Stacks -1

Lecture-45

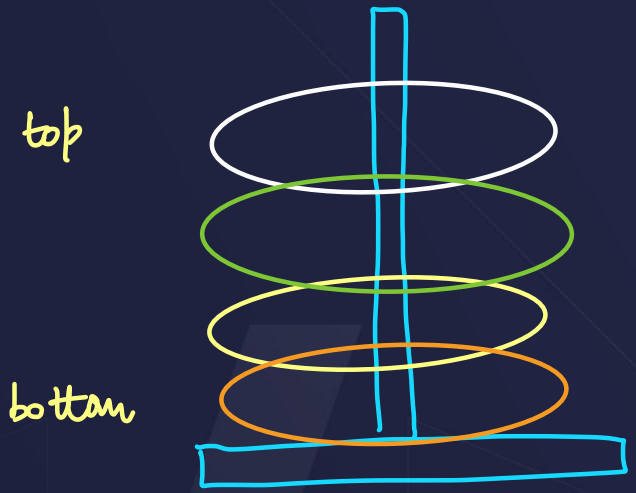
Raghav Garg

Today's checklist

1) Introduction
2) Operations performed on stacks
3) Overflow
4) Underflow
5) Array implementation of a stack
6) Linked list implementation of a stack
7) Linked list vs Array implementation
8) STL for Stack

What is a Stack?

→ Best Example : CD Rack



LIFO / FILO

(last in first out)

If I want to access the green CD what should I do?

1) Insertion of element in stack only happens at the top
2) Deletion of element in stack only happens at the top
3) get element only happens at the top

Operations on Stack

O(1) T.C. & S.C

Provides us discipline

Provides us intuition

1) st.push(val) → adds a new element at the top

2) st.pop() → removes the topmost element

3) st.top() → returns the val at the top

4) st.size() → returns the current size

→ add, delete, get at a given idx

→ T.C. & S.C

↓

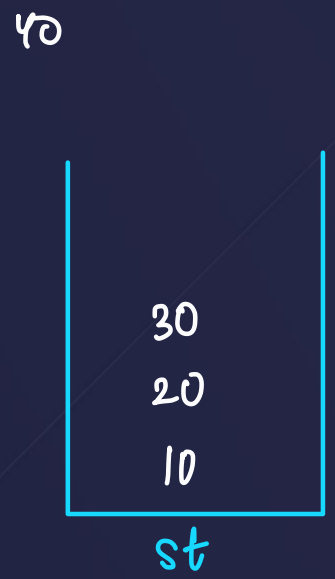
O(n)

STL for Stack

```
vector <int> v;
stack <int> st;
```

STL for Stack

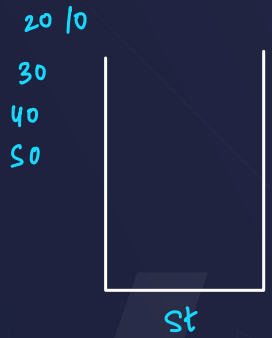
```
stack<int> st;
cout<<st.size()<<endl; //0
st.push(10); // 1
st.push(20); // 2
st.push(30); // 3
st.push(40); // 4
cout<<st.size()<<endl; // 4
st.pop(); // 3
cout<<st.size()<<endl; // 3
cout<<st.top()<<endl;
```



Output

0
4
3
30

How to print the elements of a stack ?



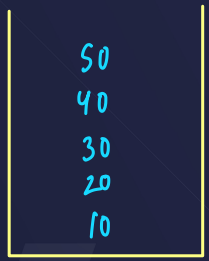
Output

50 40 30 20 10

```
while (st.size()>0){
    cout<<st.top()<<" ";
    st.pop();
}
```

How to get the elements back in Stack after printing/popping :

O(n) extra space




```
while (st.size()>0){
    cout<<st.top()<<" ";
    int x = st.pop();
    st.pop();
    temp.push(x);
}
```

