DESIGN AND ANALYSIS OF ALGORITHM

ASSIGNMENT

NAME : VIJAYALAKSHMI BAI.K

REGI NO: 192321031

COURSE CODE : CSA0669.

1) If $t_1(n) \in O(g_1(n))$ and $O(g_2(n))$, then $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$. Prove the assertions.

i) $t_1(n) \in O(g_1(n))$ means there exist constants $c_1 > 0$ and n_1 such that for all $n \geq n_1$.

$$t_1(n) \leq c_1 g_1(n)$$

ii) $t_2(n) \in O(g_2(n))$ means there exist constant $c_2 > 0$ and n_2 such that for all $n \geq n_2$.

$$t_2(n) \leq c_2 g_2(n)$$

we need to show that $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$.

Proof:

consider: $t_1(n) + t_2(n)$ and the definition of $\max\{g_1(n), g_2(n)\}$

let $g(n) = \max\{g_1(n), g_2(n)\}$

By definition of maximum, for all n:

$g(n) = \max\{g_1(n), g_2(n)\} \geq g_1(n)$

$g(n) = \max\{g_1(n), g_2(n)\} \geq g_2(n)$

given the bounds of $t_1(n)$ and $t_2(n)$:

$t_1(n) \leq c_1 g_1(n) \leq c_1 g(n)$ for all $n \geq n_1$

$t_2(n) \leq c_2 g_2(n) \leq c_2 g(n)$ for all $n \geq n_2$

to bound $t_1(n) + t_2(n)$, consider.

$t_1(n) + t_2(n)$

For $n \geq \max(n_1, n_2)$:

$t_1(n) + t_2(n) \leq c_1 g(n) + c_2 g(n) = (c_1 + c_2) g(n)$

Thus, there exists a constant $c = c_1 + c_2$ and $n_0 = \max(n_1, n_2)$ such that for all $n \geq n_0$: $t_1(n) + t_2(n) \leq g(n)$

By the definition of Big-O notation, this implies:

$t_1(n) + t_2(n) \in O(g(n))$

Since $g(n) = \max\{g_1(n), g_2(n)\}$, we have

$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

Hence, we have proven the assertion.

2. Find the time complexity of the below recurrance equations.

i) $T(n) = \begin{cases} 2T(n/2) + 1 & \text{if } n > 1 \\ 2T(n/2) + 1 & \text{if } n > 1 \end{cases}$

we can use Master's theorem for divide & conquer recurrance of the form.

$T(n) = aT(n/b) + f(n)$ where $a = 2$, $b = 2$, $f(n) = 1$

$\log_a^b = \log_2^2 = 1$

$f(n) = 1 \Rightarrow n^1 \log_a^b = n^1$

$f(n) = 1 = O(n^0)$ and $0 < 1 \Rightarrow$ case i:

$f(n) = O(n^0)$ and $0 < \log_2^2$

$T(n) = \Theta(n^k \log_a^b) = \Theta(n^1) = \Theta(n)$

$\therefore T(n) = \Theta(n)$

ii) $T(n) = \begin{cases} 2T(n-1) & \text{if } n > 0 \\ 1 & \end{cases}$

$T(n) = 2T(n-1)$ if $n > 0$

By unrolling the recurrance

$T(n) = 2T(n-1)$

$T(n-1) = 2T(n-2)$

$T(n-2) = 2T(n-3)$

continuing this, we see

$T(n) = 2T(n-1) = 2 \cdot 2T(n-2) = 2^2 \cdot 2T(n-3)$

$= 2^3 T(n-3)$

In general, after (k) steps, we have:

