10/22/23

Q = Query (Element to search)

CLASS - 9

~~LAB~~

1) 7 to 9 (CASES)
2) 10 to 12

Write Q & N instead of N & Q

	Sorted		Unsorted	
	Q=1	Q=Q	Q=1	Q=Q.
L.S	N*1, 1	N*Q, 1 (Q+N, 1) ✓	N*1, 1	N*Q, 1
B.S	$\log_2 N$, 1	$\log_2 N * Q$, 1	$N \log N + \log_2 N$, N	$N \log N + (\log_2 N) * Q$, N

PROBLEM.

15 24 19 -31 42 -10 16 19.

Q 45
16
20
-40

max (arr[i]) $\leq$ ele.

Python V.13

① while (q--)
{
int ans = -∞;
for(i: 0 -> N-1)
if (arr[i] < ele)
{ ans = ~~arr[i]~~
max(ans, arr[i]);
}
}
return ans;

Soln
Q * N ; ;

NOTE- FOR MAX VALUE AND MIN IN LANG.

C/C++	Java	Python
INTMAX → +∞	Integer. MIN_VAL	-(1<<31)
INT_MIN → -∞	Integer. MAX_VAL	(1<<31)-1

② - Sort 0 1 2 3 4 5 6 7
-3 1 10 15 16 19 19 24 42

0+7/2 = 3
0+3/2 = 1
2+3/2 = 2
4+3/2 = 3
4+4/2 = 4

B.S.

```
void BS(arr, ele); ans = -∞;
while(low <= high) → low<high for ele.
{ mid = low + high/2.
max(arr[mid]) <= ele)
ans = arr(mid);
low = mid + 1;
} else {
high = mid - 1;
}
}
```

We r going right as we want equal to number, mid=

ele = -8
0+0/2 = 0
mid = arr[mid] 16 7 -10.
high = mid - 1
low <= high 0 to 2 0+2/2 = 1
ans = -8;

PROBLEM ********

Given arr: 20 30 20 19 -16 24 20 30 19 40

Get the freq of each ele.

Op 20 → 3
-16 → 1
12 → 2
-16 → 1
35 → 0

1)

Sol ① Count / Sort (N, N)
print the 26 ele count stored in countsort at loc=20

② ① B.F (for N, 1) → Keep seeing every num, if u see ele increment counter

```
Void count (arr, ele)
        n=0
{
  for (i=0 ; i<len(arr); i++)
    {
      if (arr[i] == ele)
        { n+=1 ;
        }
    }print(ans);
}
```

② Count arr:

$-16 \le arr[i] \le 40$

$a \le arr[i] \le b$ = b a+1

count [b-a+1];

count [arr[i] - a] ++;

print (count [ele - a]);

Soln

N + Q*1 , R

N → store count arr

Q → printing

space range ≤ 10^5

This is the best soln technically if the range is $R \le 10^5$. Coz, N → every ele approached once

Q → every Query "

(N+Q*1).

③ Binary Search.

Merge Sort

0	1	2	3	4	5	6	7	8	9
-16	19	19	20	20	20	24	30	30	40.

Sorted arr:

Land at one of 20's

leb say → 5

Then phns P_1 P_2 at 5. move P_1 to left, P_2 to right and keep counting

$P_1 = 4$, $P_2 = 6$ ✓ $P_2 - P_1 - 1 \Rightarrow 6 - 2 - 1 = 3$.

$P_1 = 3$ ✓

$P_1 = 2$ ✗

Stop at the failing point.

$N \log N + Q(\log N + N)$, 1

Sort array — Binary search — 2 ptr

Code

```
void fnqsort (arr, Q)
{
    ele = 0
    arr.sort( ), N = len(arr)
    low = 0, high = N-1, mid = (low+high)/2.
while (low <= high)
    { if (arr[mid] == high)
      {  P1 = mid
         P2 = mid
         break
      }
      elif (arr[mid] > high)
      {
          high = mid-1
          low =
      }
      elif (arr[mid] < high)
      {  low = mid+1
      }
}
```

0 1 2 3 4
1 2 2 1
4-0=1

Complexity

NlogN + O(logN+N)

```
if idx == -1
  print 0
else.
while(arr[P1] == Q)        P1 >= 0 &&
  { P1 -= 1
  }
while (arr[P2] == Q)
  { P2 += 1
  }
print (P2-P1-1).
```

```
while (P1 >= 0 && P2 <= high)
{.      P1 >= 0 &&.        Q
  while( arr[P1] == mid )
    { P1 += 1
    }
  while( P2 <= high &&.      Q
         arr[P2] == Q )
    { P2 += 1
    }
}.
```

```
print(P2-P1-1).
```

(4) Main pt of time c in last prob is, 2-pts
It taking O(N). Let reduce that

-16 19 19 20 20 20 24 30 30 40.
let first occ

1) go to left & do bs for low range → get l
2) go to rt & do bs for high range → get h.