Output

50 40 30 20 10

Homework : Print elements of stack bottom to top

Comparing Arrays, Linked Lists and Stacks

T.C

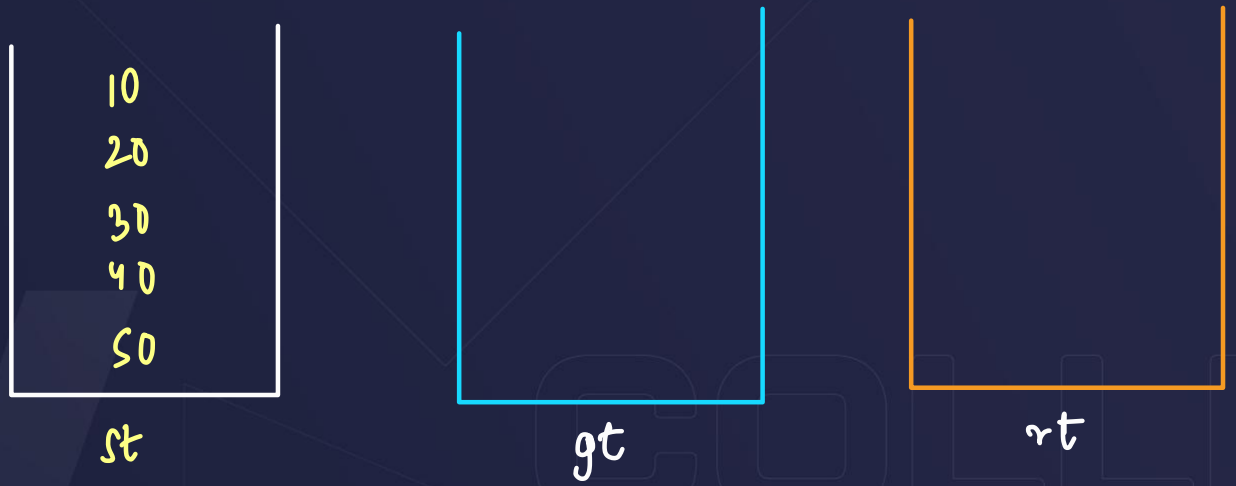
	arr	LL	stack
get	O(1)	O(n)	O(n)
insert	O(n)	O(n)	O(n)
delete	O(n)	O(n)	O(n)

S.C

	arr	LL	stack
get	O(1)	O(1)	O(n)
insert	O(n)	O(1)	O(n)
delete	O(n)	O(1)	O(n)

Q. Reverse a stack

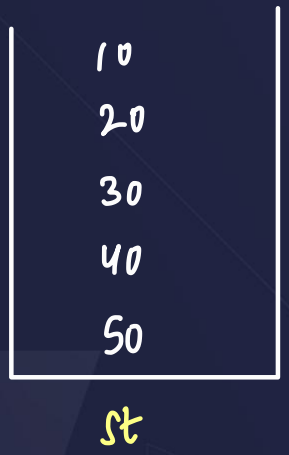
Already done [existing stack empty karke doorse stack me daal rahe hai]



