算法的时间与空间复杂度（一看就懂）


不止思考(奎哥)
公众号「 不止思考 」互联网技术、思考、认知。

算法（Algorithm）是指用来操作数据、解决程序问题的一组方法。对于同一个问题，使用不同的算法，也许最终得到的结果是一样的，但在过程中消耗的资源和时间却会有很大的区别。

那么我们应该如何去衡量不同算法之间的优劣呢?

主要还是从算法所占用的「时间」和「空间」两个维度去考量。

- 时间维度：是指执行当前算法所消耗的时间，我们通常用「时间复杂度」来描述。
- 空间维度：是指执行当前算法需要占用多少内存空间，我们通常用「空间复杂度」来描述。

因此，评价一个算法的效率主要是看它的时间复杂度和空间复杂度情况。然而，有的时候时间和空间却又是「鱼和熊掌」，不可兼得的，那么我们就需要从中去取一个平衡点。

下面我来分别介绍一下「时间复杂度」和「空间复杂度」的计算方式。

一、时间复杂度

我们想要知道一个算法的「时间复杂度」，很多人首先想到的的方法就是把这个算法程序运行一遍，那么它所消耗的时间就自然而然知道了。

这种方式可以吗？当然可以，不过它也有很多弊端。
这种方式非常容易受运行环境的影响，在性能高的机器上跑出来的结果与在性能低的机器上跑的结果相差会很大。而且对测试时使用的数据规模也有很大关系。再者，并我们在写算法的时候，还没有办法完整的去运行呢。

因此，另一种更为通用的方法就出来了：「 **大O符号表示法** 」，即 T(n) = O(f(n))

我们先来看个例子：

```
for(i=1; i<=n; ++i)
{
   j = i;
   j++;
}
```

通过「 大O符号表示法 」，这段代码的时间复杂度为：O(n) ，为什么呢?

在 大O符号表示法中，时间复杂度的公式是： T(n) = O(f(n))，其中f(n) 表示每行代码执行次数之和，而 O 表示正比例关系，这个公式的全称是：**算法的渐进时间复杂度**。

我们继续看上面的例子，假设每行代码的执行时间都是一样的，我们用 1颗粒时间 来表示，那么这个例子的第一行耗时是1个颗粒时间，第三行的执行时间是 n个颗粒时间，第四行的执行时间也是 n个颗粒时间（第二行和第五行是符号，暂时忽略），那么总时间就是 1颗粒时间 + n颗粒时间 + n颗粒时间 ，即 (1+2n)个颗粒时间，即： T(n) = (1+2n)*颗粒时间，从这个结果可以看出，这个算法的耗时是随着n的变化而变化，因此，我们可以简化的将这个算法的时间复杂度表示为：T(n) = O(n)

为什么可以这么去简化呢，因为大O符号表示法并不是用于来真实代表算法的执行时间的，它是用来表示代码执行时间的增长变化趋势的。

所以上面的例子中，如果n无限大的时候，T(n) = time(1+2n)中的常量1就没有意义了，倍数2也意义不大。因此直接简化为T(n) = O(n) 就可以了。

常见的时间复杂度量级有：

- 常数阶O(1)
- 对数阶O(logN)
- 线性阶O(n)
- 线性对数阶O(nlogN)
- 平方阶O(n^2)
- 立方阶O(n^3)
- K次方阶O(n^k)
- 指数阶(2^n)

上面从上至下依次的时间复杂度越来越大，执行的效率越来越低。

下面选取一些较为常用的来讲解一下（没有严格按照顺序）：

1. **常数阶O(1)**

无论代码执行了多少行，只要是没有循环等复杂结构，那这个代码的时间复杂度就都是O(1)，如：

```
int i = 1;
int j = 2;
```

```
++i;
j++;
int m = i + j;
```

上述代码在执行的时候，它消耗的时候并不随着某个变量的增长而增长，那么无论这类代码有多长，即使有几万几十万行，都可以用O(1)来表示它的时间复杂度。

1. 线性阶O(n)

