



算法设计与分析

作业（一）

姓 名	熊恪峥
学 号	22920202204622
日 期	2022年2月24日
学 院	信息学院
课程名称	算法设计与分析

作业（一）

目录

1	题1.2	3
2	题1.5	3
3	题1.6	4
4	题1.7	4
5	题1.8	4
6	题1.9	4
6.1	算法	4
6.2	正确性	4
6.3	时间复杂度	5
7	题1.10	5
8	题1.11	6

1 题1.2

该算法符合算法的特点。

1. **有穷性**：又穷集合有有穷个数的子集，遍历这些子集求和并判断就有有限步。因此算法能在有穷操作内完成
2. **可行性**：求出子集可以用回溯算法实现，是可行的
3. **确定性**：该算法的每一步是确定的
4. **输入、输出**：该算法输入一个整数集和一个数，输出所有和为该数的子集

2 题1.5

$FindMax(A)$ 算法每一行执行的次数如表1

表格 1: 执行一行的次数

花费	次数
c_1	1
c_3	$n-1$
c_4	$\sum_{j=2}^n t_j$
c_5	1

其中 t_j 为

$$t_j = \begin{cases} 1 & \text{当for循环的第j轮中if条件成立} \\ 0 & \text{当for循环的第j轮中if条件不成立} \end{cases}$$

则算法运行所需要的时间为

$$T(n) = c_1 + c_3 \times (n - 1) + c_4 \times \sum_{j=2}^n t_j + c_5 \quad (1)$$

当 A 中的最大值的位置在末尾时，(1)中 t_j 满足

$$t_2 = \cdots = t_n = 1$$

此时 $T(n)$ 最大，最大值为

$$\begin{aligned} T(n) &= c_1 + c_3 \times (n - 1) + c_4 \times (n - 2) + c_5 \\ &= c_1 - c_3 - 2 \times c_4 + c_5 + (c_3 + c_4) \times n \end{aligned}$$

则 $FindMax(A)$ 的时间复杂度为

$$O(n)$$

3 题1.6

$FindMax(A)$ 有循环不变量 L_j

L_j : 在 for 循环的第 j 个迭代执行前, max 中有 $A[1 \dots j-1]$ 中的最大值

初始步: 在循环开始前 $j = 2$, $max = A[1]$ 是 $A[1 \dots 1]$ 中的最大值, L_2 为真;

归纳步: 如果在循环的第 k 个迭代前 L_k 为真, 则 max 有 $A[1 \dots k-1]$ 中的最大值, 当执行迭代 k 时, 若 $A[k] > max$ 则令 $max = A[k]$ 。此时 max 有 $A[1 \dots k]$ 中的最大值。在下一迭代开始前, L_k 为真;

终止步: 此时 $j = n+1$, 由第二步的保证, 则 max 有 $A[1 \dots n]$ 中的最大值。对于任意输入 A , 此 $FindMax(A)$ 都有一个正确的输出。因此 $FindMax(A)$ 是正确的。

4 题1.7

Algorithm 1 查找最大值, 返回下标

```

1: procedure FINDMAX( $A$ )
2:    $max \leftarrow 1$ 
3:   for  $j \leftarrow 2$  to  $n$  do
4:     if  $A[j] > A[max]$  then
5:        $max \leftarrow j$ 
   return  $max$ 

```

5 题1.8

$Exp(a, n)$ 有循环不变式 L_i

L_i : 在 $while$ 循环的第 i 个迭代执行前, pow 中有 a^{i-1}

初始步: 在循环开始前 $i = 1$, $pow = 1 = a^0 = a^{i-1}$, L_i 为真;

归纳步: 如果在循环的第 k 个迭代前 L_k 为真, 则 $pow = a^{k-1}$, 当执行迭代 k 令 $pow \leftarrow pow \times a$, 此时 $pow = a^k$ 。在下一迭代开始前, L_k 为真;

终止步: 此时 $i = n+1$, 由第二步的保证, 则 $pow = a^n$ 对于任意输入 (a, n) , 此 $Exp(a, n)$ 都有一个正确的输出。因此 $Exp(a, n)$ 是正确的。

6 题1.9

6.1 算法

该算法在没找到 x 时返回0.

6.2 正确性

$Find(A, x)$ 有循环不变量 L_j

L_j : 在 for 循环的第 j 个迭代执行前, idx 中有 $A[1 \dots j-1]$ 中等于 x 的下标

Algorithm 2 查找指定定值 x ，返回下标

```

1: procedure FIND( $A, x$ )
2:    $idx \leftarrow 0$ 
3:   for  $j \leftarrow 1$  to  $n$  do
4:     if  $A[j] == x$  then
5:        $idx \leftarrow j$ 
6:       break
   return  $idx$ 

```

初始步： 在循环开始前 $j = 1$ ， $max = A[1]$ 是 $A[1 \dots 0]$ 中等于 x 的下标， L_1 为真；

归纳步： 如果在循环的第 k 个迭代前 L_k 为真，则 max 有 $A[1 \dots k-1]$ 中等于 x 值，当执行迭代 k 时，若 $A[k] == x$ 则令 $idx = k$ 。此时 max 有 $A[1 \dots k]$ 中等于 x 值。在下一迭代开始前， L_k 为真；

终止步： 此时 $j = n+1$ ，由第二步的保证，则 max 有 $A[1 \dots n]$ 中的中等于 x 值。对于任意输入 A ，此 $Find(A, x)$ 都有一个正确的输出。因此 $Find(A, x)$ 是正确的。

6.3 时间复杂度

$Find(A, x)$ 算法每一行执行的次数如表2

表格 2: 执行每一行的次数

花费	次数
c_1	1
c_2	t
c_3	$t - 1$
c_4	$t - 1$
c_5	1

则算法运行所需要的时间为

$$T(n) = c_1 + c_2 \times t + c_3 \times (t - 1) + c_4 \times (t - 1) + c_5 \quad (2)$$

当 x 位于末尾时 t 最大， $t = n + 1$ ，则

$$\begin{aligned}
 T(n) &= c_1 + c_2 \times (n + 1) + c_3 \times n + c_4 \times n + c_5 \\
 &= (c_1 + c_2 + c_5) + (c_2 + c_3 + c_4) \times n
 \end{aligned}$$

该算法的最坏时间复杂度为 $O(n)$

7 题1.10

若前者比后者快，则有

$$100n^2 \leq 2^n$$

解得

$$n \geq 14.324727836998200633849297216651$$

则从 $n = 15$ 前者比后者快.

8 题1.11

若插入排序的效率高于归并排序，则有

$$8n^2 \leq 64n \log n$$

解得

$$n \leq 6.5070996729820298949891210615877$$

则当 n 取1, 2, 3, 4, 5, 6时插入排序的效率高于归并排序.