20跨2 刘国幸 2042110228

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：

```c
#include <stdio.h>
#include <time.h>

int main()
{
    int n;
    scanf("%d", &n);

    clock_t st = clock();
    long long sum = 0;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            ++sum;
    clock_t ft = clock();
    double dt = (ft - st) * 1.0 / CLOCKS_PER_SEC;

    printf("sum = %lld\n", sum);
    printf("duration: %f s\n", dt);

    return 0;
}
```

$O(n\log n)$代码：

```
public class Main {
    public static void main(String[] args) {
        // TODO: 需要给n赋值 ↓
        int sum = 0, n = ? ;
        long smsT = System.currentTimeMillis(), snsT = System.nanoTime();
        for(int i = 1 ; i <= n; i++){
            for(int j = 1; j <= n; j+=i){
                sum += 1;
            }
        }
        long emsT = System.currentTimeMillis(), ensT = System.nanoTime();
        System.out.println("纳秒: " + (ensT - snsT) + " ns");
        System.out.println("毫秒: " + (emsT - smsT) + " ms");
    }
}
```

$O(n)$代码：

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

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n项的计算，结果对1000000007取模。

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

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：

```
public class Main {

    public static long fibonacci(Long n){
        if (n == 1){
            return 1;
        }else if(n == 2){
            return 1;
        }else{
            return fibonacci(n - 1) + fibonacci(n - 2);
        }
    }

    public static void main(String[] args) {
```

```
        long sns = System.nanoTime(), sms = System.currentTimeMillis();
        // TODO: 需要给n赋值 ↓
        long n = ? ;
        fibonacci(n);
        long ens = System.nanoTime(), ems = System.currentTimeMillis();
        System.out.println("纳秒: " + (ens - sns) + "ns");
        System.out.println("毫秒: " + (ems - sms) + "ms");

    }
}
```

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：$O(n)$

```
import java.util.Scanner;

public class Main {

    public void maxSubAry(int[] nums){
        int max = nums[0], sum = 0;
        for (int num: nums) {
            sum = sum > 0 ? sum + num : num;
            max = Math.max(sum, max);
        }
        System.out.println(max);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int length = sc.nextInt();
        int nums[] = new int[length];
        for (int i = 0; i < length; i++){
            nums[i] = Integer.parseInt(sc.next());
        }
        Main main = new Main();
        main.maxSubAry(nums);
        sc.close();
    }
}
```

代码2：$O(n^2)$

```
import java.util.Scanner;

public class NN {

    public void maxSubAry(int[] nums){
        int max = Integer.MIN_VALUE;
        for(int i = 0; i < nums.length; i++){
            int sum = 0;
            for(int j = i; j < nums.length; j++){
                sum += nums[j];
```

```
                if(max < sum) max = sum;
            }
        }
        System.out.println(max);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int length = sc.nextInt();
        int nums[] = new int[length];
        for (int i = 0; i < length; i++){
            nums[i] = Integer.parseInt(sc.next());
        }
        Main main = new Main();
        main.maxSubAry(nums);
        sc.close();
    }
}
```

代码3：$O(n^3)$

```
import java.util.Scanner;

public class Main {

    public void maxSubAry(int[] nums){
        int max = Integer.MIN_VALUE;
        for (int i = 0; i < nums.length; i++){
            for( int j = i; j < nums.length; j++){
                int sum = 0;
                for(int k = i; k <= j; k++){
                    sum += nums[k];
                }
                if (sum > max) max = sum;
            }
        }
        System.out.println(max);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int length = sc.nextInt();
        int nums[] = new int[length];
        for (int i = 0; i < length; i++){
            nums[i] = Integer.parseInt(sc.next());
        }
        Main main = new Main();
        main.maxSubAry(nums);
        sc.close();
    }
}
```