

# 算法设计与分析第一次作业

卢雨轩 19071125

2022 年 4 月 14 日

## 一、 算法时间复杂性问题

1. 试证明下面的定理：

(a) 如果  $f(n) = O(s(n))$  并且  $g(n) = O(r(n))$ , 则  $f(n) + g(n) = O(s(n) + r(n))$

证明. 因为

$$f(n) = O(s(n))$$

所以我们有：

$$\exists n_0, c_0, \forall n \geq n_0, f(n) \leq c_0 s(n) \quad (1)$$

同理：

$$\exists n_1, c_1, \forall n \geq n_0, g(n) \leq c_1 r(n) \quad (2)$$

由方程 (1), 方程 (2), 我们有：

$$\exists n_0, n_1, c_0, c_1, \forall n \geq n_0 + n_1, f(n) + g(n) \leq \max(c_0, c_1)s(n) + \max(c_0, c_1)r(n)$$

所以

$$\exists n_0, c_0, \forall n \geq n_0, f(n) + g(n) \leq c_0(s(n) + r(n))$$

$$f(n) + g(n) = O(s(n) + r(n)) \quad \square$$

(b) 如果  $f(n) = O(s(n))$  并且  $g(n) = O(r(n))$ , 则  $f(n) \times g(n) = O(s(n) \times r(n))$

证明. 因为

$$f(n) = O(s(n))$$

所以我们有：

$$\exists n_0, c_0, \forall n \geq n_0, f(n) \leq c_0 s(n) \quad (3)$$

同理：

$$\exists n_1, c_1, \forall n \geq n_0, g(n) \leq c_1 r(n) \quad (4)$$

由方程 (3), 方程 (4), 我们有：

$$\exists n_0, n_1, c_0, c_1, \forall n \geq n_0 + n_1, f(n) \times g(n) \leq c_0 c_1 s(n) r(n)$$

所以

$$\exists n_0, c_0, \forall n \geq n_0, f(n) \times g(n) \leq c_0 c_1 s(n) r(n)$$

$$f(n) \times g(n) = O(s(n) \times r(n)) \quad \square$$

## 2. 计算规模问题

假设某算法在输入规模为  $n$  时的计算时间复杂度（基本运算的次数）为：  $T(n) = 3 \times 2^n$ ，已知在 A 型计算机上实现并完成输入规模为  $n_A$  的该算法的时间为  $T$  秒，现有更先进的 B 型计算机，其运算速度为 A 型计算机的 64 倍。试求出若在先进的 B 型机上运行同一算法在  $T$  秒内能求解输入规模为多大的问题？

$$\begin{aligned} P_B &= 64P_A \\ T(n_A) &= 3 \times 2^{n_A} \\ T &= \frac{T(n_A)}{P_A} = \frac{T(n_B)}{P_B} \\ n_B &= 6 + n_A \end{aligned}$$

3. 证明  $\lg(n!) = \theta(n \lg n)$  (注：  $\lg(n!) = \theta(n \lg n)$  等价于  $\exists n_0 \in \mathbf{N}^*, c_1, c_2 \in \mathbf{R}^*, \forall n \geq n_0, c_1 n \lg n \leq \lg(n!) \leq c_2 n \lg n$ )

证明. 首先证明  $\lg(n!) \leq c_2 n \lg n$ :

$$\begin{aligned} \lg(n!) &\leq c_2 n \lg n \\ \iff \lg(1 \times 2 \times 3 \times \cdots \times n) &\leq c_2 n \lg n \\ \iff \lg 1 + \lg 2 + \cdots + \lg n &\leq c_2 n \lg n \end{aligned}$$

显然,

$$\begin{aligned} &\lg 1 + \lg 2 + \cdots + \lg n \\ &\leq \underbrace{\lg n + \lg n + \cdots + \lg n}_{n \uparrow} \\ &\leq n \lg n \end{aligned}$$

所以,  $\exists c_2 = 1$  使  $\forall n \in \mathbf{N}^*, \lg 1 + \lg 2 + \cdots + \lg n \leq c_2 n \lg n$

所以,  $\exists c_2 = 1$  使  $\forall n \in \mathbf{N}^*, \lg(n!) \leq c_2 n \lg n$ 。

再证明  $c_1 n \lg n \leq \lg(n!)$ , 设  $n_1 = \lceil \frac{n}{2} \rceil$ :

$$\begin{aligned} &\lg(n!) \\ &= \lg(1 \times 2 \times 3 \times \cdots \times n) \\ &= \lg 1 + \lg 2 + \cdots + \lg(n_1) + \lg(n_1 + 1) + \cdots + \lg n \\ &\geq \lg(n_1) + \lg(n_1 + 1) + \cdots + \lg n \\ &\geq \underbrace{\lg(n_1) + \lg(n_1) + \cdots + \lg(n_1)}_{n_1 \uparrow} \\ &\geq n_1 \lg n_1 \\ &= \left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil \end{aligned}$$

不妨设  $c_1 = 0.25$ 。当  $n \geq 4$  时：

$$(1 - 2c_1) \lg n - \lg 2 = 0.5 \lg n - \lg 2 \geq 0$$

$$(1 - 2c) \lg n \geq \lg 2$$

$$\lg n - \lg 2 \geq 2c_1 \lg n$$

$$\lg(n/2) \geq 2c_1 \lg n$$

$$\frac{1}{2} \lg\left(\frac{n}{2}\right) \geq c_1 \lg n$$

$$\frac{n}{2} \lg\left(\frac{n}{2}\right) \geq c_1 n \lg n$$

$$\left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil \geq c_1 n \lg n$$

