ALGO ASSIGNMENT 01

BILAL AHMED KHAN; 20K-0183

Sec: "B"

QUESTION 01

1	2	3	4	5	6	7	8	9
6	16	12	27	9	1	18	5	31

Step 01:

Key: ~~6~~ 16

6, 16 ‖ 12, 27, 9, 1, 18, 5, 31

Step 02:-

Key = 12

16, 6, 12 ‖ 27, 9, 1, 18, 5, 31

Step 03:- Key = 27

16, 12, 6, 27 ‖ 9, 1, 18, 5, 31

Step 04: Key = 9

27, 16, 12, 6, 9 | 1, 18, 5, 31

Step 05

Key = 1

27, 16, 12, 9, 6, 1 | 18, 5, 31

Step 06:

Key = 18

27, 16, 12, 9, 6, 1, 18 | 5, 31

Step 07:- Key = 5

27, 18, 16, 12, 9, 6, 1, 5 | 31

Step 08 :-

Key = 31

27, 18, 16, 12, 9, 6, 5, 1, 31 |

Sorted array:-

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

Time complexity of Insertion sort:-

Time complexity of Insertion sort is $O(n^2)$ because in worst case we have to do $\frac{n(n-1)}{2}$ comparision i.e $\frac{n^2-n}{2}$.

Since we take upper bound while calculating time complexities we can say that insertion sort has $O(n^2)$ time complexity

LOOP Invariant:-

The loop invariant for insertion sort is that the left side of the array is always sorted at any given moment during its execution before, during & after the algo's execution.

Loop Invariant:-

Initialization:- before 1st iteration $j=2$ the subarray $A[1 \cdots j-1]$ is sorted

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

Maintenance:-

during inner loop iterations, $A[j-1]$, $A[j-2]$ the while moves until a proper position is found for the key

- after the key is placed at its correct position the left sub array becomes sorted.

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

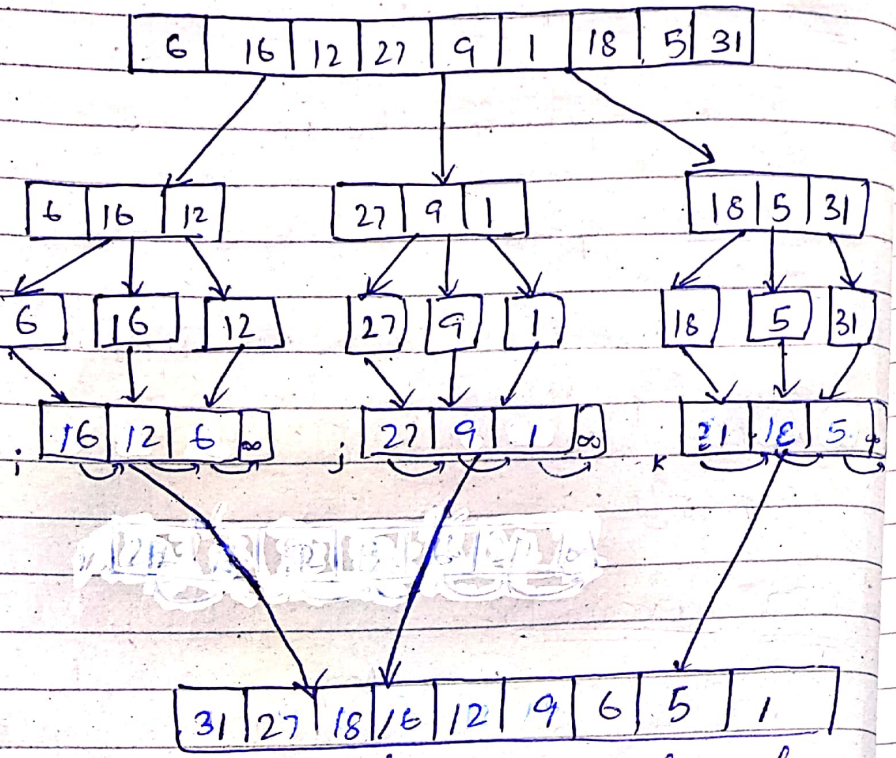
Termination:-

The loop terminates when $j > n$, the left sub array consists of all the elements in the original array in the sorted order.

