unacademy

Stack and Queue - Part II

Course on Data Structure

Pankaj Sharma • Lesson 15 • July 25, 2024

unacademy

CS & IT Engineering

Data Structure

Stack & Queue

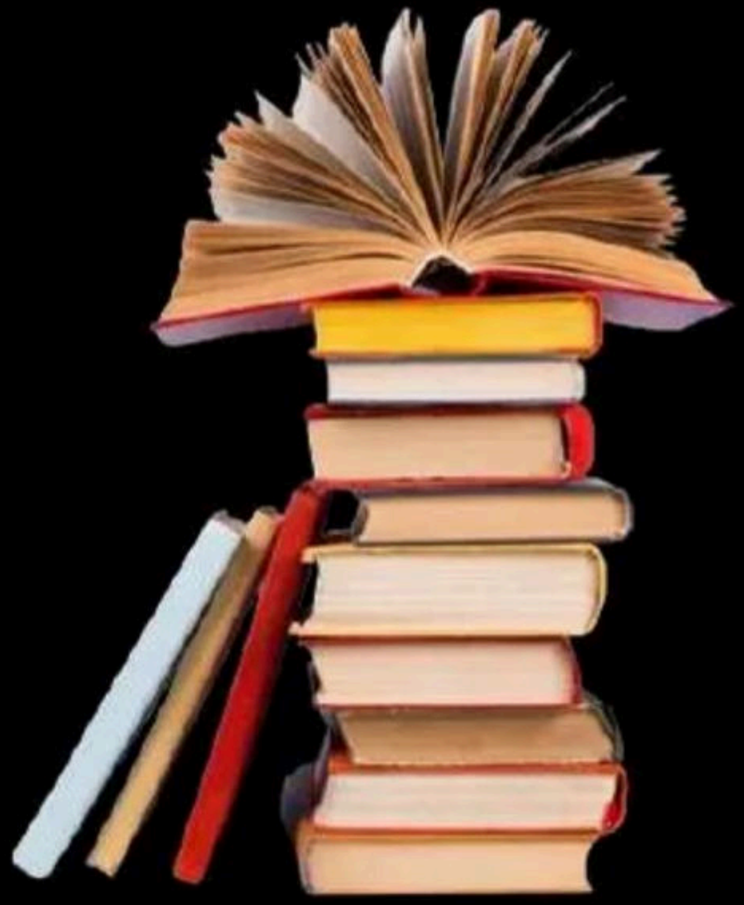
Lecture Number- 15

By- Pankaj Sir

unacademy

Topics

to be covered



1. Stack

Data structure

(1) Abstract view

- No coding
- No implementation
- No prog. lang.

- features / operations defined on a data structure.

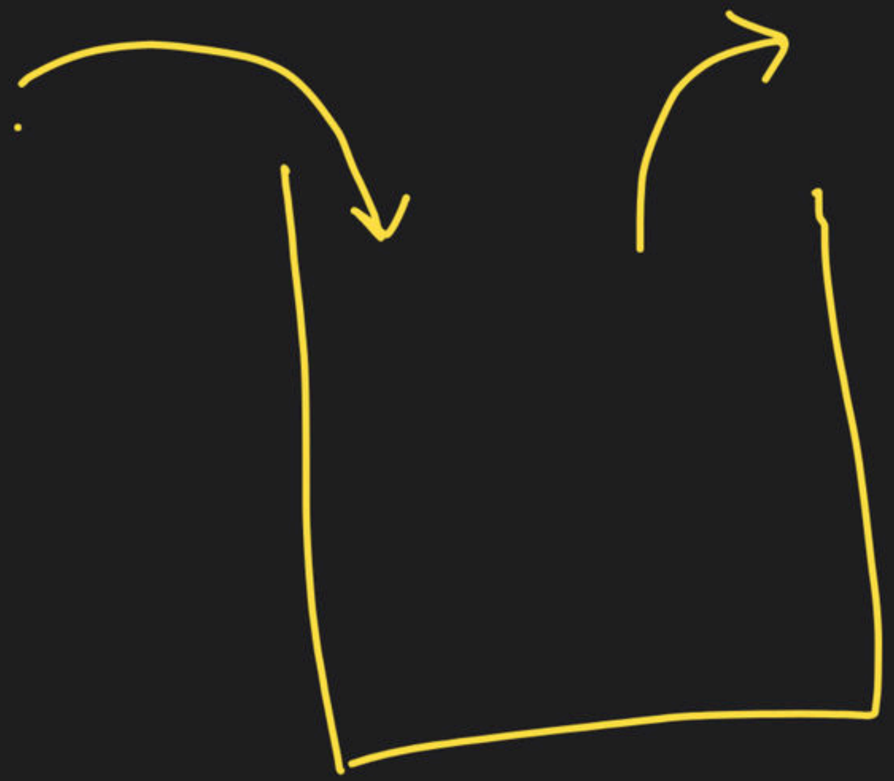
(2) Concrete view

- Implementation
- Prog. lang.

