
# 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

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 |
| $c_4$ | $\Sigma_{j=2}^{n} t_j$ |
| $c_5$ | 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

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

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$L_j$

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

**初始步：** 在循环开始前$j=2$，$max=A[1]$是$A[1\ldots1]$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$A[1\ldots k-1]$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$A[1\ldots k]$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$A[1\ldots n]$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

该算法从头遍历集合，记录到当前位置为止最大数字的下标。算法如下

**算法 1** 查找最大值，返回下标

```
1: procedure FindMax(A)
2:     max ← 1
3:     for j ← 2 to n do
4:         if A[j] > A[max] then
5:             max ← 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

该算法循环判断每一个元素是否等于$x$，在没找到$x$时返回0。

## 6.2 正确性

$Find(A,x)$有循环不变量$L_j$

**算法 2** 查找指定定值x，返回下标

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

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

**初始步：** 在循环开始前$j=1$，$max=A[1]$是$A[1\ldots 0]$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$A[1\ldots k-1]$中等于$x$值，当执行迭代$k$时，若$A[k]==x$则令$idx=k$。此时$max$有$A[1\ldots k]$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$A[1\ldots n]$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

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

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

$$n \leq 6.507099672982029894989121061587 7$$

则当$n$取$1, 2, 3, 4, 5, 6$时插入排序的效率高于归并排序。

## 9 题2.2

由max的定义

$$f(x) \leq \max(f(x), g(x))$$
$$g(x) \leq \max(f(x), g(x))$$

则

$$f(x) + g(x) \leq 2\max(f(x), g(x))$$

则有

$$\max(f(x), g(x)) \geq \frac{f(x) + g(x)}{2}$$

即

$$\max(f(x), g(x)) = \Omega(f(x), g(x)) \tag{3}$$

由非负性

$$\max(f(x), g(x)) \leq f(x) + g(x)$$

即

$$\max(f(x), g(x)) = O(f(x) + g(x)) \tag{4}$$

由(3)和(9)可得

$$\max(f(x), g(x)) = \Theta(f(x) + g(x))$$

## 10 题2.9

**必要性：** 由$f(n) = \Theta(g(n))$得

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = k, 0 < k < \infty$$

则由

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$$

得$f(n) = \Omega(g(n))$，由

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$$

得$f(n) = O(g(n))$

**充分性：** 由$f(n) = O(g(n))$且$f(n) = \Omega(g(n))$得

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$$

且

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$$

则$\exists k > 0$且$k < \infty$使

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = k$$

则$f(n) = \Theta(g(n))$

## 11 编程题

**实现思路：**如代码1。先给每人分配0.01元保底，再随机凑成吉利值。

其中$scheme_diff$是吉利数值减去0.01。在输出前的处理是四舍五入到两位小数，防止由于浮点数表示的原因导致多位小数的问题。

代码 1: 红包发放

```
1
2 import random
3 from typing import Final, List
```

```
4
5  scheme_diff: List[ int] =  list({1.65, 1.67, 16.79, 1.77, 17.79, 1.87, 18.79, 1.98,
6          5.19, 0.65, 6.59, 6.65, 0.07, 0.87, 8.79, 8.87, 0.98, 9.89, 9.98})
7
8  m, n =  input("m and n:").split()
9
10 n =  int(n)
11 m =  float(m) - n * 0.01
12
13 result =  list([0.01 for i in  range(0, n)])
14
15 for i in  range(0, n):
16         amount = scheme_diff[random.randrange(0,  len(scheme_diff))]
17         if m - amount == 0:
18                 result[i] += amount
19                 break
20         elif m - amount < 0:
21                 result[i] += m
22                 break
23         else:
24                 m -= amount
25                 result[i] += amount
26
27 result =  list([ round(r, 2) for r in result])
28
29 print(result)
```