16/8/20

1]

	0	1	2	3	4	5	6	7
ar_N:	3	10	-4	8	15	-7	23	12

True False

$K = 18, 17$

$ar(i) + ar(j) = K$

$i \neq j$

A: Brute Force $i = 0$ to N

$j = i+1$ to N

if ( )

$\Rightarrow O(N^2), 1$

It works because we are checking all the possibilities

B: Sort the array

P1 ↓ P2 ↓

ar_N: -7 -4 3 8 10 12 15 23

$ar(P1) + ar(P2)$ — $< K$, P1+ ; return tr ; $> K$, P2-

if P1 == P2 return false

$\Rightarrow O(N \log N + N), O(N)$

→ Two-Pointer Technique

① A, B (N, M) ② Merge ③ $a + b = K$

2]

	0	1	2	3	4	5	6
ar_N:	3	10	4	-6	12	18	-11

$ar(i) + ar(j) + ar(k) = K$

$i \neq j \neq k$

A: Brute Force

3 loops

$\Rightarrow O(N^3), O(1)$

B: Sort the array

0	1	2	3	4	5	6
-11	6	3	4	10	12	18

$a + b + c = S$

$a + b = S - c$

$S' = S - C$

↳ Use 1] two pointer technique to find S'

If not found, increase the pointer of c by 1.

$\Rightarrow O(N \log_2 N + N^2), O(N)$

→ SEARCHING

ar: 5 2 10 -8 3 12 15 9 -6 4

1. Linear Search: O(N), O(1)

2. Binary search: ① sort the array

	0	1	2	3	4	5	6	7	8	9
ar:	-8	-6	2	3	4	5	9	10	12	15

K = -6

② Find mid and compare with mid element and if less, increament the pointer to next mi or decrement

```
bool BSI (int ar[], int N, int K)
{
    ~~int mid = N/2;~~
    ~~if (ar[mid] == K)~~
    ~~{~~
        ~~return true;~~
    ~~}~~
int lo = 0, hi = N-1, M;
while (lo ≤ hi) {
    m = (lo+hi)/2;
    if (K < M)
    {
        hi = M-1;
    }
    else if (K > M)
    {
        lo = M+1;
    }
    else
        return true;
}
return false;
```

```
bool BSR(int ar[], int K, int
T(N)                int hi)
{
1 ←     if (lo > hi) return Fal
1 ←     int m = (lo+hi)/2;
1 ←     if (ar(m) == K)
1 ←         return true;
1 ←     if (K > ar(m))
T(N/2)
        ← return BSR(ar, K, m+1, hi);
        ~~else~~
T(N/2)
        ← return BSR(ar, K, lo, m-1);
}
```

$\therefore T(N) = T(N/2) + 1$

$\therefore T(N) = \log(N)$

$\Rightarrow O(\log N), O(1)$

	Unsorted Array		Sorted Array	
	1 Key	Q Keys	1 Key	Q Keys
near Search	N	$Q \times N$	N	$Q \times N$
inary search	$N\log N + \log N$	$N\log N + Q\log N$	$\log N$	$Q \log N$

$T(N) = T(N/2) + 1$ → Substitution, → Recurrence tree, → Master's Theorem

(simplified)

→ Master's Theorem:

$$T(N) = aT\left(\frac{N}{b}\right) + f(N) \qquad a \geq 1 \ (\mathbb{Z}^+), \quad b \geq 1$$

$$t = \log_b a, \quad f(N) = N^c$$

① $t < c$: $O(N^c)$

② $t = c$: $O(N^t \log N)$

③ $t > c$: $O(N^t)$

$$T(N) = 1 \cdot T(N/2) + N^0 \qquad c = 0$$

$$t = \log_b a = \log_2 1 = 0$$

$t = c$

$O(N^0 \log N)$

$O(\log N)$

base cond: $T(1) = 1$

1. $T(N) = 2T(N/2) + N \Rightarrow a=2, b=2, t=1, f(N)=N, c=1$; 2nd formula
2. $T(N) = 2T(N/2) + 1 \Rightarrow a=2, b=2, t=1, c=0$; 3rd formula $O(N)$
3. $T(N) = T(N/2) + N \Rightarrow a=1, b=2, t=0, c=1$; 1st formula $O(N)$
4. $T(N) = T(N/2) + 1 \Rightarrow a=1, b=2, t=0, c=0$;
5. $T(N) = T(N/2) + T(N/4) + N \times N \Rightarrow$ cannot apply master's

3]

	0	1	2	3	4	5	6	7	8	9
ar_N:	3	7	-2	15	8	12	-5	20	10	-3

x: $\max(ar(i)) \le x$
(floor (x))

Given an array and Q queries with x, find the max element in the array $\le x$.