Stack

* Linear data structure
* Deletion order ⇒ reverse order of insertion.
* Works on Last-In First out policy (LIFO)
* Both insertion & deletion are performed only at one end ⇒ Top of stack.

stack as ADT



unacademy

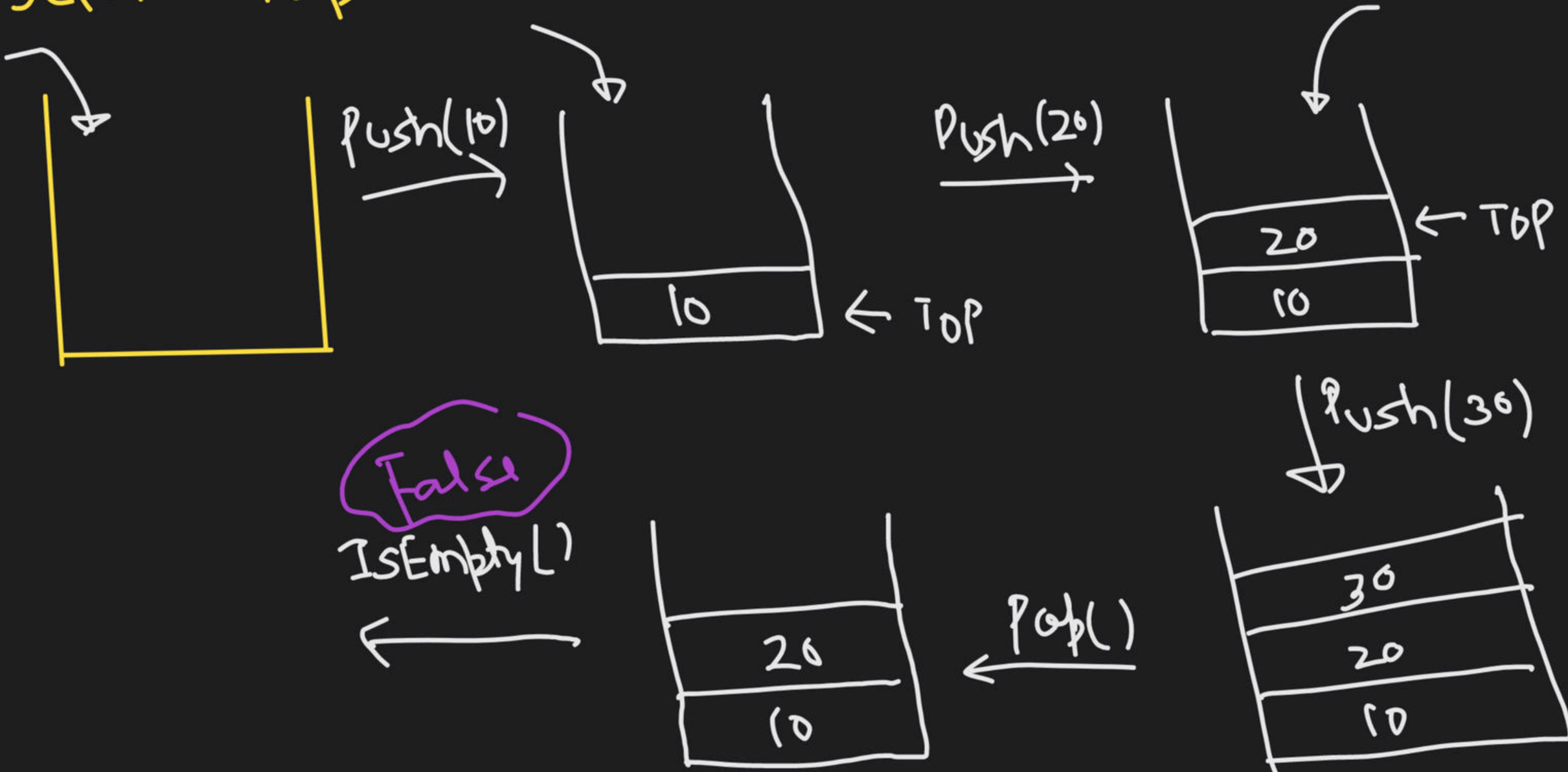
Stack of numbers

Initially Stack is Empty

Top: { Points to most recently added element }

Insert ⇒ Push

Delete ⇒ Pop

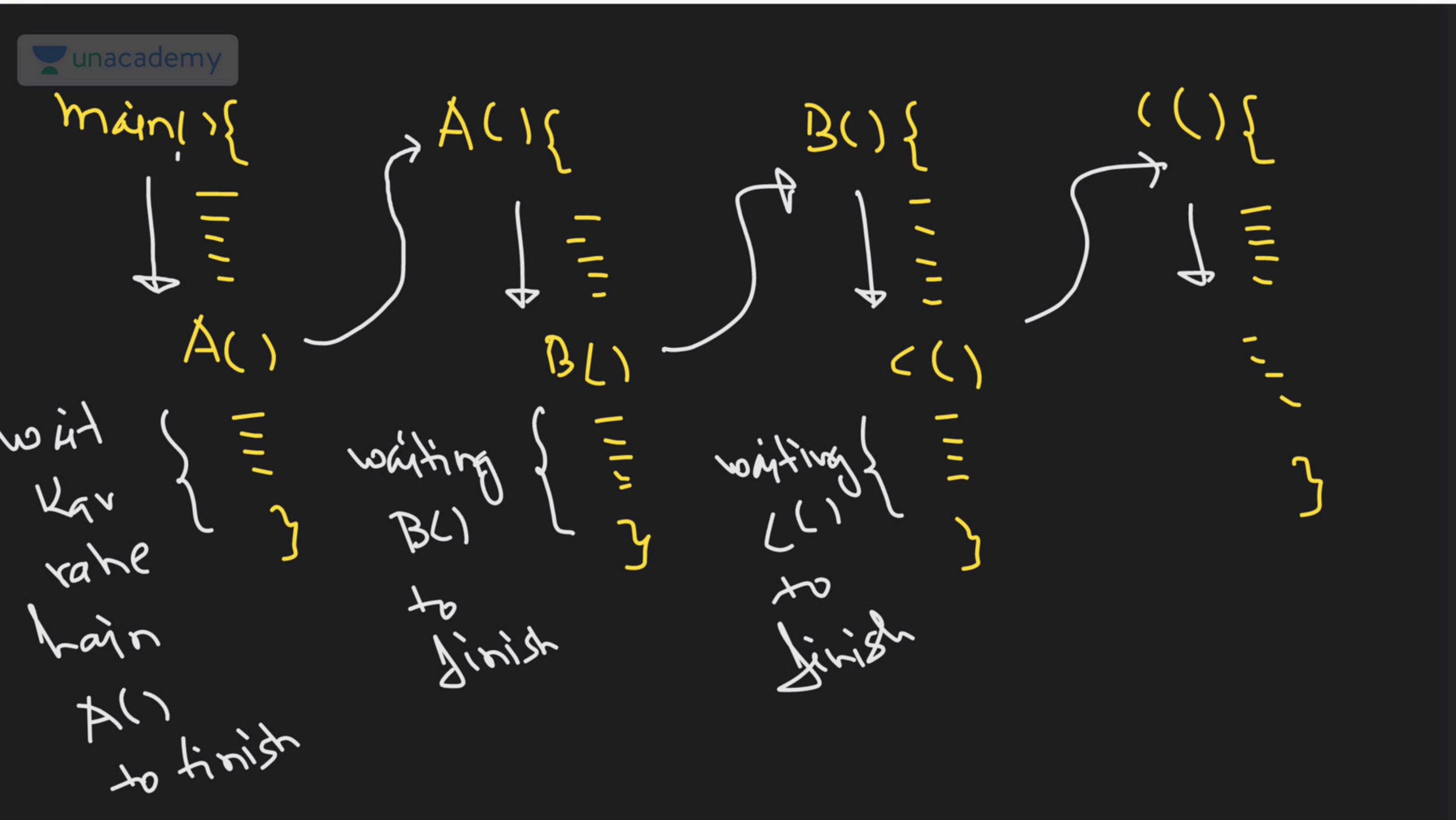