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

QUESTION 02

Array: 6, 16, 12, 27, 9, 1, 18, 5, 31



Array sorted in ascending order

Time complexity of 3-way merge sort:

The time complexity function of 3-way merge sort is given by

$$T(n) = 3T(n/3) + O(n)$$

By master theorem

$a = 3;\ b = 3;\ d = 1 \ \therefore\ a = b^d$

Therefore, the time complexity of 3-way merge sort will be

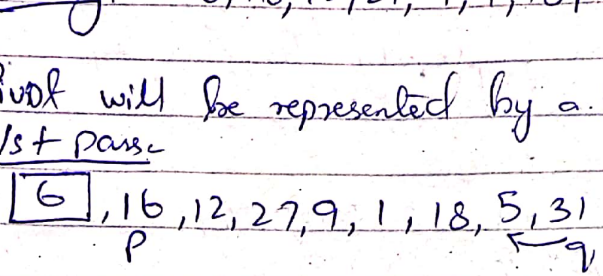
$$T(n) = n \log(n)$$

QUESTION 03:-

Array:- 6, 16, 12, 27, 9, 1, 18, 5, 31

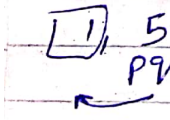
Pivot will be represented by a box around it

1st pass:-

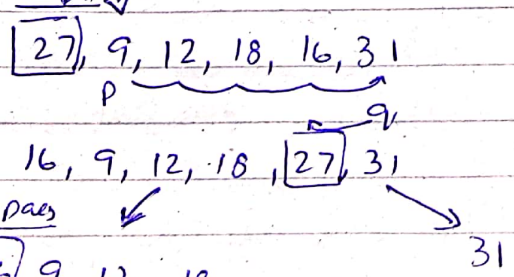


1, 5, 6, 27, 9, 12, 18, 16, 31

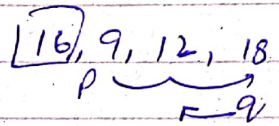
2nd pass



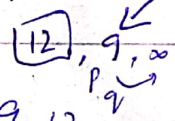
3rd pass



4th pass



5th pass



Combining elements in their sorted place we get:

Sorted array:

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

Loop Invariant:- After completion of each pass the pivot is at its correct place i.e. all elements on the left are less than the pivot & all elements on the right are greater than the pivot

- Initialization: A[0] is made the pivot of the array, it attains its correct position after 1st pass.
- Maintenance: in each subsequent pass where 1, 27, 16, 12 become pivots one by one, all of them attain their correct positions in array A.
- Termination: At termination all pivots have achieved their correct positions & consequently array is sorted.

QUESTION 04:

a) Design an $O(n \log_2 n)$ algorithm to solve this problem.

$O(n \log_2 n)$ algo to determine whether a set of $>n/2$ equivalent cards exists or not.

The problem can be solved by divide and conquer algorithm.
The input of n cards can be divided into two parts n_1 & n_2.
If both parts have a card B that equates to the majority of cards in n_1 & n_2, then we can say that the input contains $\geq n/2$ equivalent cards

Counting the majority equivalent cards in both parts of the array will take $O(n)$ time.

The splitting up of the input can be done recursively with the recurrence relation being

$$T(n) = 2T(n/2) + 2n$$
$$= 2(2T(n/4) + n) + 2n$$
$$= 4T(n/4) + 4n$$
$$= 4(2T(n/8) + n/2) + 4n$$
$$= 8T(n/8) + 6n$$
$$\vdots \qquad \vdots$$
$$= 2^K T(n/2^K) + 2Kn$$
$$= nT(1) + 2n \log n$$

