


（数据结构）十分钟搞定时间复杂度（算法的时间复杂度）

 raymondCaptain 关注

❤️ 32

👤 2017.11.02 10:17:16

📄 字数 1,644

👁 阅读 365,963

我们假设计算机运行一行基础代码需要执行一次运算。

```
1 | int aFunc(void) {
2 |     printf("Hello, World!\n"); // 需要执行 1 次
3 |     return 0; // 需要执行 1 次
4 | }
```

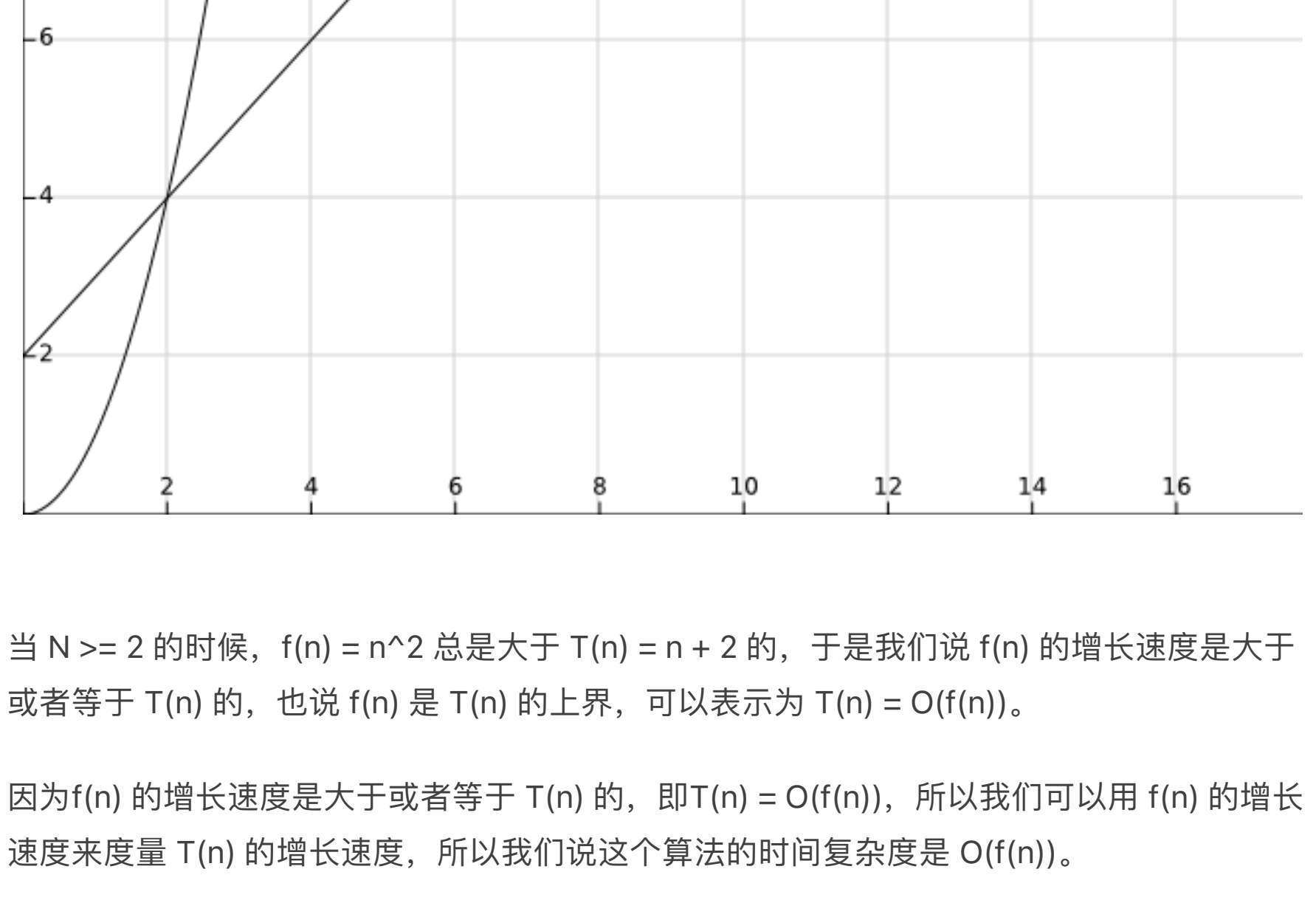
那么上面这个方法需要执行 2 次运算

```
1 | int aFunc(int n) {
2 |     for(int i = 0; i<n; i++) { // 需要执行 (n + 1) 次
3 |         printf("Hello, World!\n"); // 需要执行 n 次
4 |     }
5 |     return 0; // 需要执行 1 次
6 | }
```