IsFull() →

Applications

1) Towers of Hanoi
2) Depth First Search
3) Infix to prefix
4) Infix to postfix
5) Postfix Evaluation
6) Prefix Eval.
7) Recursion
8) Parenthesis Matching

Wait Kanwana
To delay / postponed decision

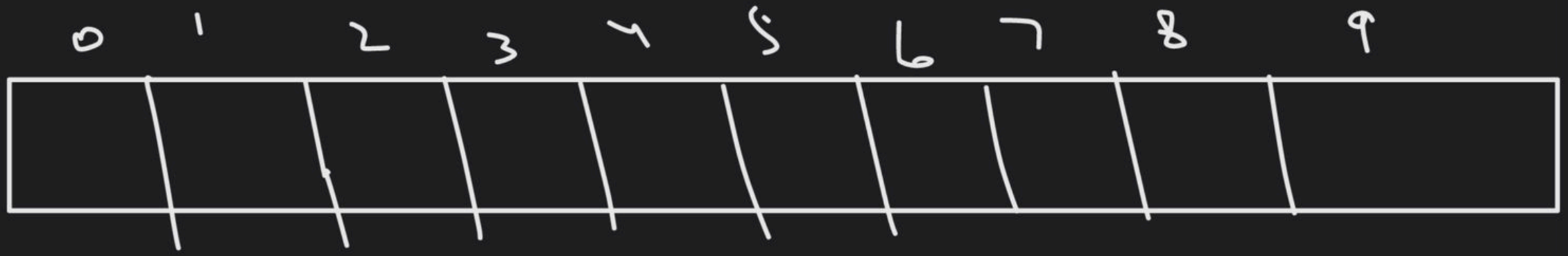

unacademy
main(){
A()
wait Kar rahe hain A() to finish
A(){
B()
waiting B() to finish
B(){
C()
waiting C() to finish
C(){
}

