任选6题

Problem 3.1
请设计一个高效的算法在n个元素中找到第三大的元素，并分析最坏情况下你的算法所需要的比较次数是多少。你的算法在执行过程中是否必须先找到最大的和次大的元素？

Problem 3.2
请设计一个算法，使得在最坏情况下，它能够只利用6次比较来找出5个元素的中位数。描述该算法的步骤，并利用决策树的形式展示你的算法。

Problem 3.3
假设对一个含有n个元素的集合，某算法只用比较来确定阶为i的元素。证明：无需另外的比较操作，它也能找到比阶为i的元素小的$i-1$个元素和比该元素大的$n-i$个元素。

Problem 3.4
假设已有一个用于选择中位数的“黑盒”算法$\mathcal{A}$，它在最坏情况下需要线性运行时间。请给出基于已有的黑盒算法$\mathcal{A}$，选择阶为任意k的元素的算法$\mathcal{B}$，要求算法$\mathcal{B}$最坏情况下也是线性时间的。

Problem 3.5 (最大的k个元素(必选))
给定有n个数的集合，现要求找出其中的前k大的k个数(得出的这k个数要求是排好序的，即知道哪个是第1 大，第2大，$\cdots$, 第k大)，请设计三个基于比较的算法，使得算法的最坏情况时间复杂度分别符合下面三种要求：

1. $O(n\log n)$
2. $O(n+k\log n)$
3. $O(n+k\log k)$

Problem 3.6 (最接近中位数的数(必选))
给定一个有n个不同整数的集合S，用M表示S的中值。请设计算法找出S中和M的大小最接近的k 个数（k远小于n）。例如，集合$S=\{6,7,50,800,900\}$，中值M是50，两个（$k=2$）和中值M最接近的数是6和7。

1. 请设计一个时间复杂度为$O(n\log n+k)$的算法；
2. 请设计一个时间复杂度为$O(n+k\log k)$的算法。

Problem 3.7 (联合中位数(必选))

1. 给定两个有序数组$A[1...n], B[1...n]$和一个整数k，请设计一个算法用$O(\log n)$的时间找到$A\cup B$中阶为k的元素(你可以假设数组中没有重复元素)；
2. 给定三个有序数组$A[1...n]$、$B[1...n]$、$C[1...n]$和一个整数k，请设计一个算法用$O(\log n)$的时间找到$A\cup B\cup C$中阶为k的元素；
3. 给定二维数组$A[1...m][1...n]$，数组的每一行都是有序的，并给定一个整数k。请设计一个算法找到数组中阶为k的元素，并分析算法的时间复杂度。

Problem 3.8 (动态发现中值)

请设计一个数据结构支持对数时间的插入操作、常数时间发现中值和对数时间删除中值(提示：利用1个最小堆和1个最大堆)。

Problem 3.9 (加权中位数(必选))

现有n个各不相同的数$x_1, x_2, \cdots, x_n$，每个数都有正权重$w_1, w_2, \cdots, w_n$，满足$\sum_{1\leq i\leq n} w_i = 1$。我们定义一个元素$x_k$为“加权中位数”，如果它满足：

$$\sum_{x_i < x_k} w_i < \frac{1}{2}, \sum_{x_i > x_k} w_i \leq \frac{1}{2}$$

1. 证明当$w_i = \frac{1}{n}$（$i = 1, 2, \cdots, n$）时，$x_1, x_2, \cdots, x_n$的中位数即为加权中位数；
2. 请设计一个基于元素的排序实现加权中位数选择的算法，要求最坏情况时间复杂度为$O(n \log n)$；
3. 请设计一个最坏情况时间复杂度为$\Theta(n)$的加权中位数选择算法。

任选9题

Problem 4.1

如果数组中的第1个元素到第i个元素值是单调递增的，而从第i个元素到最后一个元素值是单调递减的，我们称其为“单峰”数组。现在给定一个大小为n的“单峰”数组E，请设计复杂度为$O(logn)$ 的找出E 中最大元素的算法。