$T(n) = 2^k T(n-k)$

Base case, when $k = n$

$T(n) = 2^n T(0)$

$T(0) = 1 \Rightarrow T(n) = 2^n(1) = 2^n$

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

5) Big O notation : show that $f(n) = n^2 + 3n + 5$ is $O(n^2)$

Big O $\Rightarrow f(n) \leq c \cdot g(n)$

$f(n) = n^2 + 3n + 5$

Let us assume $g(n) = g x^2$

$= n^2 + 3n + 5 \leq c \cdot g(n)^2$

$\Rightarrow 1 + 3/n + 5/n^2 \leq c$

when $n = 1 \Rightarrow 1^2 + 3 + 5 = g(1)^2$

$= 9$

$3/n \leq 3$ for all $n \geq 1$

$5/n^2 \leq 5$ for all $n \geq 1$

Hence for $n \geq 1$:

$1 + 3/n + 5/n^2 \leq 1 + 3 + 5 = 9$

$\therefore c = 9$

when $n = 2$

$= 2^2 + 3(2) + 5 = g(4)$

$= 4 + 6 + 5 = 36 \Rightarrow 15 < 36$

when $n = 3$

$= g(g)$

$= 3^2 + 3(3) + 5 \Rightarrow g + g + 5 = 81$

$= 23 < 81$

$\therefore f(x) \leq c \cdot g(x)$

$\therefore$ Big (O) is satisfied

6) Big omega notation: Prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$

We have to demonstrate that constant $c > 0$ and $n_0 > 0$ such that for all $n \geq n_0$.

$$g(n) \geq c \cdot n^3$$

$$n^3 + 2n^2 + 4n \geq c \cdot n^3$$

dividing n^3 on B.S

$$1 + 2/n + 4/n^2 \geq c$$

As n grows larger, $2/n$ & $4/n^2$ becomes smaller.

for all $n \geq 1$: $2/n > 0$

$$4/n^2 > 0$$

Hence for $n \geq 1$:

$$1 + 2/n + 4/n^2 \geq 1$$

$\therefore c = 1$, for $n \geq 1$:

$$1 + 2/n + 4/n^2 \geq 1$$

We have shown that for $c = 1$ & $n_0 = 1$ the equality holds : $n^3 + 2n^2 + 4n \geq n^3$ for all $n \geq 1$

$\therefore g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$ with $c = 1$ and $n_0 = 1$

$\therefore$ Hence proved

7. Big theta notation: Determine whether $h(n) = 4n^2 + 3n$ is $\Theta(n^2)$ or not

To show $h(n) = 4n^2 + 3n$ is $\Theta(n^2)$, we need to find constants $c_1 > 0$ and $n_1 \geq 0$ such that for all $n \geq n_1$.

$$h(n) \leq c_1 \cdot n^2$$

$$4n^2 + 3n \leq c_1 \cdot n^2$$

Divide both sides by n^2

$$4 + 3/n \leq c_1$$

As n grows larger, $3/n$ become smaller for $n \geq 1$:

$$4 + 3/n \leq 4 + 3 = 7$$

$\therefore$ choose $c_1 = 7$ & $n_1 = 1$ $\because n \geq 1$

$$4n^2 + 3n \leq 7n^2$$

$\therefore h(n) = 4n^2 + 3n$ is $\Omega(n^2$

$\therefore$ find constants $c_2 > 0$ & $n_2 \geq 0$ for all $n \geq n_2$:

$$h(n) \geq c_2 \cdot n^2$$

$$4n^2 + 3n \geq c_2 \cdot n^2$$

$$4 + 3/n \geq c_2 \quad \text{(divide by } n^2\text{)}$$

n grows larger, $3/n$ become smaller for $n \geq 1$

$$4 + 3/n \geq 4$$

$\therefore c_2 = 4$ & $n_2 = 1$ $n \geq 1$

$4 + 3|n \geq 4$

$\therefore c_2 = 4 \quad \& \quad n_2 = 1 \quad n \geq 1$

$4n^2 + 3n \geq 4n^2$

This shows $h(n) = 4n^2 + 3n$ is $\Omega(n^2)$

$\therefore h(n) = 4n^2 + 3n$ is both $O(n^2)$ & $\Omega(n^2)$ we conclude

$h(n) = 4n^2 + 3n = \Theta(n^2)$

2) Let $f(n) = n^3 - 2n^2 + n$ and $g(n) = n^2$ show whether $f(n) = \Omega g(n)$ is true or false and justify your answer