Total 20's = h-l+1

Last occurance

```
low, high, mid, l=-1
while (low <= high)
{ if (arr[mid] == ele)
  { l = mid
    low = mid + 1
  }
elif (arr[mid] < el)
  { low = mid + 1
  }
else (high) mid - 1
}
return l
if l == -1
print(0)
```

(No need to check for 1st occurance as ele is not here.

First occurance

```
int low=0, high=N-1, mid, f=-1;
while(low<=high)
  if (arr[mid] == ele)
  {   f=mid;
      high=mid-1
  }
if (arr[mid] < ele)
  { low=mid+1
  }
else arr[mid]
  { high=mid-1
  }
return f
}
```

Complete Complexity

NlogN + (logN + logN)
↓ sort — ↓ f — ↓ l — ↓ L-f+1 (return ans)

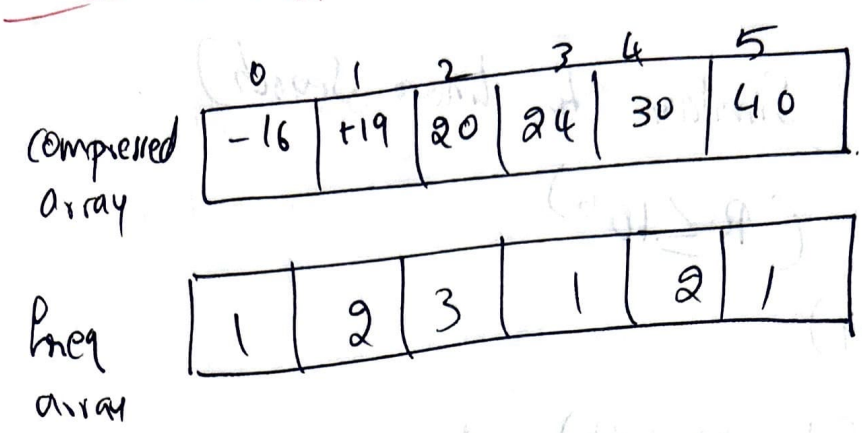
NlogN + 2(2logN + 1)
↓ sort — ↓ 2 B.S. — ↓ returning ans.

NlogN + N + 2(log$_2^N$ + 1) / N+N (space)
↓ sort — ↓ comp & freq arrays

T.C.

In B.S br(comp, 0, P2)
coz size of compressed array is P2 & not complete array.

Soln 5 : Not a regular search or sort technique

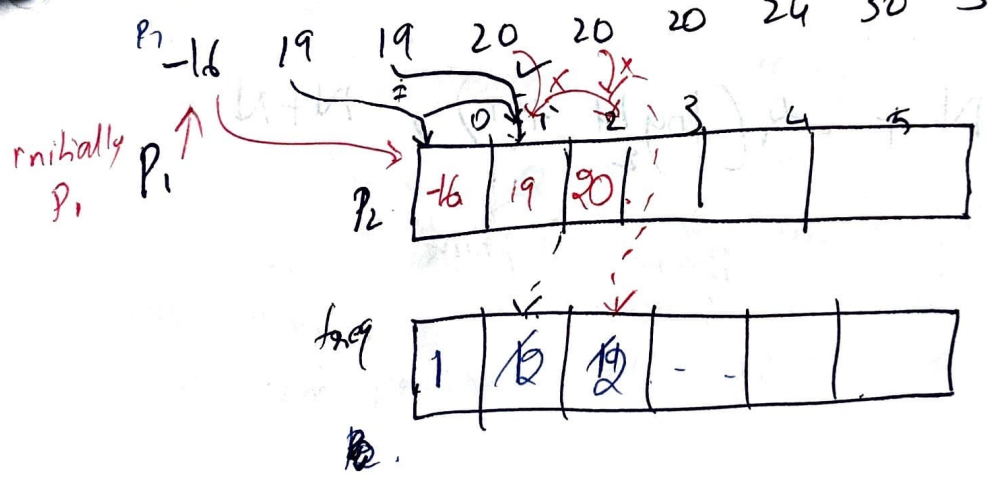


int idx = B.S (comp, 0, 5, 20)

if idx != -1 { print (freq_array[idx]) }

idx == -1 → print (0).