unacademy

Stack

① Array → sized fixed
static

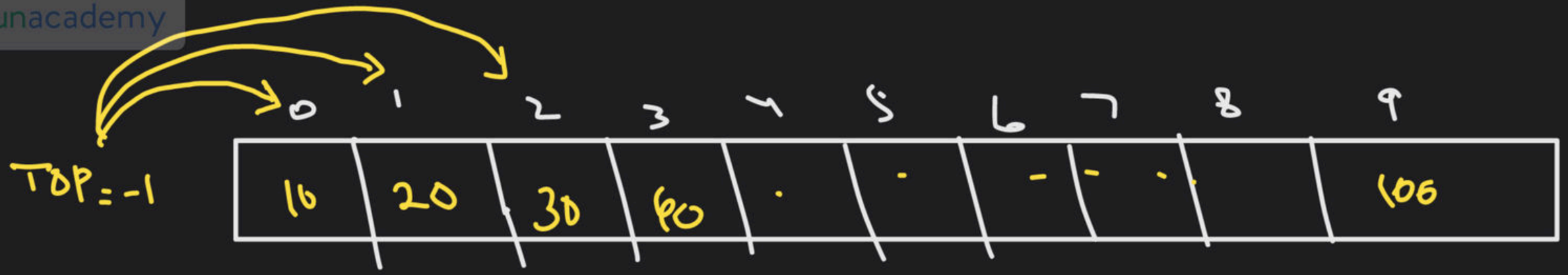
#define SIZE 10
int STACK[SIZE];



TOP: Most recently added element

Initially

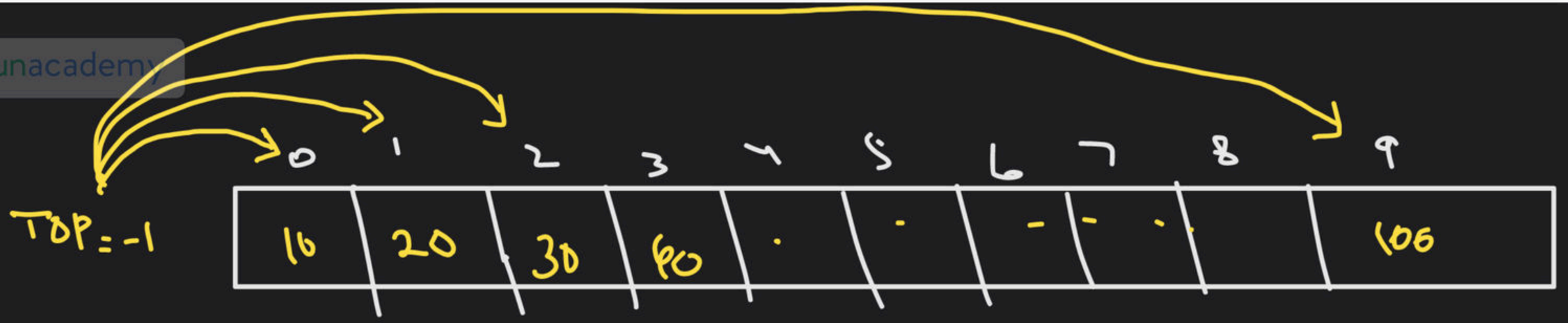
TOP = -1



1) Push(10)
2) Push(20)
3) Push(30)
:
9) Push(90)
10) Push(100)

