算法设计与分析 第1次作业

1. 时间复杂度与程序运行时间

请编写复杂度为 $O(n^2)$ 、 $O(n\log n)$ 、O(n)的任意程序,在不同问题规模下,记录运行时间,注明单位秒s或毫秒ms(写清运行代码的机器CPU配置)。

(严格来说,编写的程序,复杂度应该为 $\Theta(n^2)$, $\Theta(n \log n)$, $\Theta(n)$

CPU: Intel i3-2370M 2.40GHz

问题规模 n	$O(n^2)$	$O(n \log n)$	O(n)	
1000	0.004s	195300 ns 0 ms	10500 ns 0 ms	
10000	0.307s	2630200 ns 2 ms	108300 ns 0 ms	
20000	1.211s	1432600 ns 2 ms	211900 ns 0 ms	
40000	4.878s	3033700 ns 3 ms	446600 ns 0 ms	
100000	30.351s	4499800 ns 5 ms	2222100 ns 3 ms	
1000000	-	11718200 ns 11 ms	4861400 ns 5 ms	
10000000	-	105 ms	14263900 ns 14 ms	
100000000	-	1033 ms	125534400 ns 126 ms	
1000000000	-	10951 ms	1003376100 ns 1003 ms	

$O(n^2)$ 样例代码:

```
1 #include <stdio.h>
   #include <time.h>
 4 int main()
 5 {
        int n;
 6
        scanf("%d", &n);
 7
 8
9
        clock_t st = clock();
10
        long long sum = 0;
        for (int i = 0; i < n; ++i)
11
12
            for (int j = 0; j < n; ++j)
13
                ++sum;
        clock_t ft = clock();
14
        double dt = (ft - st) * 1.0 / CLOCKS_PER_SEC;
15
16
17
        printf("sum = %11d\n", sum);
        printf("duration: %f s\n", dt);
18
19
20
        return 0;
21
   }
```

 $O(n \log n)$ 代码:

```
public class Main {
1
 2
        public static void main(String[] args) {
 3
            // TODO: 需要给n赋值 ↓
4
            int sum = 0, n = ?;
 5
            long smsT = System.currentTimeMillis(), snsT = System.nanoTime();
 6
            for(int i = 1; i \le n; i++){
                for(int j = 1; j <= n; j+=i){
 7
 8
                    sum += 1;
9
                }
10
            }
11
            long emsT = System.currentTimeMillis(), ensT = System.nanoTime();
            System.out.println("纳秒: " + (ensT - snsT) + " ns");
12
13
            System.out.println("毫秒: " + (emsT - smsT) + " ms");
14
       }
15 }
```

O(n)代码:

```
public class Main {
 2
        public static void main(String[] args) {
 3
            // TODO: 需要给n赋值 ↓
 4
            int sum = 0, n = ?;
 5
            long smsT = System.currentTimeMillis(), snsT = System.nanoTime();
6
            for(int i = 1; i < n; i++){
7
                sum *= i;
            }
8
9
            long emsT = System.currentTimeMillis(), ensT = System.nanoTime();
            System.out.println("纳秒: " + (ensT - snsT) + "ns");
10
11
            System.out.println("毫秒: " + (emsT - smsT) + "ms");
12
        }
13
   }
```

2. 斐波那契数列计算

请用递归和递推实现斐波那契数列第加项的计算,结果对100000007取模。

```
Fibonacci numbers: F_0 = 0, F_1 = 1 F_n = F_{n-1} + F_{n-2}, (n \ge 2)
```

当n为多大时,递归计算已经明显变慢了? [当n = 41时,运行时间为851 ms; [当n = 42时,运行时间为1394 ms,超过了1 s,速度明显变慢了. 代码:

```
1
    public class Main {
2
        public static long fibonacci(Long n){
 3
 4
            if (n == 1){
 5
                return 1;
            else if(n == 2){
 6
7
                return 1;
8
            }else{
9
                return fibonacci(n - 1) + fibonacci(n - 2);
10
            }
        }
11
12
13
        public static void main(String[] args) {
```

```
14
            long sns = System.nanoTime(), sms = System.currentTimeMillis();
15
            // TODO: 需要给n赋值 ↓
            long n = ?;
16
17
            fibonacci(n);
18
            long ens = System.nanoTime(), ems = System.currentTimeMillis();
19
            System.out.println("纳秒: " + (ens - sns) + "ns");
20
            System.out.println("毫秒: " + (ems - sms) + "ms");
21
22
        }
23
    }
```

3. 最大子序和问题

请分别编写 $O(n^3)$ 、 $O(n^2)$ 、 $O(n\log n)$ 的代码,自己生成一些测试数据进行本地测试,并分别在CG平台上提交。

代码1: O(n)

```
1
    import java.util.Scanner;
2
 3
    public class Main {
 4
        public void maxSubAry(int[] nums){
 6
            int max = nums[0], sum = 0;
 7
            for (int num: nums) {
 8
                 sum = sum > 0 ? sum + num : num;
 9
                max = Math.max(sum, max);
10
11
            System.out.println(max);
        }
12
13
14
        public static void main(String[] args) {
15
            Scanner sc = new Scanner(System.in);
16
            int length = sc.nextInt();
            int nums[] = new int[length];
17
            for (int i = 0; i < length; i++){
18
19
                nums[i] = Integer.parseInt(sc.next());
20
21
            Main main = new Main();
22
            main.maxSubAry(nums);
23
            sc.close();
24
        }
25
    }
```

代码2: $O(n^2)$

```
import java.util.Scanner;
 1
 2
    public class NN {
 3
4
 5
        public void maxSubAry(int[] nums){
 6
             int max = Integer.MIN_VALUE;
 7
             for(int i = 0; i < nums.length; i++){</pre>
 8
                 int sum = 0;
 9
                 for(int j = i; j < nums.length; j++){
10
                     sum += nums[j];
```

```
11
                     if(max < sum) max = sum;
12
                }
13
            }
14
            System.out.println(max);
15
        }
16
17
        public static void main(String[] args) {
18
            Scanner sc = new Scanner(System.in);
19
            int length = sc.nextInt();
20
            int nums[] = new int[length];
            for (int i = 0; i < length; i++){
21
22
                 nums[i] = Integer.parseInt(sc.next());
23
            }
            Main main = new Main();
24
25
            main.maxSubAry(nums);
            sc.close();
26
27
        }
28
   }
```

代码3: $O(n^3)$

```
1
    import java.util.Scanner;
 2
 3
    public class Main {
 4
 5
        public void maxSubAry(int[] nums){
            int max = Integer.MIN_VALUE;
 6
 7
            for (int i = 0; i < nums.length; i++){
 8
                 for( int j = i; j < nums.length; j++){
 9
                     int sum = 0;
10
                     for(int k = i; k \le j; k++){
11
                         sum += nums[k];
12
                     }
13
                     if (sum > max) max = sum;
14
                }
15
16
            System.out.println(max);
        }
17
18
        public static void main(String[] args) {
19
20
            Scanner sc = new Scanner(System.in);
21
            int length = sc.nextInt();
22
            int nums[] = new int[length];
23
            for (int i = 0; i < length; i++){
24
                 nums[i] = Integer.parseInt(sc.next());
25
            }
            Main main = new Main();
26
27
            main.maxSubAry(nums);
28
            sc.close();
29
        }
30
   }
```