

2023 年秋季学期算法基础期中考试

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 $T(n) = \Theta(n^2 \lg n)$ 。
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。
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。

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。
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。
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。

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: $2^{\lg n} = n$)

2. 补全算法 RANDOMIZED-QUICKSORT(A 为数组)的伪代码, 并分析 RANDOMIZED-QUICKSORT($A, 1, n$) ($n = A.length$) 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 Θ 符号形式给出答案)

Algorithm 1 PARTITION(A, p, r)

```

1:  $x = A[r]$ 
2:  $i = \underline{\hspace{1cm}}$ 
3: for  $j = p$  to  $\underline{\hspace{1cm}}$  do
4:   if  $\underline{\hspace{1cm}}$  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  $\underline{\hspace{1cm}}$  then
2:    $q = \text{RANDOMIZED-PARTITION}(A, p, r)$ 
3:   RANDOMIZED-QUICKSORT( $A, p, \underline{\hspace{1cm}}$ )
4:   RANDOMIZED-QUICKSORT( $A, \underline{\hspace{1cm}}, r$ )

```

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, 并说明算法的工作原理。

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).

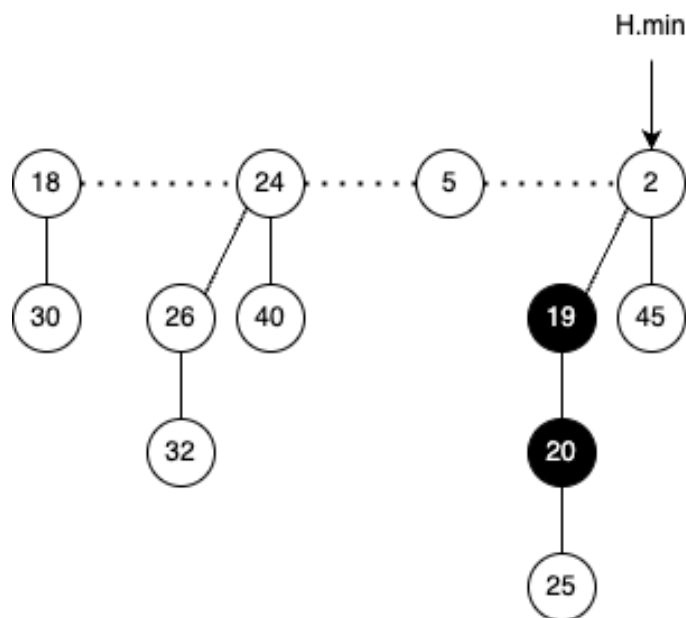


图 1: 斐波那契堆
Fig. 1 Fibonacci Heap

三、综合题（根据题目要求写出解答过程；每题 15 分，共 30 分）。

1. (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。

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。

(b) 证明：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。请补全出算法 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 4 AVL-BALANCE(x)

```

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

```

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中取得好成绩，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 αk 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）