

# 2023 年秋季学期算法基础期中考试（答案）

学号 \_\_\_\_\_ 姓名 \_\_\_\_\_

**主定理：** 令  $a \geq 1$  和  $b > 1$  是常数， $f(n)$  是一个函数， $T(n)$  是定义在非负整数上的递归式：

$$T(n) = aT(n/b) + f(n)$$

其中我们将  $n/b$  解释为  $\lfloor n/b \rfloor$  或  $\lceil n/b \rceil$ 。那么  $T(n)$  有如下渐进界：

1. 若对某个常数  $\varepsilon > 0$  有  $f(n) = O(n^{\log_b a - \varepsilon})$ ，则  $T(n) = \Theta(n^{\log_b a})$ 。
2. 若对整数  $k \geq 0$  有  $f(n) = \Theta(n^{\log_b a} \lg^k n)$ ，则  $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$ 。
3. 若对某个常数  $\varepsilon > 0$  有  $f(n) = \Omega(n^{\log_b a + \varepsilon})$ ，且对某个常数  $c < 1$  和所有足够大的  $n$  有  $af(n/b) \leq cf(n)$ ，则  $T(n) = \Theta(f(n))$ 。

**一、判断题**（根据表述判断正误，并简要说明理由；每题 5 分，共 30 分；每一题判断答案正确得 2 分，解释理由正确得 3 分。）。

1. (T, F) 递归式  $T(n) = 4T(n/4) + n \lg n$  的解满足  $T(n) = \Theta(n^2 \lg n)$ 。  
F 错误。  
由主定理可得， $f(n) = \Theta(n^{\log_4 4} \lg n)$ ，则  $T(n) = \Theta(n \lg^2 n)$ 。
2. (T, F) 堆排序和快速排序都是渐近最优的比较排序算法，他们在最坏情况下都需要做  $O(n \lg n)$  次比较。  
F 错误。  
快速排序在最坏情况下都需要做  $O(n^2)$  次比较。
3. (T, F) 选择排序的时间复杂度与输入数据的初始排列有关。  
F 错误。  
选择排序的时间复杂度是  $O(n^2)$ ，与输入数据的初始排列无关。无论输入数据是什么样的排列，选择排序都需要进行相同数量的比较和交换操作。
4. (T, F) 在使用 RANDOMIZED-SELECT 算法选择第  $k$  大的元素时，算法的期望运行时间是线性的。  
T 正确。  
RANDOMIZED-SELECT 算法是随机化的选择算法，其期望运行时间为线性。在每次划分中，算法通过随机选择一个主元素，然后将数组划分为两部分，从而实现了平均情况下的线性运行时间。通过随机性的引入，RANDOMIZED-SELECT 能够避免最坏情况下的概率，从而保证期望运行时间为  $O(n)$ 。
5. (T, F) 可以使用基于比较的算法，以  $O(n)$  的最坏时间复杂度将二叉堆转换为二叉搜索树。  
F  
如果这样的算法存在，那么对于一个无序数组，可以先用  $O(n)$  时间建堆，再调用该算法用  $O(n)$  时间转换为二叉搜索树，再用  $O(n)$  时间进行中序遍历，即得到排序后的数组，总用时为  $O(n)$ ，这与基于比较的排序算法的最坏时间复杂度下界是  $\Omega(n \lg n)$  矛盾。
6. (T, F) 如果把一棵红黑树除根节点和叶子节点外所有节点的颜色反转后，所得到的仍是合法的红黑树，那么任意一个节点的左右子树节点数相等。  
T  
由红黑树性质以及颜色反转后的性质，任意一条从根节点到叶子节点的路径上红色节点和黑色节点的数量均相等，因此所有叶子节点的高度都相同，这说明这棵树的形状是完美对称的，因此任意一个节点的左右子树节点数相等。

