Faculty of Computing , Online Examinations 2022

STUDENT NAME	**ACJD Silva**		
INDEX NUMBER (NSBM)	**22795**	**YEAR OF STUDY AND SEMESTER**	**Year 1 Semester 2**
MODULE NAME **(As per the paper)**	CS106.3- Data structures and Algorithms		
MODULE CODE	**CS106.3**		
MODULE LECTURER	**Mrs. Manoja Weerasekara**	**DATE SUBMITTED**	**16.08.2022**

For office purpose only:

GRADE/MARK	
COMMENTS	

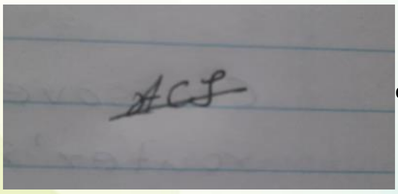
Declaration

PLEASE TICK TO INDICATE THAT YOU HAVE SATISFIED THESE REQUIREMENTS

- ✓ I have carefully read the instructions provided by the Faculty
- ✓ I understand what plagiarism is and I am aware of the University's policy in this regard.
- ✓ I declare that the work hereby submitted is my own original work. Other people's work has been used (either from a printed source, Internet or any other source), has been properly acknowledged and referenced in accordance with the NSBM's requirements.
- ✓ I have not used work previously produced by another student(s) or any other person to hand in as my own.
- ✓ I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
- ✓ I hereby certify that the individual detail information given (name, index number and module details) in the cover page are thoroughly checked and are true and accurate.

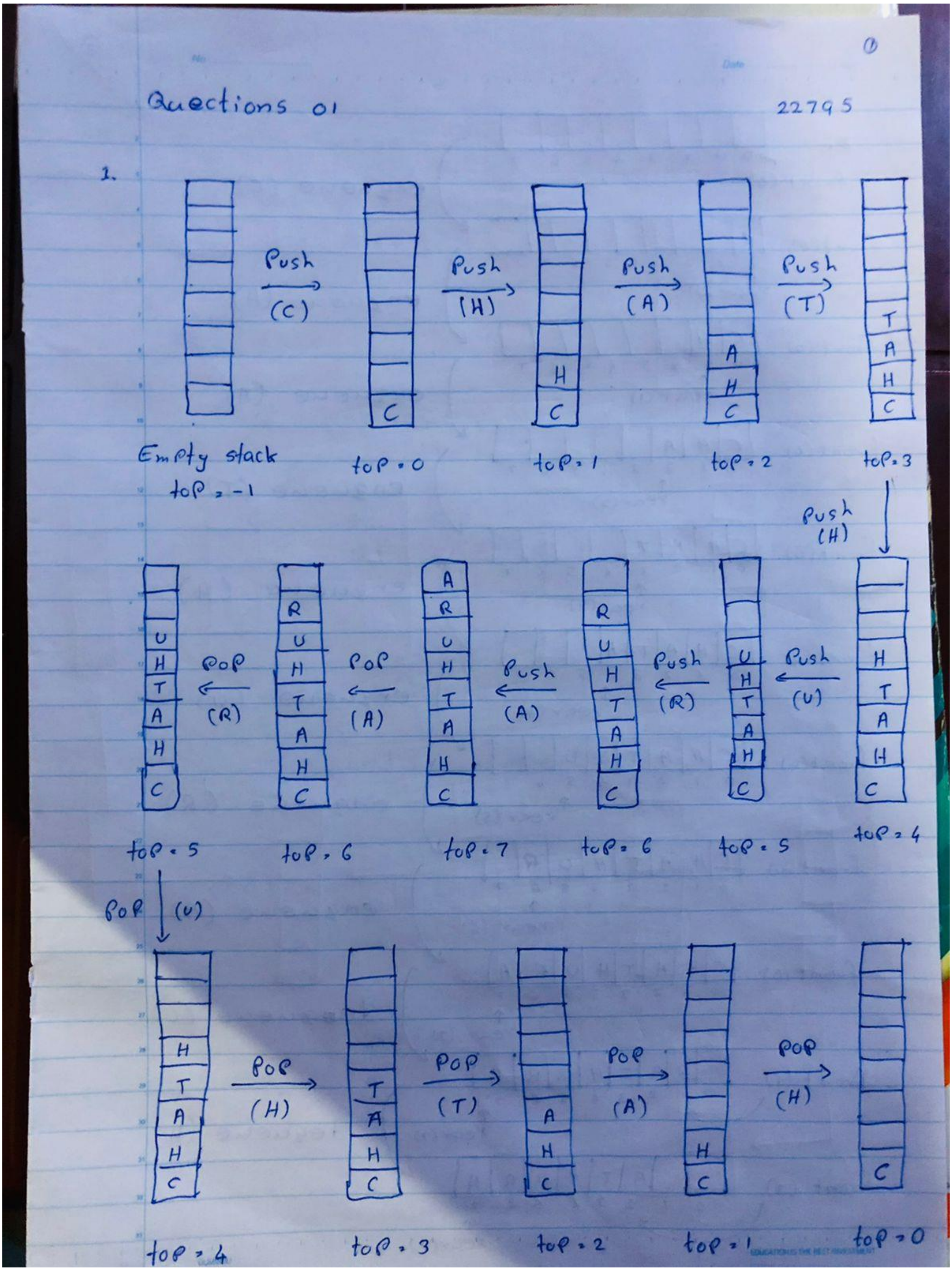
I hereby certify that the statements I have attested to above have been made in good faith and are true and correct. I also certify that this is my own work and I have not plagiarized the work of others and not participated in collusion.