这个在最开始的代码示例中就讲解过了，如：

```
for(i=1; i<=n; ++i)
{
   j = i;
   j++;
}
```

这段代码，for循环里面的代码会执行n遍，因此它消耗的时间是随着n的变化而变化的，因此这类代码都可以用O(n)来表示它的时间复杂度。

1. 对数阶O(logN)

还是先来看代码：

```
int i = 1;
while(i<n)
{
    i = i * 2;
}
```

从上面代码可以看到，在while循环里面，每次都将 i 乘以 2，乘完之后，i 距离 n 就越来越近了。我们试着求解一下，假设循环x次之后，i 就大于 2 了，此时这个循环就退出了，也就是说 2 的 x 次方等于 n，那么 x = log2^n
也就是说当循环 log2^n 次以后，这个代码就结束了。因此这个代码的时间复杂度为：**O(logn)**

1. 线性对数阶O(nlogN)

线性对数阶O(nlogN) 其实非常容易理解，将时间复杂度为O(logn)的代码循环N遍的话，那么它的时间复杂度就是 n * O(logN)，也就是了O(nlogN)。

就拿上面的代码加一点修改来举例：

```
for(m=1; m<n; m++)
{
    i = 1;
    while(i<n)
    {
        i = i * 2;
```

```
    }
}
```

1. 平方阶O(n²)

平方阶O(n²) 就更容易理解了，如果把 O(n) 的代码再嵌套循环一遍，它的时间复杂度就是 O(n²)了。


```
for(x=1; i<=n; x++)
{
   for(i=1; i<=n; i++)
    {
       j = i;
       j++;
    }
}
```


这段代码其实就是嵌套了2层n循环，它的时间复杂度就是 O(n*n)，即 O(n²)
如果将其中一层循环的n改成m，即：

```
for(x=1; i<=m; x++)
{
   for(i=1; i<=n; i++)
    {
       j = i;
       j++;
    }
}
```

那它的时间复杂度就变成了 O(m*n)

1. 立方阶O(n³)、K次方阶O(n^k)

参考上面的O(n²) 去理解就好了，O(n³)相当于三层n循环，其它的类似。

除此之外，其实还有 平均时间复杂度、均摊时间复杂度、最坏时间复杂度、最好时间复杂度 的分析方法，有点复杂，这里就不展开了。

二、空间复杂度

既然时间复杂度不是用来计算程序具体耗时的，那么我也应该明白，空间复杂度也不是用来计算程序实际占用的空间的。

空间复杂度是对一个算法在运行过程中临时占用存储空间大小的一个量度，同样反映的是一个趋势，我们用 S(n) 来定义。

空间复杂度比较常用的有：O(1)、O(n)、O(n²)，我们下面来看看：

1. 空间复杂度 O(1)

如果算法执行所需要的临时空间不随着某个变量n的大小而变化，即此算法空间复杂度为一个常量，可表示为 O(1)
举例：

```
int i = 1;
int j = 2;
++i;
j++;
int m = i + j;
```

代码中的 i、j、m 所分配的空间都不随着处理数据量变化，因此它的空间复杂度 S(n) = O(1)

1. 空间复杂度 O(n)

我们先看一个代码：

```
int[] m = new int[n]
for(i=1; i<=n; ++i)
{
   j = i;
   j++;
}
```

这段代码中，第一行new了一个数组出来，这个数据占用的大小为n，这段代码的2-6行，虽然有循环，但没有再分配新的空间，因此，这段代码的空间复杂度主要看第一行即可，即 S(n) = O(n)