二、简答题（根据题目要求写出解答过程；每题 10 分，共 40 分）。

1. 对于递归式  $T(n) = T(2n/3) + 2^{\lceil \lg n \rceil}$ ，给出  $T(n)$  的一个渐近紧确界。

（注：  $2^{\lg n} = n$ ）

解：

定义  $F(n) = F(2n/3) + n, G(n) = G(2n/3) + 2n$ ，则由主定理显然有：  $F(n) = \Theta(n), G(n) = \Theta(n)$ ，而  $n = 2^{\lg n} \leq 2^{\lceil \lg n \rceil} \leq 2^{(\lg n)+1} = 2n$ ，因此  $F(n) \leq T(n) \leq G(n)$ ，故  $T(n) = \Theta(n)$

2. 给出算法 RANDOMIZED-QUICKSORT 的伪代码，并分析 RANDOMIZED-QUICKSORT( $A, 1, n$ ) 运行过程中，在最坏和最好情况下，RANDOMIZED-QUICKSORT 分别被调用了多少次？（以  $\Theta$  符号形式给出答案）

在最坏和最好情况下，RANDOMIZED-QUICKSORT 分别被调用了  $\Theta(n)$  次。

---

**Algorithm 1** PARTITION( $A, p, r$ )

---

```
1:  $x = A[r]$ 
2:  $i = p - 1$ 
3: for  $j = p$  to  $r - 1$  do
4:   if  $A[j] \leq x$  then
5:      $i = i + 1$ 
6:      $A[i] \leftrightarrow A[j]$ 
7:  $A[i + 1] \leftrightarrow A[r]$ 
8: Return  $i + 1$ 
```

---

---

**Algorithm 2** RANDOMIZED-PARTITION( $A, p, r$ )

---

```
1:  $i = \text{RANDOM}(p, r)$ 
2:  $A[r] \leftrightarrow A[i]$ 
3: Return PARTITION( $A, p, r$ )
```

---

---

**Algorithm 3** RANDOMIZED-QUICKSORT( $A, p, r$ )

---

```
1: if  $p < r$  then
2:    $q = \text{RANDOMIZED-PARTITION}(A, p, r)$ 
3:   RANDOMIZED-QUICKSORT( $A, p, q - 1$ )
4:   RANDOMIZED-QUICKSORT( $A, q + 1, r$ )
```

---

3. 给定  $n$  个已经排好序的元素  $a_0, \dots, a_{n-1}$ ，请给出时间复杂度为  $O(n)$  的算法来判断是否存在元素对满足  $a_i = -a_j, i \neq j$ 。给出伪代码，并说明算法工作原理。

解：

做向内搜索。基本思想是：如果  $a_{left} + a_{right} > 0$ ，那么  $right--$ ，否则如果  $a_{left} + a_{right} < 0$ ，那么  $left++$ 。

---

**Algorithm 4** 向内搜索

---

```
1:  $l = 0; r = n - 1$ 
2: while  $l < n$  do
3:   if  $a_l + a_r > 0$  then
4:      $n = n - 1$ 
5:   else
6:     if  $a_l + a_r < 0$  then
7:        $l = l + 1$ 
8:     else
9:       Return True
10: Return False
```

---

4. 对于图一所给的斐波那契堆, 黑色节点表示结点的 mark 属性为 true:
- 1) 抽取最小结点 (需要考虑 CONSOLIDATE 操作), 画出操作后最后的结果。
  - 2) 在 1) 的基础上, 把关键字 25 减为 3, 画出操作后的最后的结果 (需考虑级联操作)。

Given a Fibonacci heap below. the mark attribute of black node is true:

- 1) Please give the result of extracting the minimum node (CONSOLIDATE should be considered).
- 2) Please give the result of decrease the value of key 25 down to 3, based on the result of 1).

