算法设计与分析 习题课一

任课老师：苏劲松、赖永炫

课程助教：欧阳洋、赵宇豪、龚玉雷

第六周　2025.3.28 文宣楼

目录

1. 提醒事项（3min）

2. 第一、二章部分习题（45min）

第一章提醒需要掌握的题型

二分查找 + 拓展内容

O(1)空间子数组换位【旋转数组】

O(1)空间原地归并

众数问题：哈希表、分治法、排序法

补充题：分治法

提醒：青蛙跳台阶的递归解法

目录

1. 提醒事项（3min）

2. 第一、二章部分习题（45min）

第一章提醒需要掌握的题型

二分查找 + 拓展内容

O(1)空间子数组换位【旋转数组】

O(1)空间原地归并

众数问题：哈希表、分治法、排序法

补充题：分治法

提醒：青蛙跳台阶的递归解法

1. 作业上交注意事项

作业名称：第3次作业3306_张三.pdf

上传时间：某章讲完的一周内交该章作业

代码要求：最低要求：伪代码；也可以写代码

作业答案：后续会上传到ftp对应目录

期中考试：...

2. 部分习题讲解

算法分析题 1

1-1 求下列函数的渐近表达式：

$3n^2+10n$； $n^2/10+2^n$； $21+1/n$； $\log n^3$； $10\log 3^n$

1-2 试论 $O(1)$和 $O(2)$的区别。

1-3 按照渐近阶从低到高的顺序排列以下表达式：$4n^2$，$\log n$，3^n，$20n$，2，$n^{2/3}$。又 $n!$ 应该排在哪一位？

1-4 （1）假设某算法在输入规模为 n 时的计算时间为 $T(n)=3\times 2^n$。在某台计算机上实现并完成该算法的时间为 t 秒。现有另一台计算机，其运行速度为第一台的 64 倍，那么在这台新机器上用同一算法在 t 秒内能解输入规模为多大的问题？

（2）若上述算法的计算时间改进为 $T(n)=n^2$，其余条件不变，则在新机器上用 t 秒时间能解输入规模为多大的问题？

（3）若上述算法的计算时间进一步改进为 $T(n)=8$，其余条件不变，那么在新机器上用 t 秒时间能解输入规模为多大的问题？

1-5 硬件厂商 XYZ 公司宣称他们最新研制的微处理器运行速度为其竞争对手 ABC 公司同类产品的 100 倍。对于计算复杂性分别为 n、n^2、n^3 和 $n!$的各算法，若用 ABC 公司的计算机在 1 小时内能解输入规模为 n 的问题，那么用 XYZ 公司的计算机在 1 小时内分别能解输入规模为多大的问题？

1-6 对于下列各组函数 $f(n)$和 $g(n)$，确定 $f(n)=O(g(n))$或 $f(n)=\Omega(g(n))$或 $f(n)=\theta(g(n))$，并简述理由。

（1）$f(n)=\log n^2$；	$g(n)=\log n+5$	（5）$f(n)=10$；	$g(n)=\log 10$
（2）$f(n)=\log n^2$；	$g(n)=\sqrt{n}$	（6）$f(n)=\log^2 n$；	$g(n)=\log n$
（3）$f(n)=n$；	$g(n)=\log^2 n$	（7）$f(n)=2^n$；	$g(n)=100n^2$
（4）$f(n)=n\log n+n$；	$g(n)=\log n$	（8）$f(n)=2^n$；	$g(n)=3^n$

1-7 证明 $n!=o(n^n)$。

1-8 下面的算法段用于确定 n 的初始值。试分析该算法段所需计算时间的上界和下界。

```
while(n>1)
   if(odd(n))
      n = 3*n+1;
   else
      n = n/2;
```

