ALGO Assignment 01

Mohammad Usama
20K-0190
BCS-5B

Question 01: Insertion Sort

[6, 16, 12, 27, 9, 1, 18, 5, 31]

(A) Steps

1 [16, 6, 12, 27, 9, 1, 18, 5, 31] K = 16
2 [16, 12, 6, 27, 9, 1, 18, 5, 31] K = 12
3 [27, 16, 12, 6, 9, 1, 18, 5, 31] K = 27
4 [27, 16, 12, 9, 6, 1, 18, 5, 31] K = 9
5 [27, 16, 12, 9, 6, 1, 18, 5, 31] K = 1
6 [27, 18, 16, 12, 9, 6, 1, 5, 31] K = 18
7 [27, 18, 16, 12, 9, 6, 5, 1, 31] K = 5
8 [31, 27, 18, 16, 12, 9, 6, 5, 1] K = 31

(B)

⇒ Due to nested looping of For & while loop the time complexity is $O(n^2)$.

⇒ It's correctness can be proved by loop invariant.

⇒ In Insertion sort the SubArray A[1-j-1] is always in sorted order

⇒ So the initial state is always true.

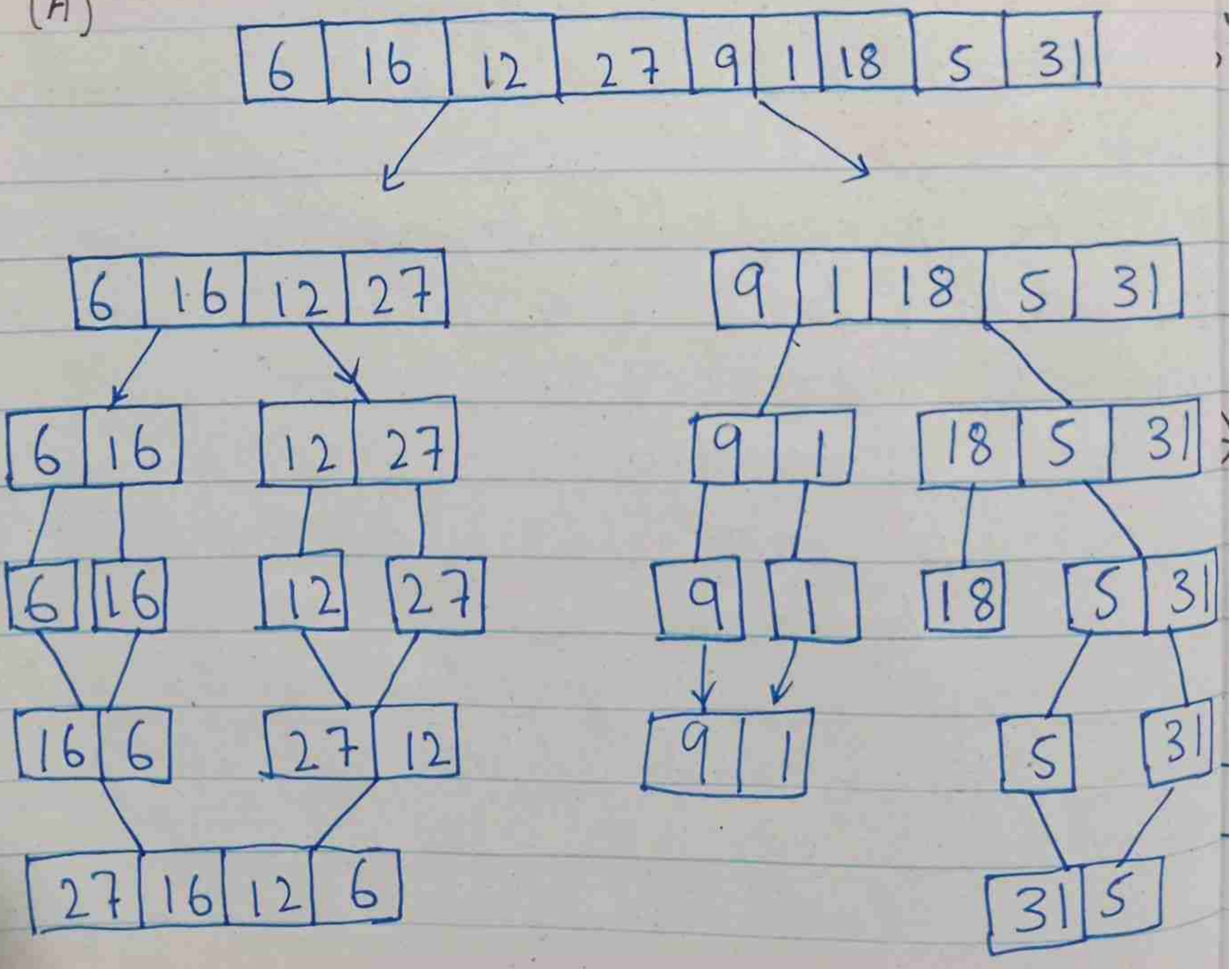
⇒ Loop invariant is true before n^{th} iteration then it will be true after $(n+1)^{th}$ iteration as well.

