CSA0669 - Design and Analysis of Algorithm

Analytical Question-2

By
R. Jagan
192321102
B Tech IT.

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

For any four arbitrary real numbers a, b, a_2, b_2 such that $a \le b$ and $a_2 \le b_2$

we have $a_1 + a_2 \le 2 \max\{b_1, b_2\}$

Since $t_1(n) \in O(g_1(n))$, then there exists some constant c_1 and non-negative integer n_1 such that

$t_1(n) \le c_1 g_1(n)$ for all $n \ge n_1$

Since $t_2(n) \in O(g_2(n))$, then there exists some constant c_2 and non-negative integer n_2 such that

$t_2(n) \le c_2 g_2(n)$ for all $n \ge n_2$

Let $c_3 = \max\{c_1, c_2\}$ and $n_0 = \max\{n_1, n_2\}$

$$\begin{aligned} t_1(n) + t_2(n) &\le c_1 g_1(n) + c_2 g_2(n) \\ &\le c_3 g_1(n) + c_3 g_2(n) \\ &= c_3\{g_1(n) + g_2(n)\} \\ &\le c_3 \max\{g_1(n), g_2(n)\} \end{aligned}$$

Hence $t_1(n) + t_2(n) \le O(\max\{g_1(n), g_2(n)\})$, with constants c and n_0 required by the O definition being $2c_3 = 2\max\{c_1, c_2\}$ and $\max\{n_1, n_2\}$ respectively

2. Find the Time Complexity of the below recurrence relation.

3. $$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 1 & \text{if } n > 1 \\ 1 & \text{otherwise.} \end{cases}$$

Master theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$a = 2$

$b = 2$

$\log_b^a = \log_2^2 = 1$

$K = 0 \qquad 1 > 0$

$\log_b^a > K$

Case 1

$\theta(n \cdot \log_b^a)$

$\theta(n \cdot 1)$

$\theta(n)$

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

Backward substitution :

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

$n = n-1$

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

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

Sub (2) in (1)

$$T(n) = 2[2T(n-2)]$$

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

$n = n-2$

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

$$= 2T(n-3) \rightarrow (4)$$

Sub (4) in (3)

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

$$T(n) = 2^3 T(n-3) \rightarrow (5)$$

$n = n-3$

$$T(n-3) = 2T[(n-3)-1)$$

$$= 2T(n-4) \rightarrow (6)$$

Sub (6) in (5)

$$T(n) = 2^3 [2T(n-4)]$$

$$= 2^4 T(n-4) \rightarrow (7)$$

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

$n - k = 0$

$\boxed{n = k}$

if $T(0) = 1$

$$T(n) = 2^k \cdot T(0)$$

$$T(n) = 2^k \cdot 1$$

$$n = k$$

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

5. Big O Notation : Show that $f(n) = n^2 + 3n + 5$ is $O(n^2)$

To prove that $f(n) = n^2 + 3n + 5$ is $O(n^2)$

we need to find constants c and n_0 such that

$$f(n) \leq c \cdot n^2 \text{ for all } n \geq n_0$$

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

For $n \geq 1$, $n^2 \geq n$... So on

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

$$f(n) = n^2 + 3n + 5 \leq 9n^2 \text{ for } n \geq 1$$

So, for $c = 9$ and $n_0 = 1$.

$$f(n) \leq c \cdot n^2 \text{ for all } n \geq n_0$$

That proves

$$f(n) \text{ is } O(n^2)$$

6. Big Omega Notation :-

prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$

To prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$ we need to find constans c and n_0 such that

$g(n) \geq c.n^3$ for all $n \geq n_0$

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

for $n \geq 1$,

$g(n) = n^3 + 2n^2 + 4n \geq n^3$

Since $2n^2$ and $4n$ are both less than n^3 when $n \geq 1$

So, for $c = 1$ and $n_0 = 1$

$g(n) \geq c.n^3$ for all $n \geq n_0$

that proves $g(n)$ is $\Omega(n^3)$

7. Big Theta Notations :-

Determine whether $h(n) = 4n^2 + 3n$ is $\theta(n^2)$ or not.

