000	20000	40000	60000	80000	100000
Algorithm 1	Iterations (*K*)	$2*10^6$	$2*10^6$	10^6	10^6	$5*10^5$	$5*10^5$	10^6	10^6
	Ticks	6875	34672	34391	68704	69140	103370	277057	3441919
	Total Time (sec)	6.875	34.672	34.391	68.704	69.14	103.37	277.057	341.919
	Duration (sec per run)	$3.4*10^{-6}$	$1.7*10^{-5}$	$3.4*10^{-5}$	$6.8*10^{-5}$	$1.38*10^{-4}$	$2.07*10^{-4}$	$2.77*10^{-4}$	$3.42*10^{-4}$
Algorithm 2 (iterative version)	Iterations (*K*)	10^6	$2*10^6$	10^6	10^6	$5*10^5$	$5*10^5$	10^6	10^6
	Ticks	47	63	31	31	16	16	47	47
	Total Time (sec)	0.047	0.063	0.031	0.031	0.016	0.016	0.047	0.047
	Duration (sec per run)	$2.4*10^{-8}$	$3.1*10^{-8}$	$3.1*10^{-8}$	$3.1*10^{-8}$	$3.2*10^{-8}$	$3.2*10^{-8}$	$4.7*10^{-8}$	$4.7*10^{-8}$
Algorithm 3 (recursive version)	Iterations (*K*)	10^6	$2*10^6$	10^6	10^6	$5*10^5$	$5*10^5$	10^6	10^6
	Ticks	125	156	94	94	47	47	94	109
	Total Time (sec)	0.125	0.156	0.094	0.094	0.047	0.047	0.094	0.109
	Duration (sec per run)	$6.3*10^{-8}$	$7.8*10^{-8}$	$9.4*10^{-8}$	$9.4*10^{-8}$	$9.4*10^{-8}$	$9.4*10^{-8}$	$9.4*10^{-8}$	$1.1*10^{-7}$

续表：

	N	200000	400000	600000	800000	1000000	2000000	10000000	100000000
Algorithm 2(iterative version)	Iterations (*K*)	10^6	10^6	10^6	10^6	10^6	10^6	10^6	10^6
	Ticks	47	62	62	62	63	63	64	78
	Total Time (sec)	0.047	0.062	0.062	0.062	0.063	0.063	0.064	0.078
	Duration (sec per run)	$4.7*10^{-8}$	$6.2*10^{-8}$	$6.2*10^{-8}$	$6.2*10^{-8}$	$6.3*10^{-8}$	$6.3*10^{-8}$	$6.4*10^{-8}$	$7.8*10^{-8}$
Algorithm 3(recursive version)	Iterations (*K*)	10^6	10^6	10^6	10^6	10^6	10^6	10^6	10^6
	Ticks	109	110	125	110	140	141	156	203
	Total Time (sec)	0.109	0.11	0.125	0.11	0.14	0.14	0.156	0.203
	Duration (sec per run)	$1.1*10^{-8}$	$1.1*10^{-8}$	$1.3*10^{-8}$	$1.3*10^{-8}$	$1.4*10^{-8}$	$1.4*10^{-8}$	$1.56*10^{-8}$	$2.03*10^{-8}$

注释：在每一次测试中 X=1.0001

IV.Analysis and Comments

1. 算法复杂度分析

a) 算法一（普通迭代）
在 for 循环中相乘 n 次，算法的时间复杂度是 O(N)

b) 算法二（二分迭代）
算法二较为巧妙，二分过程是对幂指数 N 进行二分操作，例如3^{14}，N=14，其二进制表示为 1110.可以表示为$3^{(1110)_2}$，由 8421 对应规则，$3^{14} = 3^{(1110)_2} = 3^8 * 3^4 * 3^2$；
又如$5^{19} = 5^{(10011)_2} = 5^{16} * 5^2 * 5^1$；于是，我们的二分操作是需要对幂指数进行，通过位运算>>，遍历二进制表示的每一 bit 位（相当于十进制中的除以 2），如果是 1，则乘上相应的权数。每次判断后，相应的权都会平方（t=t*t），以加入下一次迭代的运算.
在最坏的情况下，n 的对应的二进制数的每一位都是 1，一次迭代中有两次乘法，与算法三类似最坏结果也是 2logN。

c) 算法三（二分递归）
i if N is even, XN = XN / 2 × XN / 2; and if N is odd, XN = X(N – 1) / 2 × X(N – 1) / 2 × X.
我们可以得到 T(N) = T(N/2)+2（其中+2 是当 N 为奇数时最多两次乘法），我