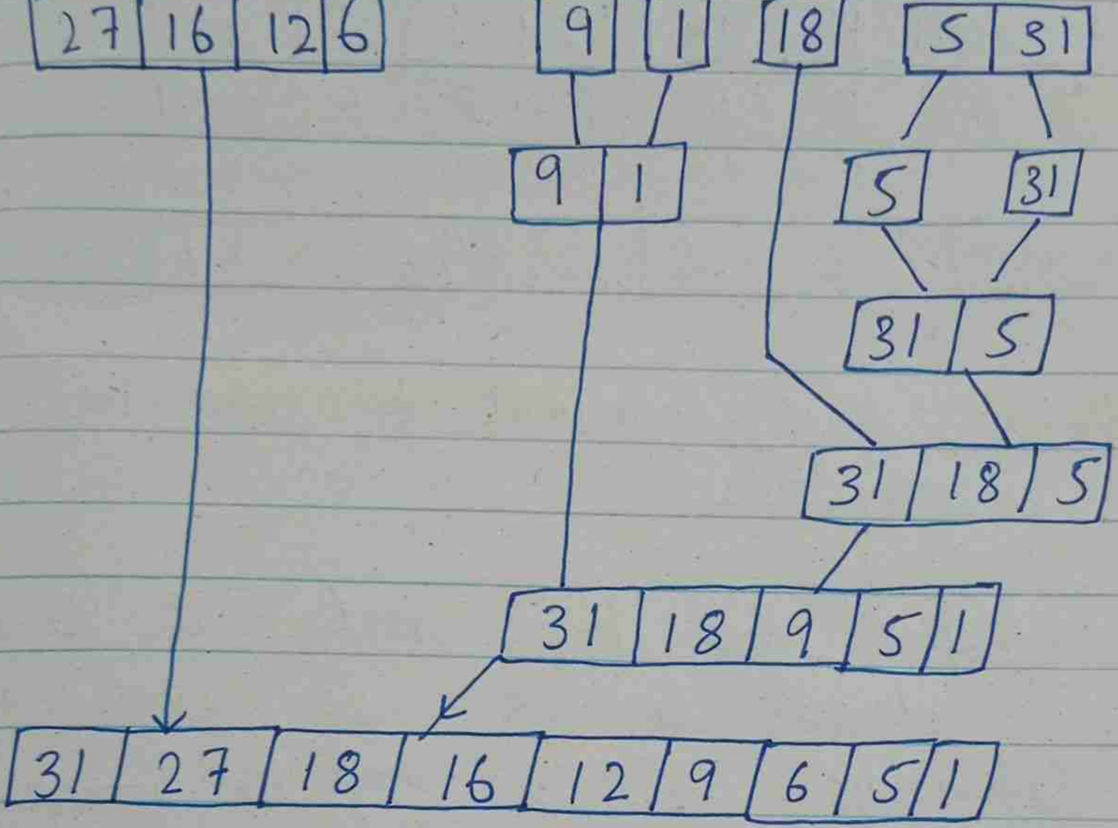
⇒ Termination of loop gives that algorithm is correct.

⇒ Since it's holds all three conditi[ons] we say that this algorithm is correct.

Questions 02 Merge Sort

(A)





) Rather than splitting in two parts if we split in three at each level so the time complexity will be $O(n \log_3 n)$.

uestion 03 Quick sort

6	16	12	27	9	1	18	5	31
		P						

16	27	18	31	12	6	9	1	5
P				P		P		

27	18	31	16	12	9	6	1	5
	P		P	P	P	P		

27	31	18	16	12	9	6	1	5
P	P	P	P	P	P	P		P

31	27	18	16	12	9	6	5	1

⇒ Using loop invariant to show it's correctness.