这个方法需要 $(n + 1 + n + 1) = 2n + 2$ 次运算。

我们把 算法需要执行的运算次数 用 输入大小n 的函数 表示，即 T(n) 。
此时为了 估算算法需要的运行时间 和 简化算法分析，我们引入时间复杂度的概念。

定义：存在常数 c 和函数 f(N)，使得当 $N \geq c$ 时 $T(N) \leq f(N)$ ，表示为 $T(n) = O(f(n))$ 。
如图：



当 $N \geq 2$ 的时候， $f(n) = n^2$ 总是大于 $T(n) = n + 2$ 的，于是我们说 $f(n)$ 的增长速度是大于或者等于 $T(n)$ 的，也说 $f(n)$ 是 $T(n)$ 的上界，可以表示为 $T(n) = O(f(n))$ 。

因为 $f(n)$ 的增长速度是大于或者等于 $T(n)$ 的，即 $T(n) = O(f(n))$ ，所以我们可以用 $f(n)$ 的增长速度来度量 $T(n)$ 的增长速度，所以我们说这个算法的时间复杂度是 $O(f(n))$ 。

算法的时间复杂度，用来度量算法的运行时间，记作: $T(n) = O(f(n))$ 。它表示随着 输入大小n 的增大，算法执行需要的时间的增长速度可以用 $f(n)$ 来描述。

显然如果 $T(n) = n^2$ ，那么 $T(n) = O(n^2)$ ， $T(n) = O(n^3)$ ， $T(n) = O(n^4)$ 都是成立的，但是因为第一个 $f(n)$ 的增长速度与 $T(n)$ 是最接近的，所以第一个是最好的选择，所以我们说这个算法的复杂度是 $O(n^2)$ 。

那么当我们拿到算法的执行次数函数 T(n) 之后怎么得到算法的时间复杂度呢？

1. 我们知道常数项对函数的增长速度影响并不大，所以当 $T(n) = c$ ，c 为一个常数的时候，我们说这个算法的时间复杂度为 $O(1)$ ；如果 $T(n)$ 不等于一个常数项时，直接将常数项省略。

```
1 | 比如
2 | 第一个 Hello, World 的例子中 T(n) = 2，所以我说那个函数(算法)的时间复杂度为 O(1)。
3 | T(n) = n + 29，此时时间复杂度为 O(n)。
```

2. 我们知道高次项对于函数的增长速度的影响是最大的。 n^3 的增长速度是远超 n^2 的，同时 n^2 的增长速度是远超 n 的。同时因为要求的精度不高，所以我们直接忽略低此项。

```
1 | 比如
2 | T(n) = n^3 + n^2 + 29，此时时间复杂度为 O(n^3)。
```

3. 因为函数的阶数对函数的增长速度的影响是最显著的，所以我们忽略与最高阶相乘的常数。

```
1 | 比如
2 | T(n) = 3n^3，此时时间复杂度为 O(n^3)。
```

综合起来：如果一个算法的执行次数是 T(n)，那么只保留最高次项，同时忽略最高项的系数后得到函数 f(n)，此时算法的时间复杂度就是 $O(f(n))$ 。为了方便描述，下文称此为 大O推导法。

由此可见，由执行次数 T(n) 得到时间复杂度并不难，很多时候困难的是从算法通过分析和数学运算得到 T(n)。对此，提供下列四个便利的法则，这些法则都是可以简单推导出来的，总结出来以便提高效率。

1. 对于一个循环，假设循环体的时间复杂度为 O(n)，循环次数为 m，则这个循环的时间复杂度为 $O(n \times m)$ 。

```
1 | void aFunc(int n) {
2 |     for(int i = 0; i < n; i++) { // 循环次数为 n
3 |         printf("Hello, World!\n"); // 循环体时间复杂度为 O(1)
4 |     }
5 | }
```

