1 基本算法

1.1 尺取法 (双指针)

用以解决序列的区间问题, 一般有两个要求:

1. **序列是有序的, 需要先对序列进行排序**

2. 问题与序列的区间有关, 操作两个或多个指针 i, j 表示区间

在 Python 中, 用 while 实现较为方便

扫描方向:

反向扫描, 左右指针: i, j 的方向相反;

同向扫描, 快慢指针: i, j 的方向相同, 但是扫描速度一般不同, 可以形成一个大小可变的滑动窗口

1.1.1 反向扫描

找指定和的整数对

问题: 输入 n ($n <= 100000$) 个整数, 放在数组 a[] 中. 找出其中的两个数, 它们之和等于整数 m. (假定肯定有解).

输入:

第 1 行是数组 a[], 第 2 行是 m

Sample input

21 4 5 6 13 65 32 9 23

28

Sample output

5 23

```
# 哈希 复杂度为O(n), 但是需要较大的哈希空间
a = list(map(int, input().split()))
m = int(input())
s = set(a)
outed = set()
for item in s:
  if m - item in s and item not in outed:
    print(item, m - item)
    outed.add(m-item)
```

```
# 尺取法 复杂度为O(n log_2~n), 其中, 排序的复杂度为O(log_2~n), 检查的复杂度为O(n)
a = list(map(int, input().split()))
a.sort()
m = int(input())

# 双指针
i, j = 0, len(a) - 1
while (i < j):
  s = a[i] + a[j]
  # s < m: i增加1, 之后的s>=当前s
  if s < m:
    i += 1
  elif s > m:
    j -= 1
  else:
    print("{} {}".format(a[i], a[j]))
    i += 1
```

判断回文串

输入: 第 1 行输入测试实例个数, 之后每行输入一个字符串

输出: 是回文串输出 yes, 不是输出 no

```
n = int(input())
for i in range(n):
  s = str(input())
  i, j = 0, len(s) - 1
  while i < j:
    flag = False
    if s[i] == s[j]:
      flag = True
    else:
      flag = False
      break
    i += 1
    j -= 1
  if (flag):
    print('yes')
  else:
    print('no')
```

1.1.2 同向扫描

使用尺取法产生滑动窗口

寻找区间和

给定一个长度为 n 的正整数数组 a[] 和一个数 s, 在数组中找一个区间, 使得该区间的数组元素之和等于 s. 输出区间的起点和终点位置

第 1 行输入数组长度 n, 第 2 行输入数组, 第 3 行为 s

Sample input

15

6 1 2 3 4 6 4 2 8 9 10 11 12 13 14

6

Sample output

0 0

1 3

5 5

6 7

初始值 i = j = 0

如果 sum = s: 输出一个解, sum 减去 a[i], i++

如果 sum < s: j++, sum + a[j]

如果 sum > s: sum - a[i], i++

```python
n = int(input())
a = list(map(int, input().split()))
s = int(input())

sum = a[0]
i, j = 0, 0
while i < n and j < n:
    if sum == s:
        print("{} {}".format(i, j))
        sum -= a[i]
        i += 1
        j += 1
        sum += a[j]
    elif sum < s:
        j += 1
        sum += a[j]
    elif sum > s:
        sum -= a[i]
        i += 1
```

数组去重

给出一个数组, 输出去除重复元素之后的数组

```python
# 哈希, 数据多或者数值过大时需要占用大量的空间
a = list(map(int, input().split()))
s = set(a)
unique_a = list(s)
print(unique_a)
```

```python
# 尺取法
a = list(map(int, input().split()))
# a排序, 使得相同元素排列在一起
a.sort()
n = len(a)
# 双指针均从0开始
i, j = 0, 0
# j始终指向无重复元素部分的最后一个元素
while i < n and j < n:
    # 若i和j指向的元素不同, j++, 将i指向的元素复制到j上
    # EX: 1 2(j) 3(i) 3
    # -> 1 2 3(j, i) 3
    # EX: 1 2 3(j) 3 4(i)
    # -> 1 2 3 3(j) 4(i)
    # -> 1 2 3 4(j) 4(i)
    if a[i] != a[j]:
        j += 1
        a[j] = a[i]
    i += 1
unique_a = a[0:j+1]
print(unique_a)
```

找相同数对

给出一串数字和一个数字 C, 要求计算出所有 A - B = C 的数对的个数 (不同位置的数字一样的数对算不同的数对)

输入: 2 行, 第 1 行输入整数 n 和 C, 第 2 行输入 n 个整数

输出: 满足 A - B = C 的数对的个数

Sample Input

6 3

8 4 5 7 7 4

Sample Output

5

```python
n, c = map(int, input().split())
a = list(map(int, input().split()))
a.sort()

i, j, k = 0, 0, 0
ans = 0

for i in range(n):
    # j, k指向相同元素区间的起点和终点后1个元素
    # 寻找的对象是区间内的元素 - a[i] = C
    while j < n - 1 and a[j] - a[i] < c:
        j += 1
    while k < n and a[k] - a[i] <= c:
        k += 1
    if a[j] - a[i] == c and a[k-1] - a[i] == c and k - 1 >= 0:
        ans += k - j
print(ans)
```