哈希表

字典具有相同的键和值，即使键的顺序不同，也将返回True

```python
dict1 = {'a': 1, 'b': 2, 'c': 3}
dict2 = {'b': 2, 'c': 3, 'a': 1}

print(dict1 == dict2)  # True
```

有效的字母异位词

```python
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        if len(s)!=len(t):return False
        def build(s):
            mp={}
            for c in s:
                mp[c]=mp.get(c,0)+1
            return mp

        s = build(s)
        t = build(t)

        return s==t

# 法二
 class Solution(object):
    def isAnagram(self, s: str, t: str) -> bool:
        from collections import Counter
        s = Counter(s)
        t = Counter(t)
        return s==t
```

字母异位词分组

输入： strs = `["eat", "tea", "tan", "ate", "nat", "bat"]`
输出： [["bat"],["nat","tan"],["ate","eat","tea"]]

```python
from collections import defaultdict

class Solution:
    def groupAnagrams(self, strs: List[str]) -> List[List[str]]:
        mp = collections.defaultdict(list)

        for st in strs:
            key = "".join(sorted(st))
            mp[key].append(st)

        return list(mp.values())

#法二
class Solution:
    def groupAnagrams(self, strs: List[str]) -> List[List[str]]:
        mp = collections.defaultdict(list)

        for st in strs:
            counts = [0] * 26
            for ch in st:
                counts[ord(ch) - ord("a")] += 1
            # 需要将 list 转换成 tuple 才能进行哈希
            mp[tuple(counts)].append(st)

        return list(mp.values())
```

找到字符串中所有字母异位词

输入: s = "cbaebabacd", p = "abc"
输出: [0,6]
解释:
起始索引等于 0 的子串是 "cba", 它是 "abc" 的异位词。
起始索引等于 6 的子串是 "bac", 它是 "abc" 的异位词。

```python
class Solution:
    def findAnagrams(self, s: str, p: str) -> List[int]:
        s_len, p_len = len(s), len(p)

        if s_len < p_len:
            return []

        ans = []
        s_count = [0] * 26
        p_count = [0] * 26

        for i in range(p_len):
            s_count[ord(s[i]) - 97] += 1
            p_count[ord(p[i]) - 97] += 1

        if s_count == p_count:
            ans.append(0)

        for i in range(s_len - p_len):
            s_count[ord(s[i + p_len]) - 97] += 1
            s_count[ord(s[i]) - 97] -= 1

            if s_count == p_count:
                ans.append(i + 1)

        return ans
```

两个数组的交集

```python
class Solution:
    def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:
        mp={}
        for i in nums1:
            mp[i]=mp.get(i,0) + 1

        ans={}

        for i in nums2:
            if i in mp:
                ans[i]=ans.get(i,0)+1
        return list(ans.keys())

# 法二    return list(set(nums1) & set(nums2))
```

快乐数

> **输入：** n = 19
> **输出：** true
> **解释：**
> $1^2 + 9^2 = 82$
> $8^2 + 2^2 = 68$
> $6^2 + 8^2 = 100$
> $1^2 + 0^2 + 0^2 = 1$

```python
class Solution:
    def isHappy(self, n: int) -> bool:
        s = set()
        while n != 1:
            n = sum(int(i) ** 2 for i in str(n))
            if n in s:
                return False
            s.add(n)
        return True
```

四数相加 II

```python
from collections import Counter

class Solution:
    def fourSumCount(self, A: List[int], B: List[int], C: List[int], D: List[int]) -> int:
        countAB = Counter(u + v for u in A for v in B)
        ans = 0
        for u in C:
            for v in D:
                if -u - v in countAB:
                    ans += countAB[-u - v]
        return ans
```

最小操作次数使数组元素相等

逆向思考：其中一个数减1

```python
class Solution:
    def minMoves(self, nums: List[int]) -> int:
        min_num = min(nums)
        res = 0
        for num in nums:
            res += num - min_num
        return res
```

三数之和

```python
class Solution:
    def threeSum(self, li: List[int]) -> List[List[int]]:
        li.sort()
        ans = []

        for i in range(len(li)-2): #-2
            if li[i] > 0:break
            if i>0 and li[i]==li[i-1]:continue

            l,r = i+1,len(li)-1
            while l<r:
                s = li[i]+li[l]+li[r]
                if s<0:
                    l += 1
                    while l<r and li[l] == li[l-1]:l += 1
                elif s>0:
                    r -= 1
                    while l<r and li[r] == li[r+1]:r -= 1
                else:
                    ans.append([li[i],li[l],li[r]])
                    l+=1
                    r-=1
                    while l<r and li[l] == li[l-1]:l += 1
                    while l<r and li[r] == li[r+1]:r -= 1
        return ans
```

四数之和

```
class Solution:
    def fourSum(self, nums: List[int], target: int) ->
List[List[int]]:
        nums.sort()
        ans = []

        for i in range(len(nums)-3): # -3
            # if nums[i]>0:break
            if i>0 and nums[i]==nums[i-1]:continue

            for j in range(i+1, len(nums)-2):
                if j>i+1 and nums[j]==nums[j-1]:continue

                l,r=j+1,len(nums)-1
                while l<r:
                    tot = nums[i]+nums[j]+nums[l]+nums[r]
                    if tot==target:
                        ans.append([nums[i],nums[j],nums[l],nums[r]])
                        l+=1
                        r-=1
                        while l<r and nums[l]==nums[l-1]:
                            l+=1
                        while l<r and nums[r]==nums[r+1]:
                            r-=1
                    elif tot>target:
                        r-=1
                        while l<r and nums[r]==nums[r+1]:
                            r-=1
                    else:
                        l+=1
                        while l<r and nums[l]==nums[l-1]:
                            l+=1
        return ans
```