此时时间复杂度为 $O(n \times 1)$ ，即 $O(n)$ 。

2. 对于多个循环，假设循环体的时间复杂度为 O(n)，各个循环的循环次数分别是 a, b, c,...，则这个循环的时间复杂度为 $O(n \times a \times b \times c \dots)$ 。分析的时候应该由里向外分析这些循环。

```
1 | void aFunc(int n) {
2 |     for(int i = 0; i < n; i++) { // 循环次数为 n
3 |         for(int j = 0; j < n; j++) { // 循环次数为 n
4 |             printf("Hello, World!\n"); // 循环体时间复杂度为 O(1)
5 |         }
6 |     }
7 | }
```

此时时间复杂度为 $O(n \times n \times 1)$ ，即 $O(n^2)$ 。

3. 对于顺序执行的语句或者算法，总的时间复杂度等于其中最大的时间复杂度。

```
1 | void aFunc(int n) {
2 |     // 第一部分时间复杂度为 O(n^2)
3 |     for(int i = 0; i < n; i++) {
4 |         for(int j = 0; j < n; j++) {
5 |             printf("Hello, World!\n");
6 |         }
7 |     }
8 |     // 第二部分时间复杂度为 O(n)
9 |     for(int j = 0; j < n; j++) {
10 |         printf("Hello, World!\n");
11 |     }
12 | }
```

此时时间复杂度为 $\max(O(n^2), O(n))$ ，即 $O(n^2)$ 。

4. 对于条件判断语句，总的时间复杂度等于其中 时间复杂度最大的路径 的时间复杂度。

```
1 | void aFunc(int n) {
2 |     if (n >= 0) {
3 |         // 第一条路径时间复杂度为 O(n^2)
4 |         for(int i = 0; i < n; i++) {
5 |             for(int j = 0; j < n; j++) {
6 |                 printf("输入数据大于等于零\n");
7 |             }
8 |         }
9 |     } else {
10 |         // 第二条路径时间复杂度为 O(n)
11 |         for(int j = 0; j < n; j++) {
12 |             printf("输入数据小于零\n");
13 |         }
14 |     }
15 | }
```

此时时间复杂度为 $\max(O(n^2), O(n))$ ，即 $O(n^2)$ 。

时间复杂度分析的基本策略是：从内向外分析，从最深层开始分析。如果遇到函数调用，要深入函数进行分析。

最后，我们来练习一下

一、基础题

求该方法的时间复杂度

```
1 | void aFunc(int n) {
2 |     for (int i = 0; i < n; i++) {
3 |         for (int j = i; j < n; j++) {
4 |             printf("Hello World\n");
5 |         }
6 |     }
7 | }
```

参考答案：
当 $i = 0$ 时，内循环执行 n 次运算，当 $i = 1$ 时，内循环执行 $n - 1$ 次运算.....当 $i = n - 1$ 时，内循环执行 1 次运算。
所以，执行次数 $T(n) = n + (n - 1) + (n - 2) \dots + 1 = n(n + 1) / 2 = n^2 / 2 + n / 2$ 。
根据上文说的大O推导法 可以知道，此时时间复杂度为 $O(n^2)$ 。

二、进阶题

求该方法的时间复杂度

```
1 | long aFunc(int n) {
2 |     if (n <= 2) {
3 |         return 1;
4 |     } else {
5 |         return aFunc(n - 1) + aFunc(n - 2);
6 |     }
7 | }
```

参考答案：
显然运行次数， $T(0) = T(1) = 1$ ，同时 $T(n) = T(n - 1) + T(n - 2) + 1$ ，这里的 1 是其中的加法算一次执行。
显然 $T(n) = T(n - 1) + T(n - 2)$ 是一个斐波那契数列，通过归纳证明法可以证明，当 $n \geq 1$ 时 $T(n) < (5/3)^n$ ，同时当 $n > 4$ 时 $T(n) > (3/2)^n$ 。
所以该方法的时间复杂度可以表示为 $O((5/3)^n)$ ，简化后为 $O(2^n)$ 。
可见这个方法所需的运行时间是以指数的速度增长的。如果大家感兴趣，可以试下分别用 1，10，100 的输入大小来测试下算法的运行时间，相信大家会感受到时间复杂度的无穷魅力。

👍 668人点赞

🗉 数据结构

🔍

更多精彩内容下载简书APP



"小礼物走一走，来简书关注我"

👍 赞赏支持

👤 共3人赞赏

 raymondCaptain 创造有价值的东西

总资产24 (约1.45元) 共写了2547字 获得694个赞 共313个粉丝

关注




怎样轻松学日语 近视了怎样恢复 孩子钢琴表现 深度学习算法 测孩子智商 孩子厌学怎么办

写下你的评论...

