算分作业 2

24.03.04

2.7

(1) 算法的 Python 实现如下:

```python
def solve(a):
    a.sort()
    n = len(a)
    for i in range(n - 1 - n // 2):
        if a[i] == a[i + n // 2]:
            return True
    return False
```

(2) 注意到主元素一定是中位数, 因此先找出中位数, 再测试它的出现次数是否超过 $n/2$ 即可. 我们知道求中位数是 $O(n)$ 的, 因此整个算法是 $O(n)$ 的.

(3) 先考虑 n 为偶数的情况. 将元素分成两组进行淘汰, 即两两比较, 若相等, 保留一个, 若不等, 全部丢弃. 容易证明, 若原数组含有主元素, 那么经过一轮淘汰后它仍是主元素, 但反之不成立. 继续这样做下去, 若最终不剩下元素, 则原数组没有主元素; 若最终剩下一个元素, 则只有它可能是原数组的主元素, 计算它在原数组中的出现次数即可. 当 n 为奇数时, 会有一个元素轮空, 先计算该元素在数组中的出现次数, 若超过一半, 则只有该元素可能是原数组的主元素, 在原数组中计算它的出现次数, 然后算法结束; 否则, 该元素一定不是原数组的主元素, 将它丢弃, 继续淘汰即可. 这个算法满足 $T(n) \le T(n/2) + O(n)$, 即 $T(n) = O(n)$.

2.12

建立一张哈希表并插入 A 的所有元素, 这一步是平均 $\Theta(n)$ 的. 然后对 B 的每个元素, 检查它是否存在于哈希表, 若存在就加入 C, 否则不加入, 这一步是平均 $\Theta(m)$ 的. 因此总时间是平均 $\Theta(n+m) = \Theta(n)$ 的.

算法伪代码如下:

```
输入: A[1..n] 和 B[1..m]
输出: C = A ∩ B
H = 空哈希表
C = 空集合
for i in A do H.insert(i)
for i in B do
    if H.contains(i) then
        C.append(i)
return C
```

2.15

(1) 算法 A 最坏是 $\Theta(n\sqrt{n})$ 的, 算法 B 最坏是 $\Theta(n \log n)$ 的.

(2) 先找到数组中第 i 大的数 a, 再遍历数组找到所有不小于 a 的数, 最后对这些数排序并输出即可. 这个算法的每一步都是 $O(n)$ 的, 因此总时间也是最坏 $O(n)$ 的. 算法伪代码如下:

```
输入: S[1..n], i
输出: S 中前 i 大的数, 从大到小排列
a = Select(S, n-i+1)
A = 空数组
for s in S do
    if s >= a then
        A.append(s)
A.sort(reverse=true)
return A
```

2.16

(1) 显然将 x, y 取为数组中的最大和最小值即可, 这个算法是 $O(n)$ 的.

(2) 先对数组排序, 然后计算所有相邻元素的差值, 取出差值最小的那一对元素作为 x, y 即可, 这个算法是 $O(n \log n)$ 的.

2.18

先求出所有点到原点的距离, 与点的编号一起记录, 然后找到其中距离第 $\lfloor \sqrt{n} \rfloor$ 小的元素, 再遍历数组取出所有距离不超过它的元素, 最后排序并输出即可. 每一步都是 $O(n)$ 的, 因此总时间是 $O(n)$ 的. 伪代码与 2.15 基本相同, 就不再次给出了.

2.23

先找出数组中第 $\lfloor n/4 \rfloor$ 小和第 $\lfloor n/4 \rfloor$ 大的元素, 分别记为 a, b. 数组中处于 (a, b) 的元素即为近似中值, 遍历数组检查是否存在即可. 每一步都是 $O(n)$ 的, 因此总时间是 $O(n)$ 的.

2.27

注意到仓库的横纵坐标对总路程的影响是独立的, 分别将其达到最小即可. 显然, 将仓库的横坐标分别取为所有商店横坐标的一个中位数时, 横向总路程最短, 纵坐标同理. 求中位数是 $O(n)$ 的, 因此这个算法是 $O(n)$ 的.

补充题 1

设这 n 个数按顺序为 $a_1, \cdots, a_n$.

注意到, 若 $i < j < k$, $a_j < a_i, a_k$, 则在 (i, k) 必定存在极小值点.

不妨设 $a_1 < a_n$.

若 $a_1 < a_2$, 则 a_1 就是极小值, 算法结束. 否则, $a_1 > a_2$. 记 $m = \lfloor \frac{n+1}{2} \rfloor$, 考虑 a_m.

若 $a_m > a_2$, 则在 $(1, m)$ 必定存在极小值点. 否则, $a_m < a_2 < a_1 < a_n$.

若 $a_m > a_{m-1}$, 则在 $(1, m)$ 必定存在极小值点. 若 $a_m > a_{m+1}$, 则在 (m, n) 必定存在极小值点. 若都不成立, 则 a_m 就是极小值, 算法结束.

像这样, 经过常数次比较可以把区间长度缩小一半, 因此整个算法是 $O(\log n)$ 的.

补充题 2

算法伪代码如下:

```
Test(a[1..n])
输入: n 片芯片的数组 a, 其中好芯片不少于坏芯片
输出: 好芯片或 -1 (代表未找到)
if n == 0 return -1
if n 为奇数
    从 a 中任取一片, 记为 t
    res = Test(去掉 t 的 a)
    return res == -1 ? t : res
else
    将芯片两两分组测试, 若结果为两好则任丢弃一片, 否则全部丢弃
    return Test(剩下的芯片)
```

改进算法的正确性其实是显然的, 要点如下:

- 当 n 为偶数时, 算法与原算法相同, 子问题满足前提条件 (好芯片不少于坏芯片). 当 n 为奇数时, 必有好芯片严格多于坏芯片, 任去掉一片后仍满足前提条件.
- 容易证明, 在该算法下, 当子函数返回 -1 时, 子函数的输入必满足好坏芯片数相等, 因此最近的因 n 为奇数去掉的那片芯片一定是好芯片.