Date: 16.08.2022 **E- Signature:



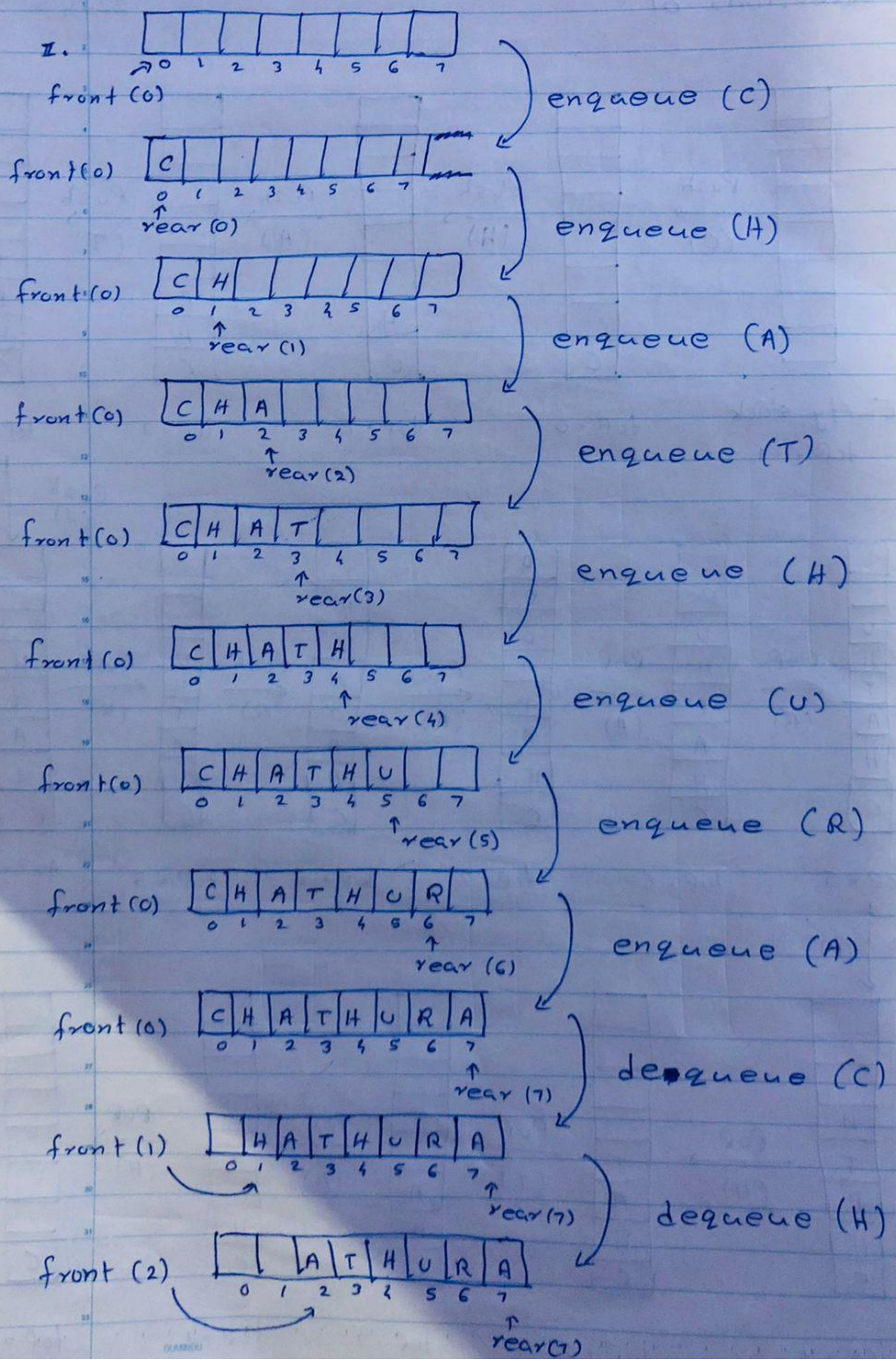
of

Question 1



1
Questions 01
22795
1.
Push (C)
Push (H)
Push (A)
Push (T)
Empty stack
top = -1
top = 0
top = 1
top = 2
top = 3
Push (H)
Pop (R)
Pop (A)
Push (A)
Push (R)
Push (U)
top = 5
top = 6
top = 7
top = 6
top = 5
top = 4
Pop (U)
Pop (H)
Pop (T)
Pop (A)
Pop (H)
top = 4
top = 3
top = 2
top = 1
top = 0

II.


front (0)
enqueue (C)
C
rear (0)
enqueue (H)
C H
rear (1)
enqueue (A)
C H A
rear (2)
enqueue (T)
C H A T
rear (3)
enqueue (H)
C H A T H
rear (4)
enqueue (U)
C H A T H U
rear (5)
enqueue (R)
C H A T H U R
rear (6)
enqueue (A)
C H A T H U R A
rear (7)
dequeue (C)
front (1)
H A T H U R A
rear (7)
dequeue (H)
front (2)
A T H U R A
rear (7)
0 1 2 3 4 5 6 7

2. struct queue {

```
    int q [size] ;
    int rear = -1 ;
    int front = 0 ;

} st ;

void enqueue (int item)

  {
     st. rear ++ ;
     st. q [st. rear] = item ;

  }

Stract stack {

     int s [size];
     int top ;

  } st ;

  void push (int item) {

     st. top ++ ;
     st. s [st. top] = item ;

  }
```

