算法设计与分析 LECTURE 1

第一周作业-Solution

LECTURER: 杨启哲 LAST MODIFIED: 2023 年 12 月 23 日

1. (教材习题 1.2) 令数组 $A[1,\dots 2000] = 1, 2\dots, 2000$, 用算法 $BinarySearch$ 搜索下列元素时, 执行了多少次比较运算?

   (a) -3 (b) 1 (c) 1000 (d) 2000

解答. 注意到 $BinarySearch$ 的 mid 更新规则为 $\lfloor (low+high)/2 \rfloor$, 其中需要进行的比较次数有 $low \le high$, $j = 0$, $x = a[mid]$ 和 $x < a[mid]$, 因此计算可得:

- (a) 41 次
- (b) 40 次
- (c) 3 次
- (d) 43 次

如果只考虑 $x = a[mid]$ 和 $x < a[mid]$, 即不考虑循环的判断比较则计算可得:

- (a) 20 次
- (b) 19 次
- (c) 1 次
- (d) 21 次

□

2. (教材习题 1.14) 用 $True$ 或者 $False$ 填空:

$f(n)$	$g(n)$	$f(n) = O(g(n))$	$f(n) = o(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$
$2n^3 + 3n$	$100n^2 + 2n + 100$				
$50n + \log n$	$10n + \log\log n$				
$50n \log n$	$10n \log\log n$				
$\log n$	$\log^2 n$				
$n!$	5^n				

解答. 计算可得:

$f(n)$	$g(n)$	$f(n) = O(g(n))$	$f(n) = o(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$
$2n^3 + 3n$	$100n^2 + 2n + 100$	0	0	1	0
$50n + \log n$	$10n + \log\log n$	1	0	1	1
$50n \log n$	$10n \log\log n$	0	0	1	0
$\log n$	$\log^2 n$	1	1	0	0
$n!$	5^n	0	0	1	0

□

3. (教材习题 1.18) 请找到两个单调递增函数 $f(n)$ 和 $g(n)$，使得 $f(n) \neq O(g(n))$ 并且 $g(n) \neq O(f(n))$。

解答. 考察下列两个函数 $f(n), g(n)$:

$$f(n) = \begin{cases} n^{2n}, & n\text{为偶数} \\ n^{2n+1}, & n\text{为奇数} \end{cases}, \quad g(n) = \begin{cases} n^{2n-1}, & n\text{为偶数} \\ n^{2n}, & n\text{为奇数} \end{cases},$$

注意到:

- 当 n 为奇数时，$\frac{f(n)}{g(n)} = n \to \infty$.
- 当 n 为偶数时，$\frac{g(n)}{f(n)} = n \to \infty$.

所以我们有 $f \neq O(g)$, $g \neq O(f)$。下面证两个函数都是单调递增的:

- 当 $n = 2k$ 时，我们有:

$$\begin{aligned} f(n) - f(n-1) &= (2k)^{4k} - (2k-1)^{4k-1} > 0 \\ g(n) - g(n-1) &= (2k)^{4k-1} - (2k-1)^{4k-2} > 0 \end{aligned}$$

- 当 $n = 2k+1$ 时，我们有:

$$\begin{aligned} f(n) - f(n-1) &= (2k+1)^{4k+3} - (2k)^{4k} > 0 \\ g(n) - g(n-1) &= (2k+1)^{4k+2} - (2k)^{4k-1} > 0 \end{aligned}$$

□

4. (教材习题 1.32) 考虑如下算法 COUNT6:

算法 1.18: COUNT6

输入: 正整数 n
输出: 第 6 步的执行次数 *count*

```
1: count ← 0
2: for i ← 1 to ⌊log n⌋ do
3:     for j ← i to i + 5 do
4:         for k ← 1 to i^2 do
5:             count ← count + 1
6:         end for
7:     end for
8: end for
```

(1) 第 6 步的执行了多少次?
(2) 要表示算法的时间复杂性，O 和 Θ 哪个符号更合适? 为什么?
(3) 算法的时间复杂性是什么?

解答. 算法一共有 3 个循环，第一个循环执行了 $\lfloor \log n \rfloor$ 次，第二个循环每次执行了 6 次，第三个循环每次执行了 i^2 次。

1. 第 6 步执行了 $6\lfloor\log n\rfloor\sum_{i=1}^{\lfloor\log n\rfloor} i^2$ 次。
2. Θ 是一个更好的符号，因为这不仅表示了算法至多这么快，也表示了算法至少这么快，更为精确。但很多时候我们只能估计出一个运行的时间，这其实只说明了至多这么快，所以经常我们用大 O 表示。
3. 由第一问可知第 6 步执行的次数为 $6\lfloor\log n\rfloor\sum_{i=1}^{\lfloor\log n\rfloor} i^2$，因此算法的时间复杂性为 $\Theta(\log^3 n)$.

