Problem-977

[12 9 1 0 4 16 36] ← asc order

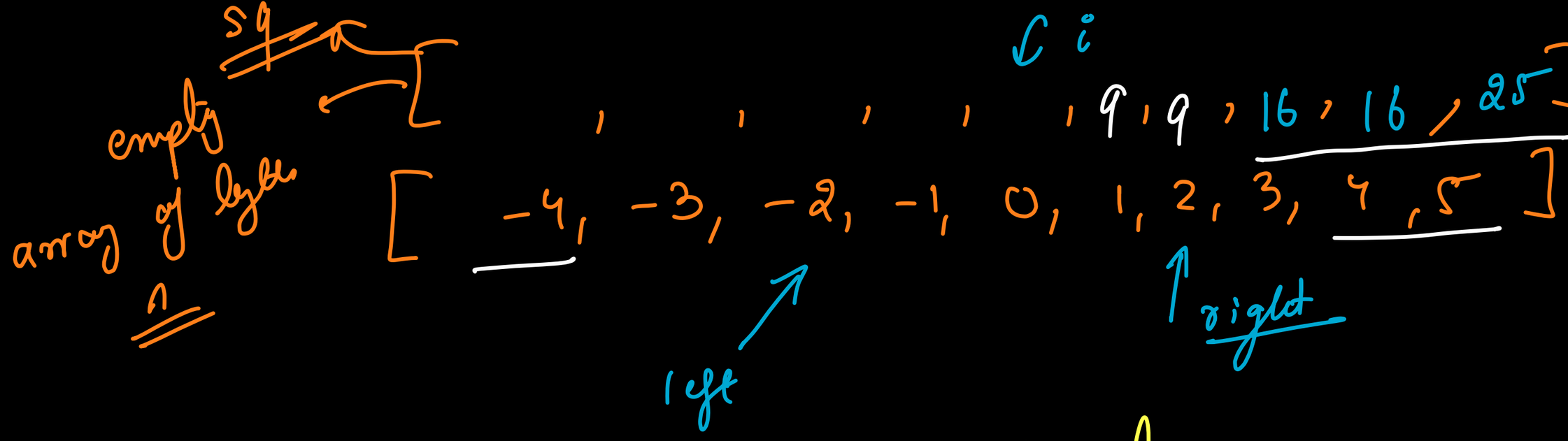
[-4, -3, -1, 0, 2, 4, 6]

return a new array

[0, 1, 4, 9, 16, 16, 36]

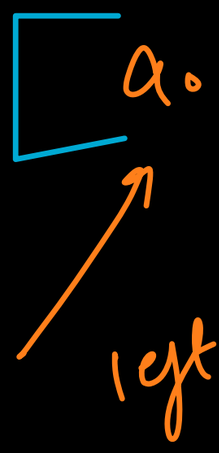
Brute force → To create a new array, & store square of each value in the original array. & then use an interval Sort func^n to array the elements in inc order.

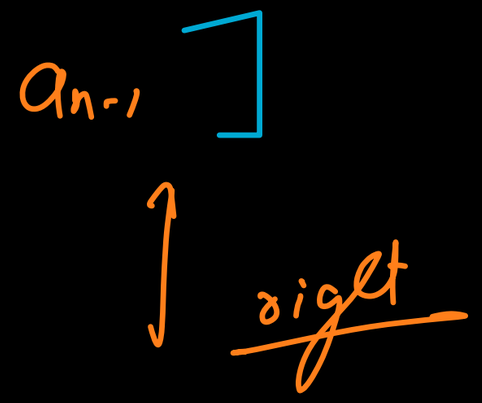
```
result = []
for(i=0 ; i<n ; i++) {
    result.push(arr[i] ** 2)
}
result.sort(                    )
```



whenever we square a no, it will be always a +ve no.