(B) ⇒ Before the loop starts the condi is satisfied, because the pivot an subarrays A[P-i] and A[i+1, —, j-1] are empty.

⇒ When the loop is running, if A[j] ≤ pivot, then A[j] and A[i+1] are swapped and then i and j are incremented.

If A[j] > pivot then only j increments.

Question 05

MinMaxSum (A)
$n = A.length/2$
Quicksort (A, 1, A.length) // sort A in $n \log_2 n$ times
$i = i+1$
For $j=1$ to $n/2$
Sum[i] = A[j] + A[n-j+1]
$i = i+1$
MaxSum = INT_MIN
For $j=1$ to n
if Sum[j] > maxsum
maxsum = Sum[j]
return maxsum

Question : 06

(A) Prove that $n^3 - 2n + 1 = O(n^3)$

$f(n) \le cn^3$

$= n^3 - 2n + 1 \le cn^3$

$= n^3 + 1n^3 \le cn^3$

$= 2n^3$

$c = 2$

Now if we check for $c=2$ and $n_0 = 1$

$$2(1)^3 \leq (2)(1)^3$$

Thus, this proves that $n^3 - 2n + 1 = O(n^3)$

for $c=2$ and $n_0 = 1$

(B) Prove that $5n^2 \log_2 n + 2n^2 = O(n^2 \log_2 n)$

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

$$5n^2 \log_2 n + 2n^2 \leq cn^2 \log_2 n$$

$$5n^2 \log_2 n + 2n^2 \leq 5n^2 \log_2 n + 2n^2 \log_2 n$$

$$\leq 7n^2 \log_2 n$$

$\boxed{c=7}$

for all $n \geq 2$

Hence proved $5n^2 \log_2 n + 2n^2 = O(n^2 \log_2 n)$

for $\boxed{c=7}$ and for all $n \geq 2$

$\boxed{n_0 = 2}$

Question 07:

To measure the efficiency of an algorithm we take account of the scalability and the time & space complexity to find the complexity we need to take account

of the input size.
The Big O, Ω, $\emptyset$ are used to retake growth of functions.

Big O:- The graph of $f(x)$ remains to right of $c \cdot g(x)$ after the point of x_0 we say that $f(x)$ is Big (O) of $g(x)$.

Big Ω :- The graph of $f(x)$ bounds from below the graph of $c \cdot g(x)$ we say that $f(x)$ is in Big Omega of $g(x)$ after the point x.

Big θ :- If the graph $f(x)$ bound $g(x)$ from above and below then we say that $f(x)$ is in Big Theta of $g(x)$ after the point x_0.

The Asymtotic bounds are the limit of the function.

Question 08

(A) $T(n) = 2T(n/3) + cn^2$

$a = 2$, $b = 3$ & $d = 2$

Since $a < b^d$

i.e: $2 < 3^2$

Applying 1st Condition:

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

$T(n) \in O(n^2)$

(B) $T(n) = 4T\left(\frac{n}{3}\right) + c.n$

$a = 4$, $b = 3$ & $d = 1$

Since $a > b^d$

i.e: $4 > 3^1$

So third Condition applies

$O(n^{\log_b a}) = O(n^{\log_3 4})$

$T(n) \in O(n^{\log_3 4})$

(C) $T(n) = 8T(n/2) + c.n^3$

$a = 8$, $b = 2$ & $d = 3$

Since $a = b^d$

ie, $8 = 2^3$

So the Second Condition applie

$O(n^d \log n) = O(n^3 \log n)$

$\boxed{T(n) \in O(n^3 \log n)}$

Question 09:

A)

$$T(n) = 2T(n/3) + n^2 \quad , \{T(1) = 1\}.$$

$\longrightarrow$ eq 1

$$T(n/3) = 2T(n/9) + n^2/9$$

put $T(n/3)$ in eq ①

$$T(n) = 2[2T(n/9) + n^2/9] + n^2$$

$$T(n) = 4T(n/9) + 2n^2/9 + n^2 \quad - ②$$

put $n = n/9$ in eq ①

$$T(n/9) = 2T(n/27) + n^2/81$$

Now put $T(n/9)$ in eq ②

$$T(n) = 4\cdot[2T(n/27) + n^2/81] + 2n^2/9 + n^2$$

$$= 8T(n/27) + 4n^2/81 + 2n^2/9 + n^2$$

$$= 8T(n/27) + n^2(4/81 + 2/9 + 1)$$

$$= 2^k T\left(\frac{n}{3^k}\right) + n^2\left(\frac{1-(2/9)^k}{7/9}\right)$$

$$T(1) \Rightarrow \frac{n}{3^k} = 1 \Rightarrow n = 3^k \Rightarrow k = \log_3 n$$

