

ACM 大一新生培训 Lecture 1

Hongliang Yang

广东外语外贸大学 ACM 实验室

2022 年 10 月 15 日

ACM 概述

如何学习 ACM

复杂度理论

O 记号

更多的复杂度计算的例子

数组

一维数组

二维数组


结构体数组

字符串数组

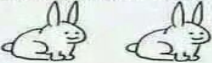
作业

做好充分的心理准备

先假设你有一只兔子。



假设有人又给了你另一只兔子。



现在，数一下你所拥有的兔子数量，你会得到结果是两只。也就是一只兔子加一只兔子等于两只兔子，也就是一加一等于二。

$$1 + 1 = 2$$

这就是算术的运算方法了。

那么，现在你已经对算术的基本原理有了一定了解。就让我们来看一看下面这个简单的例子，来把我们刚刚学到的知识运用到实践中吧。

试试看！
例题 1.7

$$\log \Pi(N) = \left(N + \frac{1}{2}\right) \log N - N + A - \int_N^{\infty} \frac{B_1(x) dx}{x}, \quad A = 1 + \int_1^{\infty} \frac{B_1(x) dx}{x}$$

$$\log \Pi(s) = \left(s + \frac{1}{2}\right) \log s - s + A - \int_0^{\infty} \frac{B_1(t) dt}{t+s}$$

工具网站

- ▶ 开始的第一步：学习科学的上网技巧
- ▶ <https://ikuuu.dev/>
- ▶ <https://www.google.com/>
- ▶ <https://github.com/ACM-Goldsmith/XCPC-Nurturance>

复杂度

算法：计算方法

一个问题可能有很多种算法来解决，但耗时有显著差异。

冒泡排序一个长度为 100000 的数组，耗时几分钟；快速排序耗时不到一秒钟。

如何衡量算法的运行时间？

E.g. 1

```
1  int getSum(int x) {  
2      sum = 0;  
3      for (int i = 1; i <= n; i++)  
4          for (int j = 1; j <= n; j++)  
5              sum += i * j;  
6      return sum;  
7  }
```

E.g. 1

```
1 int getSum(int x) {  
2     sum = 0;  
3     for (int i = 1; i <= n; i++)  
4         for (int j = 1; j <= n; j++)  
5             sum += i * j;  
6     return sum;  
7 }
```

$$n + n + n + \dots + n = \sum_{i=1}^n n = O(n^2)$$

E.g. 2

```
1  int getSum(int x) {  
2      sum = 0;  
3      for (int i = 1; i <= n; i++)  
4          for (int j = i; j <= n; j++)  
5              sum += i * j;  
6      return sum;  
7  }
```


E.g. 2

```
1 int getSum(int x) {  
2     sum = 0;  
3     for (int i = 1; i <= n; i++)  
4         for (int j = i; j <= n; j++)  
5             sum += i * j;  
6     return sum;  
7 }
```

$$1 + 2 + 3 + \dots + n = \sum_{i=1}^n i = \frac{n(n+1)}{2}$$

O 记号

n 很大的时候, n^2 与 $\frac{n(n+1)}{2}$ 区别不太大。

O 记号是“最坏情况下, 程序执行操作的总次数”。

如果是多个项相加, 只取最大的项。

省略掉所有的常数。

例子

- ▶ $\frac{n(n+1)}{2}$ 是 $O(n^2)$ 的
- ▶ $3n - 10$ 是 $O(n)$ 的
- ▶ $n^3 + 200n^2 + n + 233$ 是 $O(n^3)$ 的
- ▶ 2333333 是 $O(1)$ 的
- ▶ $233n^{100} - 1000000n + 1$ 是
- ▶ $1 + 2 + 4 + 8 + \cdots + 2^n$ 是

O 记号

n 很大的时候, n^2 与 $\frac{n(n+1)}{2}$ 区别不太大。

O 记号是“最坏情况下, 程序执行操作的总次数”。

如果是多个项相加, 只取最大的项。

省略掉所有的常数。

练习

- ▶ $233n^{100} - 1000000n + 1$ 是
- ▶ $1 + 2 + 4 + 8 + \cdots + 2^n$ 是
- ▶ $123 \log_2(n) + 566 \ln(n) + 10 \lg(n)$ 是

O 记号

n 很大的时候, n^2 与 $\frac{n(n+1)}{2}$ 区别不太大。

O 记号是“最坏情况下, 程序执行操作的总次数”。

如果是多个项相加, 只取最大的项。

省略掉所有的常数。

练习

- ▶ $233n^{100} - 1000000n + 1$ 是 $O(n^{100})$ 的
- ▶ $1 + 2 + 4 + 8 + \cdots + 2^n$ 是 $O(2^n)$ 的
- ▶ $123 \log_2(n) + 566 \ln(n) + 10 \lg(n)$ 是 $O(\log n)$ 的

更多的复杂度计算的例子

E.g. 3

```
1  int getSum(int x) {  
2      sum = 0;  
3      for (int i = 1; i <= n; i++)  
4          for (int j = n; j <= n + 5; j++)  
5              sum += i * j;  
6      return sum;  
7  }
```

E.g. 3

```
1  int getSum(int x) {  
2      sum = 0;  
3      for (int i = 1; i <= n; i++)  
4          for (int j = n; j <= n + 5; j++)  
5              sum += i * j;  
6      return sum;  
7  }
```

注意到，第 4-5 行是 $O(1)$ 的，所以总体的时间复杂度为 $O(n)$

复杂度理论总结

- ▶ 我们以 O 记号来表示程序的复杂程度。
- ▶ 时间复杂度是**操作次数**，空间复杂度是使用的**内存大小**。
- ▶ 复杂度只考虑**数量级**，如 $n^3 + 200n^2 + n + 233$ 是 $O(n^3)$ 的
- ▶ 时间复杂度越大，程序跑的时间就越长。

关于时间复杂度

可以估计，电脑一秒之内执行 1 亿次运算。

根据经验，如果时间复杂度算出来在 $2000w$ 以下，一般可以认为 $1s$ 之内能跑出来。

一维数组的语法

可以认为是数列！

二维数组的语法

可以认为是矩阵！（类型相同）

结构体数组的语法

可以认为是矩阵！（类型不同）

字符串数组的语法

以字符为元素的数组！

第一次讲课作业

- ▶ 作业地址：
<https://www.luogu.com.cn/contest/87757#problems>
邀请码: g5os
- ▶ 汇报地址：
<https://docs.qq.com/sheet/DUE9ucE9XaFphSUV3>
- ▶ 国庆礼包: <https://vjudge.net/contest/519420>