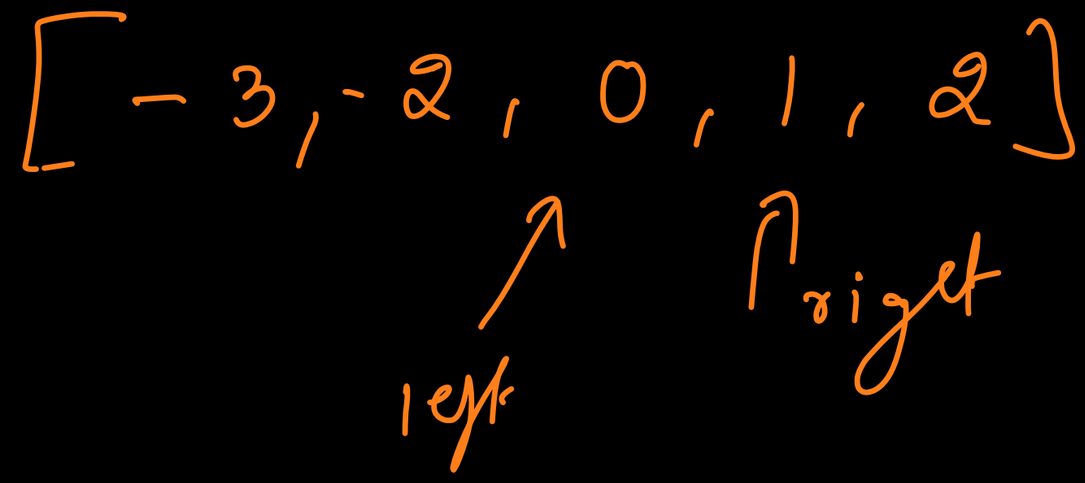
↳ because we have +ve & -ve no, the first & last element when sq up given the candidate for the largest element.

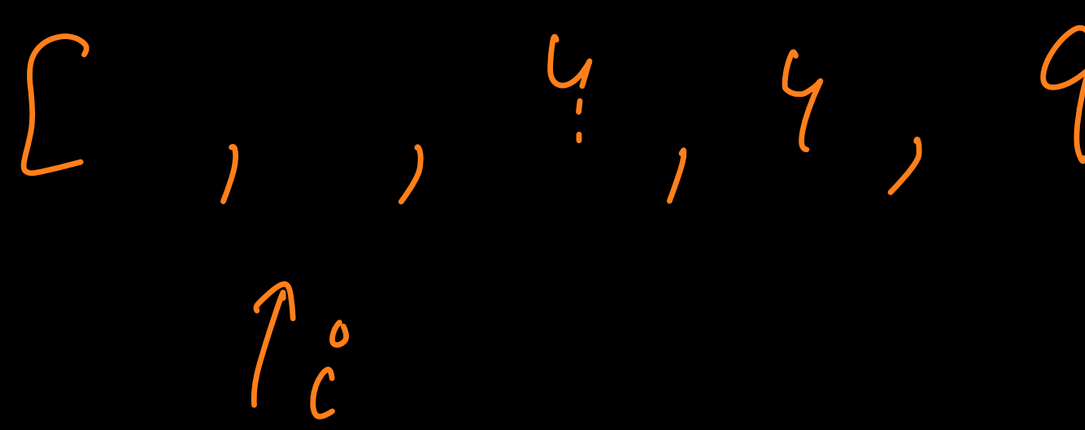




if we see the extreme ends, then sg of our g then is the biggest root.

```
left = 0
right = n - 1
for ( i = n-1 ; i >= 0 ; i--) {
    if ( nums[left] ** 2 < nums[right] ** 2) {
        result[i] = nums[right] ** 2;
        right--;
    } else {
        result[i] = nums[left] ** 2;
        left++;
    }
}
```





Problem 15

(-1, 0, 1, 2, -1, -4)

[0, 0, 0]

[0, 0, ...]

(-1, 0, 1)

(-1, -1, 2)

(0, 1, -1)

2

[0, 0, 0, 0]

1 → [0, 0, 0]

(a, b, c) → asc order

$a + b + c = 0$

$a + b = -c$ → sum of a pair equal to a target.

$-1, 0, 1, 2, -1, 4$

⇓

$[-1, -1, 0, 1, 2, 4]$ → asc

i j k
c

target = $-c$

$= -(-1)$

$= 1$

$[-1, -1, 2]$

$[-1, 0, 1]$

target $= -c$

$= -0$

$= 0$

2^c

$[-1, -1, -1, 0, 1, 1, 2, 2, 4]$

target $= -c$

$= -(-1)$

$= 1$

$[-1, -1, 2]$

$[-1, 0, 1]$

[0, 0, 0, 0]

c = 0

target = -c

= -0

= 0

[0,0,0]

```
for (c=0 ; c<n ;  c++) {
    if ( c==0 || nums[c] != nums[c-1]) {
```

// 2 sum logic

curr elmt is diff than prev

```
    }
}
```

problem 249 → anagrams are permutation of each other.