$$T(n) = 2^{\log_3 n} + \frac{n^2}{7/9} - \frac{(2/9)^{\log_3 n}}{7/9} n^2$$

Dominating term:- $9/7\, n^2$

$T(n) \in \theta(n^2)$

(B) $T(n) = 4T(n/3) + n, [T(1) = 1]$ — ①

$T(n/3) = 4T(n/9) + n/3$

Put in eq ①

$T(n) = 4[4T(n/9) + n/3] + n$
$T(n) = 16T(n/9) + 4n/3 + n$ — ②

~~T(n/8)~~ $T(n/9) = 4T(n/27) + n/9$
put $T(n/9)$ in eq ②

$T(n) = 16[4T(n/27) + n/8]$
$+ 4n/3 + n$

$= 64T(n/27) + 16n/9 + 4n/3 + n$
$= 64T(n/27) + n(1 + 4/3 + 16/9)$

$= 4^k T(n/3^k) + n\left(1 + \frac{4}{3} + \frac{16}{9} + \dots \frac{4^{k-1}}{3^{k-1}}\right)$

$= 4^k T(n/3k) + n\left(\frac{1 - (4/3)^k}{1 - 4/3}\right)$

$4^k \; T(n/3^k) - 3n + 3n(4/3)^k$

$T(1) = 1 \Rightarrow k = \log_3 n$

$4^{\log_3 n} - 3n + 3n\left(\frac{4^{\log_3 n}}{3^{\log_3 n}}\right)$

$4^{\log_3 n} - 3n + 3n\left(\frac{4^{\log_3 n}}{n}\right) =$

$4^{\log_3 n} - 3n + 3(4^{\log_3 n})$

$4^{\log_3 n} - 3n + 3\left(4^{\frac{\log_4 n}{\log_4 3}}\right).$

$4^{\log_3 n} - 3n + 3(n^{1/\log_4 3})$

$T(n) = 4^{\log_3 n} - 3n + 3n^{\log_3 n}$

Dominating term $= 3n^{\log_3 4}$

$T(n) \in \Theta(n^{\log_3 4})$

Question 10

(A) $f(n) = O(g(n))$ and $f(n) = \Omega(h(n))$,

then $g(n) = h(n) = \Omega(f(n))$

True:

let $f(n) = n^2$, $g(n) = n^3$, $h(n) = n$

then $f(n) \in O(n^3)$ and $f(n) \in \Omega(n)$

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

$$n^3 + n \in \Omega(f(n)) \Rightarrow n^3 + n \in \Omega(n^2)$$

(B)

$$\text{Max}\,[f(n), g(n)] \in \theta\,[f(n) + g(n)]$$

True

let $f(n) = n$, $g(n) = n^2$

$$\text{Max}\,(n, n^2) = n^2$$

$$n^2 \in \theta\,(n + n^2)$$

(C) $f(n) = O(g(n))$ and $g(n) = \Omega(g(n$

this implies that

$$c_1\,(g(n)) \leq f(n) \leq c_2\,g(n)$$

$\Rightarrow$ This will be true if $O(g(n))$

$= \Omega\,(g(n))^2 = O(g(n))^2$

True

eg:

$$f(n) = n+1 \quad , \quad g(n) = n$$

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

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

Hence true

Question 04:

(A) $O(n \log_2 n)$ algorithm

Function checkMajority (c, M)
i = count = 0
⇒ while i less than length of M if (M[i]) is not actual c) and (equivalentTest)(M[i], c) == true) then:
count ++
Endif
Endwhile
if count is greater than half the length of M then:
Return true
Else:
Return false
Endif

```
function divide_and_find(M)
    if M.length = 1
        Return M[0]
    Else if M.length = 2
        If equivalentTest (M[0], M[1])
                == true
            Return M[0] or M[1]
Divide M
 M1 = assign first half
 M2 = assign second half
 c = divide_and_find (M1)
 if c is returned then
        found = checkMajority (c, M)
        if found = true then
            Return c

        Else
            c = divide_and_find (M2)
            found = check Majority (c, M)
        If found = true then
                return c

        Else
            Return "not found"
        End if
      End if
 End if
 Return "not found"
```