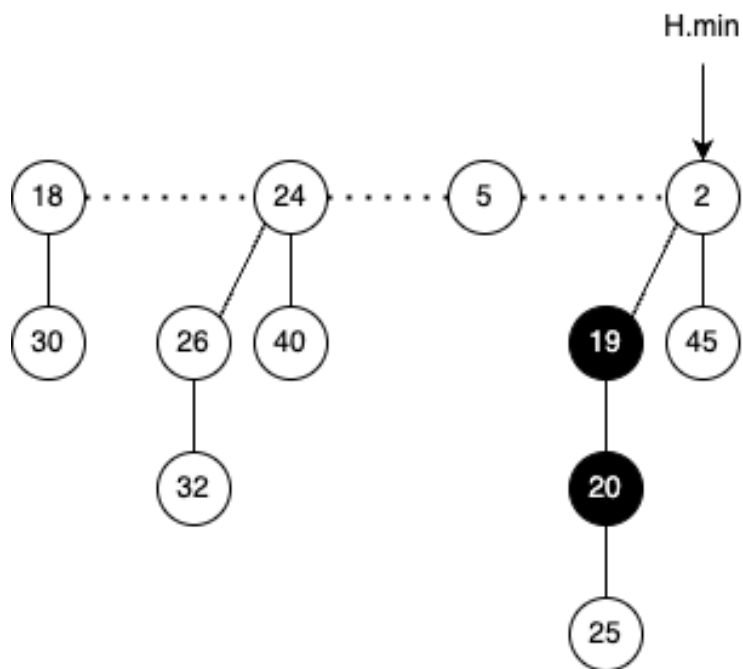
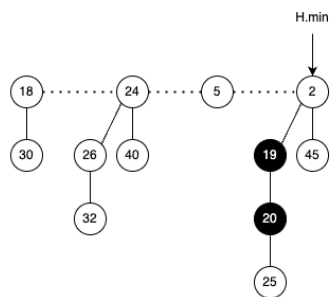


图 1: 斐波那契堆  
Fig. 1 Fibonacci Heap

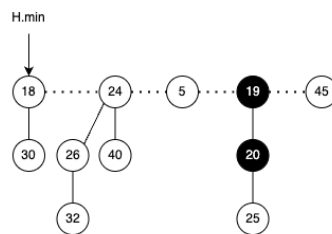
1)

本题度数为1的树的组合方式共有3种。所以有三种解法。其中19为根结点的解法，19和20均为marked，其余的做法中，仅20为marked结点。

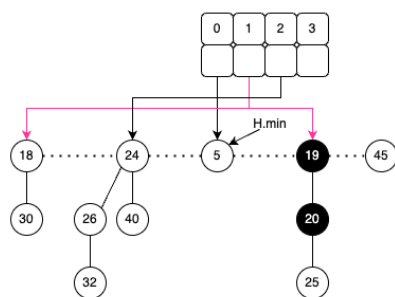
1)



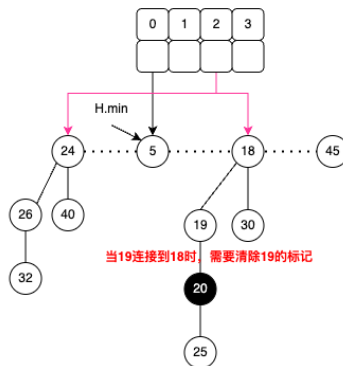
2)



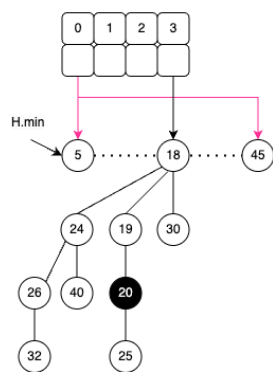
3)



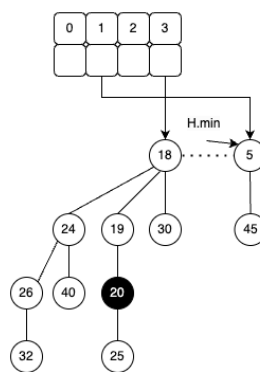
4)



5)

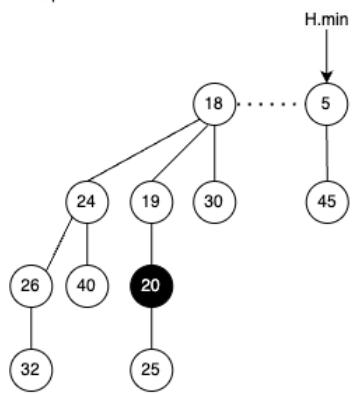


6)

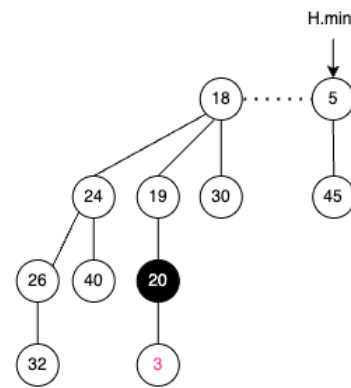


2)