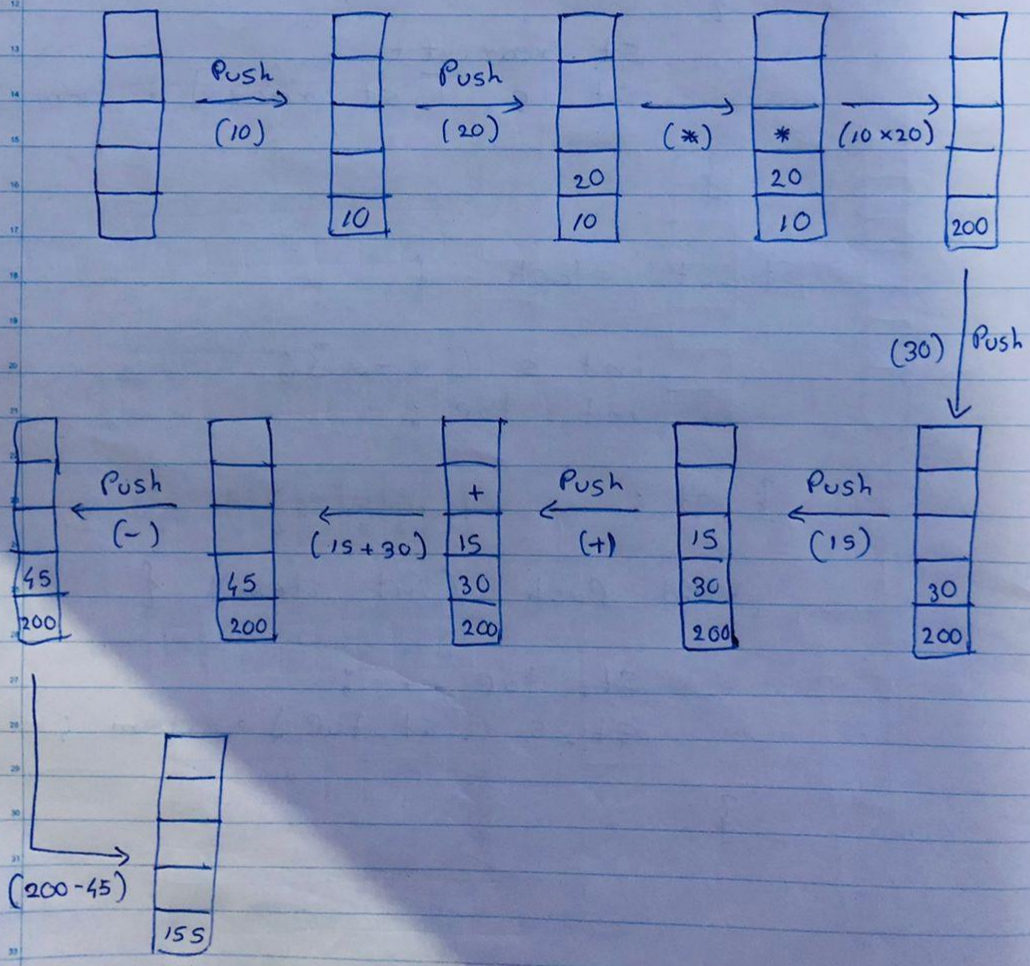
iv. DFS

S
A
B
D
C

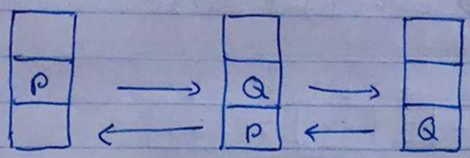
BFS

S	A	B	C	D

v.



vi.



Questions 02

22795

2. Linear Search :

search key (sk) = 8 let i = 0

0	1	2	3	4	5	6
32	53	21	8	85	3	71

sk = 8

arr [0] == sk
False.

i = 1

i++ ;

32	53	21	8	85	3	71

arr [1] == sk
False

i = 2

i++ ;

32	53	21	8	85	3	71

arr [2] == sk
False

i = 3

i++ ;

32	53	21	8	85	3	71

arr [3] == sk
True.

∴ arr [3] = 8

Binary Search :-

* for binary search , array must be sorted first.

0	1	2	3	4	5	6
32	53	21	8	85	3	71

after sorting :-

0	1	2	3	4	5	6
3	8	21	32	53	71	85

applying binary search :-

no. of elements = 7

mid value = $\frac{0-6}{2}$ = 3

Search key (sk) = 8

let i = mid value.

0	1	2	3	4	5	6
3	8	21	32	53	71	85

i = 3

arr[3] == sk
False.

sk < mid value.
∴ i-- ;

0	1	2	3	4	5	6
3	8	21	32	53	71	85

i = 2

arr [2] == sk
False.

i-- ;

0	1	2	3	4	5	6
3	8	21	32	53	71	85

i = 1

arr [1] == sk
True.

