排序

1.1 课本 2.3-1 使用图 2-4 作为模型，说明归并排序在数组 A=<3，41，52，26，38，57，9，49>上的操作。

（不好画图）

<table>
<tr><td colspan="8">3 9 26 38 41 49 52 57</td></tr>
<tr><td colspan="4">3 26 41 52</td><td colspan="4">9 38 49 57</td></tr>
<tr><td colspan="2">3 41</td><td colspan="2">26 52</td><td colspan="2">38 57</td><td colspan="2">9 49</td></tr>
<tr><td>3</td><td>41</td><td>52</td><td>26</td><td>38</td><td>57</td><td>9</td><td>49</td></tr>
</table>

1.2 课本 7.1-1 参照图 7-1 的方法，说明 PARTITION 在数组 A=<13，19，9，5，12，8，7，4，21，2，6，11>上的操作过程。

加粗的与斜体的表示参照图 7-1 的阴影区域。红色的是 x。

斜体的为浅阴影($\le x$)，粗的为深阴影($> x$)

13	**19**	9	5	12	8	7	4	21	2	6	11
9	**13**	**19**	5	12	8	7	4	21	2	6	11
9	*5*	**13**	**19**	12	8	7	4	21	2	6	11
9	*5*	**13**	**19**	**12**	8	7	4	21	2	6	11
9	*5*	*8*	**13**	**19**	**12**	7	4	21	2	6	11
9	*5*	*8*	*7*	**13**	**19**	**12**	4	21	2	6	11
9	*5*	*8*	*7*	*4*	**13**	**19**	**12**	21	2	6	11
9	*5*	*8*	*7*	*4*	**13**	**19**	**12**	**21**	2	6	11
9	*5*	*8*	*7*	*4*	*2*	**13**	**19**	**12**	**21**	6	11
9	*5*	*8*	*7*	*4*	*2*	*6*	**13**	**19**	**12**	**21**	11
9	*5*	*8*	*7*	*4*	*2*	*6*	11	**13**	**19**	**12**	**21**

1.3 课本 8.2-1 参照图 8-2 的方法，说明 COUNTING-SORT 在数组 A=<6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2>上的操作过程。

原先：

a	6	0	2	0	1	3	4	6	1	3	2

c	0	0	0	0	0	0	0

计数并累加：

a	6	0	2	0	1	3	4	6	1	3	2

c	2	4	6	8	9	9	11

填入：

1.

b	x	x	x	x	x	2	x	x	x	x	x

c	2	4	5	8	9	9	11

2.

b	x	x	x	x	x	2	x	x	3	x	x

c	2	4	5	7	9	9	11

3.

b	x	x	x	1	x	2	x	x	3	x	x

c	2	3	5	7	9	9	11

…

最后：

b	0	0	1	1	2	2	3	3	4	6	6

1.4 课本 8.3-1 参照图 8-3 的方法，说明 RADIX-SORT 在下列英文单词上的操作过程：COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX.

第一次：

S	E	A
T	E	A
M	O	B
T	A	B
D	O	G
R	U	G
D	I	G
B	I	G
B	A	R
E	A	R
T	A	R
C	O	W
R	O	W
N	O	W
B	O	X
F	O	X

第二次：

T	A	B
B	A	R
E	A	R
T	A	R
S	E	A
T	E	A
D	I	G
B	I	G
M	O	B
D	O	G
C	O	W
R	O	W
N	O	W
B	O	X
F	O	X
R	U	G

第三次：

B	A	R
B	I	G
B	O	X
C	O	W
D	I	G
D	O	G
E	A	R
F	O	X
M	O	B
N	O	W
R	O	W
R	U	G
S	E	A
T	A	B
T	A	R
T	E	A

1.5 课本 8.4-1 参照图 8-4 的方法，说明 BUCKET-SORT 在数组 A=<0.79，0.13，0.16，0.64，0.39，0.20，0.89，0.53，0.71，0.42>上的操作过程。