To determine whether $f(n) = n^3 - 2n^2 + n$ is $\Omega(g(n))$ where $g(n) = n^2$, we need to show that there exist $c > 0$ and $n_0 \geq 0$ such that $n \geq n_0$

$f(n) \geq c \cdot g(n)$

$n^3 - 2n^2 + n \geq c \cdot (-n^2) \Rightarrow n^3 - 2n^2 + n \geq -cn^2$

$n^3 + (c-2)n^2 + n \geq 0$

consider $c = 3$

$n^3 + (3-2)n^2 + n = n^3 + n^2 + n$

For all $n \geq 1$:

$n^3 + n^2 + n \geq 0$

$\therefore n^3, n^2$ and n are all +ve for $n \geq 1$. inequality holds true.

Thus $c = 3$ and $n_0 = 1$

$f(n) : n^3 - 2n^2 + n \geq 3\ (\cdot n^3) = \cdot 3n^2$

$f(n) \geq c \cdot g(n)$ for all $n > n_0$ we conclude

$f(n) : n^3 - 2n^2 + n$ is $\Omega(n^2)$

$\therefore$ the statement $f(n) = \Omega(g(n))$ is true.

9) Determine whether $h(n) = n\log n + n$ is in $\Theta(n\log n)$ Prove a rigorous proof for your conclusion.

$h(n) = n\log n + n$ is $O(n\log n)$ we need to find constant $c_1 > 0$ and $n_1 \geq 0$ such that for all $n \geq n_1$:

$h(n) \leq c_1 \cdot n\log n$

let's consider the expression $h(n) : n\log n + n$:

$n\log n + n \leq c_1 \cdot n\log n$.

we can factor out $n\log n$:

$n\log n + n = n(\log n + 1)$

So we need:

$n(\log n + 1) \leq c_1 \cdot n\log n$.

Divide both sides by n (for $n > 0$):

$\log n + 1 \leq c_1 \cdot \log n$.

Now, we can find c_1:

$\log n + 1 \leq c_1 \cdot \log n$.

Divide both sides by $\log n$ (for $\log n > 0$):

$$1 + \frac{1}{\log n} \leq c_1$$

As n grows larger, $\frac{1}{\log n}$ Become smaller ∴ for sufficiently large n, $\frac{1}{\log n} \leq 1$ eg thus:

$$1 + \frac{1}{\log n} \leq 2.$$

Hence, we can choose $c_1 = 2$ and $n_1 \geq 2$ (to ensure $\log n \geq 1$)

this shows that $n\log n + n \leq 2n\log n$. Thus $h(n) = n\log n + n$

is $O(n\log n)$

Proof for $\Omega(n\log n)$:

To show that $h(n) = n\log n + n$ is $\Omega(n\log n)$, we need to find constants $c_2 > 0$ and $n_2 \geq 0$ such that for all $n \geq n_2$:

$$h(n) \geq c_2 \cdot n\log n$$

Consider the expression $h(n) = n\log n + n$.

$$n\log n + n \geq c_2 \cdot n\log n.$$

We can factor out $n\log n$:

$$n\log n + n = n(\log n + 1)$$

$$n\log(n+1) \geq c_2 \cdot n\log n.$$

÷ n on b.s (for $n \geq 0$):

$$\log n + 1 \geq c_2 \log n.$$

Now, we can find c_2.

$$\log n + 1 \geq c_2 \log n.$$

÷ by log n on b.s.