∴ arr [1] = 8 //

ii. Application of bubble sort :

0	1	2	3	4	5	6
32	53	21	8	85	3	71

let i = 0 , j = 0 ;

i = 0 ;

0	1	2	3	4	5	6
32	53	21	8	85	3	71

iteration = 1
(J = 0)

arr [1] > arr [0] ; true.

i = 1 ;

0	1	2	3	4	5	6
32	53	21	8	85	3	71

arr [2] > arr [1] ; False
swap

i = 2 ;

0	1	2	3	4	5	6
32	21	53	8	85	3	71

arr [3] > arr [2] ; False
swap

i = 3 ;

0	1	2	3	4	5	6
32	21	8	53	85	3	71

arr [4] > arr [3] ; True.

i = 4 ;

0	1	2	3	4	5	6
32	21	8	53	85	3	71

arr [5] > arr [4] ; false.
swap

i = 5 ;

0	1	2	3	4	5	6
32	21	8	53	3	85	71

arr [6] > arr [5] ; False.
swap

0	1	2	3	4	5	6
32	21	8	53	3	71	85

i = 0 ; iteration = 2 [j = 1]

32	21	8	53	3	71	85

arr [1] > arr [0] ; false
swap

i = 1 ;

0	1	2	3	4	5	6
21	32	8	53	3	71	85

arr [2] > arr [1] ; flase
swap

i = 2 ;

0	1	2	3	4	5	6
21	8	32	53	3	71	85

arr [3] > arr [2] ; True.

i = 3 ;

0	1	2	3	4	5	6
21	8	32	53	3	71	85

arr [4] > arr [3] ; False.
Swap.

i = 4 ;

0	1	2	3	4	5	6
21	8	32	3	53	71	85

arr [5] > arr [4] ; True.

i = 5 ;

0	1	2	3	4	5	6
21	8	32	3	53	71	85

arr [6] > arr [5] ; True.

i = 0 ;

0	1	2	3	4	5	6
21	8	32	3	53	71	85

iteration = 3
(j = 2)

arr [1] > arr [0] ; False
swap.

i = 1 ;

0	1	2	3	4	5	6
8	21	32	3	53	71	85

arr [2] > arr [1] ; True.

i = 2 ;

0	1	2	3	4	5	6
8	21	32	3	53	71	85

arr [3] > arr [2] ; False.
swap.

i = 3 ;

0	1	2	3	4	5	6
8	21	3	32	53	71	85

arr [4] > arr [3] ; True.

i = 4 ;

0	1	2	3	4	5	6
8	21	3	32	53	71	85

arr [5] > arr [4] ; True.

i = 5 ;

0	1	2	3	4	5	6
8	21	3	32	53	71	85

arr [6] > arr[5] ; True.

Iteration = 4
(j = 3)

i = 0 ;

0	1	2	3	4	5	6
8	21	3	32	53	71	85

arr [1] > arr [0] ; True.

i = 1 ;

0	1	2	3	4	5	6
8	21	3	32	53	71	85

arr [2] > arr [1] ; False.
Swap.

i = 2 ;

0	1	2	3	4	5	6
8	3	21	32	53	71	85

arr [3] > arr [2] ; True.

i = 3 ;

0	1	2	3	4	5	6
8	3	21	32	53	71	85

arr [4] > arr [3] ; True.

i = 4 ;

0	1	2	3	4	5	6
8	3	21	32	53	71	85

arr [5] > arr [4] ; True.

i = 5 ;

8	3	21	32	53	71	85

Iteration = 5
(j = 4)

i = 0 ;

0	1	2	3	4	5	6
8	3	21	32	53	71	85

arr [1] > arr [0] ; false.
Swap

i = 1 ;

3	8	21	32	53	71	85

Sorted

Selection sort :-

0	1	2	3	4	5	6
32	53	21	8	85	3	71

min = 0 ;

0	1	2	3	4	5	6
3	53	21	8	85	32	71

Iteration = 1

if arr [min] < arr[i] ;
Swap.