1. Overflow condition

```
void Push(int x){
    TOP = TOP+1 ;
    STACK[TOP] = x ;
}
```



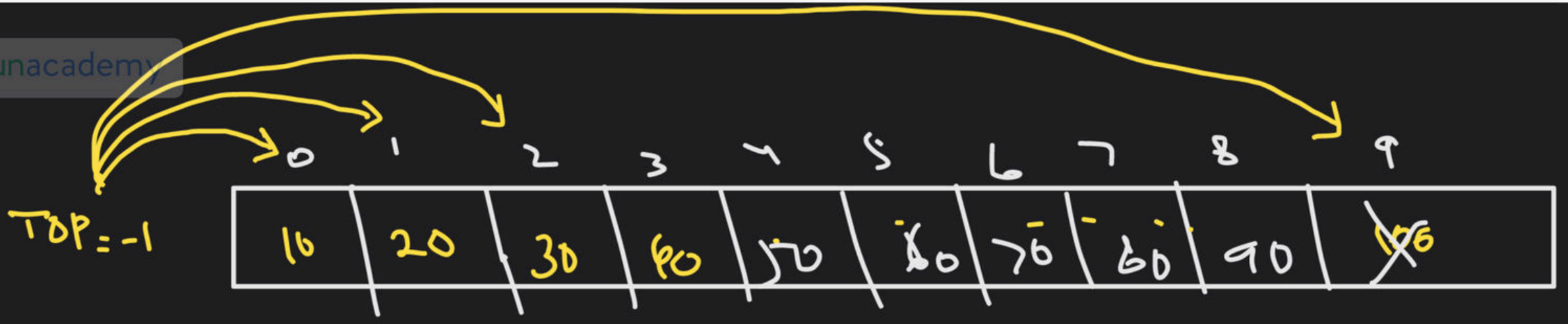
No loop ✗
No recu ✗

Constant
O(1)

```
void push(int x){
if(Top == SIZE-1){
    pf("overflow");
    return;
}
Top = Top+1;
STACK[Top) = x; }
```

SIZE 10

last valid index = 9 (SIZE-1)



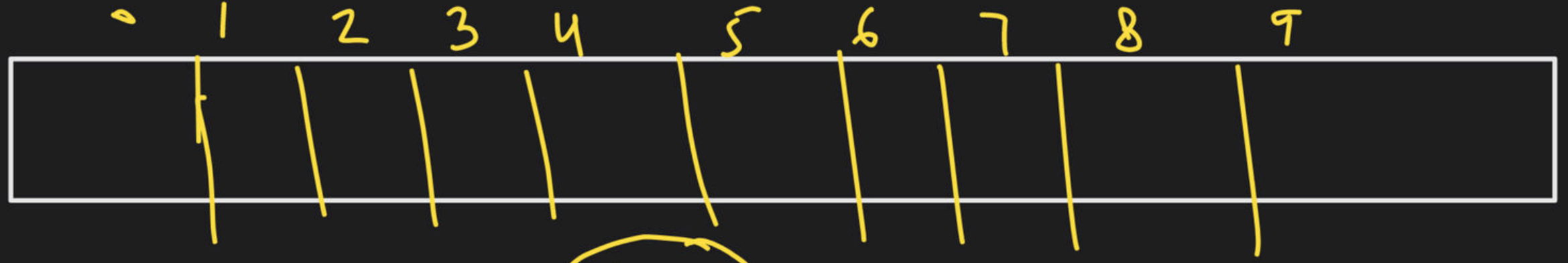
Input/output

int Pop() {

```
int temp;
temp = STACK[TOP];
TOP = TOP - 1;
return temp;
}
```

10, 20, 30, ---, 70, 80, 90, 100

TOP = -1



int pop() {

int temp;

if (TOP == -1) {

return INT_MIN;

}

temp = STACK[TOP];

Top--;

