SQP1 Stack Implementation Using Array

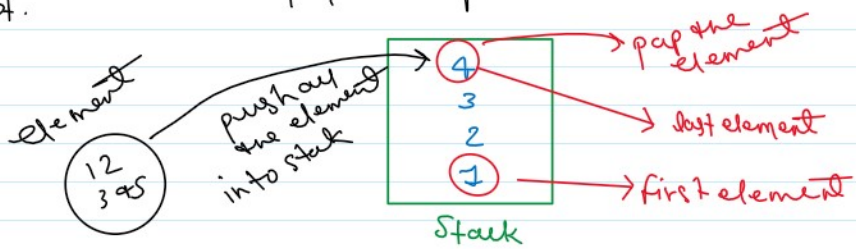
Tuesday, November 8, 2022 2:35 PM

Now we know that what is stack it is a data structure that Allow the (LIFO) property

where;-

L → last
I → in
F → first
O → out

from interview point of view it is a very important data structure and here we learn some most popular problem based on stack and queue and its concept.



#push
push means put the data into the stack

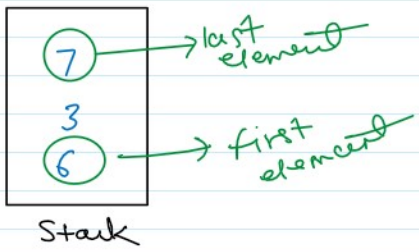
Pop
pop means through out the data from the stack

Some Important Operation on Stack

push(6)
→ that means push (6) into the stack

push(3)
→ that means push(3) into the stack

push(7)
→ that means push(7) into the stack



In stack basically we have different Operation
→ push()
→ pop()
→ top()/peek()
→ size()
→ empty()

#top()
top means give the top element from the stack / give that element that we insert into the stack last (element)

Stack. top(7) ✗
Stack.top() ✓

#pop()
pop means delete the top most element from the stack or delete that element that we insert last time into the stack.

that we insert last time into the stack.

→ that removed the element as well as return the last element from the stack

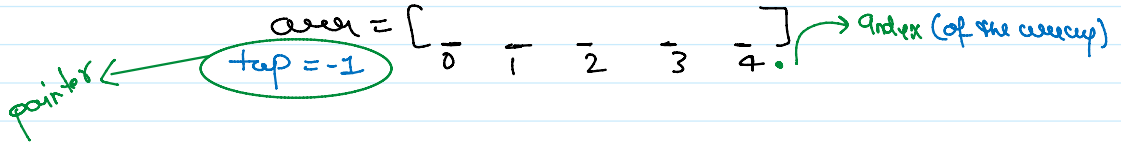
Implementation Stack using Array

→ In this lecture we learn how to implement stack using array data structures

→ Now first we fixed the size of the array as declared the array length
→ after the Instantiation of the array creating the pointer and assigne to the right side of the 0th index

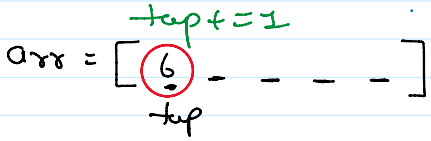
Example

array.size = (5)

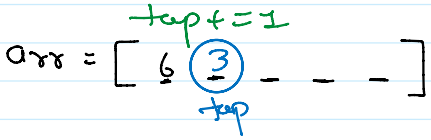


Apply Operation

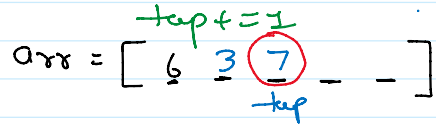
push(6)
→ once any one say push(6) that means basically increment arr pointer (+1) and put the data into the array



push(3)
→ again follow same Approach increment arr pointer (+1) and put the data into the array



push(7)
→ again follow same Approach increment arr pointer (+1) and put the data into the array



→ that is the way push() Operation work into the array using the stack Concept

→ that is the way push() Operation works into the array using the Stack Concept

top()
top() means print the last element that we insert into the array and return to the user

↳ Now where is the top pointer that is the our top element

top() → (7)
So we return directly
our[top]

pop()
pop() means remove the last element from the array

↳ delete the last element from the array and reduced the our top pointer by (-1)

top -= 1
our = [6 (3) _ _ _]
top

→ that is one way How to Implement Stack using array data Structure

Implement Point

→ Remember that every push Operation check the length of the array is < then declared length so push() Operation perform other wise not perform

→ always remember check every time the pop Operation our pointer Not < 0 because every pop Operation we reduce our pointer by (-1)

→ every top Operation we will checked that our pointer value is always >=0 and also check pop() and top() Operation our Stack should be not Empty

→ our Stack Size is pointer+1 always

→ if my pointer == -1 that means my stack is Empty

TimeComplexity = O(N)

↳ because we traverse (N) time for Stack Operation

Space Complexity = O(N)

↳ because we stored the our data into the array data

Space Complexity = O(N)

→ because we stored the given data into the array data structure

#Implementation Python

```
# gfg: https://practice.geeksforgeeks.org/problems/implement-stack-using-array/1
# lc: https://leetcode.com/problems/design-a-stack-with-increment-operation/
class Satck:

    def __init__(self):
        self.stackSize = 10000
        self.topPointer = -1
        self.stack = []

    def push(self,x):
        if len(self.stack) < self.stackSize:
            self.stack.append(x)
            self.topPointer+=1
        else:
            print("Stack if Full")

    def top(self):
        if self.topPointer > -1:
            return (self.stack[self.topPointer])

    def pop(self):

        if self.topPointer > -1:
            x = self.stack.pop(self.topPointer)
            self.topPointer-=1
            return x

    def size(self):
        return self.topPointer+1

    def isEmpty(self):
        return self.topPointer == -1

ans = Satck()
ans.push(10)
ans.push(7)
ans.push(3)

print(ans.top())
print(ans.isEmpty())
print(ans.size())

print(ans.pop())
print(ans.size())
```

#Implementation Java

```
class Stack{

    int stackSize = 10000;
    int[] stack = new int[stackSize];
    int topPointer = -1;

    void push(int x){
        topPointer+=1;
        stack[topPointer] = x;
    }

    int top(){
        if(topPointer == -1){
            return -1;
        }
        return stack[topPointer];
    }

    int pop(){
        int x = stack[topPointer];
        topPointer-=1;
        return x;

    }

    int size(){
        return topPointer+1;
    }
    boolean isEmpty() {
        return topPointer == -1;
    }
}
```