1-9 证明：如果一个算法在平均情况下的计算时间复杂性为 $\theta(f(n))$，则该算法在最坏情况下所需的计算时间为 $\Omega(f(n))$。

符号的基本定义

常见的时间复杂度排序

给一个函数求时间复杂度

递归函数的时间复杂度

主定理

下面那条规则是正确的？（ ）

A. $\{f(n)=O(F(n)), g(n)=O(G(n))\}\Rightarrow f(n)/g(n)=O(F(n)/G(n))$

B. $\{f(n)=O(F(n)), g(n)=O(G(n))\}\Rightarrow f(n)/g(n)=\Theta(F(n)/G(n))$

E. $\{f(n)=\Omega(F(n)), g(n)=\Omega(G(n))\}\Rightarrow f(n)/g(n)=\Omega(F(n)/G(n))$

D. $\{f(n)=\Theta(F(n)), g(n)=\Theta(G(n))\}\Rightarrow f(n)/g(n)=\Theta(F(n)/G(n))$

以下代码段的时间复杂性是（ ）.

```
for (j=1; j<=n;j++)
    for (k=n; k>=1; k/=2)
        count++;
```

A. $O(n^2)$ B.O(nlogn) C. O(logn) D. O(n)

2. 部分习题讲解

2-3　改写二分搜索算法。

请改写二分搜索算法，使得当搜索元素 x 不在数组中时，返回小于 x 的最大元素位置 i 和大于 x 的最小元素位置 j。设 a[0:n−1]是已排好序的数组。当搜索元素在数组中时，i 和 j 相同，均为 x 在数组中的位置。

```
template<class Type>
int BinarySearch(Type a[], const Type& x, int n) {          // 在a[0]<=a[1]<=...<=a[n-1]中搜索x
  // 找到x时返回其在数组中的位置，否则返回-1
  int  left = 0;          int  right = n-1;
  while (left <= right) {
    int  middle = (left+right)/2;
    if (x == a[middle])
      return middle;
    if (x > a[middle])
      left = middle+1;
    else
      right = middle-1;
  }
  return-1;                                                 // 未找到x
}
```

教科书的二分查找：简单，容易理解，通用性不足

2. 部分习题讲解

更简单&更重要的问题：找有序数组到第一个大于等于target的数字的下标
寻找排序数组中，某个给定元素应该插入的位置。

https://leetcode.cn/problems/binary-search/

https://leetcode.cn/problems/find-first-and-last-position-of-element-in-sorted-array/description/

实现中的问题 [编辑]

> 尽管二分查找的基本思想相对简单，但细节可以令人难以招架 ... — 高德纳[2]

当乔恩·本特利将二分搜索问题布置给专业编程课的学生时，百分之90的学生在花费数小时后还是无法给出正确的解答，主要因为这些错误程序在面对边界值的时候无法运行，或返回错误结果。[16]1988年开展的一项研究显示，20本教科书里只有5本正确实现了二分搜索。[17]不仅如此，本特利自己1986年出版的《编程珠玑》一书中的二分搜索算法存在整数溢出的问题，二十多年来无人发现。Java语言的库所实现的二分搜索算法中同样的溢出问题存在了九年多才被修复。[18]

参考标库写法：

c++ 的 lower_bound

python的二分查找标准库 bisect_left

```python
a = [2, 3, 4, 7, 7, 7, 8, 10]
ans = bisect_left(a, 7)  # 第一个大于等于7的元素

print(ans)  # 3

ans = bisect_right(a, 7)  # 第一个大于等于7的元素
print(ans)  # 6

def bisect_left_impl(a, target):
    l, r = 0, len(a)-1
    while l <= r:
        m = (l + r) // 2
        if a[m] < target:
            l = m + 1  # 上课的时候填空
        else:  # >=
            r = m - 1
    return l

ans = bisect_left_impl(a, 7)  # 3
print(ans)
```

