ALGO ASSIGNMENT 1

Name: Muhammad Musab
Roll No: 20K-0226 Section: 5D

Question No: 1

Steps of Insertion Sort:-

~~Step 1~~

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

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

Step 2:
[16, 12, 6, 27, 9, 1, 18, 5, 31] K = 12

Step 3:
[27, 16, 12, 6, 9, 1, 18, 5, 31] K = 27

Step 4:
[27, 16, 12, 9, 6, 1, 18, 5, 31] K = 9

Step 5:
[27, 16, 12, 9, 6, 1, 18, 5, 31] K = 1

Step 6:
[27, 18, 16, 12, 9, 6, 1, 5, 31] K = 18

Step 7:
[27, 18, 16, 12, 9, 6, 5, 1, 31] K = 5

Step 8:
[31, 27, 18, 16, 12, 9, 6, 5, 1] K = 31

(Done)

(b) Due to the Nested looping of for and while loop in the code, the time complexity is $O(n^2)$. We can prove it's correctness by loop invariant.

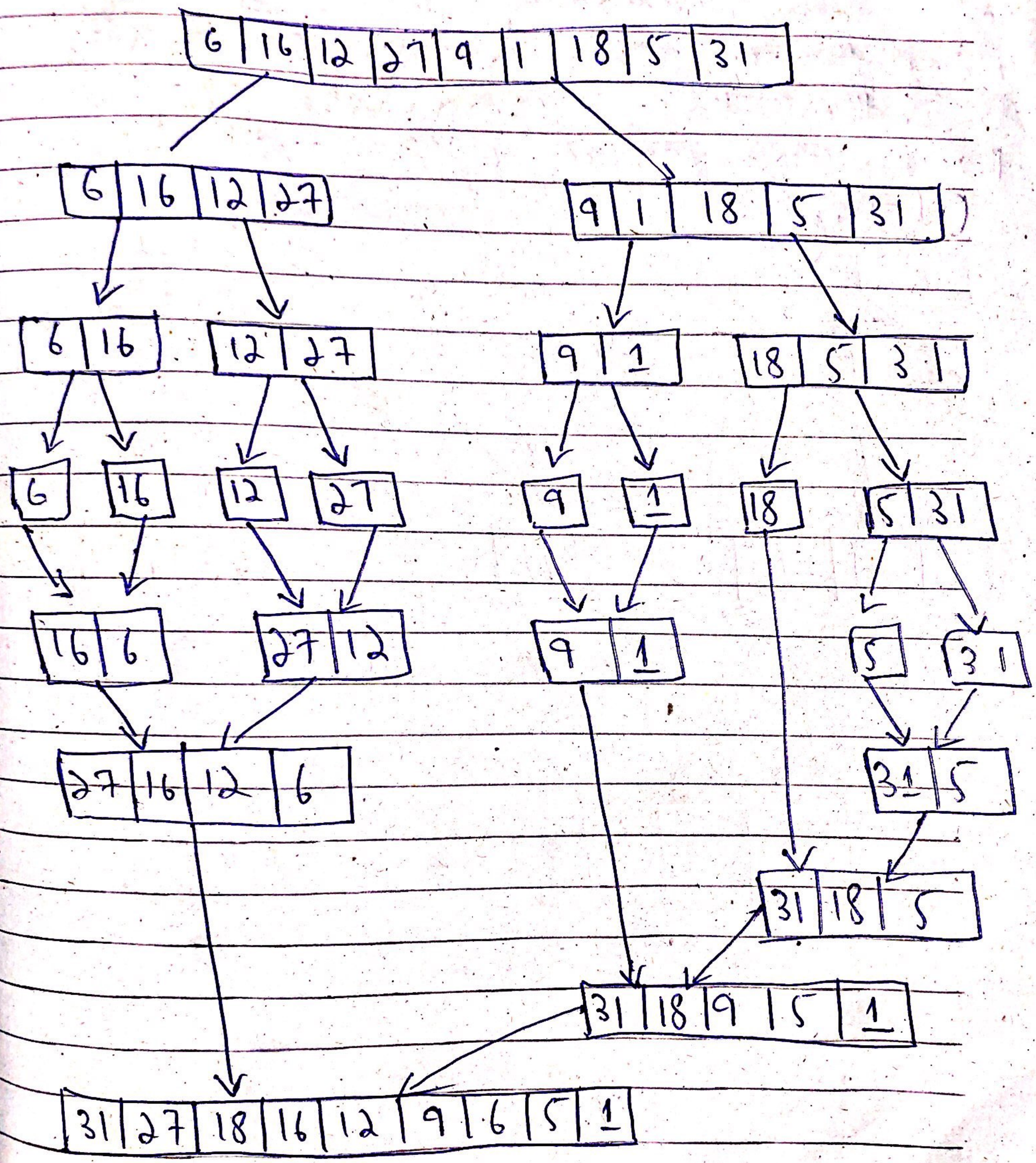
The insertion sort says that the subarray $A[1 - j-1]$ the array is always in sorted order.