以上，就是对算法的时间复杂度与空间复杂度基础的分析，欢迎大家一起交流。

欢迎继续看 算法之

不止思考(奎哥)：算法一看就懂之「数组与链表」
zhuanlan.zhihu.com

和

不止思考(奎哥)：算法一看就懂之「堆栈」
zhuanlan.zhihu.com

和

不止思考(奎哥)：算法一看就懂之「队列」
zhuanlan.zhihu.com

，以及

不止思考(奎哥)：算法一看就懂之「递归」
zhuanlan.zhihu.com

算法与数据结构相关文章会持续更新，欢迎关注。

> 本文原创发布于微信公众号「不止思考」，欢迎关注，交流更多的 互联网认知、工作管理、大数据、Web、区块链技术。

编辑于 2019-09-10

算法　算法与数据结构

文章被以下专栏收录

不止思考
互联网、思考、认知。微信公众号「不止思考」

进入专栏

推荐阅读

BFS的套路二：识别BFS

BFS（广度优先搜索）的识别套路：问题的核心是一个图或者树，有结点及连接的特性。通过扩展结点，可能在搜索的过程中得到结果。当前结点的所有扩展结点，其属性与当前结点相关，且具有相

风清扬　发表于程序员算法...

谈算法与数据结

bange...　发

28 条评论　切换为时间排序

写下你的评论...

你瞒我瞒　11 个月前
可以讲的很清楚感谢

5

Zachary 9 个月前

假设循环x次之后，i 就大于 “n” 了，此时这个循环就退出了，也就是说 2 的 x 次方等于 n，那么 x = log2^n

9

 Arain 回复 Zachary 5 个月前

这个是不是写错了，公式应该是x=log2（n），2是底数

2

 HowKing 回复 Arain 5 个月前

底数无关紧要 可以省略 如果是5也一样 时间复杂度只是描述算法耗时的程度而已 不是计算具体时间

2

展开其他 2 条回复

竟然重名 8 个月前

简洁易懂，鼓掌

赞

 大宝剑之力 7 个月前

我看了java数据结构，上面写了，空间复杂度是以时间复杂度为上限的，这是不是某种情况来说，空间复杂度是等于时间复杂度的?

赞

 周大侠Jay 7 个月前

我可以转到我的公众号里吗😄

赞

 不止思考(奎哥) (作者) 回复 周大侠Jay 6 个月前

可以

赞

wenxin667 6 个月前

大致上是明白了，不过这段话是不是有误啊【假设每行代码的执行时间都是一样的，我们用1颗粒时间 来表示，那么这个例子的第一行耗时是1个颗粒时间，第三行的执行时间是 n个颗粒时间】 无论第几行 都应该是 1个颗粒时间啊 一共n行，就是n个颗粒时间 不该是 1+2n 啊?

5

 修心 6 个月前

我记得以前调试for循环的时候，会调到for，for的一行应该不是一个颗粒时间吧?

2

2sheep2simple 6 个月前

for 循环里的i 不需要声明吗?

赞

玉米片 5 个月前

我一直以为，一看就懂的后面还有一句话

赞

心瘾雨 回复 **玉米片** 3 天前

哈哈哈哈(ಡωಡ)hiahiahia

赞

Brizer 5 个月前

通俗易懂，非常优秀

赞

靠脸吃饭的我好饿 4 个月前

幡然醒悟，谢谢大佬

赞

鱼黄 4 个月前

。。。我想问为什么第一个循环是1颗粒，这个循环不也要执行n次吗？我是小白，刚开始学[捂脸]

2

孤叶飞雪 回复 **鱼黄** 13 天前

有同样的疑惑

赞

李小龙的李 3 个月前

通俗易懂，感谢!

赞

熊阿福 3 个月前

已收藏，谢谢大佬

赞

千夜晓梦 3 个月前

想到一个问题，时间复杂度与空间复杂度成反比么

赞

hjacky He 2 个月前

很清晰

赞

知乎用户 2 个月前

写的很清晰

赞

知乎用户 1 个月前

“我们试着求解一下，假设循环x次之后，i 就大于 2 了”，这里应该是大于 n

赞

七罪魔魂 25 天前

感谢

赞

zhu123 16 天前

看了很多相关视频都没看懂，看了您的讲解，我终于会算了，感谢

赞

1 2 下一页