2. 部分习题讲解

2-8　$O(1)$空间子数组换位算法。

设 a[0:n−1]是有 n 个元素的数组，k（0≤k≤n−1）是一个非负整数。试设计一个算法将子数组 a[0:k−1]与 a[k:n−1]换位。要求：算法在最坏情况下耗时 $O(n)$，且只用到 $O(1)$的辅助空间。

https://leetcode.cn/problems/rotate-array/　　轮转数组多种方法的时间复杂度分析

1. 循环换位法：每次整体向右边移动1位， 移动k次
2. 三次反转法
3. 新开辟一个数组，把旧的数字放到新的位置
4. without gcd? with gcd

   {1, 2, 3, 4, 5} shiftright 2 -> ans {4, 5, 1, 2, 3}

   {1, 2, 3, 4} shiftright 2

上课会展示代码讲解

nums = "----->-->"; k =3
result = "-->----->";
reverse "----->-->" we can get "<--<-----"
reverse "<--" we can get "--><-----"
reverse "<-----" we can get "-->----->"
this visualization help me figure it out :)

2. 部分习题讲解

2-1　众数问题。

问题描述：给定含有 n 个元素的多重集合 S，每个元素在 S 中出现的次数称为该元素的重数。多重集 S 中重数最大的元素称为众数。例如，$S=\{1, 2, 2, 2, 3, 5\}$。多重集 S 的众数是 2，其重数为 3。

算法设计：对于给定的由 n 个自然数组成的多重集 S，计算 S 的众数及其重数。

数据输入：输入数据由文件名为 input.txt 的文本文件提供。文件的第 1 行为多重集 S 中元素个数 n；在接下来的 n 行中，每行有一个自然数。

结果输出：将计算结果输出到文件 output.txt。输出文件有 2 行，第 1 行是众数，第 2 行是重数。

法1：哈希表

1.统计元素出现次数：使用哈希表（字典）记录每个元素及其出现的次数。

2.在遍历集合的过程中，跟踪当前的最大重数及其对应的元素。每次更新哈希表后，检查当前元素的重数是否超过已记录的最大值。如果是，则更新最大值和众数。

时间复杂度：O(n)