10 个桶，每个桶范围为 0.1

```
[
[]
[ 0.13, 0.16 ]
```

```
[ 0.2 ]
[ 0.39 ]
[ 0.42 ]
[ 0.53 ]
[ 0.64 ]
[ 0.71, 0.79 ]
[ 0.89 ]
[]
]
```

1.6 假设对于 n 个不同的元素$x_1, x_2 ... x_n$有正加权值$w_1, w_2 ... w_n$，有$\sum(w_i) = 1$，我们定义加权中位数x_k为满足以下条件的元素：

$\sum_{x_i<x_k} w_i < \frac{1}{2}$ 且 $\sum_{x_i>x_k} w_i \geq \frac{1}{2}$

1.6.1 (a) 证明，当$w_i = \frac{1}{n}$时，$x_1, x_2 ... x_n$的中位数是加权中位数。

中位数的左边和右边的数字个数 $\leq \lfloor \frac{n}{2} \rfloor$。因此每一边的数字的权和 $\leq \frac{1}{2}$

1.6.2 (b)设计一种基于排序的算法求解加权中位数，要求其最差算法复杂度小于$O(n \lg n)$。证明设计算法的时间复杂度

首先对元素按照其值进行排序，然后计算加权前缀和，直到找到第一个加权前缀和大于等于 1/2 的元素。这个元素就是加权中位数。

该算法的时间复杂度分为排序和计算加权前缀和两个部分。排序的时间复杂度为$O(n \log n)$，而计算加权前缀和的时间复杂度为$O(n)$。因此，总体时间复杂度为 $O(n \log n)$

1.6.3 (c) 考虑一维快递中心选址问题。我们给出 n 个不同点$p_1, p_2 ... p_n$（即 n 个数值），分别拥有权重$w_1, w_2 ... w_n$，我们需要找到一个点 p（p 可以是任意一个点，不一定是 n 个给定点中的一个），要求最小化$\sum_{i=1}^{n} w_i d(p, p_i)$，这里 d 是距离，定义为$p - p_i$。证明， 序列的加权中位数即是所求的 p 点。（提示：p 这里是连续变量，其导数为 0 时，有极值）

$$f(p) = \sum_{i=1}^{n} w_i \left| p - p_i \right|$$

$$f'(p) = \sum_{i=1}^{n} w_i \operatorname{sgn}(p - p_i) = \sum_{x_i<p} w_i - \sum_{x_i>p} w_i$$

当 p 为加权中位数时，$\sum_{x_i<p} w_i$ 与 $\sum_{x_i>p} w_i$ 最接近，得 $f'(p)$ 最小，即为 $f(p)$ 的极值。

1.6.4 (d) 进一步的，我们考虑二维的快递中心选址问题，此时$p_i = (x_i, y_i)$，采用曼哈顿距离$d(p, p_i) = |x - x_i| + |y - y_i|$。给出一种算法复杂度为$O(n \lg n)$的算法来解决该问题，写出相应的伪代码。

sort-3.py

```python
from dataclasses import dataclass


@dataclass
class point:
    x: int
    y: int
    w: float
```

```python
    def manhattan(self, other) -> int:
        return abs(self.x - other.x) + abs(self.y - other.y)


def calc(s: list[point]):
    s.sort(key=lambda x: point(0, 0, 0).manhattan(x))
    accu = 0
    index = 0
    while index < len(s):
        accu += s[index].w
        if accu >= 0.5:
            break
        index += 1

    print(s[index])


calc(
    [
        point(1, 1, 0.2),
        point(1, 2, 0.3),
        point(1, 3, 0.05),
        point(2, 4, 0.15),
        point(3, 5, 0.3),
    ]
)

# point(x=1, y=2, w=0.3)
```

1.7 （不做要求）课本第 **2** 章思考题：**2-1**。这道思考题实际上就是 **Timsort** 的介绍。请大家思考这个问题，完成习题部分，以及给出伪代码（或是程序）。

1.8 （**leetcode** 题目）**leetcode** 题库 **88**、**21**、**23**、**147**、**148**、**41**、**75**、**34**、**74**。

https://github.com/lxl66566/OJ/tree/main/leetcode_cn