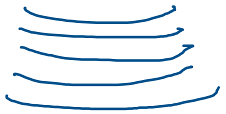
Stacks, July 23

23 July 2020 18:54

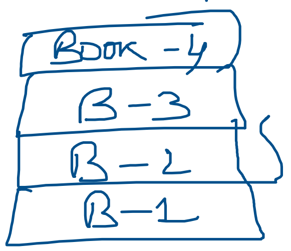
What is a Stack? → Pile of Things

Plate



↳ FILO

↳ LIFO



→ LIFO

Stack

In Computers → Function Calls → Memory Stack

Ctrl - z → Undo → Stack

Code Parser

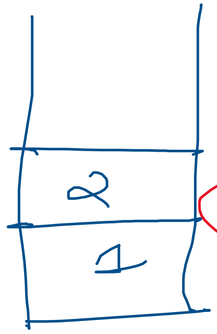
↳ { { }

Checking Parenthesis ⇐
is done using

Stacks ✓

ABSTRACT DATA TYPE → list of operations

Stack ADT



Stack

O(1)

1. Insert → Push
2. Delete → Pop [Deleting + Returning the deleted]
3. Read the topmost element → Peek / Top
4. Check if stack is empty → is Empty → True/False

Array ✓

Linked Lists

Implementation of Stack using Array

Size = 10

top

0	1	2	3	4	5	6	7	8	9
1	2	3	~~4~~	5	0	0	0	0	0

$\boxed{top = -1}$ Initially

```
boolean isEmpty()
{ if (top == -1) return true;
  return false;
}

void insert(int data)
{ top = top + 1;
  arr[top] = data
}

int peek()
```

[illegible]

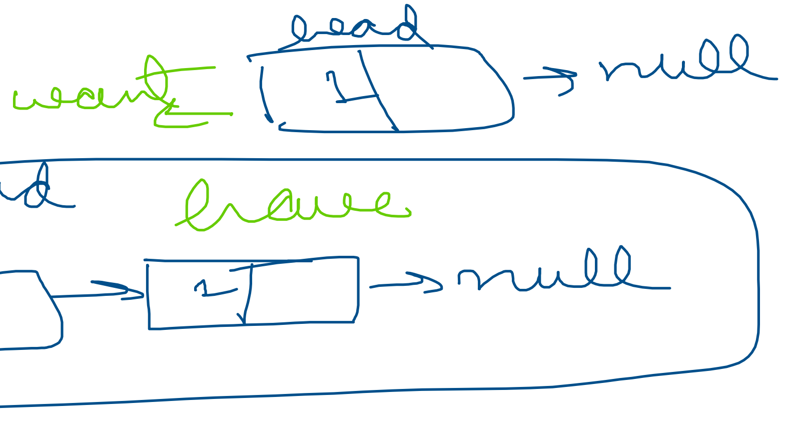
```
{
   return arr[top];
}

int pop()
{  int x = arr[top];
      top--;
   return x;
}
```

Implementation of Stack using Linked Lists

4 functions

push
pop → O(1)

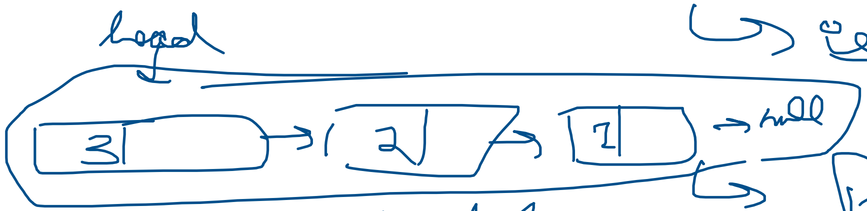


is Empty
peek

↳ ~~InsertAtHead~~

↳ head = head.next (Pop)

↳ isEmpty   if (head == null)



↳ Peek → (return head.data)

```
int x = head.data;
head = head.next;
return x;
```

Balanced Paranthesis

→ String

list	{ }	✓
{ (No Pair)	{]	×
Not open	) (	×
{ [	{ [ ] }	✓
{ [	{ [ ) }	×
{	{ } [ ]	✓

① length of String
Odd → false
even → Continue

② Create a Stack of character

③ Traverse the String
→ If opening braces
↳ Store it

LIFO
type
Stack

{ [} ×

{ ( ) }] ×

~~2~~

{ } [[×

→ If closing braces
1. Stack not to be empty
2. Match as pair with top of stack
3. if No Match return false

4. Stack to be empty therefore
if Not → false