精彩评论 1

 vennnnny 78楼 2020.04.08 20:44
从大一到现在十年时间，总算有点明白这个东西怎么计算的了。。感动

 0error_0warning 2020.05.28 21:10
你不是一个人...五年了，终于看懂了...

 BigHead_81ad 01.23 00:53
@0error_0warning @0error_0warning 哈哈，这评论把我逗乐了！专门登录来评论下，虽然还没看文章，但冲你这个评论我决定看一下，希望我也能搞懂。。我可不止5年了。。


👤 添加新评论

全部评论 140


 开发应以大桶为重 96楼 2020.12.19 22:47

 骑着毛驴逗你玩儿 95楼 2020.10.22 07:55

整体还不错，就是有一个矛盾冲突的地方，就是第二个md和第四个md，for循环内的O(1)到底是相加，还是相乘？

 炒蛋蛋 01.12 16:34
 $O(n^2) + O(n^3) + O(n^4) = O(n^4)$ ，这样理解对吗

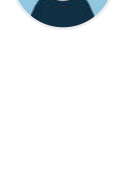
👤 添加新评论


 819699f5a40f 94楼 2020.10.05 17:31
很清楚，终于搞懂了！


 小mingmingming 93楼 2020.09.27 19:16
好b (╯▽╰) d

 智人一千 91楼 2020.08.17 12:02
定义：存在常数 c 和函数 f(N)，使得当 $N \geq c$ 时 $T(N) \leq f(N)$ ，表示为 $T(n) = O(f(n))$ 。


按照定义： $f(n) \geq T(n)$ ，和下面的转换规则：舍弃常数，舍弃低阶是矛盾的，因为 $f(n) < T(n)$ ，所以这句话有误

 char_hu 90楼 2020.08.13 20:13
整理得很不错！

 Lc1991 89楼 2020.08.11 16:19
为了搞懂时间复杂度，又去翻出高中时期的对数，数列，相关知识

 coder_YJ 88楼 2020.07.27 11:42
写的非常好，清晰明了，查了下百度谷歌的时间复杂度，差点睡着

 HanOba 87楼 2020.07.14 09:43
感谢分享。

 三国韩信 86楼 2020.07.10 18:42
那个斐波那契数列的，没看懂，绕晕在里面了

被以下专题收入，发现更多相似内容

Other 算法 数据结构与算法 Fuck IO... AI-Tech 算法

我要提高编程技术 展开更多

推荐阅读

编程之算法时间复杂度 更多精彩内容

算法复杂度 时间复杂度 空间复杂度 什么是时间复杂度 算法执行时间需通过依据该算法编制程序在计算机上运行时所消耗...

KODIE 阅读 2,008 评论 0 赞 9

算法的时间复杂度和空间复杂度-总结 转自: https://blog.csdn.net/zolalad/article/details/11848739 ...

王皓199207 阅读 1,321 评论 1 赞 3

算法的时间复杂度和空间复杂度 通常，对于一个给定的算法，我们要做 两项分析。第一是从数学上证明算法的正确性，这一步主要用到形式化证明的方法及相关...

西域小妈 阅读 1,162 评论 0 赞 11

数据结构——《大话数据结构》之时间复杂度 声明：该文章中内容为《大话数据结构》一书中的内容 1 函数的渐进增长 我们

现在来声明一下，两个算法A和B哪个好。...

Jack_Chao 阅读 880 评论 0 赞 13

最好的时光在前面 《我是个年轻人，我心境不太好》，这是一个看书名就让我产生强烈共鸣的书，它的推荐语对我的胃口：“我”26岁，是个研...

👤 赞赏 阅读 97 评论 0 赞 0



绝对不要告诉别人哦

说已上策

raymondCaptain 关注

总资产24 (约1.45元)

定义一个存放16进制数的数组 阅读 4,306

算法设计的要求和注意点 阅读 3,763

推荐阅读

如何用7只老鼠检测100杯水之哪一杯是有毒的？ 阅读 1,405

字节跳动三面offer到手，面试官都问了啥呢？ 阅读 6,335

【labuladong的算法小抄】滑动窗口算法 阅读 817

阿墨哥试答：你连个排序算法都讲不明白？出门右拐吧！ 阅读 1,085

快手三面（Java岗），意向已拿，盘点一下面试官都问了啥呢？ 阅读 19,413

绝对不要告诉别人哦

说已上策