Hint : Use two extra stacks

Q. Reverse a stack

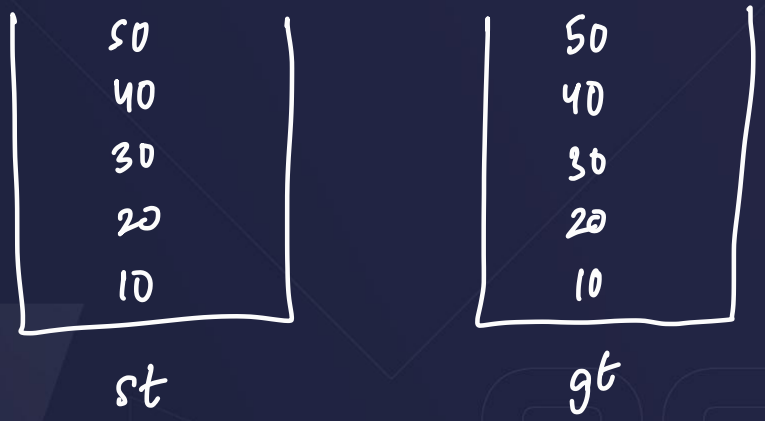
Using an extra array



	0	1	2	3	4
vector<int> v = {	50	40	30	20	10 }

Q. Copy stack into another stack in same order

Homework

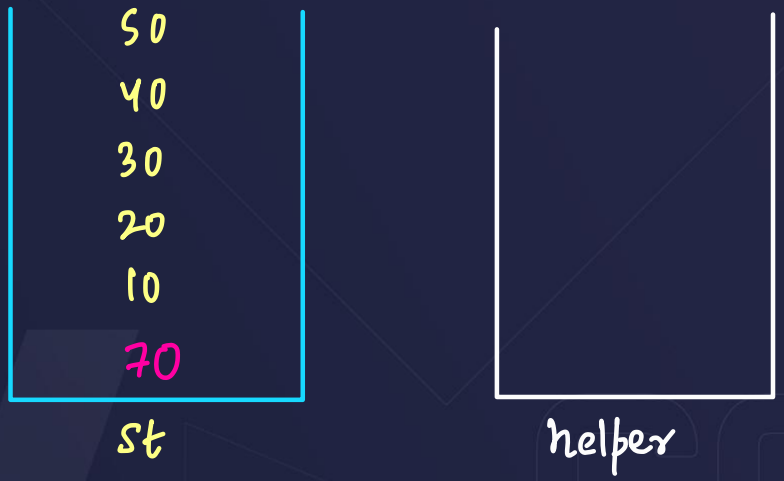


Q. Display stack

done ✓

Q. Push element at bottom / any index

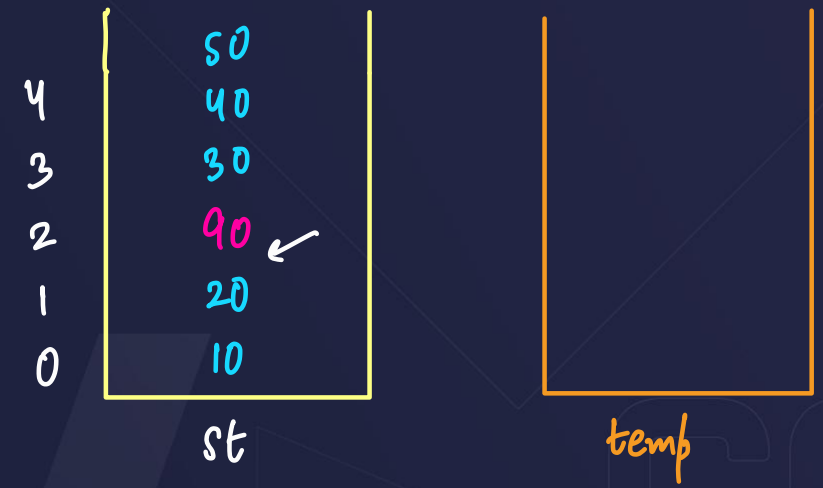
push at bottom (70)



Hint : Use another stack

Q. Push element at bottom / any index

pushAtIndex(st, 2, 90)




```
while (st.size > idx)
    temp.push(st.top());
    st.pop()
}
st.push(val)
```

