斐波那契数列实验报告

万宗祺 201600090059

April 28, 2018

Contents

1 问题描述

Fibonacci 数列可由式 $a_k = a_{k-1} + a_{k-2}, k = 3, 4, \cdots$ 生成，其中初值 $a_1 = a_2 = 1$, 要编写函数计算该数列的第 n 项，要求：

1. 函数执行时首先测试输入参数，确保函数能正确调用；
2. 分别使用递归和循环的方式实现，如 $y = fibr(n)$ 递归方式，$y = fibl(n)$ 循环方式；
3. 针对不同的 n 值，分别调用两种函数进行计算，并使用 tic 和 toc 函数计时，比较两种方式的计算时间，并分析原因。

2 实验过程

下面给出 fibr.m 与 fibl.m 的代码，内含详细的注释

注. 由于 *listing* 宏包不支持中文注释，注释只能写英文了

```
% fibr.m
function fib = fibr(n)
%递归计算
if n==1 || n==2
% Boundary conditions at the end of recursion
    fib = 1;
else
    fib = fibr(n-1)+fibr(n-2);
% If do not get to the boundary, continue to call the function
end
```

```
% fibl.m
function fib = fibl(n)
% Calculate item n circularly
fibb = [1,1];% set initial value
for i=3:n
    fibb = [fibb fibb(i-1)+fibb(i-2)]; % calculate a new item
end
fib = fibb(n);
```

接下来，针对不同的 n，分别调用两种函数进行计算并计时，脚本代码如下:

```
%test.m
for n=21:25
    tic
    fibr(n);
```

```
    toc
    tic
    fibl(n);
    toc
end
```

3 结果分析

测试了计算 n 从 21 到 25，两个函数分别需要的时间，结果以表格展示如下

n	21	22	23	24	25
递归方式用时 (s)	0.107574	0.160348	0.240618	0.389781	0.633730
循环方式用时 (s)	0.000206	0.000067	0.000046	0.000071	0.000073

由此可以看出，循环方式所需要的时间远远少于递归方式，分析其原因，应该是递归方式进行了大量的重复计算，而循环方式顺推到 n，对序列中每一个项只计算了一遍，而递归方式却是随着 n 增大，计算次数以指数方式增长。