所以，对于  $n_0 = 4$ ，对于  $\forall n \geq n_0$ ， $\exists c_1 = 0.25$  使  $c_1 n \lg n \leq \lg(n!)$

综上， $\exists n_0 = 4, c_1 = 0.25, c_2 = 1, \forall n \geq n_0, c_1 n \lg n \leq \lg(n!) \leq c_2 n \lg n$ 。

即  $\lg(n!) = \theta(n \lg n)$

□

## 二、 算法设计与分析问题

### 1. 最大值和最小值问题的最优算法

给定  $n$  个实数存放于一维数组  $A$  中，试设计一个算法在最坏情况下用  $3n/2 - 2$  次的比较找出  $A$  中的最大值和最小值（为简化，可假设  $n$  为偶数）。

答：见 [算法 1](#)

### 2. 聪明的判定问题

给定  $n$  个正整数，它们存放于一维数组  $S$  中，且  $S$  中的正整数从小到大有序排列，试设计一个算法，判定  $S$  中是否存在这样两个整数  $a$ 、 $b$ ，其和恰好与给定值  $x$  相等（要求算法的时间复杂性为  $O(n)$ ）。

答：见 [算法 2](#)

### 3. 快速排序问题

著名的快速排序算法对于计算机类专业的学生已不陌生，算法是递归实现的。请按照快速排序的思想，设计一个非递归的（即迭代的）快速排序算法。（提示：可利用栈）。

答：见 [算法 3](#)

---

**Algorithm 1** Find the max and min value of an array

---

```

1: procedure FIND( $A$ ) ▷ An array  $A$ 
2:    $mins \leftarrow$  Empty Array
3:    $maxs \leftarrow$  Empty Array
4:   for  $i, j \leftarrow$  paired values from  $A$  do ▷ Go through array  $A$  and divide it into two value sets.
5:     if  $i \geq j$  then
6:        $mins.PushBack(j)$ 
7:        $maxs.PushBack(i)$ 
8:     else
9:        $mins.PushBack(i)$ 
10:       $maxs.PushBack(j)$ 
11:    end if
12:  end for
13:  if  $A$  has not iterated value. then
14:     $mins.PushBack(\text{not iterated value})$ 
15:     $maxs.PushBack(\text{not iterated value})$ 
16:  end if
17:   $min \leftarrow mins_0$ 
18:   $max \leftarrow maxs_0$ 
19:  for  $i \leftarrow 1$  to  $|mins| - 1$  do ▷ Find the minimal value from mins using regular way
20:    if  $mins_i < min$  then
21:       $min \leftarrow mins_i$ 
22:    end if
23:  end for
24:  for  $i \leftarrow 1$  to  $|maxs| - 1$  do ▷ Find the maxnimal value from maxs using regular way
25:    if  $maxs_i > max$  then
26:       $max \leftarrow maxs_i$ 
27:    end if
28:  end for
29:  return  $max, min$ 
30: end procedure

```

---

---

**Algorithm 2** Test if an array  $S$  has two value that sum to  $x$ 

---

```
1: procedure TEST( $S$ ) ▷ An array  $S$ 
2:    $i \leftarrow 0$ 
3:    $j \leftarrow |S| - 1$ 
4:   while  $i \leq j$  do
5:     if  $S_i + S_j = x$  then
6:       return True
7:     else if  $S_i + S_j < x$  then
8:        $i \leftarrow i + 1$ 
9:     else
10:       $j \leftarrow j - 1$ 
11:    end if
12:  end while
13:  return False
14: end procedure
```

---

---

**Algorithm 3** Non-recursive quick sort

---

```

1: procedure TEST( $S$ ) ▷ An array  $arr$ 
2:    $beq, end \leftarrow$  Empty Stack
3:    $beq.Push(0)$ 
4:    $end.Push(|arr|)$ 
5:   while  $beq$  not empty do
6:      $L \leftarrow beq.top()$ 
7:      $R \leftarrow end.top() - 1$ 
8:     if  $L < R$  then ▷ Partition  $[L, R)$ 
9:        $piv \leftarrow arr_L$ 
10:      while  $L < R$  do ▷ Move piv around
11:        while  $arr_R \geq piv$  and  $L < R$  do
12:           $R \leftarrow R - 1$ 
13:        end while
14:        if  $L < R$  then
15:           $arr_L = arr_R$ 
16:           $L \leftarrow L + 1$ 
17:        end if
18:        while  $arr_L \leq piv$  and  $L < R$  do
19:           $L \leftarrow L + 1$ 
20:        end while
21:        if  $L < R$  then
22:           $arr_R = arr_L$ 
23:           $R \leftarrow R - 1$ 
24:        end if
25:      end while ▷ Now, piv should get to L.
26:       $arr_L \leftarrow piv$  ▷ And we should partition  $[L+1, end.top())$  and  $[beq.top(), L)$ 
27:       $beq.push(L + 1)$  ▷ Because we didn't pop  $beq$ , we just need to push  $L+1$ 
28:       $temp \leftarrow end.pop()$  ▷ And insert current  $L$  under  $end.top()$ 
29:       $end.push(L)$ 
30:       $end.push(temp)$ 
31:    else
32:       $beq.pop()$ 
33:       $end.pop()$ 
34:    end if
35:  end while
36: end procedure

```

---