$$1 + \frac{1}{\log n} \geq c_2$$

As n grows larger, $\frac{1}{\log n}$ become smaller ∴ for sufficiently large, $n \cdot \frac{1}{\log n} \leq \frac{1}{2}$ and thus,

$$1 + \frac{1}{\log n} \geq \frac{3}{2}$$

Hence, we choose $c_2 = \frac{3}{2}$ & $n_2 \geq e^2$, this shows

$$n \log n + n \geq \frac{3}{2} n \log n.$$

Thus, $h(n) = n \log n + n$ is $\Omega(n \log n)$

∵ we have shown that $h(n) = n \log n + n$ is both $O(n \log n)$ and $\Omega(n \log n)$. we conclude that $h(n) = n \log n + n$ is $\Theta(n \log n)$

10. Solve the following recurrence relation and find the order of growth for solution

$$T(n) = 4T(n/2) + n^2, \quad T(1) = 1$$

$T(n) = aT(n/b) + f(n)$ where $a = 11, b = 2$, $f(n) = n^2$

$f(n) = n^k \log_n^p$

$= n^2 \log_n^p$

$\log_a^b = \log_n^2 = 2$

$f(n) = n^2$

$f(n) = \theta(n^2)$

If $f(n) = \theta(n^k \log_a^b)$

$T(n) = \theta(n^k \log(\log n))$

Thus, $T(n) = \theta(n^2 \log n)$

1. Given an array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] Integers, find the maxi and mini product that can be obtained by multiplying 2 Integer from the array.

Sort the array:

[-9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

Candidate for maxi product:

→ $10 \times 11 = 110$

→ $-9 \times -8 = 72$

For min Product

⇒ [illegible] = [illegible]

⇒ [illegible] = -99

⇒ [illegible] = [illegible]

Max = [illegible]

Min = -99

12 Demonstrate Binary Search method to search key 23 from the array arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}

Pseudocode

```
def bs (arr, key):
    start = 0
    end = len(arr) - 1
    while ( start <= end ):
        mid = (start + end) // 2
        if (arr[mid] == key):
            return mid
        elif arr[mid] < key
            start = mid + 1
        else:
            end = mid - 1
return -1
```

8) start = 0 , end = 9.

mid = 4

16 < 23 , so search in the right half.

i) 23, 38, 56, 72, 91.

start = 5 end = 9

mid = 7

56 > 23 , search in the left half.

