Question:- 1 (A)

• Time Complexity:-

→ Passes:-

```
i = 0 {
    {   j = 0 => sum[0] += entry[0][0];
        :
        j = n+1 => sum[0] += entry[0][1];
    }
}
i = 1
    j = 0 => sum[1] += entry[1][0];
    j = n-1 => sum[1] += entry[1][j-1];
i = 2
    j = 0 => sum[2] += entry[2][0];
    j = n-1 => sum[2] += entry[2][n-1]
i = 3
    j = 0 => sum[3] += entry[3][0]
    j = n-1 => sum[3] += entry[3][n-1]
i = 4
    j = 0 => sum[4] += entry[4][n-1];
```

$\Rightarrow [(n+1)(n+1)] + n^2$

$\Rightarrow 2n^2 + 2n + 1 = f(n) = O(n^2)$ Ans.

Question: 2(B)

• Passes:

$i = 0 \rightarrow j = 0$

$i = 1 \rightarrow j = 0 \Rightarrow m += 0$
$j = 1$

$i = 2 \rightarrow j = 0 \Rightarrow m += 0$
$j = 1 \Rightarrow m += 1$

$i = 3 \rightarrow j = 0 \Rightarrow m += 0$
$j = 1 \Rightarrow m += 1$
$j = 2 \Rightarrow m += 2$

$i = 4 \rightarrow j = 0 \Rightarrow m += 0$
$j = 1 \Rightarrow m += 1$
$j = 2 \Rightarrow m += 2$
$j = 3 \Rightarrow m += 3$

• Time complexity:

$$f(n) = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

$f(n) = O(n^2)$ Ans

C)

Passes:

i = 1 → tot += 1
i = 1 × 2
i = 2 → tot += 2
i = 2 × 2
i = 4 → tot += 4
i = 4 × 2
i = 8 → tot += 8
i = 8 × 2
i = 16 → tot += 16
i = 16 × 2

Time complexity:

$f(n) = O(\log n)$

Due to i increases twice time on every iteration.

D)

Passes

i = 32 → tot += 32
i = 16 → // += 16
i = 8 → // += 8
i = 4 → // += 4
i = 2 → // += 2

Time complexity

$f(n) = O(\log n)$

f)

Passes:

Same as Part (A)

Time complexity.

$f(n) = (n^2 + n + 1)(n+1)$

$f(n) = O(n^3)$

E)

Passes

i = 0 → j = 0, n-1
i = 1 → j = 0, n-1
i = 2 → j = 0, n-1
i = 3 → j = 0, n-1
i = 4 → j = 0, n-1

Time complexity

$f(n) = O(n^2)$

G)

Passes.

i=0 → K=0, n-1
i=1 → K=0, n-1
i=2 → K=0, n-1
i=3 → K=0, n-1
i=4 → K=0, n-1

Time complexity

$f(n) = O(n^2)$

H)

Passes

i=0 → j=0, sqr(n)-1
i=1 → j=0, sqr(n)-1
i=2 → j=0, sqr(n)-1
i=3 → j=0, sqr(n)-1
i=4 → j=0, sqr(n)-1

Time complexity

$f(n) = O(n \times sqrt(n))$

I)

Passes

i=0
i=1
i=2
i=3
i=4

Time complexity.

$f(n) = O(n)$

J)

Passes

i=0
i=1
i=2
i=3
i=4 → i=0
i=1
i=2
i=3

Time complexity

$f(n) = O(n^2)$

K)

Passes

i=0 → j=0
i=1 → j=0
i=2 → j=0,1
i=3 → j=0,1,2
i=4 → j=0,1,2,3

Time complexity

$f(n) = O(n^2)$

Question: 2

A) Time complexity.

$$f(n) = [(n+1)(n+1)] + n^2$$
$$= 2n^2 + 2n + 1$$
$$= O(n^2)$$

B) Time complexity:

$$f(n) = [(n+1)(2n+1)] + n^2 \quad \because i = n$$
$$= 3n^2 + 3n + 1$$
$$= O(n^2)$$

c) Time complexity:

$f(n) = O(n \log n)$ → 'i' increases (i) twice times & (n) times.

Question: 3

A) Time complexity:

$$f(n) = ((\log n) \times (n+1)) + n \log n$$
$$= O(n \log n).$$ → here also (i) is increases 2 times.

B) Time complexity:

$$f(n) = [(n+1)(n+1)(n+1)] + n^3$$
$$= O(n^3)$$ → dominating term is (n^3).

Question: 4

A)

↳ Insertion is the best choice to its efficiently with nearly sorted data. Selection & Bubble sort are less efficient for this scenario due to their higher numbers of unnecessary comparisions & swaps.

b)

↳ Insertion sort is more optimal for sorting a user's recently Played songs by Playback time due to its efficiently with small lists & nearly sorted elements. leading to inefficiently for a lists of 20-30 items.

c)

- Insertion sort → $O(n^2)$ in worst case, in best case $O(n)$
- Selection sort → $O(n^2)$ initial order of elements {all same best, average}
- Bubble sort → $O(n^2)$ " " " " $O(n)$ is best.

Question: 5

A) Suresh

Bubble sort:

suresh
usresh
usersh
usehrs
usehs
sruesh
sreush
sresuh
sresuh
rsesuh
ressuh.
erssuh
ehrssu Ans

Selection sort:

suresh
rusesh
rsuesh
rseush
resush
ressuh
resshu
ersshu
ehssru
ehrssu

• Insertion sort:

suresh
~~sruesh~~
sruesh
rsesuh
resshu
ersshu
ehssru
ehrssu Ans.

B)

1) In-Place Sorting:

- Insertion sort: Yes
- Bubble Sort: Yes
- selection sort: Yes

2) online Sorting:

- Insertion sort: Yes
- Bubble Sort: NO
- selection sort: NO

3) Stable Sorting:

- Insertion sort: Yes
- Bubble sort: Yes
- selection sort: NO