1. $h(n) = 4n^2 + 3n$ is $O(n^2)$

For $n \geq 1$, $h(n) \leq 4n^2 + 3n^2$

(since $3n$ is less than n^2 when $n \geq 1$)

For this simplifies to $h(n) \leq 7n^2$

for $n \geq 1$

Therefore, $h(n)$ is $O(n^2)$

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

For $n \geq 1$, $h(n) \geq 4n^2$

$h(n)$ is $\Omega(n^2)$ (since $3n$ is positive)

it is $\theta(n^2)$

8 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.

$n = 1$

$f(1) = 1^3 - 2(1)^2 + 1$
$= 1 - 2 + 1$
$= 0$

$g(1) = (-n)^2$
$= -1^2$
$= 1$

$n = 2$

$f(2) = 2^3 - 2(2)^2 + 2$
$= 8 - 8 + 1$
$= 2$

$g(2) = (-2)^2$
$= 4$

$n = 3$

$f(3) = 2^3(3)^2 + 3$
$= 27 - 9 + 3$
$= 21$

$g(3) = (-3)^2$
$= 9$

$n = 4$

$f(4) = 4^3 - 2(4)^2 + 4$
$= 64 - 32 + 4$
$= 36$

$g(n) = (-4)^2$
$= 16$

$n = 5$

$f(5) = 5^3 - 2(5)^2 + 5$
$= 40$

$g(n) = (-5)^2$
$= 25$

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

So it is best case according to asymptotic notation

$f(n) = \Omega(g(n))$.

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

1. Upper bound (O notation):

we need to find c_1 and n_0 such that

$h(n) \leq c_1 \cdot n \log n$ for all $n \geq n_0$

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

$= n \log n + n \log n$ (since $\log n$ is increasing)

$= 2n \log n$

Now Let $c_1 = 2$, then $h(n) \leq 2n \log n$ for all $n \geq 1$

so, $h(n)$ is $O(n \log n)$

2. Lower bound (Ω notation):

we need to find c_2 and n_0 such that

$h(n) \geq c_2 \cdot n \log n$ for all $n \geq n_0$

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

$\geq \frac{1}{2} \cdot n \log n$ (for $n \geq 2$)

let $c_2 = \frac{1}{2}$ then $h(n) \geq \frac{1}{2} \cdot n \log n$

for all $n \geq 2$, so $h(n)$ is $\Omega(n \log n)$

3. Combining Bounds:

since $h(n)$ is both $O(n \log n)$ and $\Omega(n \log n)$.

it is also $O(n \log n)$

Thus $h(n) = n \log n + n$ is in $\Theta(n \log n)$

10. Solve the Following recurrence relations and find the order of growth for solutions

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

$T(n) = aT(n/b) + f(n)$

$a = 4$

$b = 2$ $\log_b^a = \log_4^2 = 2$

$K = 2$

$2 = 2$

$\log_b^a = K$

Case 2

$p > -1$ $\Theta(n^K \log n^{p+1})$

$\Theta(n^2 . \log n^{1+1})$

$\Theta(n^2 . \log_n^2)$

$T(n) = \Theta(n^2 \log(n))$

The order of growth for the solution is $n^2 . \log(n)$.