ignoring constants we can write

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

$\because n/2^K = 1$
$n = 2^K$
$\log_2 n = K$
$K = \log n$

Algorithm:-

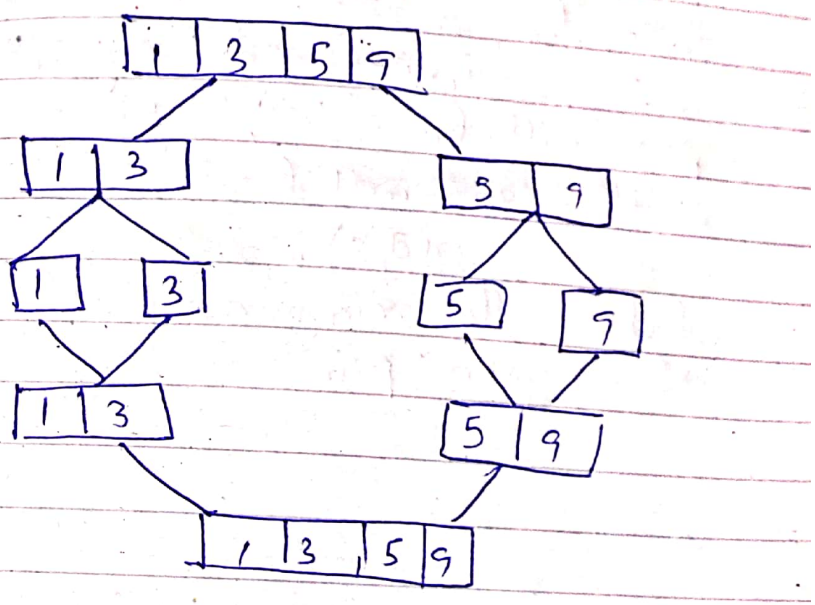
```
Equivalent_cards (cards):
  if |cards| == 1  return this card
  if |cards| == 2
        if card[0] == card[1]
        return card[0]

S1 = first n/2 elements of cards
S2 = last n/2 elements of cards
// call algo on S1
if Equivalent_cards (S1)
    returns a card
    test against all |cards|
if Equivant_cards (S2)
    returns a card
    test against all |cards|
return the card with the majority
        equivalence
```

QUESTION NO.5:

We divide our solution into 2 parts
1) Sorting
2) Pairing.

1) Sorting:-

For sorting, merge sort (divide & conquer technique will be used since it) sorts the data in nlogn complexity. The algorithm works as follows



The time complexity is given by:

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

By master theorem we can say

$T(n) = n \log n$

ii) **Pairing:-**

For making partitions that ~~maximize the minimum~~ minimize the maximum sum of a pair will be given by $(i, n-i)$ i.e for $(1, 3, 5, 9)$ the pairs will be $|(1,9), (3,5)|$ & the sums will be

$(10, 8)$

which is the minimum possible max sum of a given pair.

QUESTION 06:- (a)

Proof:-

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

$n^3 - 2n + 1 \leq c \cdot n^3$ (ignoring $-2n$)

$n^3 + 1 \leq c \cdot n^3$

$n^3 \leq n^3$ for $n \geq 0$

$1 \leq n^3$ for $n \geq 1$

$\therefore n^3 + 1 \leq n^3 + n^3$ for $n \geq 1$

$n^3 + 1 \leq 2n^3$

Thus

$c = 2$ $n_0 = 1$

QUESTION 06 (B)

Proof:-

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

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

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

$5n^2 \log_2 n \leq 5n^2 \log_2 n$ for $n \geq 0$

$2n^2 \leq 2n^2 \log_2 n$ for $n \geq 2$

Thus $c = 7$ $n_0 = 2$

QUESTION 07:-

- When we talk about an algorithms efficiency, we are talking about its scalability
- The speed of an algo is measured relative to its input, & growth in # of operations as the size of input increases
- Three types of bounds are used to analyze an algorithm's speed.
  1) Big O 2) Big Ω 3) Big Θ
- They are used to give an idea of the relative growth of a function

- $f(x)$ belongs to $O(g(x))$ such that for a constant c, x_0 $f(x) \leq c * g(x)$ for all $x \geq x_0$
- $f(x)$ belongs to $\Omega(g(x))$ such that for constants c, x_0 $f(x) \geq c * g(x)$ for all $x \geq x_0$
- $f(x)$ belongs to: $\Theta(g(x))$ such that constants $c_1, c_2, x_0 \geq 0$ such that $c_1 * g(x) \geq f(x) \geq c_2 * g(x)$ for all $x \geq x_0$