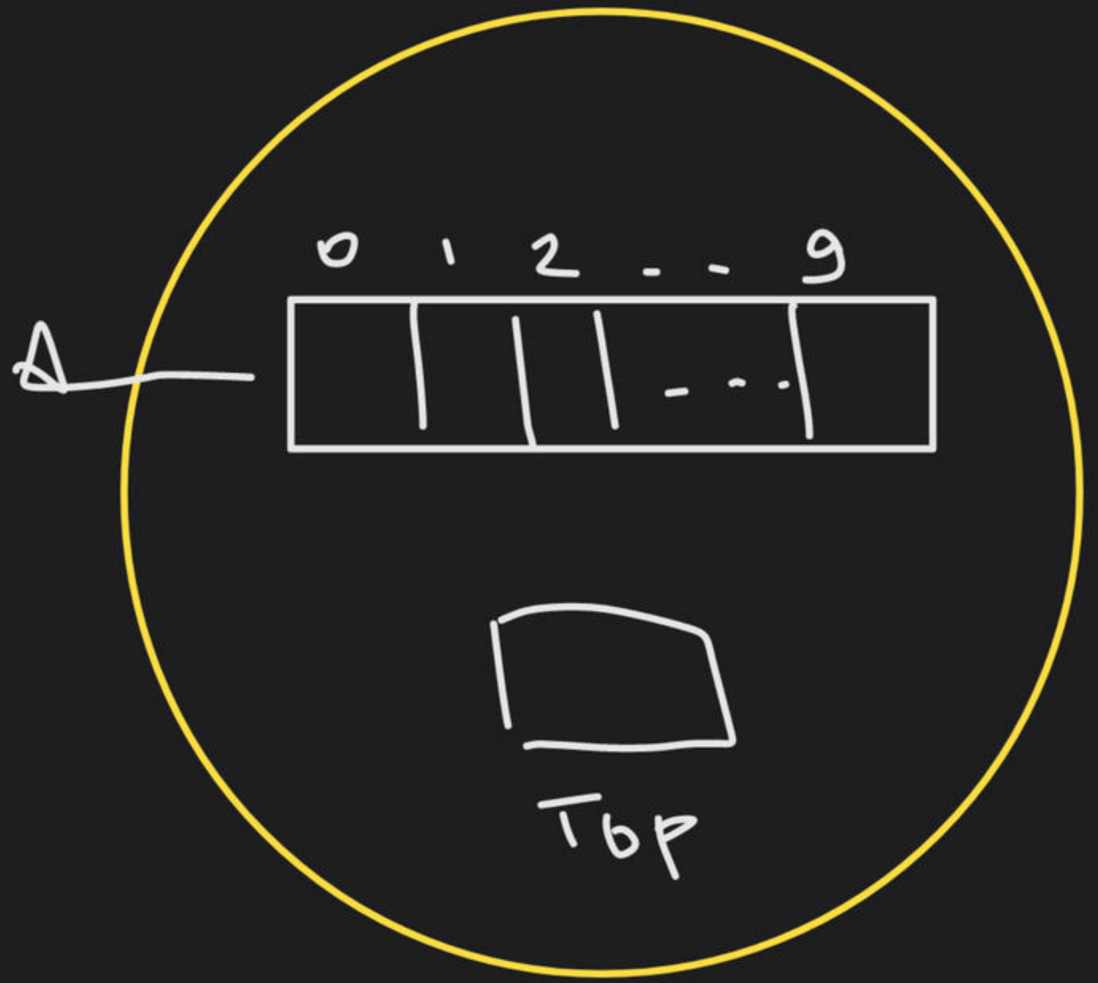
return temp; }

lonstant

```
#define SIZE 10
int STACK[SIZE];
int Top = -1;
void Push(int x) {
    if (TOP == SIZE-1)
        return;
    Top++;
    STACK[Top] = x; }
int Pop() {
    temp = STACK[Top];
```