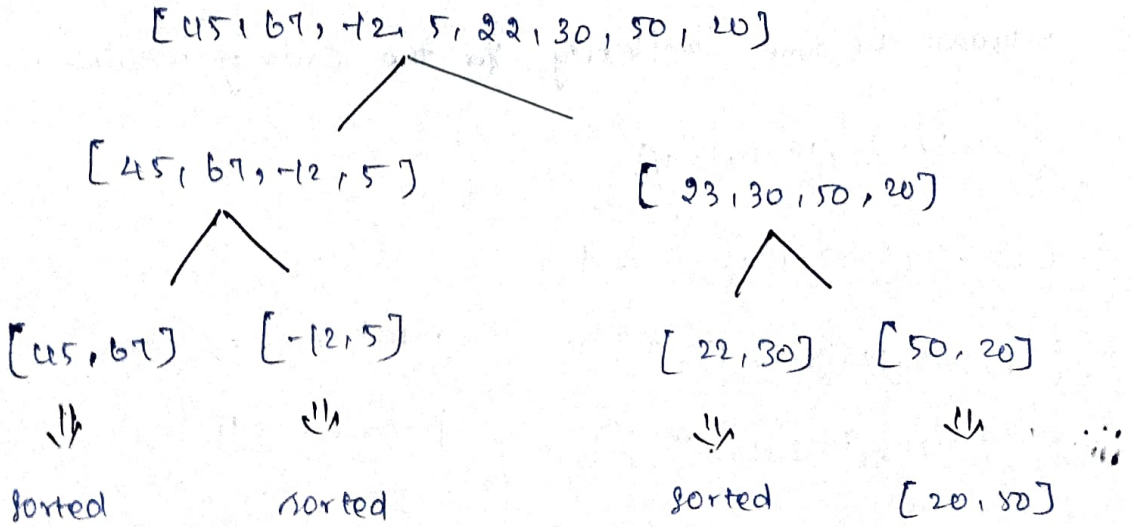
ii) 23, 38

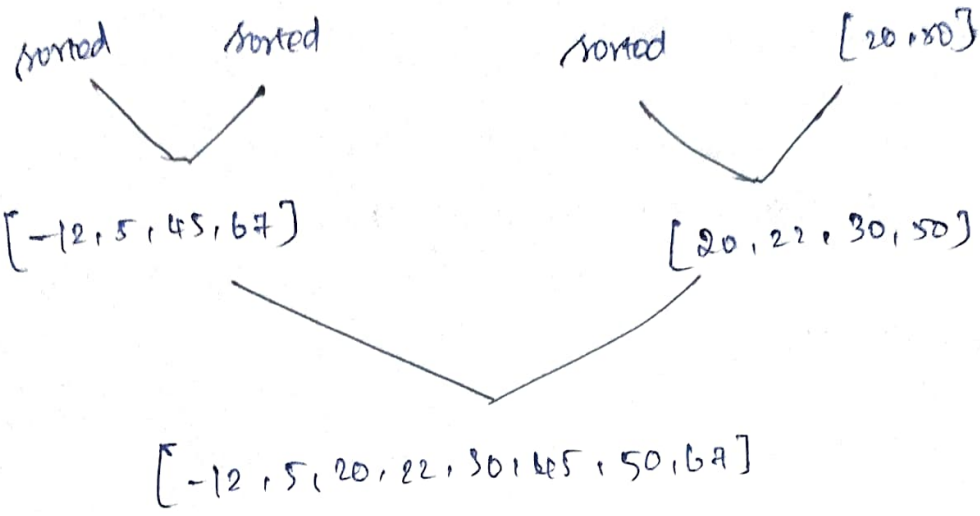
start = 5 end = 6

mid = 5

23 = 23 key found at index 5

13) Apply merge sort and order the list of 8 element, a = (45, 67, -12, 5, 22, 30, 50, 20) . Set up a recurrence relation for the no of key comparison made by merge sort





Recurrance relation

$$T(n) = 2T(n/2) + n.$$

$$\log_a^b = \log_2^2 = 1$$

$$f(n) : n = n^k \Rightarrow k = 1, p = 1$$

$$\log_a^b = k \Rightarrow \text{case(i)} : p > -1 : \Theta(n^k (\log_n^{p+1}))$$

$$T(n) = \Theta(n \log n)$$

4) Find the no of times to perform swapping for selection sort Also estimate the time complexity for the order of notation sort

S: (12, 7, 5, -2, 18, 6, 13, 4).

i) [-2, 7, 5, 12, 18, 6, 13, 4]

ii) [-2, 4, 5, 12, 18, 6, 13, 7]

iii) [-2, 4, 5, 6, 18, 12, 13, 7]

iv) [-2, 4, 5, 6, 7, 12, 13, 18]

Total no of swaps performed = 4.

Best case for selection sort involves $O(n^2)$ comparison worst case also involves $O(n^2)$ comparison and $O(n)$ swaps. Average case also involves $O(n^2)$ comparison and $O(n)$ swaps.

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

15) find the index of the target value 10 using binary search from the following list of [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