Problem 4.2

给定一个有n个互不相同的有序整数$\{a_1, a_2, \cdots, a_n\}$的序列。请设计算法判断是否存在某个下标$i$满足$a_i = i$。例如，$\{-10, -3, 3, 5, 7\}$，有$a_3 = 3$；在$\{2, 3, 4, 5, 6, 7\}$ 不存在满足条件的i。

Problem 4.3

数组$A[1..n]$存放的是n个有序的互不相同的正整数。请设计算法找出A中缺少的最小的正整数。例如，$A = [1, 2, 4, 5]$，缺少的最小的正整数是3。

Problem 4.4 (不同哈希表的存储效率(必选))

哈希表H在闭哈希(closed-address hashing)方法下是一个链表头组成的数组，而在开哈希(open-address hashing)方法下是一个关键字的数组。假设一个关键字需要一个单位的存储，而一个链表节点需要两个单位的存储，其中一个存放关键字，另一个存放链表节点的指针。考虑在闭散列下如下的负载因子：0.25,0.5,1.0,2.0。设h_C是使用闭哈希时哈希表的大小(即链表头数组初始时有多少空位)。

1. 请计算闭哈希表的空间消耗。假设同样的空间用于开散列表，它的负载因子会是多少？
2. 假设一个关键字要占据4个单位的存储，而一个表节点需要5个单位的存储(4个用于关键字，一个用于指针)，请再次计算上一问中的问题。

Problem 4.5

对下面每个问题给出符合要求的算法，可以在你的算法中使用已有的排序算法。

(a) S是由n个整数组成的数组，并且未排序。请给出算法用来找到整数对$x, y \in S$并且$|x - y|$ 最大，最坏情况下，你的算法的复杂度为$O(n)$。

(b) S是由n个整数组成的有序数组。请给出算法用来找到整数对$x, y \in S$并且$|x - y|$最大，最坏情况下，你的算法的复杂度为$O(1)$。

(c) S是由n个整数组成的数组，并且未排序。请给出算法用来找到整数对$x, y \in S$并且$|x - y|(x \neq y)$最小，最坏情况下，你的算法的复杂度为$O(nlogn)$。

(d) S是由n个整数组成的有序数组。请给出算法用来找到整数对$x, y \in S$并且$|x - y|(x \neq y)$ 最小，最坏情况下，你的算法的复杂度为$O(n)$。

Problem 4.6

令M是一个$m \times n$的矩阵，矩阵中的每一行元素从左到右按照升序排列，每一列的元素从上到下按照升序排列。请给出高效的算法在矩阵中查找给定的元素x（x可能不存在），并给出最坏情况下所需要的比较次数。

Problem 4.7 (寻找给定和的两个元素(必选))
给定由n个实数组成的集合S和一个实数值a。请给出算法判定在S中是否存在两个元素的和为a。

(a) 假设S是未排序的。给出复杂度为$O(nlogn)$的算法。

(b) 假设S是排完序的。给出复杂度为$O(n)$的算法。

Problem 4.8 (红黑树的平衡性(必选))
任意的一棵RB_h或者ARB_h的black height是良定义的，为h。现在另T为一棵RB_h，A为一棵ARB_h，请证明下列命题：

(a) T至少有2^h-1个内部黑色节点。

(b) T至多有4^h-1个内部节点。

(c) 任意黑色节点的深度至多是它的black depth的两倍。

(d) A至少有2^h-2个内部黑色节点。

(e) A至多有$(4^h)/2-1$个内部节点。

(f) 任意黑色节点的深度至多是它的black depth的两倍。

Problem 4.9
采用闭地址散列的方式避免哈希表的冲突问题。现在假设有k个关键字需要插入该哈希表中（有n个槽），每个关键字等可能地插入每个槽中。求恰好有k 个关键字插入某个特定槽的概率。