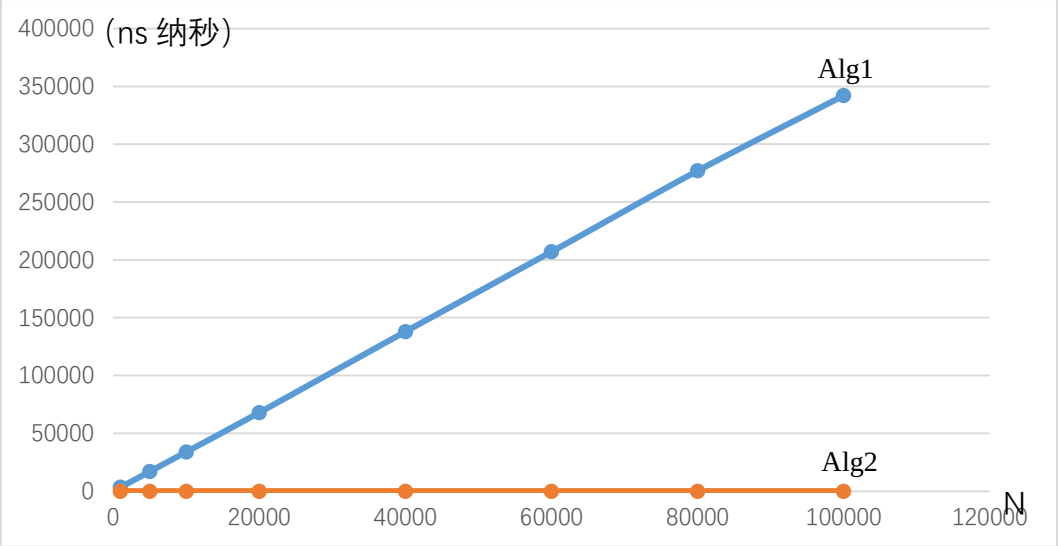
们令 T(1) = 1 units, 当 N = 2^k，$T(2^k) = T(2^{k-1}) + 2$,依次递推可得，$T(2^k) = 1+2k$,再令 $m = 2^k$,可得，$T(m) = 1+2\log m$，即 O(log N)；当 N 为奇数时，最坏的结果是 2log N。

2. 迭代次数

迭代次数最终确定为10^5~10^6，迭代次数为10^4在 N = 10^5和 80000 时，运行速度同样太快，算法二、三无法测出时间和 tick 数。而迭代次数为10^5~10^6可以使时间较好显示，同时运行时间不算太久。