Problem

2 stack

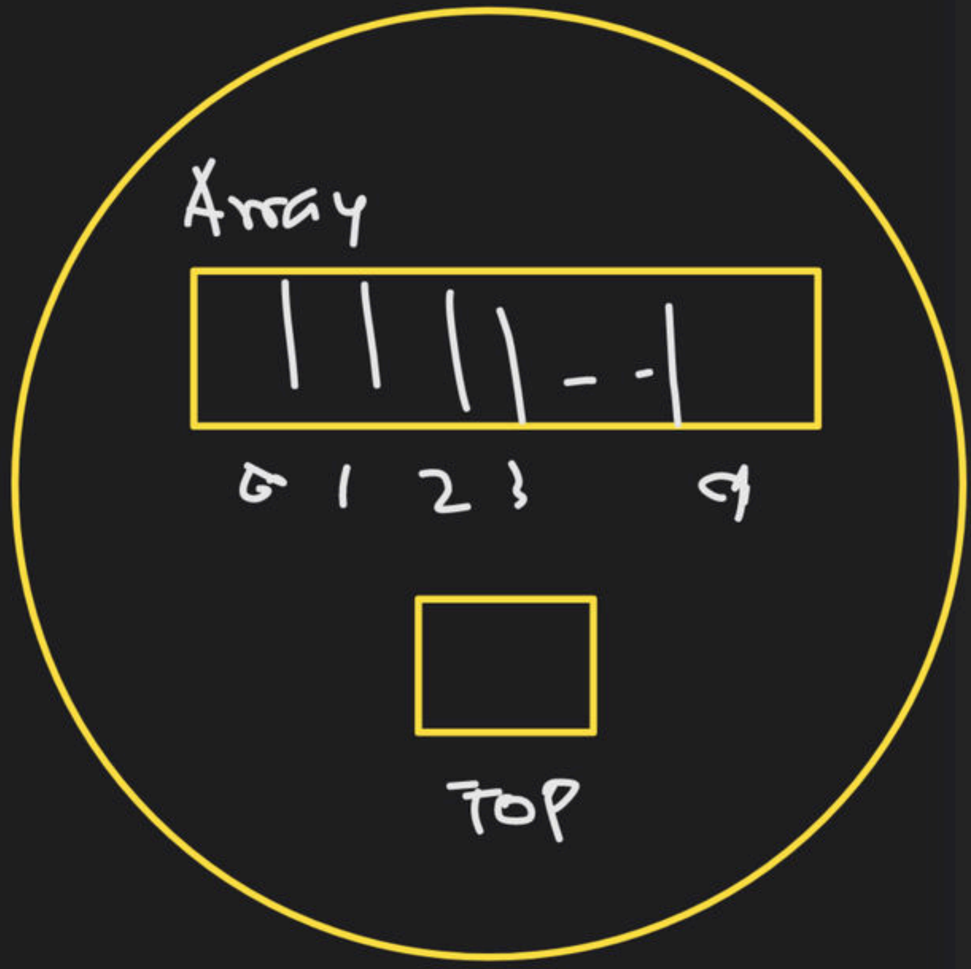


unacademy

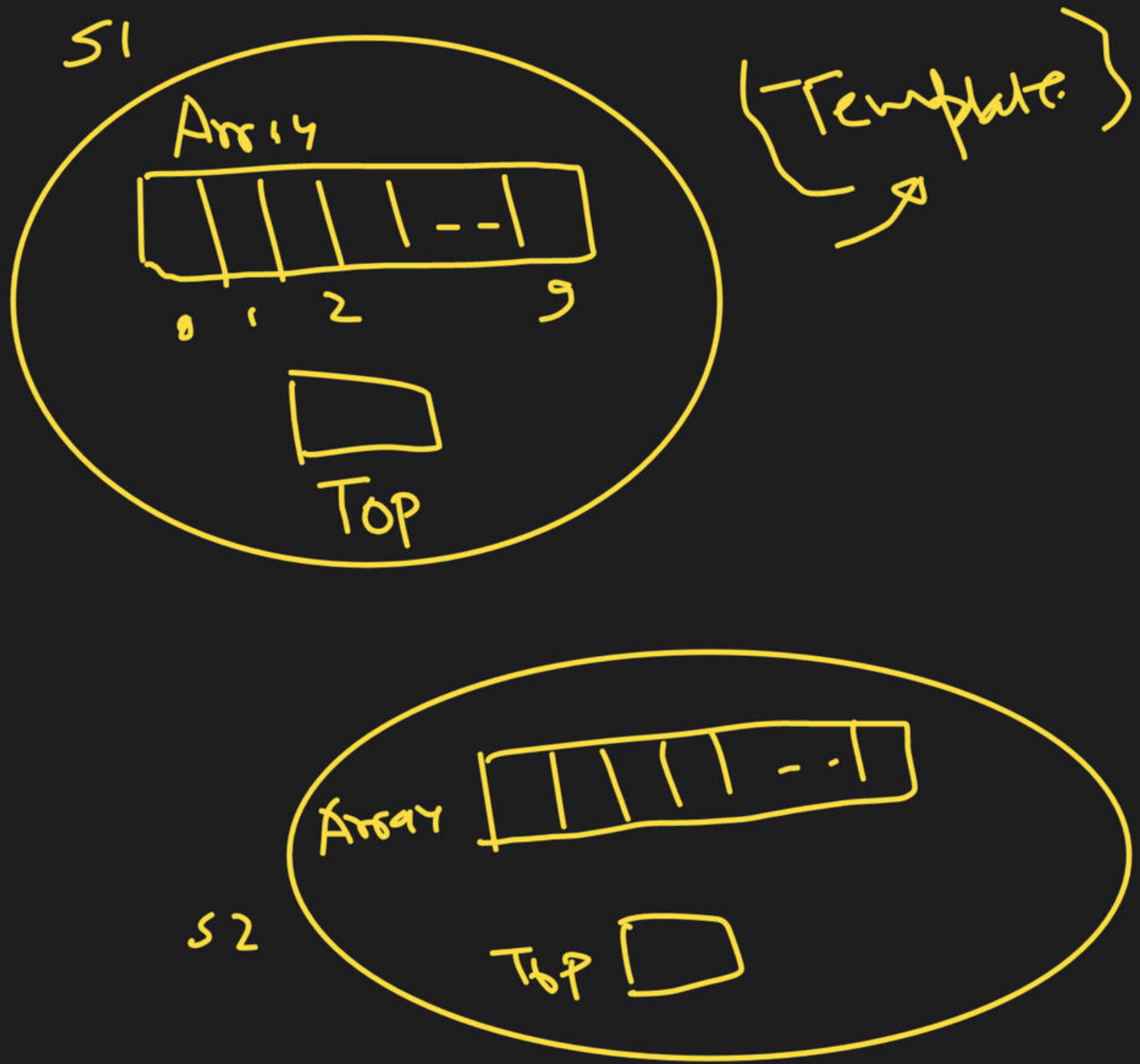
NO memory is allocated

```
#define SIZE 10
struct STACK{
  int Array[SIZE];
  int TOP; };
```

```
void main(){
  struct STACK s;
  s.Top=-1;
  =
}
```



```
#define SIZE 10
struct STACK{
  int Array[SIZE];
   int  Top; };
void main(){
  struct STACK S1,S2;
  S1.Top=-1;
  S2.Top=-1;
```



