Assignment - 1

Name: Md. Minhazul Mowla

ID: 23201390

Section: 13

Course: CSE221

Submission Date: 7/3/25

Ans. to the Ques. No.-1

a

loop-1,

step	i
0	$\frac{n}{2}\times\left(\frac{1}{6}\right)^0$
1	$\frac{n}{2}\times\left(\frac{1}{6}\right)^1$
2	$\frac{n}{2}\times\left(\frac{1}{6}\right)^2$
...	...
k	$\frac{n}{2}\times\left(\frac{1}{6}\right)^k$

$$\frac{n}{2}\times\left(\frac{1}{6}\right)^k \leq 1$$

$$\Rightarrow \log_6\left(\frac{n}{2}\right) \geq k$$

$$\therefore k = \log_6\left(\frac{n}{2}\right) \approx \log_6(n)$$

$$\therefore O(\log_6(n))$$

loop-2,

step	i
0	2
1	$8 = 2\times 4^1$
2	$32 = 2\times 4^2$
...	...
k	2×4^k

$$2\times 4^k \leq n$$

$$\cancel{\Rightarrow \log_4}$$

$$\Rightarrow 4^k \leq \frac{n}{2}$$

$$\Rightarrow \log_4\left(\frac{n}{2}\right) \geq k$$

$$\therefore k = \log_4\left(\frac{n}{2}\right) \approx \log_4(n)$$

$$\therefore O(\log_4(n))$$

loop - 3,

step	k
0	0
1	0
...	...
m	0

$\therefore O(\infty)$

$\therefore O(\log_6 n \times \log_4 n \times \infty)$

$\therefore O(\infty)$

b

loop - 1,

steps	i
0	n
1	n-1
2	n-2
...	...
$\cancel{k=}n$	$\cancel{n-k}$ 1

$\therefore O(n)$

loop - 2,

step	j
0	2 = 2+0
1	3 = 2+1
2	4 = 2+2
...	...
k	n = 2+k

$\therefore O(n)$

loop-3.

step	i
0	$\frac{n}{2}\times(\frac{1}{6})^0$
1	$\frac{n}{2}\times(\frac{1}{6})^1$
...	...
k	$\frac{n}{2}\times(\frac{1}{6})^k$

$\therefore O(\log_6(n))$

loop-4,

step =	j
0	2
1	$8 = 2\times4^1$
...	...
k	2×4^k

$\therefore O(\log_4(n))$

loop-5.

step	k
0	0
1	0
...	...
m	0

$\therefore O(\infty)$

$\therefore O(n\times n + \log_6 n \times \log_4 n \times \infty)$

$\therefore O(\infty)$

Scanned with CamScanner

Ans. to the Ques. No - 2

Python:

a

```
def bs_f_idx (arr, n = len(arr), val):
    l = 0
    r = n-1
    ans = -1
    while l<=r:
        m = (l+r)//2
        if arr[m] == val:
            ans = m
            r = m-1
        elif arr[m] < val:
            l = m+1
        else:
            r = m-1
    return ans
```

b

```
def bs_l_idx(arr, n=len(arr), val):
    l = 0
    r = n-1
    ans = -1
    while l<=r:
        m = (l+r)//2:
        if arr[m] == val
            ans = m
            l = m+1
        elif arr[m] < val:
            l = ~~mid+~~ m+1
        else:
            r = m-1
    return ans
```

def count(arr, n = len(arr), val):
    f = bs_f_idx(arr, n, val)
    if f != -1:
        l = bs_l_idx(Arr, n, val)
        ~~return(f, l - f + 1)~~
    ~~return f~~
        ~~return {f, l~~
        return f, l-f+1
    return f, 0

Ans. to the Ques. No.-3

Yes. the algorithm will work even though ~~its not~~ s the array is not sorted

	L	R	m
i.	0	7	3
ii	0	2	1

Ans. to the Ques. No.-4

a

```
l = 0
r = len(arr)-1
while l<r:
    m = (l+r)//2
    if arr[m]< A[m+1]:
        l = m+1
    else:
        r = m
print(arr[l])
```

b

step	element
0	$n = \frac{n}{2^0}$
1	$\frac{n}{2} = \frac{n}{2^1}$
2	$\frac{n}{4} = \frac{n}{2^2}$
...	...
k	$\frac{n}{2^k}$

$$\frac{n}{2^k} = 1$$

$$\therefore k = \log_2 n$$

$$\therefore T(n) = O(\log_2 n) \quad : [O(1) \text{ ignored}]$$

Ans. to the Ques. No.-5

a

python

```
def LinearSearchToFindSquareRoot(key):
    result = -1
    for i in range(1, key + 1):
        if i * i <= key:
            result = i
        else:
            break
    return result
```

b

Python

```
def LinearSearchToFindSquareRoot (key):
        l = 0
        r = key
        result = -1
        while l <= r:
            m = (l+r)//2
            if m*m <= key:
                result = m
                l = m+1
            else:
                r = m-1
        return result
```

Ans to the Ques. No.-6

a

If there's more than one search in the same testcase, sort at binary search works ~~more~~ faster than linear searching everytime.

b

Adding a certain integer to ~~make all neg~~ every element in order to make a non-negative integer array. Do the count sort. Substract the added integer from every element.

c

There's both float and negative integers in the given list.

i. multiply ~~with~~ 10 with every element

ii. find the smallest integer(-51) and add 51 to every integer

iii. do count sort

iv. substract 51 from every integer

v. float divide with 10 to every element

d

Both merge sort and quick sort are good options. But as the time complexity of quick sort on worst case is $O(N^2)$, I would say merge sort is the best option.

e

arr = [1, 2, 3, 4, 5, 6, 8 ..... 100]

Ans to the Ques No.-7

Python

```
n = len(arr)
if (n-1) % 2 == 0:
        e = arr[n-1 : : -2]
else:
        e = arr[n-2 : : -2]

o = arr[1 : : 2]
ans = []
i = 0
j = 0
while   i < len(e)   and   j < len(o) :
        if   e[i] < o[j]:
            ans += [e[i]]
        ~~else~~  i += 1
        else:
            ans += [o[i]]
            j += 1

while  i < len(e) :
    ans += [e[i]]
    i += 1

while  j < len(o):
    ans += [o[i]]
    j += 1

print(ans)
```