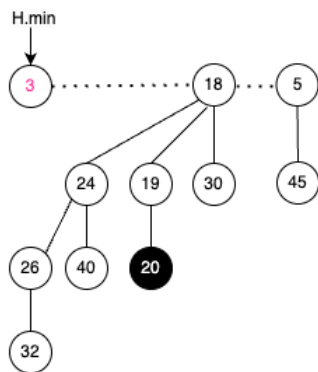
step 1



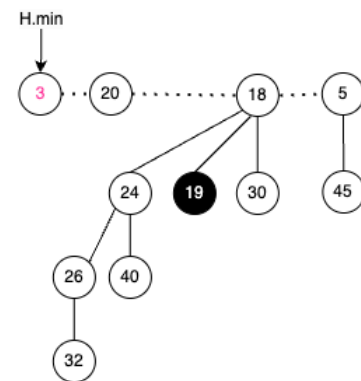
step 2



step 3



step 4



三、综合题 (根据题目要求写出解答过程; 每题 15 分, 共 30 分)。

1. 线性时间排序算法

(a) 说明如何以  $O(n)$  的最坏时间复杂度对  $[1, n^2]$  范围内的  $n$  个整数排序。假设整数已经在你需要的基上, 也就是说, 你不需要对它们进行进制转换。

(b) 对于常数  $m \in \mathbb{N}^+$ , 给定长度为  $n(n > m)$  的数列  $Z$ , 已知  $Z$  中每个元素都以这样的方式生成:  $Z_i = X_i + Y_i$ , 离散随机变量  $X_i$  在  $\{0, \frac{1}{m}, \frac{2}{m}, \dots, \frac{m-1}{m}\}$  中随机均匀取值, 连续随机变量  $Y_i$  在  $(0, \frac{1}{n})$  上随机均匀取值, 所有  $X_i, Y_i$  均独立。请设计一个桶排序算法, 以  $O(n)$  的期望时间复杂度对  $Z$  进行排序。你只需给出算法简要流程, 不需要对这一时间复杂度进行证明。

提示: 第二问中, 为达到  $O(n)$  的期望时间复杂度, 你需要设计  $\Theta(n)$  个桶, 并尽可能均匀地将  $Z$  中的元素分配到桶中。

解:

(a) 以  $n$  为基底使用基数排序。由于数字以  $n$  为基底, 因此位数的范围为 1 到  $n$  (准确地说是 0 到  $n-1$ ), 所以  $k = n$ 。由于  $n^2 = 100_n$ , 所以  $d = 3$ 。故基数排序的时间复杂度为  $\Theta(d \cdot n + d \cdot k) = \Theta(3n + 3n) = \Theta(n) \in O(n)$ 。

(b) 因为  $Y_i < \frac{1}{n} < \frac{1}{m}$ , 易验证, 对任意  $Z_i$ ,  $X_i = \lfloor mZ_i \rfloor$ , 因此可以用这种方式将  $Z_i$  对应的  $X_i, Y_i$  分解出来。建立  $m$  个“大桶”, 每个“大桶”里有  $n$  个“小桶”。对于每个  $Z_i$ , 根据  $X_i$  的值决定它进入哪个大桶, 再根据  $Y_i$  的值决定它进入哪个小桶。因为  $X_i$  和  $Y_i$  都是均匀的, 因此  $Z_i$  在期望意义上被均匀地分配到桶中, 从而具有线性时间复杂度。

具体而言, 对于桶  $B(x, y), x \in [0, m-1], y \in [0, n-1]$ , 其内放置的元素范围应为  $\left(\frac{x}{m} + \frac{y}{n^2}, \frac{x}{m} + \frac{y+1}{n^2}\right)$