unacademy

Push → In which stack we want to insert

Pop → From which stack

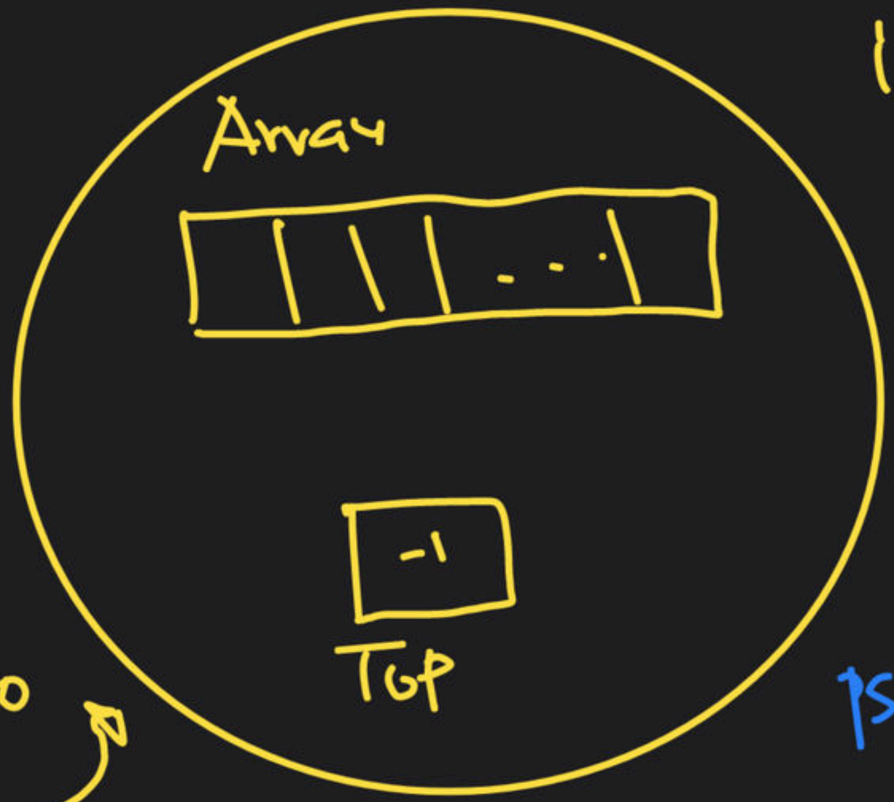
```
void main()                void Push (struct STACK t, int x)
{                          {
  ===
      call by value
  Push(s1, 10);
    ==
  }                        }  <=
```

```
void main(){

struct STACK S1,S2;
S1.Top=-1;
S2.Top=-1;
Push(&S1,10);

}
```

≡

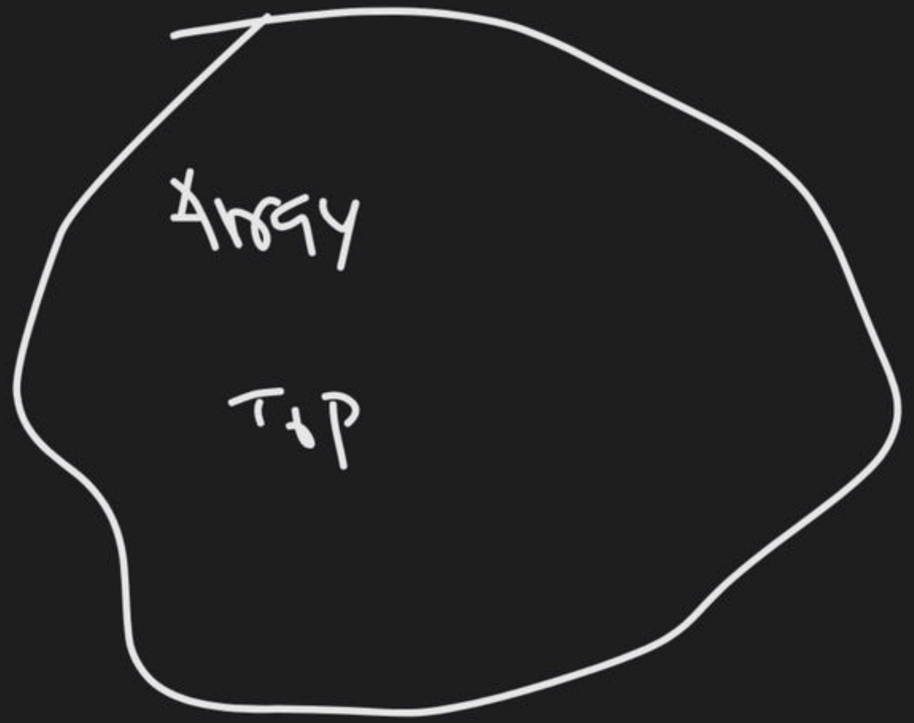



```
void Push(struct STACK *ps,
          int x)
{
if(PS->Top==SIZE-
    return;
Top=~~Top~~+1;
PS->Top=PS->Top+1;
~~Array~~(Top)=x
PS->Array[PS->Top]=x;
}
```

Structure

P ⇒ Pointer to structure

PS ⇒



P → member1

P → member2

PS → Array [PS → Top]

unacademy

THANK YOU!

Here's to a cracking journey ahead!