□

5. (鸡蛋掉落) 假设现在有一幢 N 层高的楼和一些鸡蛋。对于这些鸡蛋来说，存在一层楼 T, 使得当这些鸡蛋从 T 层楼或更高的楼层摔落下去时鸡蛋会碎，反之鸡蛋则不会碎。你现在的目标是在下述条件下设计算法找到这个楼层 T:

   (1) 你只有 1 个鸡蛋，但有 T 次机会。
   (2) 你有 $\log N$ 个鸡蛋和 $\log N$ 次机会。
   (3) 你有 $\log T$ 个鸡蛋和 $2\log T$ 次机会。
   (4) 你有 2 个鸡蛋和 $2\sqrt{N}$ 次机会。

   (一个小小的挑战：能不能将第四种情况的机会次数减少到只跟 T 有关？即找到一个常数 c，使得你有 2 个鸡蛋和 $c\sqrt{T}$ 次机会的情况下找到 T。)

解答. 这道题的关键在于 N 和 T 的分别。

1. 算法非常简单，从 1 楼开始，逐层向上扔鸡蛋，直到鸡蛋碎了为止。这样最多需要 T 次。
2. 基于二分思想的算法，从 $\lfloor\frac{N}{2}\rfloor$ 层楼开始扔鸡蛋，有两种情况：
   - 鸡蛋碎了，说明 T 在 1 到 $\lfloor\frac{N}{2}\rfloor$ 之间，此时剩余 $\log N-1$ 个鸡蛋和 $\log N-1$ 次机会，重复上述二分过程。
   - 鸡蛋没碎，说明 T 在 $\lfloor\frac{N}{2}\rfloor+1$ 到 N 之间，此时剩余 $\log N-1$ 个鸡蛋和 $\log N-1$ 次机会，重复上述二分过程。
3. 依旧基于二分算法，但现在要求的是 $\log T$，因此我们需要先确定 T 的大小。

   (1) 从 $1, 2, 2^2, \ldots$ 楼层开始扔鸡蛋，直到鸡蛋碎了为止，此时 T 在 2^{k-1} 到 2^k 之间，其中 k 为扔鸡蛋的次数。
   (2) 仿照第二问的算法，在 $2^{k-1}\sim 2^k$ 之间的楼层找到确切的 T.

   显然算法的第一步需要消耗 $\log T$ 次机会和 1 个鸡蛋，因此由第二问的结论可知，一共至多消耗 $\log T$ 个鸡蛋和 $2\log T$ 次机会。
4. 鸡蛋变少了，次数变多了。因此我们可以设计如下的算法：

   (1) 从 $\lfloor\sqrt{n}\rfloor, 2\lfloor\sqrt{n}\rfloor, \ldots$ 逐层向上扔鸡蛋，直到鸡蛋碎了为止。这样最多需要 $\lfloor\sqrt{n}\rfloor$ 次。
   (2) 第一步确定 T 的范围在 $k\lfloor\sqrt{n}\rfloor$ 到 $(k+1)\lfloor\sqrt{n}\rfloor$ 之间，因此我们可以在这个范围内逐层向上扔鸡蛋，直到鸡蛋碎了为止。这样最多需要 $\lfloor\sqrt{n}\rfloor$ 次。

   从而我们可以用 2 个鸡蛋和 $2\sqrt{n}$ 次机会找到 T。
5. 和第 3 问的想法类似，想要做到 2 次机会和 $c\log T$ 次尝试，我们首先要确定 T 的范围，因此我们可以设计如下的算法，其中令 S_i 表示 $1+2+\cdots+i$ 的和：

   (1) 从 $1+S_1, 1+S_2, \ldots$ 逐层向上扔鸡蛋，直到鸡蛋碎了为止。这样最多需要 k 次。

(2) 从 $1+S_{k-1}+1$ 层开始逐逐层向上扔鸡蛋，直到鸡蛋碎了为止，此时便找到了相应的 T.

注意到在第一个鸡蛋碎的时候我们有：

$$2+\frac{(k-1)k}{2}=1+S_{k-1}+1\leq T\leq 1+S_k$$

从而我们有 $k\leq\sqrt{2}\cdot(\sqrt{T}-1)$，因此令 $c=2\sqrt{2}$ 即满足要求。

□