11. 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 maximum and minimum product that can be obtained by multiplying two integers 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]

Candiates for maximum product

$10 \times 11 = 110$

$-9 \times -8 = 72$

Candiates for minimum product

$-9 \times -8 = +72$

$-9 \times 11 = -99$

$-8 \times 11 = -88$

Max. product = 110

Min product = -99

12. Demonstrate Binary Search Method to search key = 23

Form the array arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91

low = 0

high = 9

mid = $\frac{low + high}{2}$

= $\frac{0+9}{2}$ = 4

a[mid] == key

a[4] = 23

16 != 23

16 < 23

low = mid + 1

5	6	7	8	9
23	38	56	72	91

low = 5 high = 9

mid = $\frac{5+9}{2} = \frac{14}{2} = 7$

a[mid] == key

a[7] == 23

56 != 23

56 > 23

high = mid - 1

5	6	7
23	38	56

l = 5 h = 7

m = $\frac{5+7}{2}$ = 6

a[6] == 23

38 != 23

38 > 23

h = mid - 1

5

| 23 |

L = 5 h = 5

mid = $\frac{5+5}{2}$ = 5

a [5] == key

23 == 23

Return the position of the Key 5

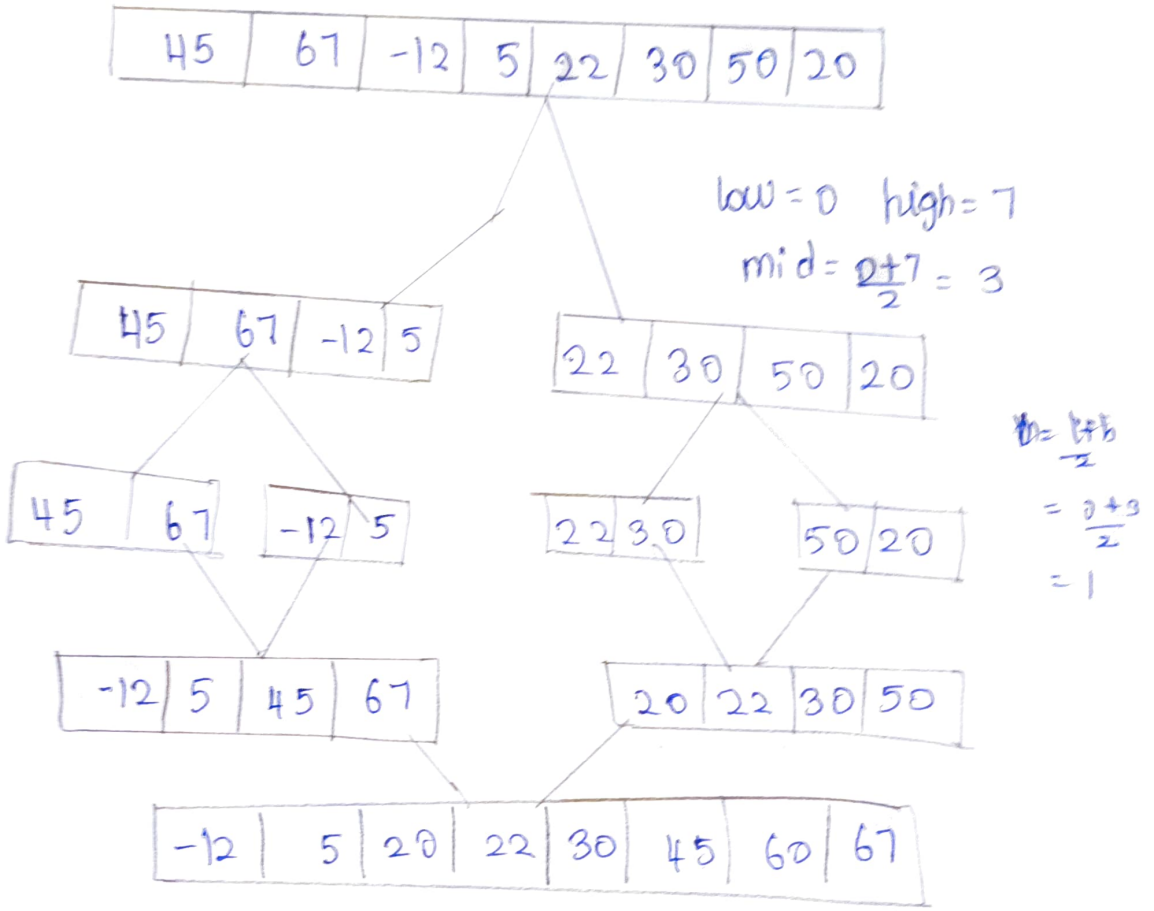
pseudocode ::

```
binary_search (a, n, key)
  low = 0
  high = n-1
while (low <= high);
    mid = (high + low) // 2
if   a [mid] == key
      return mid
     a [mid] > key
        high = mid - 1
     a [mid] < key
        low = mid + 1
return -1   [If not found]
```