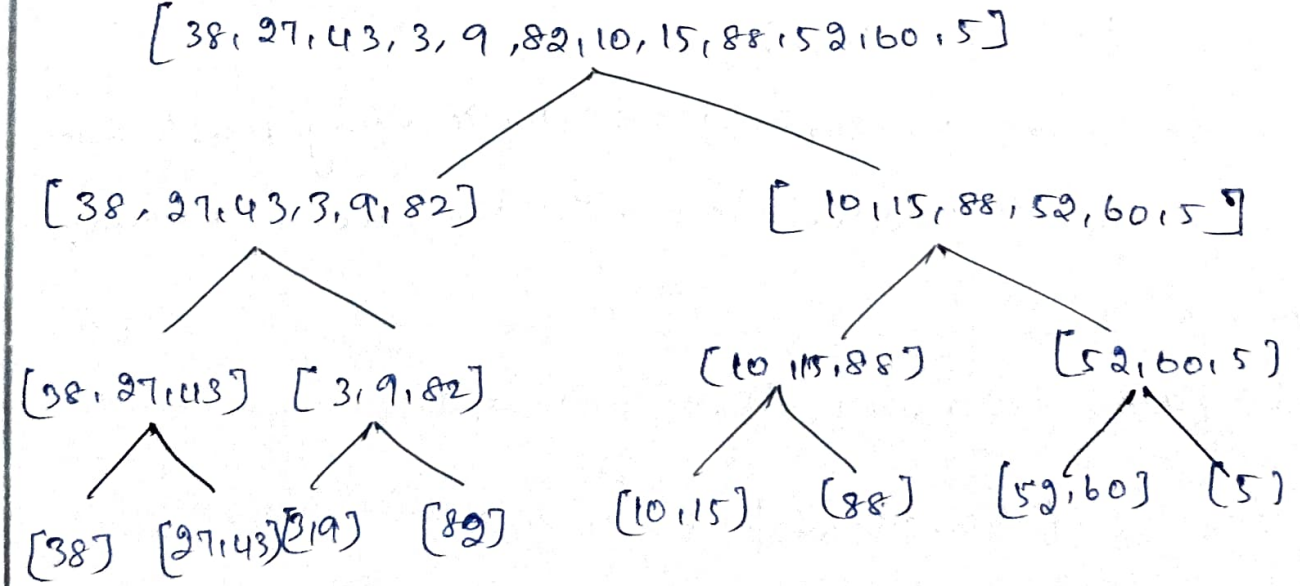
Initially → start = 0, end = 9.

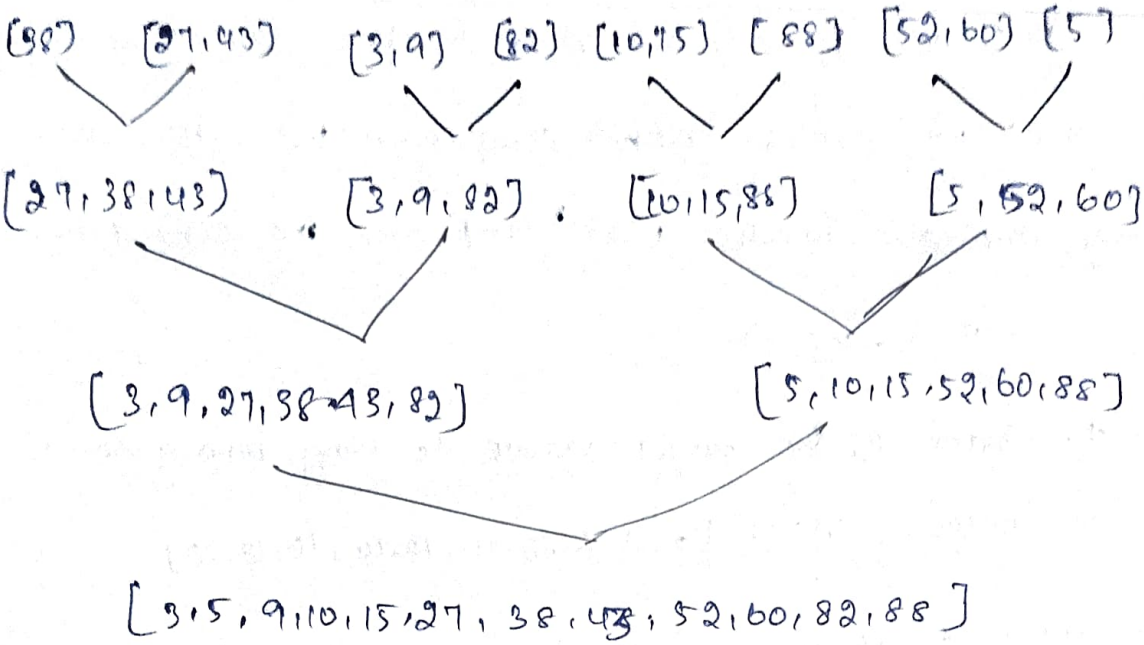
$$\text{mid} = \frac{\text{start} + \text{end}}{2} = \frac{0+9}{2} = 4.$$

list[mid] = list[4] = 10

Target value is at index 4.