2. 二叉搜索树一定程度上可以提高搜索效率，但在最坏情况下，仍需要  $O(n)$  时间完成一次搜索，因此需要引入平衡二叉搜索树。AVL 树是一种高度平衡的二叉搜索树：对于每个结点，其左子树与右子树的高度差不超过 1。为实现一棵 AVL 树，需要在每个节点内维护一个额外的属性： $height[x]$  为结点  $x$  的高度。此外，我们还定义平衡因子 (Balance Factor)： $BF[x]$  为结点  $x$  左右子树的高度差，取值为 0、1、-1，分别对应着左右子树等高，左子树比右子树高，右子树比左子树高。

(a) 已知一颗 AVL 树上的结点  $x'$ ， $BF[left[x']] = 0$ ， $BF[right[x']] = 1$ ，试问  $BF[x']$  的值。

$BF[x']$  的值可能是 0、1 或 -1。

(b) 证明：一棵有  $n$  个结点的 AVL 树高度为  $O(\lg n)$ 。

对于斐波那契数列有  $F(0) = 1$ ， $F(1) = 1$ ， $F(2) = 2$ ， $F(n) = F(n-1) + F(n-2)$ 。

设  $T(h)$  为高度为  $h$  的 AVL 树的最少结点数，则有  $T(1) \geq F(1)$ ， $T(2) \geq F(2)$ 。

又有  $T(h) \geq T(h-1) + T(h-2) + 1 \geq F(h-1) + F(h-2) + 1 > F(h)$ 。

又有  $1.6^h \leq F(h) \leq 2^h$ ，故有  $T(h) \geq 1.6^h$ 。所以，一棵有  $n$  个结点的 AVL 树高度为  $O(\lg n)$ 。

(c) 在一棵 AVL 树中插入一个结点，首先以二叉搜索树的顺序把该结点放在适当的位置上。此时，可能某些结点的平衡因子可能为 2 或 -2。请给出算法 AVL-BALANCE( $x$ ) 的伪代码，输入一棵以  $x$  为根的子树，其左右子树都是高度平衡的，若  $BF[x]$  为 2 或 -2，将其转变为高度平衡的，返回调整后的子树的根，否则直接返回  $x$ 。

---

**Algorithm 5** AVL-BALANCE( $x$ )

---

```

1: if  $|BF[x]| \leq 1$  then
2:   Return  $x$ 
3: else if  $BF[x] = 2$  then
4:    $y = left[x]$ 
5:   if  $height[left[y]] < height[right[y]]$  then
6:     LEFT-ROTATE( $y$ )
7:   Return RIGHT-ROTATE( $x$ )
8: else
9:    $y = right[x]$ 
10:  if  $height[left[y]] > height[right[y]]$  then
11:    RIGHT-ROTATE( $y$ )
12:  Return LEFT-ROTATE( $x$ )

```

---

#### 四、附加题（根据题目要求写出解答过程，共 10 分）。

为了在算法基础课程期中考试取得好成绩，小花同学在考前狂刷《算法导论》课后习题。在不刷任何题的情况下，小花能够考  $S$  分。《算法导论》中共有  $n$  道题，刷第  $i$  题能使她的期中考试成绩提升  $s_i$  ( $s_i \geq 0$ )。但与此同时，刷题会产生疲劳，进而影响考试发挥，如果小花一共刷了  $k$  道题，则她会因为疲劳而损失  $\alpha k$  分 ( $\alpha \geq 0$ )。小花不喜欢跳着刷题，也不喜欢倒着刷题，更不喜欢重复刷题，因此她只会从某一题开始刷连续的若干题。假设小花希望让自己的最终分数落入  $[l, r]$ ，请问她有多少种刷题方案？试设计一个具有  $O(n \lg n)$  最坏时间复杂度的算法解决该问题。（在小花所处的平行世界中，考试的分数未必是  $0 \sim 100$  的整数，因此你不能使用计数排序等对数据范围有限制的方法）

解：令  $D_k = \sum_{i=1}^k s_i - \alpha k$ ，当小花刷题的范围是  $(x, y]$  时，她的最终得分为  $S + D_y - D_x$ 。因此问题可转化为：满足  $x < y$  且  $D_y - D_x \in [l - S, r - S]$  的数对  $\langle x, y \rangle$  的个数。与逆序对数量类似，这可以在归并排序的过程中顺便统计，或者通过维护一棵红黑树进行统计。详见[区间和的个数](#)