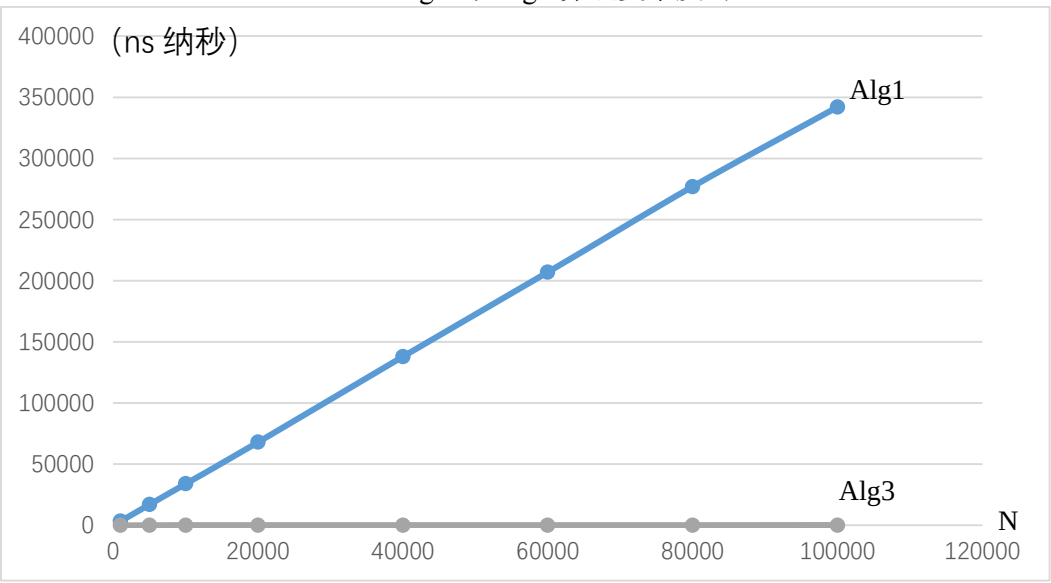
3. 分析与评价

Alg1 与 Alg2 算法复杂度的对比



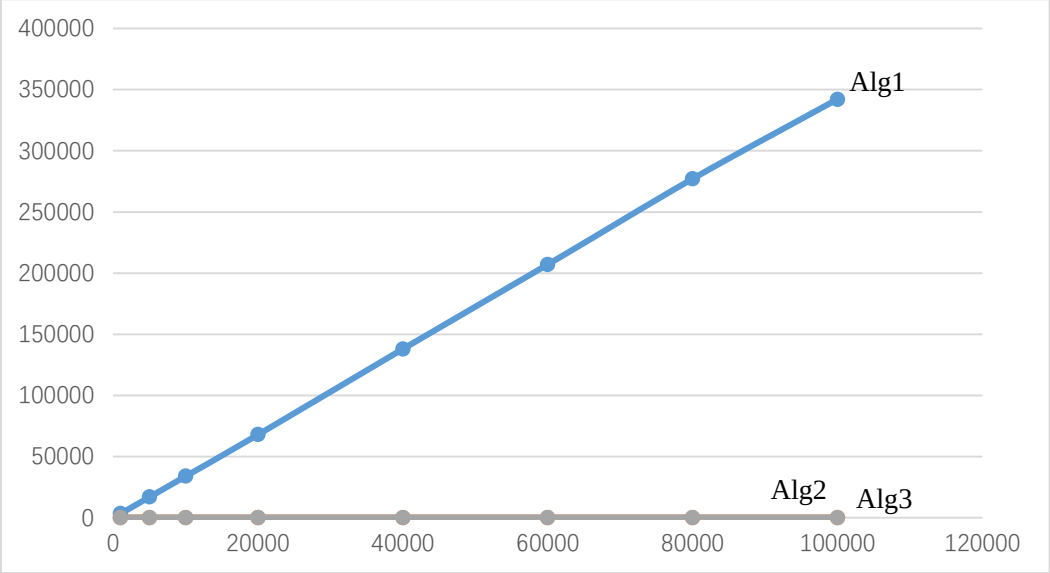
图表 1

Alg1 与 Alg3 算法复杂度对比



图表 2

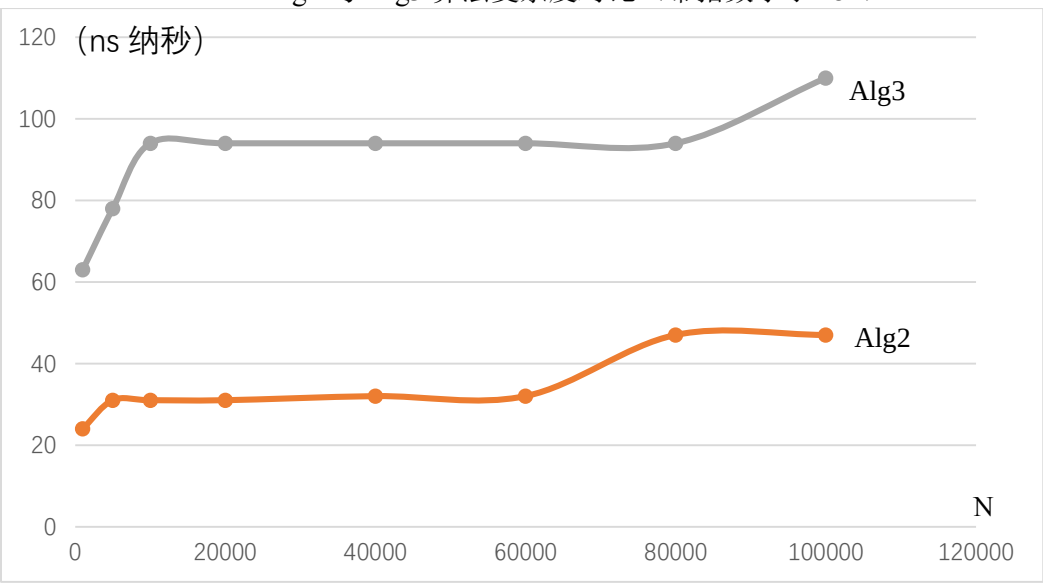
Alg1、Alg2、Alg3 算法复杂度对比



图表 3

分析 1：由表格 1、2、3 可知，算法一时间复杂度是 O(N)，相比于算法一，算法二算法三的时间复杂度明显偏小，而且在纵轴刻度较大的情况下，无法比较两者的时间复杂度。因此需要单独进行比较。