Time Complexity ::

O(n log n).

13. Apply merge sort and order the list of 8 elements.
Data d= (45, 67, -12, 5, 22, 30, 50, 20) set up a recurrence relation for the number of key comparisons made by mergesort.



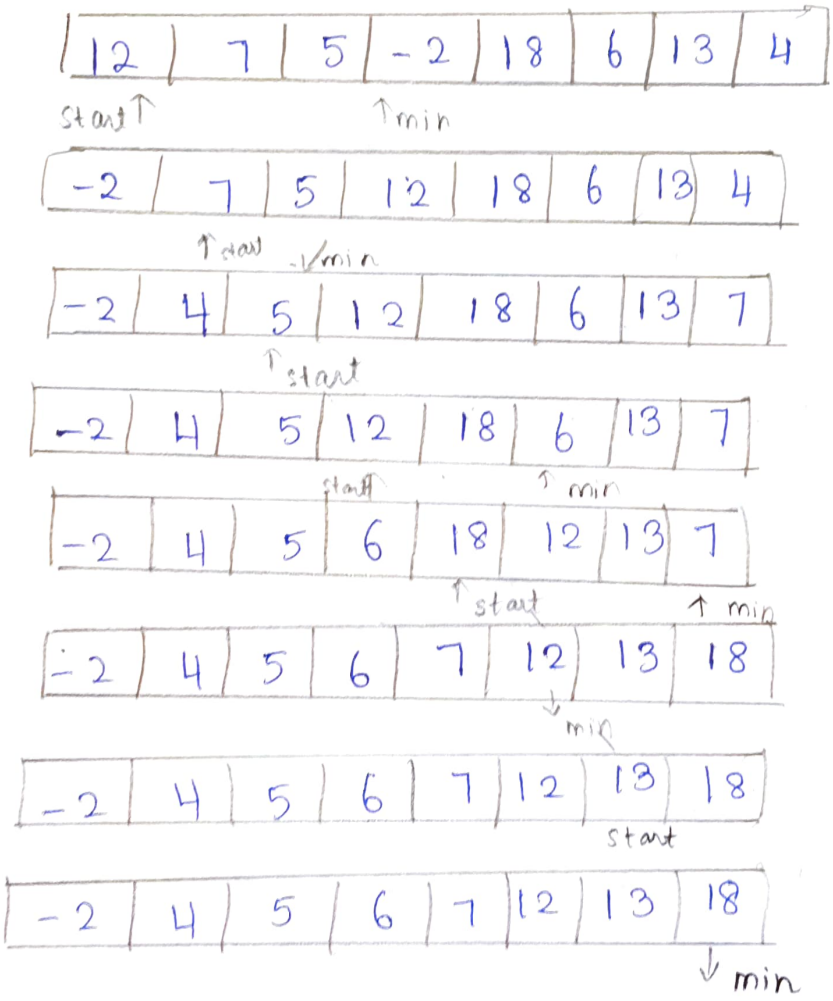
∴ The sorted list is

-12, 5, 10, 22, 30, 45, 60, 67

Time Complexity : $O(n \log n)$

Recurrence Relation : $T(n) = 2T(n/2) + (n-1)$.