0	1	2	3	4	5	6
3	8	21	53	85	32	71

Iteration = 2

0	1	2	3	4	5	6
3	8	21	53	85	32	71

if arr [min] < arr [i] ;
Swap

0	1	2	3	4	5	6
3	8	21	32	85	53	71

Iteration = 3

0	1	2	3	4	5	6
3	8	21	32	53	85	71

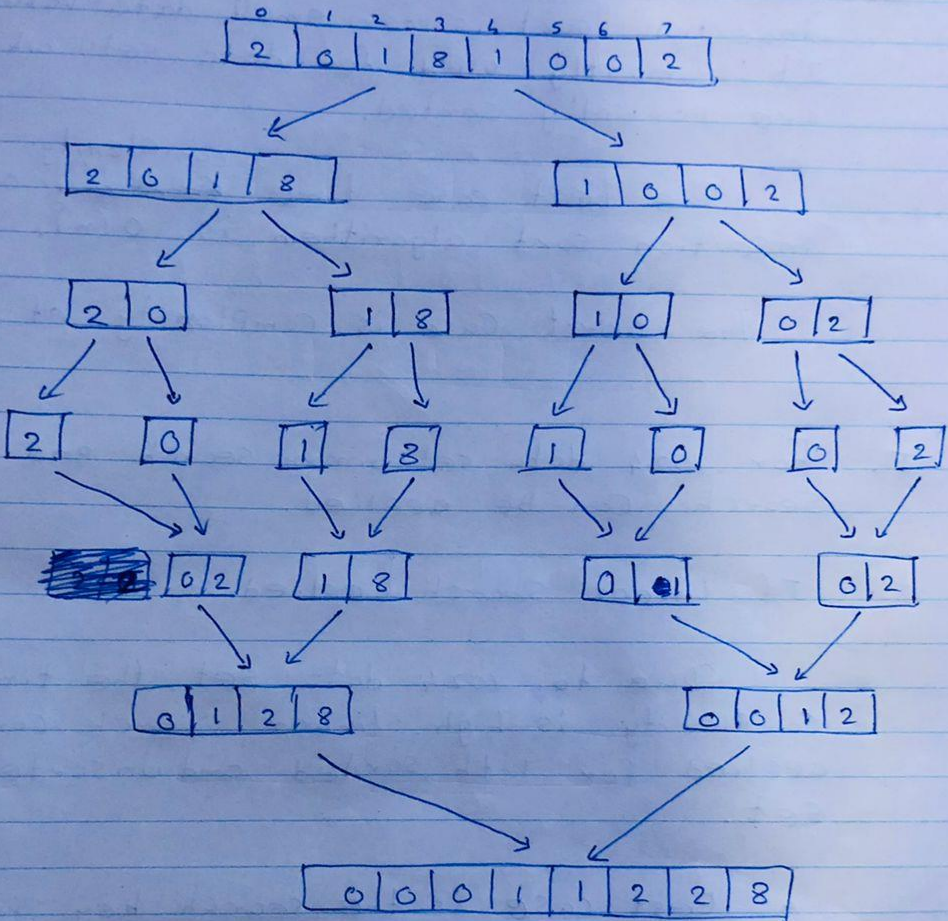
if arr [min] < arr[i] ;
Swap.

0	1	2	3	4	5	6
3	8	21	32	53	71	85

Iteration = 4

Sorted

iii. Merge Sort :



IV. The best and most efficient to implement insertion sort is for small data values. It is mostly used for data sets which are partially sorted.

The best case time ~~Completly~~ complexity of insertion sort algorithem is $O(n)$.

The worst case is complexity of $O(n^2)$.

V. For 1024 data set binary search and linear Search can be applied.

If linear Search applied. :-

Due to 1024 data set the time Complexity is high. linear Search Can be applied for both sorted and unsorted data set.

* Best case is the search key in first element. ~~of the data set~~.

* Best case Complexity is $O(1)$.

* The worst case is the search key in last element of the data set.

worst Case complexity is $O(n)$.

If binary search is applied :-

In here binary search only can be applied to sorted data list.

∴ Appliying binary search for sorted 1024 data set is more efficient than linear search.

* Best case of the binary search for this 1024 data set is search key is in 512^{th} data.

Best case : O(1).

* worst case scenario for the binary search for 1024 data set is search key is in 1024^{th} or first data.