- Generally input size of algorithms is +ve integer

QUESTION 08:-

i) $T(n) = 2T(\frac{n}{2}) + c.n^2$

$a = 2;\ b = 3;\ d = 2$

$a < b^d$

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

$\boxed{T(n) = O(n^2)}$

ii) $T(n) = 4T(\frac{n}{3}) + cn$

$a = 4;\ b = 3;\ d = 1$

$a > b^d$

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

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

$\boxed{T(n) = O(n^{1.9})}$

iii) $T(n) = 8T(n/2) + cn^3$

$a = 8;\ b = 2;\ c = 3$

$a = b^d$

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

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

QUESTION 09

i) $T(n) = 2T(n/3) + n^2 \quad (T(1) = 1)$

$T(n) = 2\left(2T(n/9) + n^2/9\right) + n^2$

$= 4T(n/9) + \frac{2n^2}{9} + n^2$

$= 4\left(2T(n/27) + \frac{n^2}{81}\right) + \frac{2n^2}{9} + n^2$

$= 8T(\frac{n}{27}) + \frac{4n^2}{81} + \frac{2n^2}{9} + n^2$

$= 2^k T(\frac{n}{3^k}) + \left(1 + \frac{2}{9} + \frac{4}{81}\right) n^2$

$1 + \frac{2}{9} + \frac{4}{81} + \cdots \quad \because x < 1$

$1 + x + x^2 + \cdots x^n = \frac{1}{1-x} = \frac{1}{1-2/9} = 9/7$

$= 2^k T(\frac{n}{3^k}) + \frac{9}{7} n^2$

$\frac{n}{3^k} = 1$

$n = 3^k$

$\log_3 n = k$

$= 2^{\log_3 n} T(1) + \frac{9}{7} n^2$

$= 2^{\log_3 n} + \frac{9}{7} n^2$

$\because n^2$ is the greater degree thus by ignoring other terms.

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

2) $T(n) = 4T(n/3) + n \quad (T(1) = 1)$

$T(n) = 4\left(4T(n/9) + n/3\right) + n$

$= 16T(n/9) + \frac{4}{3}n + n$

$= 16\left(4T(n/27) + n/9\right) + \frac{4}{3}n + n$

$= 64T\left(\frac{n}{27}\right) + \frac{16}{9}n + \frac{4}{3}n + n$

$= 64T\left(\frac{n}{27}\right) + \left(1 + \frac{4}{3} + \frac{16}{9}\right)n$

$r = 4/3 \quad \because r > 1$

$= \frac{r^{n+1} - 1}{r - 1} = \frac{(4/3)^{n+1} - 1}{4/3 - 1}$

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

$= 3\left[(4/3)^{n+1} - 1\right]$

$= 3\left(\frac{4}{3}\right)^n \cdot 4/3 - 3$

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

$\Rightarrow = 64T\left(\frac{n}{27}\right) + \left[4\left(\frac{4}{3}\right)^n - 3\right]n$

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

$n/3^K = 1$

$n = 3^K$

$\log_3 n = K$

$= 4^{\log_3 n} T(1) + 4\left(\frac{4}{3}\right)^{\log_3 n} - 3n$

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

$= n^{1.26} + 4n^{-0.28} - 3n$

Ignoring lower order terms.

$T(n) = O(n^{1.26})$

QUESTION 10

a) Let $f(n) = n^2$, $g(n) = n^3$ & $h(n) = n$

$f(n) \in O(g(n))$

$f(n) \in O(n^3)$

&

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

$f(n) \in \Omega(n^2)$

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

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

Thus the statement is TRUE

b) Let $f(n) = n^2$, $g(n) = n$

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

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

Thus the statement is true

c)

$f(n) = O(g(n))$ & $f(n) = \Omega(g(n))$

this implies that

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

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

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

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

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

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

Thus true