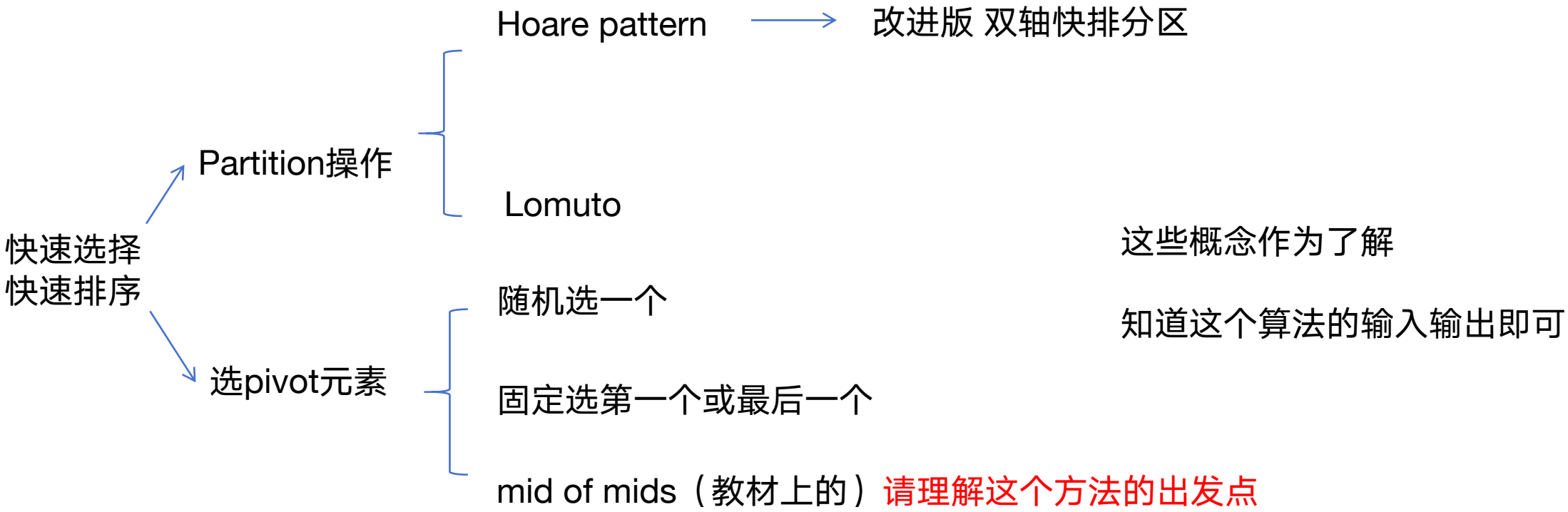
空间复杂度：O(m) < O(n)，m为集合中不同元素的数量。

法3 摩尔投票 O(n), O(1)

众数问题法2：分治法

补充预备知识：快速选择算法、荷兰国旗（变种快排partition）

快速选择算法：在O(n)时间找到无序数组的第k小元素，递归分治算法



众数问题：分治法

https://www.bilibili.com/video/BV1hr4y1y7w2/

补充预备知识：快速选择算法、荷兰国旗（变种快排partition）

荷兰国旗算法：给一个数字x,一次遍历，将数组划分为三个区域，小于x的，等于x的，大于x的，返回两个边界

x= 5

[2 3 5 7 0 1 4 5 6]

[2 | 3 5 7 0 1 4 5 6]

[2 3 | 5 7 0 1 4 5 6]

[2 3 | 5 7 0 1 4 5 6]

[2 3 | 5 6 0 1 4 5 | 7]

[2 3 | 5 5 0 1 4 | 6 7]

[2 3 | 5 5 0 1 4 | 6 7]

[2 3 0 | 5 5 1 4 | 6 7]

绿色维护小于x区域的右边界，红色维护大于x区域的右边界，蓝色是探测指针，当前数字

1. cur数字 < x: 当前数字和小区下一个交换，小区右移，当前下一个

2. cur数字等于x，cur直接+

3。 cur数字大于x，cur和大区前一个交换，大区域左移，cur不动

[2 3 0 1 | 5 5 4 | 6 7]

[2 3 0 1 4 | 5 5 | 6 7]

法2：分治

1. 找出数组的中位数，利用Partition函数将数组划分为小于中位数和大于中位数的两部分。
2. 计算中位数的重数，更新众数。
3. 递归计算左右两部分中位数的重数及更新众数。

时间复杂度：O(nlogn)

空间复杂度：O(logn)。

```
void mode(int l, int r) {
   int  l1, r1;
   int  med = median(a, l, r);
   split(a, med, l, r, l1, r1);
   if(largest < r1-l1+1)
      largest = r1-l1+1, element = med;
   if (l1-l > largest)
      mode(l, l1-1);
   if (r-r1 > largest)
      mode(r1+1, r);
}
```

median 函数 就是前面的 O(n) 时间选择算法

split就是前面的三分区算法（荷兰国旗算法）

补充题：

设$T[1:n]$是一个含有n个元素的数组。如果元素x的出现次数超过$n/2$，称元素x为数组T的主元素。

(1) 如果这n个元素存在序关系，比如n个整数

(2) 如果这n个元素不存在序关系，比如n个坐标

请分别针对上述两种情况，分别设计时间复杂性为$O(n)$的分治算法，判断该数组里是否有主元素。

不存在序关系的——摩尔投票更合适

（1）若x是T的主元素，则x一定是T的中位数。

1. 利用线性时间选择算法找到T的中位数median
2. 遍历数组，计算median的重数，通过重数判断median是否是主元素

```
int master(T)
{
     int median = Select(T, 1, n, (n+1)/2);  //求中位数
     int cnt = 0
      //计算中位数出现次数
     for (int i = 1; i <= n; i ++)
         if (T[i] == median) cnt ++;
     if (cnt > n/2)
        return median
     else return -1;
}
```