18 → 15

10 → 10

17 → 15

-3 → -5

25 → 20

-20 → INT_MIN(-2^{31})

11 → 10

A: Brute Force :

```
int floor(int ar[], int x, int N) {
    int ans = Integer.MIN_VALUE;
    for(int i=0; i<N; i++)
    {
        if (ar[i] ≤ x)
        {
            if(ans < ar[i])
            {
                ans = ar[i];
            }
        }
    }
    return ans;
}
```

$\Rightarrow O(Q \times N)$, $O(1)$

B: Sort it and do binary search $\Rightarrow O(N\log N + Q\log N)$, $O(1)$

	0	1	2	3	4	5	6	7	8	9
Sorted array:	-13	-5	-2	3	7	8	10	12	15	20

```
int floor(int ar[], int x, int N)
{
    int lo = 0, hi = N-1, m, ans = INT_MIN;
    while ( lo <= hi )
    {
        m = lo + hi/2;
        if (ar(m) > x)
        {
            hi = m-1;
        }
        else
        {
            ans = ar(m);
            lo = m+1;
        }
    }
    return ans;
}
```

4]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ar_N :	1	3	5	3	3	8	-2	10	3	15	15	-7	-2	20	3

Q: x : int(x)

10	1
3	5
15	2
12	0
3	5
-2	2

A: Linear search and count

$O(Q+N)$, 1

B: Count Array

$O(N+Q)$, $O(R)$

C: Merge Sort and binary search and expand

$$O(\underbrace{N\log_2 N}_{\text{sort}} + Q(\underbrace{\log_2 N}_{\text{binary search}} + \underbrace{N}_{\text{checking on both sides if } x \text{ found}})),\ O(N)$$

D: Merge Sort and Two-pointer with linear search

$$O(\underbrace{N\log_2 N}_{\text{sort}} + Q\times N),\ O(N)$$

E: Merge Sort and Two-pointer with binary search

ar_N: -7 -2 -2 1 3 3 3 3 3 5 8 10 15 15 20

lo = m+1   P1 BS1   P2 BS2   hi = m-1

for BS1 : ans = m
hi = m-1

for BS2 : ans = m
lo = m+1

then return P2 - P1 + 1

$$O(N\log_2 N + Q(2\log_2 N)),\ O(N)$$

F: Sort the array and sort the queries as well

							$ans_Q[\,]$					
br_Q:	-2	3	3	10	12	15	2	5	5	1	0	2
ar_N:	-7	-2	-2	1	3	3	3	3	3	5	8	10	15	20

Traverse br and ar and if it matches, until x is present increment the count and move to next element in br, if the value in br is > ar, then check previous br and if previous x is not same as current then count = 0 print 0, and move to next element in br.

Since order in $ans_Q[\,]$ is not as same as required, we have to compare with original queries and then print ans

complexity:

for two pointer in both array and queries

$$O\left(N\log_2 N + Q + Q\log Q + N+Q + Q\log Q\right), \quad N+Q+Q+Q+Q$$

$$O(N+Q)$$

- $N\log_2 N$ → sorting the array
- Q → copy queries
- $Q\log Q$ → sort queries
- $N+Q$ → for two pointer in both array and queries
- $Q\log Q$ → binary search for ans with original

G1: Sort the array and travers and make 2 arrays A and B and then binary search x in A and print corresponding element in B.

A : −7 −2 1 3 5 8 10 15 20

B : 1 2 1 5 1 1 1 2 1

$$N\log_2 N + N + Q\log_2 N, \quad N + N + N$$

- $N\log_2 N$ → Sort
- N → traversing array and forming B.
- $Q\log_2 N$ → binary search on x
- N → merge
- N → A
- N → B

H: HashMap

1. Insert ar(i) → in Hashmap
2. Search x in HashMap
   - → print cnt (if found)
   - → print 0 (if not found)

$a+b=k$

1] C: Binary Search

$a+b=k$

$b=k-a$

↑

binary search b

ar_N	-9	-4	3	10	12	15	20	25

K = 18	
a	b
-9	27 x
-4	22 x
3	15 ✓

K = 24	
-9	33 x
-4	28 x
3	21 x
10	14 x
12	12 ✓

but $i=j$ in this case, this should return false

```
bool solve (int ar[], int N, int k){
N→MS (ar, 0, N-1);
→for(int i=0; i<N; i++)
        a = ar(i);
        b = k-a;
N → if (BSR (ar, b, i+1, N-1) == T)
        return True;
return false;
```

that's why we start from $i+1$

$\Rightarrow O(N\log N + N\log N),\ O(N)$

$a+b=k$
- N^2 (BF)
- $N\log N + N$ (2 ptr)
- $N\log N + N\log N$ (BS) (2)

$a+b+c=k$
- N^3
- $N\log_2 N + N\times N$
- $N\log N$
- $N^2\log N$

(2)

$a-b=k$
- N^2
- $N\log_2 N + \boxed{??}$ TODO LIST
- $N\log N + N\log N$

(2)

Implement (2)
(2), (2)