∴ worst case : O (log n).

16

Question 03 22795.

I.

a. base case = A if (n<2).
recusive call = return fib (n-1) + fib (n-2)

b. 6

c. 5

II. Out Put 01 = 1 2 3 4 5 6 7 8 9 10

Out Put 2 = 10 9 8 7 6 5 4 3 2 1

III. functio addup to (n) {

return n * (n+1) / 2 → runs only one time

}

∴ time complexity O(1).

function addupto (n) {

let total = 0; ← C_1

for (let i=1 ; i<=n ; i++) {

C_2 C_3 C_4

total += 1 ← C_6

} C_5

return total ; ← C_7.

$C_1 \rightarrow$ runs only one tine

$C_2 \rightarrow$ runs only one time.

$C_3 \rightarrow$ runs $(n+1)$

$C_4 \rightarrow$ runs $(n+1)$

$C_5 \rightarrow$ runs (n) time

$C_6 \rightarrow$ runs (n) time

$C_7 \rightarrow$ runs (1) time.

time complexing $= C_1 * 1 + C_2 * 1 + C_3 (n+1) + C_4 * (n+1) + C_5 * n + C_6 * n + C_7 * 1$

$= C_3 (n+1) + C_4 (n+1) + C_5 n + C_6 n$

IV.

a. $5 + 0.001 n^3 + 0.025 n$

$O(n^3)$

b. $500 n + 100 n^{1.5} + 50 n \log_{10}{}^{n}$

c. $0.3 n + 5 n^{1.5} + 2.5 n^{1.75}$

$O(n^{1.75})$

d. $n^2 \log_2 n + n(\log_2 n)$

$O(n^2 \log n)$

e. $n \log_3 n + n \log_2 n$

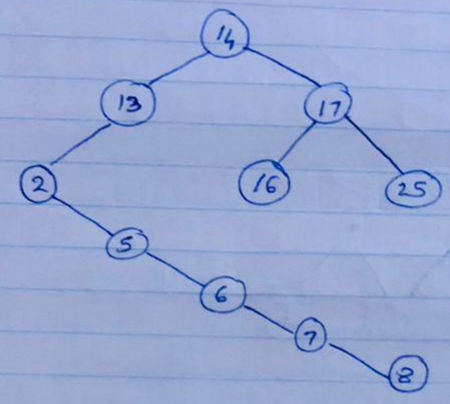
$O(n \log n)$

Questions 04

(18).

22795

(a)



(b) In Order : 2, 5, 6, 7, 8, 13, 14, 16, 17, 25

Pre Order : 14, 13, 2, 5, 6, 7, 8, 17, 16, 25

Post Order : 8, 7, 6, 5, 2, 13, 16, 25, 17, 14

(c) leaf Node : 8, 16, 25

total value of l.v : 49 //

(d) There is no node called 15 if we consider node 16 Path

{14, 17, 16}

(e) depth of 16 → 3 //

(f) 7

(g)

$n = 2^h - 1$

$16 = 2^h - 1$

$17 = 2^h$ //

It's not a Perfect Binary tree.

$n = 2^h - 1$

$63 = 2^h - 1$

$64 = 2^h$

$2^6 = 2^h$

$h = 6$ //

It's a Perfect Binary tree.

$n = 2^h - 1$

$127 = 2^h - 1$

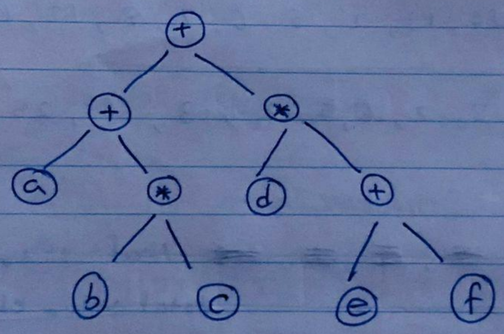
$128 = 2^h$

$2^7 = 2^h$

$h = 7$ //

It's a Perfect Binary tree.

(h) $a + (b * c) + d * (e + f)$



$(5 - x) * y + 6 / (x + z)$