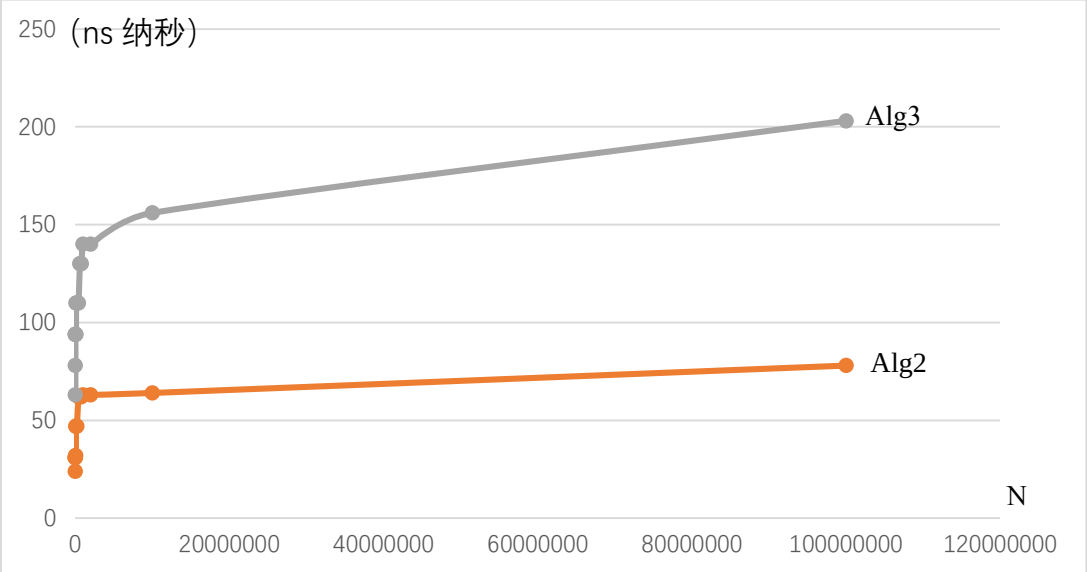
Alg2 与 Alg3 算法复杂度对比（幂指数小于$\mathbf{10^5}$）



图表 4

分析 2：当幂指数小于等于10^5时,随着 N 的增大，两者用时增长幅度较小，无法从增长类型看出时间复杂度的增长规律。同时，递归算法比迭代用时明显较多。

Alg2 与 Alg3 算法复杂度对比（高幂指数≤10^8）



图表 5

分析 3:当幂指数继续增大时，两者均表现了明显的趋势：当 N 较小时（相对较小），时间增长较快；当 N 较大时，随着 N 的增加，两者增幅均不大。两者的时间复杂度表现出对数增长的趋势，即 O(log N)，与我们的理论分析相符。同时注意到，虽然两者增幅不大，但是，算法三增长略快于算法二，且明显大于算法二所用时间，原因很明显，算法三递归调用的时间随着递归次数的增加，其调用函数、堆栈操作所用时间将超过计算本身的时间，导致时间明显增加。

结论：

a) 验证了算法一、二、三时间复杂度分别为 O(N)、O(log N)、O(log N)
b) 通过二分思想求幂可以使时间大大减小
c) 当幂指数 N 逐渐增大时，函数递归对所用时间影响越大，导致用时增加。

4. 评价：

1)本次实验较为成功，没有出现变量溢出等情况，一方面是算法的正确性；另一方面是变量定义的合理性，程序中可能较大的整型变量都定义为 long int，避免溢出发生。

2) 一个好的算法不仅要求代码简易，同时需要时空复杂度上的优越，递归在大部分情况下，可以是算法简易，但是时间复杂度，需要具体问题具体分析。分而治之的编程思想在算法设计中非常有用。在思路相同的情况下，迭代要比递归节省时间。

3) 团队交流是解决问题的最快途径

V.Appendix：Source Code

Please check it in code file

VI.Declaration

We hereby declare that all the work done in this project titled "projectl" is of our independent effort as a group.

VII.Duty Assignments:

Programmer: 耿飙3150105222

Tester: 邓墨琳 3150103457

Report Writer: 崔瑜翔 3150105215