sanket ✗
sam ✗

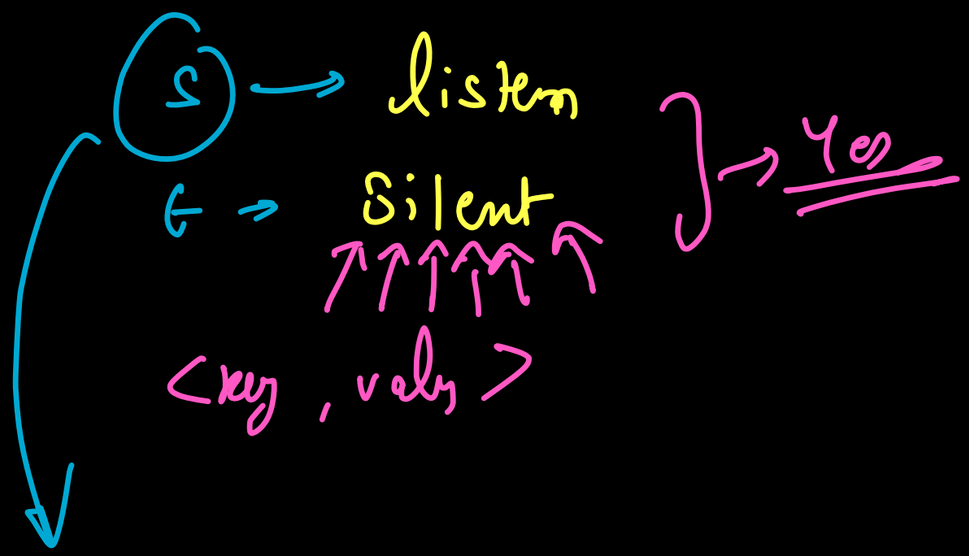
silent ↙
listen ↙] → eilnst, eilnst]

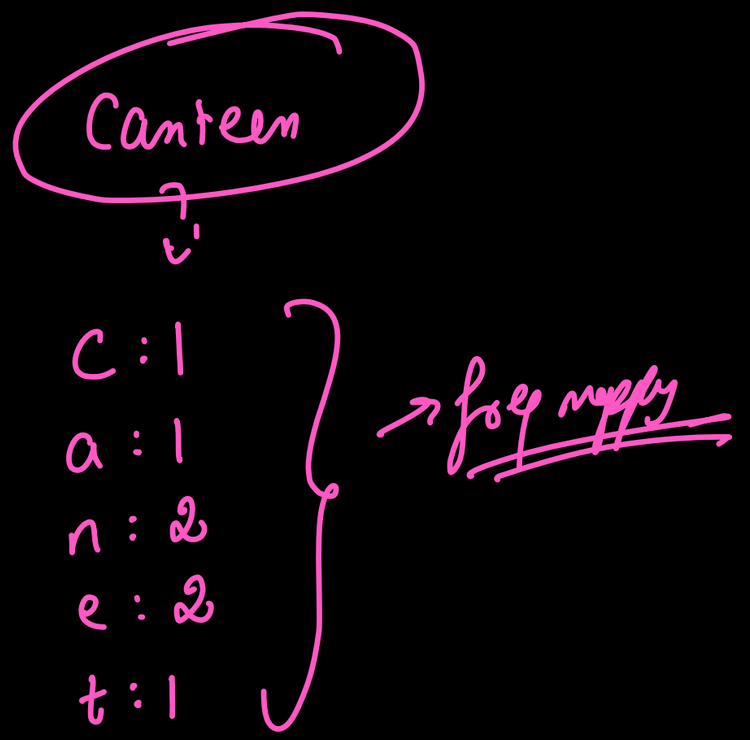
Brute force → Sort both the words.

how to optimise

↳ in anagrams, the same set of chars are present with same freq.

how about we prepare a freq mapping ??





<key : value>

s → anagram →

t → nagaram ↳ true

s → rat →

t → car

false ✗

r : 1

a : 1

t : 1

Problem 217

↳ Log map of integers

[1, 2, 1, 3, 2, 6]

{
1 : 2
2 : 1
3 : 1
6 : 1
}

>1

true

[1, 2, 3, 4]

{
1 : 1
2 : 1
3 : 1
4 : 1
}

false

[11, 2, 7, 15] target = 9

const

{ 11:1
2:1
7:1
15:1
3

→ freq map

(a, b)

a + b = target

↑ fix

a = target − b

mp[target − b] != = ends