SEC F TEST

Q. consider $T(n)_{best} = K_1 n + K_2$

$T(n)_{worst} = K_1 n^2 + K_2 n + K_3$

for a sorting algorithm 'A' and we have a function of A, titled 'Sort'

```
Div_solv (A)
    n = length (A)
    S1 = Sort (A[1 to n/2])
    S2 = Sort [A [n/2 + 1 to n])
    S = merge (S1, S2)
    return S
```

* merge has running time αn

Find T(n) for best and worst case

• DRY RUNNING:-

(BEST CASE)

A = [1, 2, 3, 4, 5, 6]

1. n = 6
2. S1 = Sort [1, 2, 3] = [1, 2, 3]
3. S2 = Sort [4, 5, 6] = [4, 5, 6]
4. S = merge (S1, S2) = [1, 2, 3, 4, 5, 6]
5. return [1, 2, 3, 4, 5, 6]

Lines of code	Frequency (Worst case) best
1	1
2	$K_1(n/2) + K_2$
3	$K_1(n/2) + K_2$
4	$K_1 n + K_2$
5	1

$$T(n) = 1 + K_1 \frac{n}{2} + K_2 + K_1 \frac{n}{2} + K_2 + K_1 n + K_2 + 1$$

$$T(n) = \frac{2K_1 n}{2} + 3K_2 + 2 + K_1 n$$

$$T(n) = 2K_1 n + 3K_2 + 2$$

• DISCUSSION:-

- Best case complexity of div-sol function is O(n)
- Div-sol function in best case grows linearly.
- means that, minimum time which a code can take

Date

• DRY RUNNING :-

(WORST CASE).

A = [6, 5, 4, 3, 2, 1]

1. n = 6
2. S_1 = Sort [6, 5, 4] = [4, 5, 6]
3. S_2 = Sort [3, 2, 1] = [1, 2, 3]
4. S = merge(S_1, S_2) = [1, 2, 3, 4, 5, 6]
5. return [1, 2, 3, 4, 5, 6]

LINES OF CODE	FREQUENCY
1	1
2	$K_1(n/2)^2 + K_2(n/2) + 1$
3	$K_1(n/2)^2 + K_2(n/2) + 1$
4	$K_1 n + K_2$
5	1

$$T(n) = 1 + K_1 \frac{n^2}{4} + K_2\left(\frac{n}{2}\right) + 1 + K_1 \frac{n^2}{4} + K_2 \frac{n}{2} + 1 + K_1 n + K_2 + 1$$

$$T(n) = 2K_1 \frac{n^2}{4} + 2K_2 \frac{n}{2} + K_1 n + K_2 + 4$$

$$T(n) = K_1 \frac{n^2}{2} + K_2 n + K_1 n + K_2 + 4$$

$$T(n) = K_1 \frac{n^2}{2} + (K_2 + K_1) n + K_2 + 4$$

• Discussion:-

- Worst case complexity of div_sol function is $O(n^2)$
- Div-sol function in worst case grows quadratically.
- means that, maximum time which a code can take