* The initial state is always true.
* The loop invariant is true before nth iteration then it will be true after $(n+1)^{th}$ iteration as well.
* The termination of loop gives that algorithm is correct.

Since all these conditions hold, we conclude that this algorithm is correct.

Question No: 2 (Merge Sort)

Part (a):



(b) At each level, if we split array into three parts instead of two and then combining all of them using three way merge subroutine, so the time complexity of three-way merge sort will be $\Theta(n \log_3 n)$.

Question No: 3 {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

Now, for showing it's correctness, we are using loop invariant.

(1) Initialization:- Before the loop starts the condition is satisfied, because the pivot and subarrays $A[p \ldots i]$ and $A[i+1, \ldots, j-1]$ are empty.

(2) When the loop is running, if $A[j] \leq$ 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 No: 4 :-

Part(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
    Endwhil
    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 = checkMajority(c, M)
          If found = true then
              Return c
          Else
              Return 'not found'
          Endif
      Endif
  Endif
  Return 'not found'
```

Question No: 5

```
MinMaxSum(A)
    n = A.length/2
    Quicksort (A, 1, A.length) //Sort A in nlog2n 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 No: 6

Part (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 $n_0 = 1$

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

Thus, this is proved that $n^3 - 2n + 1 = O(n^3)$ for $c = 2$ and $n_0 = 1$

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

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

$$5n^2 \log_2 n + 2n^2 \le c\, n^2 \log_2 n$$

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

$$\le 7n^2 \log_2 n$$

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

Thus proved $5n^2 \log_2 n + 2n^2 = O(n^2 \log_2 n)$ for $c = 7$ and for all $n \ge 2$

$\boxed{n_0 = 2}$.

Question No: 7

To measure the efficiency of an algorithm we take account of the scalability and the time and space complexities. To find time complexity we need to take account of the input size. The Big O, Ω, Ø are used to relate growth of functions.

Big O:- The graph of f(n) remains to right of c·g(n) graph after the point K_0 we say that f(n) is Big (O) of g(n)

Big Ω: The graph of f(n) bounds from below the graph of c·g(n) we say that f(n) is in Big Omega of g(n) after the point K.

Big Ø:- If the graph f(n) bound g(n) from above and below then we say that fK is in Big Theta of g(n) after the point K_0.

The Asymptotic bounds are the limit of the functions.

Question No: 8 Master Theorem

Part (a): $T(n) = 2T(n/3) + cn^2$

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

Since $a < b^d$

i.e; $2 < 3^2$

so applying 1st condition.

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

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

Part (b) $T(n) = 4T\left(\frac{n}{3}\right) + c \cdot n$

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

Since $a > b^d$

i.e $4 > 3^1$

so third condition applies

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

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

Part (c): $T(n) = 8T(n/2) + cn^3$

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

Since $a = b^d$

$8 = 2^3$

so the second condition applies

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

$$T(n) \in O(n^3 \log n)$$

Question No: 1. Iteration Method

Part (a): $T(n) = 2T(n/3) + n^2$, $\{T(1) = 1\}$ → eq 1

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

Put $T(n/3)$ in eq (1)

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

$T(n) = 4T(n/9) + 2n^2/9 + n^2$ — (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)

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

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

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

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

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

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

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

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

Part (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)

$T(n/9) = 4T(n/27) + n/9$

Put $T(n/9)$ in eq (2)

$T(n) = 16[4T(n/27) + n/9] + 4n/3 + n$

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

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

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

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

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

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

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

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

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

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

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

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

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

Question No: 10

Part(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 $f(n) \in \Omega(n)$

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

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

Part(b): $\max(f(n), g(n)) \in \Theta(f(n) + g(n))$

True

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

$\max(n, n^2) = n^2$

$n^2 \in \Theta(n + n^2)$

Part(c): $f(n) = O(g(n))$ and $g(n) = \Omega(g(n))$

this implies that

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

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

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

True

for example:

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

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

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

Hence true