时间复杂度：线性时间选择O(n) + 计算重数O(n) = O(n)

```
function findMajorityUnordered(nums):
    if nums is empty:
        return False
    candidate, _ = divideAndConquerVote(nums, 0, length(nums) - 1)
    count = 0
    for num in nums:
        if num == candidate:
            count += 1
    return count > length(nums) / 2

function divideAndConquerVote(nums, left, right):
    if left == right:
        return (nums[left], 1)  // 只有一个元素，候选元素是自己，票数为1

    mid = (left + right) / 2
    (left_cand, left_count) = divideAndConquerVote(nums, left, mid)
    (right_cand, right_count) = divideAndConquerVote(nums, mid + 1, right)

    if left_cand == right_cand:
        return (left_cand, left_count + right_count)  // 相同元素，票数相加

    return (left_cand, left_count - right_count) if left_count > right_count
           else (right_cand, right_count - left_count)
```

$$T(n) = 2T(n/2) + O(1)$$

时间复杂度：O(logn)+O(n)=O(n)，空间复杂度O(logn)

2-7　集合划分问题。

问题描述：n 个元素的集合$\{1, 2, \cdots, n\}$可以划分为若干非空子集。例如，当 n=4 时，集合$\{1, 2, 3, 4\}$可以划分为 15 个不同的非空子集如下：

$\{\{1\}, \{2\}, \{3\}, \{4\}\}$　　$\{\{1, 3\}, \{2, 4\}\}$
$\{\{1, 2\}, \{3\}, \{4\}\}$　　$\{\{1, 4\}, \{2, 3\}\}$
$\{\{1, 3\}, \{2\}, \{4\}\}$　　$\{\{1, 2, 3\}, \{4\}\}$
$\{\{1, 4\}, \{2\}, \{3\}\}$　　$\{\{1, 2, 4\}, \{3\}\}$
$\{\{2, 3\}, \{1\}, \{4\}\}$　　$\{\{1, 3, 4\}, \{2\}\}$
$\{\{2, 4\}, \{1\}, \{3\}\}$　　$\{\{2, 3, 4\}, \{1\}\}$
$\{\{3, 4\}, \{1\}, \{2\}\}$　　$\{\{1, 2, 3, 4\}\}$
$\{\{1, 2\}, \{3, 4\}\}$

算法设计：给定正整数 n，计算出 n 个元素的集合$\{1, 2, \cdots, n\}$可以划分为多少个不同的非空子集。

先看一个转化的问题：

有一个n个元素的集合 {1, 2, 3, 4 ...n} 现在需要把他划分成一些非空子集 总共有几种划分方式？

$$B(n) = \sum_{k=1}^{n} S(n, k)$$

思考：用右边的方式计算斯特林数，再求和总的时间复杂度是多少？

https://oi-wiki.org/math/combinatorics/bell/

```go
package main

import "fmt"

// 有一个n个元素的集合 {1, 2, 3, 4 ...n} 现在需要把他划分成一些非空子集 总共有几种划分方式?
// example: n = 4
// { {1} {2} {3} {4} }
// { {1, 2} {3, 4} }
// { {1} {2, 3, 4} }
// { {2} {1, 3, 4} }
// ...

// {1 .. a} a 个元素，划分成 b 个部分，有多少种方案
func F(a int, b int) int {
    if a == 0 || b == 0 {
        return 0
    }
    if a == 1 && b == 1 {
        return 1
    }
    return F(a-1, b-1) + F(a-1, b)*b
}

func main() {
    a := 4
    for i := 1; i <= 4; i++ {
        fmt.Printf("F(4, %v): %v\n", i, F(a, i))
    }
}
```

```
F(4, 1): 1
F(4, 2): 7
F(4, 3): 6
F(4, 4): 1
```