14. Find the no. of times to perform swapping for selection sort. Also estimate the time complexity for the order of notation set S(12 7 5 -2 18 6 13 4).



Sorted list :- -2, 4, 5, 6, 7, 12, 13, 8

usually, the number of swaps required will be n-1. But for this question there only 4 swaps

Time complexity :- $O(n^2)$.

It is n^2 in all three cases.

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

0	1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18	20

low = 0 high = 9

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

a [mid] == key

a [4] == 10

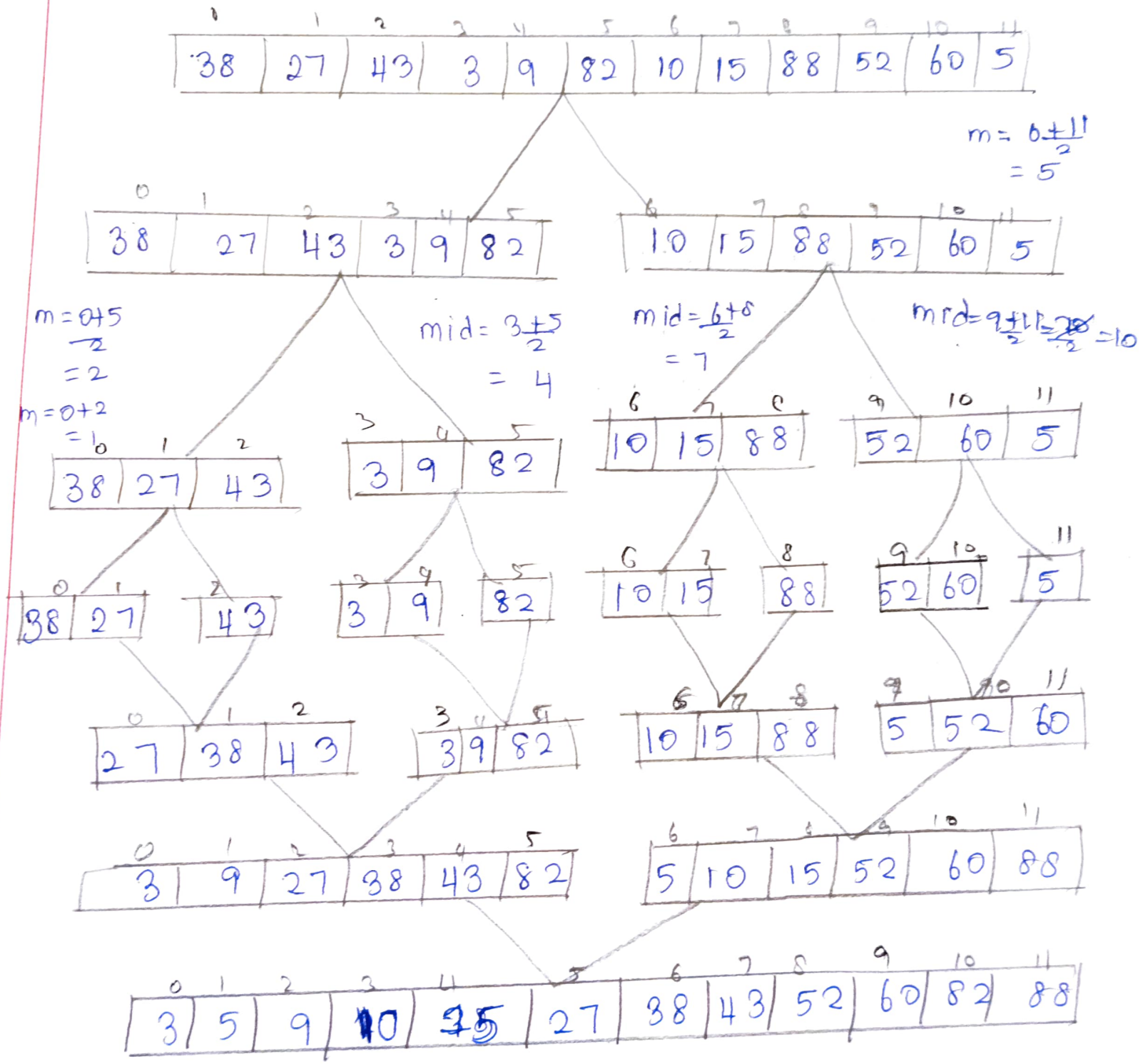
10 == 10

Return the position of the key 4.