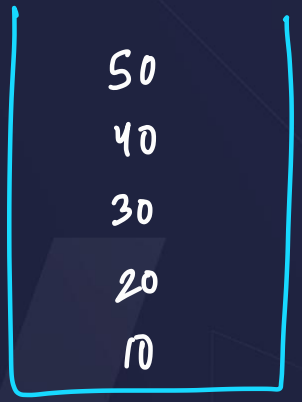
Q. Reverse stack recursively

S.C. = O(n) → Call stack

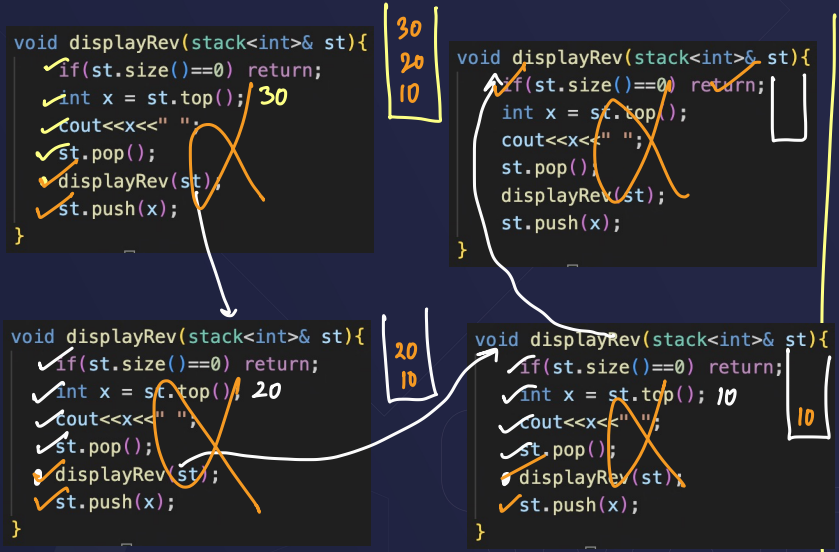
1

Display a stack using recursion

Rev display



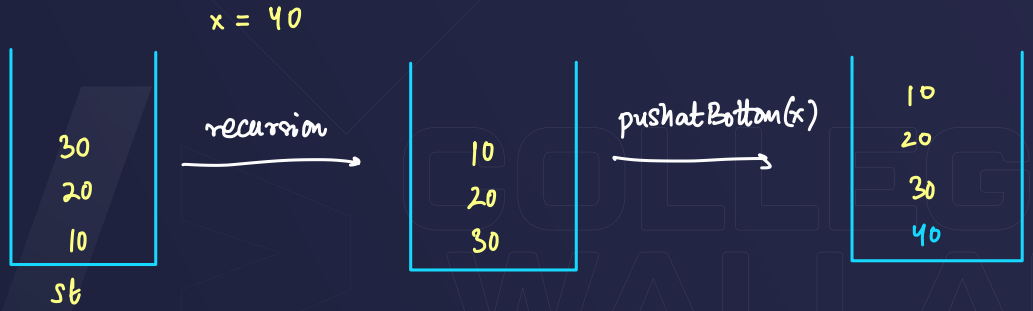

```
void displayrec(st){
    cout << st.top();
    int x = st.top();
    st.pop();
    displayrec(st);
    st.push(x);
}
```


void displayRev(stack<int>& st){
if(st.size()==0) return;
int x = st.top();
cout<<x<<" ";
st.pop();
displayRev(st);
st.push(x);
}
30
20
10
Output
30 20 10

Push at Bottom : (Recursive)

displayRec → print mat karao & base case me add

'Reversing a stack'



Overflow

If your stack is full, then if you try to push an element

Underflow

If in an empty stack, & try to perform these 2 functions → st.pop()
st.top()

Array / Vector Implementation

idx = -1 0 1 0

```
st.push(70)
st.push(40)
st.pop()
```

```
int size(){
    return idx+1;
}
```

```
void push(int val){
    idx++;
    arr[idx] = val;
}
void pop(){
    idx--;
}
int top(){
    return arr[idx];
}
```

Linked List Implementation → 'Zabardast'

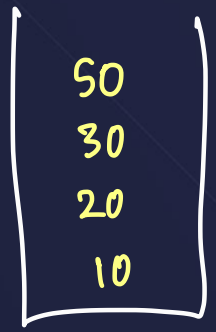
```
class Stack {
    Node* head;
    int size;
    Stack(){
        head = NULL;
        size = 0;
    }
    void push(int val);
    {
        Node* temp = new Node(val);
        temp->next = head;
        head = temp;
        size++;
    }
```

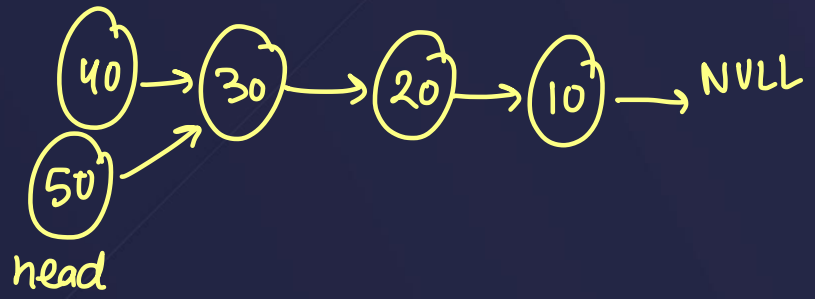
```
void pop()
{
    if(head==null){
    head = head->next;
    size--;
}
```

Linked List Implementation

O(1) → push, pop, top

head = NULL





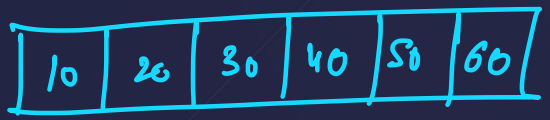

```
push(10)
push(20)
push(30)
push(40)
```

```
cout << top();
pop()
push(50)
```

Linked List VS ~~Vector~~ Array Implementation

- Unlimited Size ✓

- Display → O(1) space



Stack - Maza aa gaya

THANK YOU!