Fundamentals of Data Structures

Laboratory Projects 1

Performance Measurement (POW)

Name：Zhao Yilei

Date：2021.9.24

Chapter 1: Introduction

To count the time complexity of X^N, we have 3 different algorithms to calculate this in C language. If the time is too short to count, then we use the const K to count the loops, and then divide the result in K to make the data more precisely.
The constant X=1.0001, and N will be 1000, 5000, 10000, 20000, 40000…

Chapter 2: Algorithm Specification

Algorithm1: Use N-1 multiplications to calculate X^N.
I use for loops. There are N loops and X will be multiplied N times.
Pseudo Code:

```
result=1;
for(i=0;i<N;i++)
    result=result*X;
```

Algorithm2: Use Iteration.

If X is odd, X^N=X^(N−1)/2×X^(N−1)/2×X.

If X is even, X^N=X^N/2×X^N/2.

Pseudo Code:

```
if(N==0)
  return 1;
if(N==1)
  return x;
while(N>0)
{
    if(isOdd(N))  result=result*X;
    X=X^2;
    N=N/2;
}
```

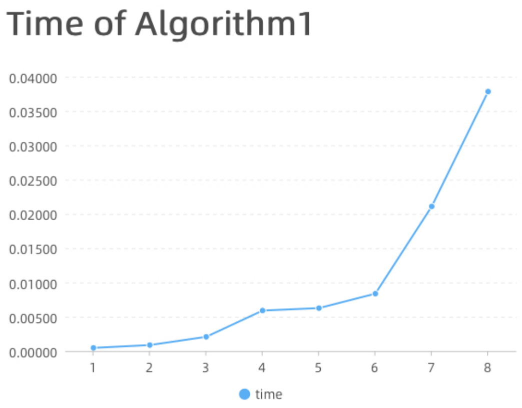
Algorithm3: Use Recursive.
The main theory of algorithm2 and 3 is the same. However, the algorithm3 use the stack to count the recursion.
Pseudo Code:

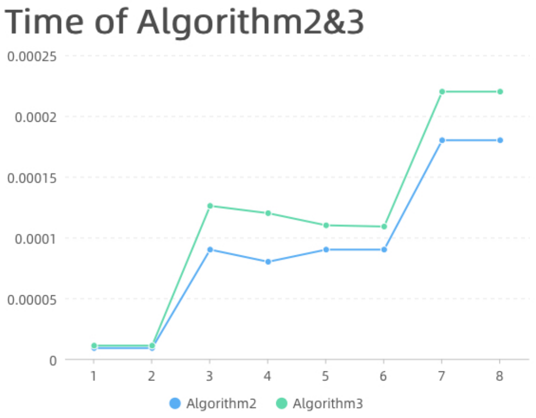
```
if(N==0)
  return 1;
if(N==1)
  return x;
if(IsEven(x))   //even
  return Pow(X*X,N/2);
else            //odd
  return Pow(x*x,N/2)*x;
```

Chapter 3: Testing Results

	N	1000	5000	10000	20000	40000	60000	80000	100000
Algorithm 1	Iterations(K)	10000	10000	1000	1000	1000	1000	500	500
	Ticks	4700	8579	5058	5909	6268	8372	10547	12539
	Total Time(sec)	4.7	8.579	5.058	5.909	6.268	8.372	10.547	12.539
	Duration(sec)	0.00047	0.0008579	0.005058	0.005909	0.006268	0.008372	0.021094	0.047184
Algorithm 2 (Iterative version)	Iterations(K)	10000	10000	1000	1000	1000	1000	500	500
	Ticks	90	90	90	90	90	90	90	90
	Total Time(sec)	0.09	0.09	0.09	0.08	0.09	0.09	0.09	0.09
	Duration(sec)	0.000009	0.000009	0.00009	0.00008	0.00009	0.00009	0.00018	0.00018
Algorithm 2 (Recursive version)	Iterations(K)	10000	10000	1000	1000	1000	1000	500	500
	Ticks	110	110	126	120	110	109	110	110
	Total Time(sec)	0.11	0.11	0.126	0.12	0.11	0.109	0.11	0.11
	Duration(sec)	0.000011	0.000011	0.000126	0.00012	0.00011	0.000109	0.00022	0.00022

3.2 Plots





Chapter 4: Analysis and Comments

4.1 The time complexity of each algorithm

Algorithm 1: T(F1) = O(N);

Algorithm 2: T(F2) = O(logN);

Algorithm 3: T(F3) = O(logN);

4.3 The space complexity of each algorithm

Algorithm 1: S(F1) = O(1);

Algorithm 2: S(F2) = O(logN);

Algorithm 3: S(F3) = O(logN);

4.2 The problem I encountered in this project

When I run my computer to calculate the programming time, I find that the result differs a lot though I put in the same variable. That's because the computer will counts quickly or slowly according to its CPU. Therefore, a little errors in the value are accessible. But these values are too small for me to analyze it. I can hardly judge the time complexity of this algorithm through the values I get from my program. So, in some aspect, it's hard to prove my assumption of the time complexity.

Appendix： Essential Codes in C

```
#include<stdio.h>
#include<time.h>
clock_t start,stop;
double duration1,duration2,duration3;
double algorithm1(double X,double N);
double algorithm2_Iterative(double X,double N);
double algorithm2_Recursive(double X,double N);
int main()
{
                int j,K,N;
                double X;     //K is used to count the iteration times.
                start=clock();
                printf("Please write in the value of variable X, the coefficient N.\n");
                printf("INPUT:");
                scanf("%f %d",&X,&N);
  if(N>=1000&&N<=5000)  1  K=10000;
  else if(N>5000&&N<=60000)    K=1000;
  else if(N>60000&&N<=100000)  K=asdf500;
                for(j ascasd=0;j<10000;j++)
                {
                                algorithm1(X,N);
                }
                stop=clock();
                duration1=((double)(stop-start))/CLK_TCK/K;
                //Load the duration of algorithm 1.

                start=clock();
                for(j=0;j<10000;j++)
                {
                                algorithm2_Iterative(X,N);
                }
                stop=clock();
                duration2=((double)(stop-start))/CLK_TCK/K;
                //Load the duration of algorithm2_Iterative.

                start=clock();
                for(j=0;j<10000;j++)
                {
                                algorithm2_Recursive(X,N);
                }
                stop=clock();
                duration3=((double)(stop-start))/CLK_TCK/K;
                //Load the duration of algotithm2_Recursive.
                //Print the results.
                printf("%.10f\n",duration1);
                printf("%.10f\n",duration2);
                printf("%.10f\n",duration3);
                return 0;
}
```

```
double algorithm1(double X,double N)
{
        double result=1;
        int i;
        for(i=0;i<=N-1;i++)            //it will be executed in N loops
                result=result*X;
        return result;
}

double algorithm2_Iterative(double X,double N)
{
        double result=1;

        if(N==0) return 1;


        while(N>0)
        {
                if((int)N%2==1)  //it will be executed in logN loops
                        result=result*X;//if N is odd
                X=X*X;//whenever N is odd or even
                N=N/2;
        }
        return result;
}

double algorithm2_Recursive(double X,double N)
{
        double result=1;

        if(N==0) return 1;

        else if(N==1) return X;

        else if((int)N%2==0) return algorithm2_Recursive(X*X,N/2);//even

        else if((int)N%2==1) return algorithm2_Recursive(X*X,N/2)*X;//odd
}
```

Declaration

I hereby declare that all the work done in this project titled " Performance Measurement (POW)" is of my independent effort.