pseudocode :-

```
binary_search (array, size of array, key)
    low = 0
    high = size - 1
while (low <= high)
    mid = (high + low)/2
if a [mid] == key
    return mid
a [mid] > key
    high = mid - 1
a [mid] < key
    low = mid + 1
return -1 (If not found)
```

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 analyze complexity of the algorithm.



∴ The Sorted list is 3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88.

17. Sort the array 64, 34, 25, 12, 22, 11, 90 using Bubble Sort. what is the time complexity of selection sort in best, worst and average cases?

pseudocode:

```
partition(low, high)
if l<h
mid = h+l/2
partition (l, mid)
partition (mid+1, h)
merge (l, mid, h)
end if
```

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

By using master theorem

$a = 2 \quad b = 2 \quad k = 1$

$\log_b^a = \log_2^2 = 1$

Comparing $\log_a^b = k$

$\log_a^b = k$

case 2 $p < -1$

$O(n^k \log^{p+1} n)$

$= O(n \log n)$

∴ Time Complexity $O(n \log n)$.

18. Sort the Array 64, 34, 25, 12, 22, 11 using selection sort what is the time complexity of selection sort in the best, worst and average cases?

64 34 25 12 22 11 90

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

34 25 12 22 11 64 90

phase I

---

34 25 12 22 11 64 90

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

25 12 22 11 34 64 90

phase II

---

25 12 22 11 34 64 90

12 25 22 11 34 64 90

12 22 25 11 34 64 90

12 22 11 25 34 64 90

12 22 11 25 34 64 90

phase III

12 22 11 25 34 64 90

12 22 11 25 34 64 90

12 11 22 25 34 64 90

12 11 22 25 34 64 90

phase IV

---

12 11 22 25 34 64 90

11 12 22 25 34 64 90

11 12 22 25 34 64 90

phase V

---

11 12 22 25 34 64 90

11 12 22 25 34 64 90

phase VI

∴ the sorted list 11, 12, 22, 25, 34, 64, 90.

Time Complexity $O(n^2)$

Selection sort :

Best Case : $O(n^2)$

Worst Case : $O(n^2)$

Average Case : $O(n^2)$

19. 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 algorithm.

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

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

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

27 38 ~~43~~ 43 9 82 10 15 88 52 60 5

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 9 10 15 27 38 43 52 60 82 88

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

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

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

∴ The Sorted list is

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

Pseudocode:

```
Insertion_sort(a, size of a)
    for i in range [2, size)          -> n-2
        key = a[i]                    -> 0
        j = i-1                       -> 1
        while j>0 and a[j]>key        -> n*(n-2)
            a[j+1] = a[j]             -> 0
            j -= 1                    -> 0
        a[j+1] = key                  -> 0
```

Time complexity:

$$T(n) = n*(n-2) + n-2 + 1 + 1 + 1 + \cdots + 1 + 1$$

$$= n^2 - 2n + n - 2 + 6$$

$$= n^2 - n + 4$$

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

Time complexity is $O(n^2)$.

20. 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. sort the following elements using insertion sort using Brute Force Approach strategy analyze Complexity of the algorithm.

Pseudocode

```
Insertion_sort (a, size of a)
  for i in range [2, size);
    key = a[i]
     j = j-1
  while j>0 and a[j] > key
    a[j+1] = a[j]
       j -= 1
  a[i+1] = key
   return a
```

Time complexity:

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