16) Sort the following elements using Merge sort divide and Conquer strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyse complexity of algorithm.





Recurrence Relation

$$T(n) = 2T(n/2) + n$$

$$\log_a^b = \log_2^2 = 1$$

$$f(n) = n \Rightarrow n^k \Rightarrow k=1, \; p=1$$

$$\log_a^b = k \Rightarrow \text{case 1}$$

$$p > -1 : \Theta(n^k \log_n^{p+1})$$

$$T(n) = \Theta(n \log n)$$

(7) Sort the array 64, 34, 25, 12, 22, 11, 90 Using bubble sort what is TC of Selection sort in the best, worst & average case?

1st pass [34, 64, 25, 12, 22, 11, 90]

[34, 25, 64, 12, 22, 11, 90]

[34, 25, 12, 64, 22, 11, 90]

[34, 25, 12, 22, 64, 11, 90]

[34, 25, 12, 22, 11, 64, 90]

2nd pass [25, 34, 12, 22, 11, 64, 90]

[25, 12, 34, 22, 11, 64, 90]

[25, 12, 22, 34, 11, 64, 90]

[25, 12, 22, 11, 34, 64, 90]

3rd pass [12, 25, 22, 11, 34, 64, 90]

[12, 22, 25, 11, 34, 64, 90]

[12, 22, 11, 25, 34, 64, 90]

4rd pass [12, 11, 22, 25, 34, 64, 90]

5th pass [11, 12, 22, 25, 34, 64, 90]

Best case : $O(n^2)$

Worst case : $O(n^2)$

Avg case : $O(n^2)$

(8) Sort the array 64, 25, 12, 22, 11 using selection sort. What is the time complexity of selection in the best, worst and average cases?

1st pass: [11, 25, 12, 22, 64]

2nd pass: [11, 12, 25, 22, 64]

3rd pass: [11, 12, 22, 25, 64]

4th pass: 25 & 64 is swapped.

Sorted: [11, 12, 22, 25, 64]

Best case: $O(n^2)$

worst case: $O(n^2)$

Average case: $O(n^2)$

(9) Sort the following elements using Insertion sort using Brute force Approach strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and Analyze complexity of the algor.

[27, 38, 43, 3, 9, 82, 10, 15, 88, 5, 2, 60, 5]

[27, 38, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5]

[3, 27, 38, 43, 9, 82, 10, 15, 88, 82, 60, 5]

[3, 9, 27, 38, 43, 82, 10, 15, 88, 52, 60, 5]

[3, 9, 10, 27, 38, 43, 82, 15, 88, 52, 60, 5]

[3, 9, 10, 15, 27, 38, 43, 52, 82, 88, 60, 5]

[3, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88, 5]

[3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88]

Best case : $O(n)$

Worst case : $O(n^2)$

Average case : $O(n^2)$

Give an array [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers. Sort the following elements Using insertion sort Using Brute force approach Strategy analyse complexity of the algorithm

a) [-2, 4, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9]

[-2, 3, 4, 5, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9]

[-3, 2, 3, 4, 5, 10, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9]

[-5, -2, 2, 3, 4, 5, 10, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9]

[-5, -2, 2, 3, 4, 5, 8, 10, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9]

[-5, -3, -2, 2, 3, 4, 5, 8, 10, 6, 7, -4, 1, 9, -1, 0, -6, -8, -11, 9]

[-5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, -6, -8, -11, -9]

[-9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Worst case : $O(n^2)$

Best case : $O(n)$

Average case : $O(n^2)$