Problem 4.10
以下是程序自动分析中的一个问题。对于一组变量$x_1,\cdots,x_n$，给定一些形如“$x_i=x_j$”的等式约束和形如“$x_i\neq x_j$”的不等式约束。这些约束是否能同时满足？

例如，如下一组约束：

$$x_1=x_2,x_2=x_3,x_3=x_4,x_1\neq x_4$$

是无法同时满足的。请给出一个有效的算法，判断关于n个变量的m个约束是否可同时满足。

Problem 4.11
假设有一个数组$A[1...n]$，它满足以下两个条件：$A[1]\geq A[2]$；$A[n-1]\leq A[n]$。当元素$A[x]$ 并不大于它的两个邻居（$A[x-1]\geq A[x],A[x+1]\geq A[x]$）时，我们称$A[x]$为局部最小。例如，在下面的数组中有六个局部最小元素。

9	7	7	2	1	3	7	5	4	7	3	3	4	8	6	9

(a) 我们可以用$O(n)$的时间通过扫描一遍数组找到一个局部最小元素。请设计一个用$O(logn)$的时间找到一个局部最小元素的算法。

(b) 在给定的边界条件下，请说明至少存在一个局部最小元素。

Problem 4.12 (平方根：计算还是寻找？(必选))
假设你只可以使用加法和移位操作。请给出一个高效的算法计算$\lceil\sqrt{N}\rceil$。

1. 给定一个自然数N有n个比特，请给出复杂度为$O(n^2)$的算法计算$\lceil\sqrt{N}\rceil$。
2. （选做）请将你的算法改进到$o(n^2)$。

Problem 4.13 (平摊分析(必选))
现在有一个由数组组成的集合，数组i的大小是$2i$。每个数组不是空（没有元素）就是满的（填满元素）。例如, 11 个元素存储在下面的数组里：

$$\begin{aligned} A0 &: [a_1] \\ A1 &: [a_2, a_3] \\ A2 &: empty \\ A3 &: [a_4, a_5, \cdots, a_{11}] \end{aligned}$$

现在插入一个新的元素，称作a_{12}。首先创建一个新的大小为1的数组存放a_{12}，现在我们查看$A0$ 是否为空，如果$A0$为空，那么就另这个新数组成为$A0$；如果不为空（如上面的例子），就将这个新数组和$A0$合并为一个新的数组（在上面的例子中$A0$就变为$[a_1, a_{12}]$），并且再继续查看$A1$是否为空，如果$A1$为空, 就另这个新的$A0$成为$A1$；如果不为空，则将其和$A1$合并，并且继续查看$A2$，反复执行上述操作。所以，在上面的例子中插入a_{12}，我们最终得到的新的数组的集合是：

$$\begin{aligned} A0 &: empty \\ A1 &: empty \\ A2 &: [a_1, a_2, a_3, a_{12}] \\ A3 &: [a_4, a_5, \cdots, a_{11}] \end{aligned}$$

现在我们另创建一个新的大小为1的数组的开销是1，合并两个大小分别为m的数组的开销为$2m$，所以上面的例子所需要的总开销是1+2+4。请用平摊分析的方法分析插入操作的复杂度。

Problem 4.14
使用两个后进先出的栈可以实现一个先进先出的队列。假设有三个操作，push（压栈）、pop（出栈）和empty（判断是否为空），每个操作的代价都是1。队列的两个操作可以按照如下方式实现：

- Enqueue(x)入队:将x元素压入栈1。如果栈1满，那么返回错误。
- Dequeue()出队:如果栈2为空，将栈1的元素依次出栈并压入栈2。然后，将栈2的元素出栈，并返回结果。如果栈2与栈1均为空，返回错误。

请回答下面两个问题：

(a) 解释这个方法的正确性。

(b) 利用平摊分析法分析该算法的时间复杂度。