How to compress the array




```
int comp[N], freq[N]
comp[0] = arr[0]
freq[0] = 1
P1 = 0 ; P2 = 0
while (P1 < N)
{ if (arr[P1] == comp[P2])
    { freq[P2] += 1
    }
  else
    { P2 += 1
      comp[P2] = arr[P1]
      freq[P2] = 1
    } P1 += 1
}
```

```
while (q--)
{
int idx = bs
bs(comp, 0, P2, ele)
if (idx != -1)
    print (freq[id
```

Comp : N log N + N | , N + N

Solution:

1) $Q * N$, 1 [Similar to linear Search]

2) $N + Q * 1$, R $(R \leq 10^5)$
(count array)

3) $N \log_2 N + Q(\log_2 N + N)$, 1
Sort arr — B.S. — 2 ptr.

4). $N \log N + Q(\log_2 N + \log_2 N + 1)$, 1
first occurance — last occurance — L-F+1

5) $N \log_2 N + N + Q(\log_2 N + 1)$, $N+N$
Sort array — compressed freq. — Binary search — print — space comp array freq array

⑥ Take Queries in an array.

Q.	20	-16	19	-16	35

Sort

-16	-16	19	20	35

Note: Queries can repeat. ~~20 30 20 19 -16 24 20 30 19 40.~~

-16 -16 19 20 35 → Compare with array.

-16 19 19 20 20 20 24 30 30 40.

freq

1	1	2	3	0.

20
-16
19 → 0.
-16
35

In freq compare query with array if present count++=1 move next in

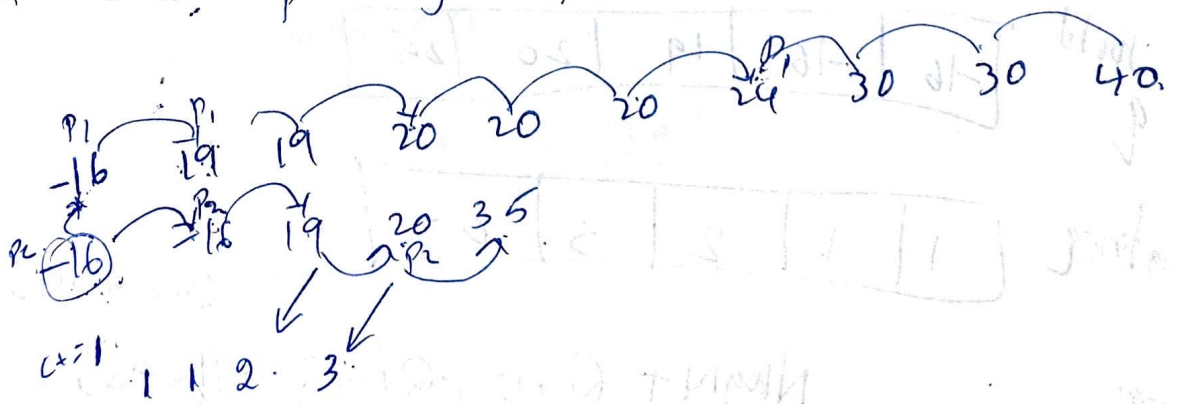
1) Compare $Q \xrightarrow{(P_2)}$ array(P_1)

2) If present c++=1, move P_1

3) Compare Q → arr (P_2)

If fails. move P_2. update freq

In the end printing o/p order is failing.



① Handling the query seq

1) Query & index array

2) // sort both query & index array

// sort can be done using merge sort

if (arr[P1] ≤ arr[P2])

{ inde

3) fill (freq) answer array using 2 ptr technique on sorted original array & sorted queries array.

4) ~~Unsort the index array.~~ // sort on sorted index array & freq array

② ~~Easier~~ soln

Binary search. [sorted q

↓

and print corresponding freq value in freq array.

q: | 20 | -16 | 19 | -16 | 35 |

sorted q: | -16 | -16 | 19 | 20 | 25 |

| 1 | 1 | 2 | 3 | 0 |

(ans array)

$N \log N + Q + Q + Q \log Q + (N+Q) + Q(Q \log Q + 1)$

BS on sorted q

$N\log N + Q + Q + Q\log Q + (N+Q) + Q*(\log Q + 1)$

- $N\log N$ → sort the main array
- Q → 2 Queries arrays. (Q, freq)
- $Q\log Q$ → sort q array
- $(N+Q)$ → 2-ptrs
- $Q*(\log Q + 1)$ → B.S on sorted queries array

Space:

(7) LAST AND IMPORTANT APPROACH

Hashing {20:3, 30:2, 19:2, -16:1, 24:1, 40:1}

Based on Query for each query go into hash and print the count.

NOTE

1) Hashing takes constant time for insert, ~~delete~~ search

So overall complexity

$N*1 + Q*1$, N ⇒ space for hash table.

(Insertion) Searching