

Name :- Zubayer Ahmed Zidhan Laskar, Roll :- 200710007062

Subject :- Data Structures & Algorithms, Date :- 29/10/2021

Semester :- 3rd

CIF-1

1. Ans:- A stack is a type of data structure in which new elements are added and removed from the same side of the array called "top". A stack exhibits LIFO (Last In First Out) property.

The push algorithm for a stack implemented using an array is as follows:-

Procedure push (S, maxsize, top, element)

1. If ($top == maxsize - 1$)
2. Print("Stack Overflow").
3. Return.
4. EndIf
5. $top := top + 1$
6. $S[top] := element$.
7. Return.

2. Ans:- A queue is a type of linear data structure where all the insertions are made at one end of the ~~are~~ list (known as "rear"), and, all the deletions are made at the other end of the list (called "front"). A queue exhibits FIFO (First In First Out) property.

The algorithm for delete operation on a circular queue implemented using array is as follows.

Procedure Dequeue (Q, front, rear, maxsize)

1. If (front == -1) Then
2. Write ("Queue Underflow")
3. Return.
4. EndIf
5. Write ("The element to be deleted is: " Q[front]);
6. If (front == rear) Then
7. front := -1
8. rear := -1
9. Else
10. front := (front + 1) % maxsize
11. EndIf
12. Return

3.) Ans: Asymptotic complexity of an algorithm is the analysis of the concerned algorithm when the size of its input/output is very high (∞).

Given, 'N' is the max. size of the array
Let, 'n' be the no. of elements present in the array.

The algorithm for insertion at a particular posⁿ is as follows

~~SPART~~
 Procedure Insert ($N, n, \text{element}, \text{position}, \text{arr}$)

1. If $n == N$, Then

2. Return

3. Else

4. Repeat for i in $n, n-1, n-2, \dots, \text{position}$

5. ~~$\text{arr}[i] := \text{arr}[i+1]$~~
 ~~$\text{arr}[i+1] := \text{arr}[i]$~~

6.

3. > Ans: [Contd.]

The algorithm for insertion at a particular position is as follows:-

Procedure Insert ($N, n, \text{element}, \text{position}, \text{arr}$)

1. If $n == N$, Then

2. Return

3. Else

4. Repeat for i in $n, n-1, n-2, \dots, \text{position}$

5. $\text{arr}[i] := \text{arr}[i+1]$

6. $\text{arr}[\text{position}] := \text{element}$

7. $n++$

8. Return.

9. EndIf

$$\text{Frequency count} = 1 + (n - \text{position} + 1) + (n - \text{position} + 1) + 1 + 1$$

$$= 2(n - \text{position} + 1) + 3 \approx 2n + 3 \quad [\text{if } n \gg \text{pos}]$$

The worst case will be when ~~we~~ we have to insert at index 0, ~~and~~ in that case total steps

$$= 1 + (N+1) + (N+1) + 1 + 1$$

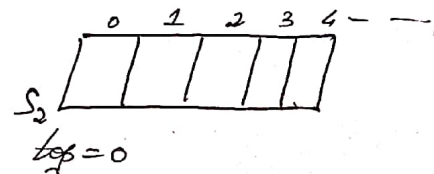
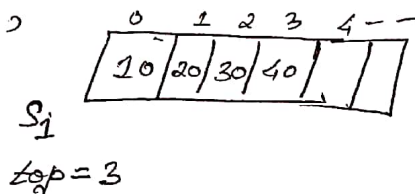
$$= 2N + 5, \text{ which is of the order } O(N)$$

∴ Worst case scenario is ~~is~~, $O(N)$

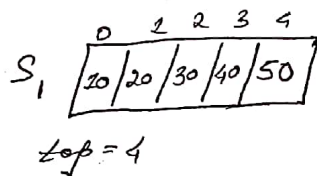
4) Ans: The minimum no. of stacks required for implementing a queue is two (2).

Let the two stacks be S_1, S_2 .

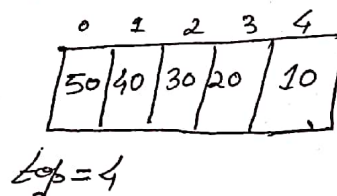
Initially,



Insert 50



Now pop from S_1 , and push all to S_2



So, ~~50~~ '10' was the first entered element is at the top of S_2 , and so in case of deletion '10' will be deleted first, fulfilling the FIFO criteria of a queue.

⑤

And, new elements will be added to S_1 and elements will be popped from S_2 , when S_2 is empty, all elements are popped from S_1 and pushed to S_2 . And, thus queue is implemented using two stacks.

5.) Ans - We can use as a character (char) stack say, S . And then we can traverse the string expression (input) and for every starting bracket i.e. " $($ " we push it to the stack. And for every closing bracket " $)$ " we pop from the stack. ~~The popped element matches with " $)$ "~~ ~~the we continue to the next character in the~~

If on reaching the end of the string there are still some elements left in the stack, then the string of parentheses is not balanced, else, the string is balanced.

— x —>