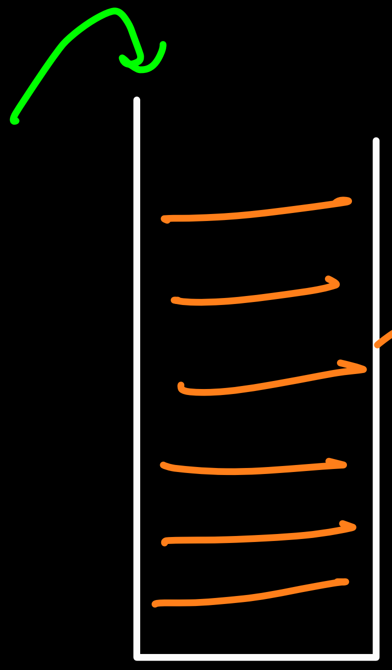
Stacks



stack of plate

closed

LIFO

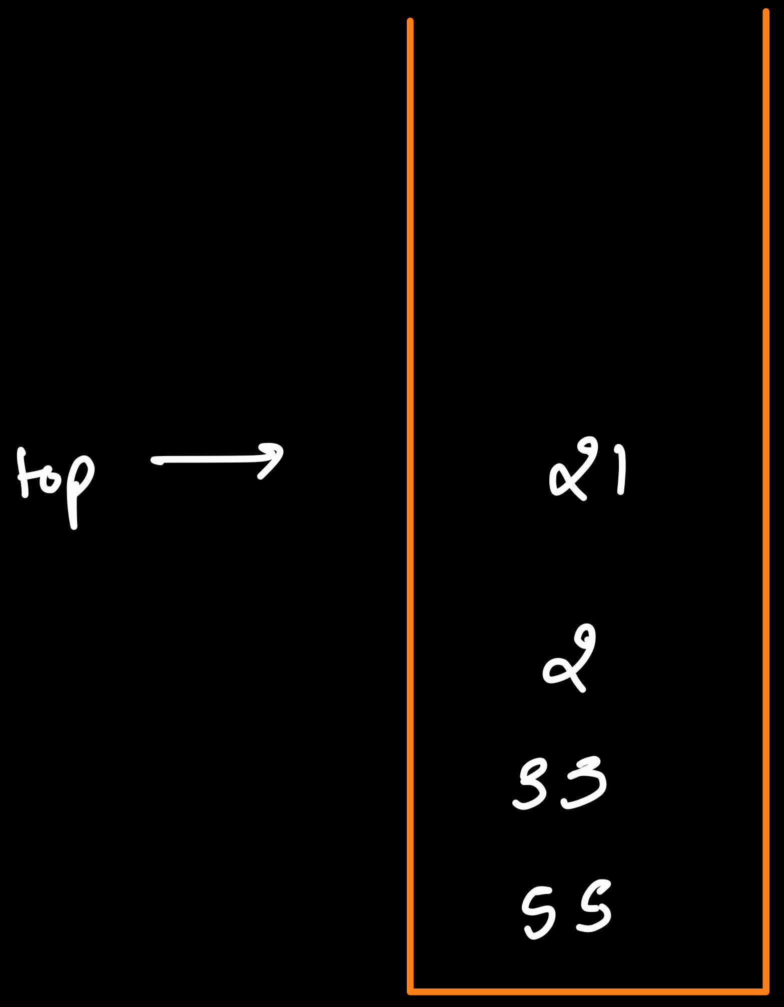
last in first out

memory → call stack

- linear data structure
- data can be only accessed from one end of the stack
- apart from the topmost element no other element is directly accessible in the stack.

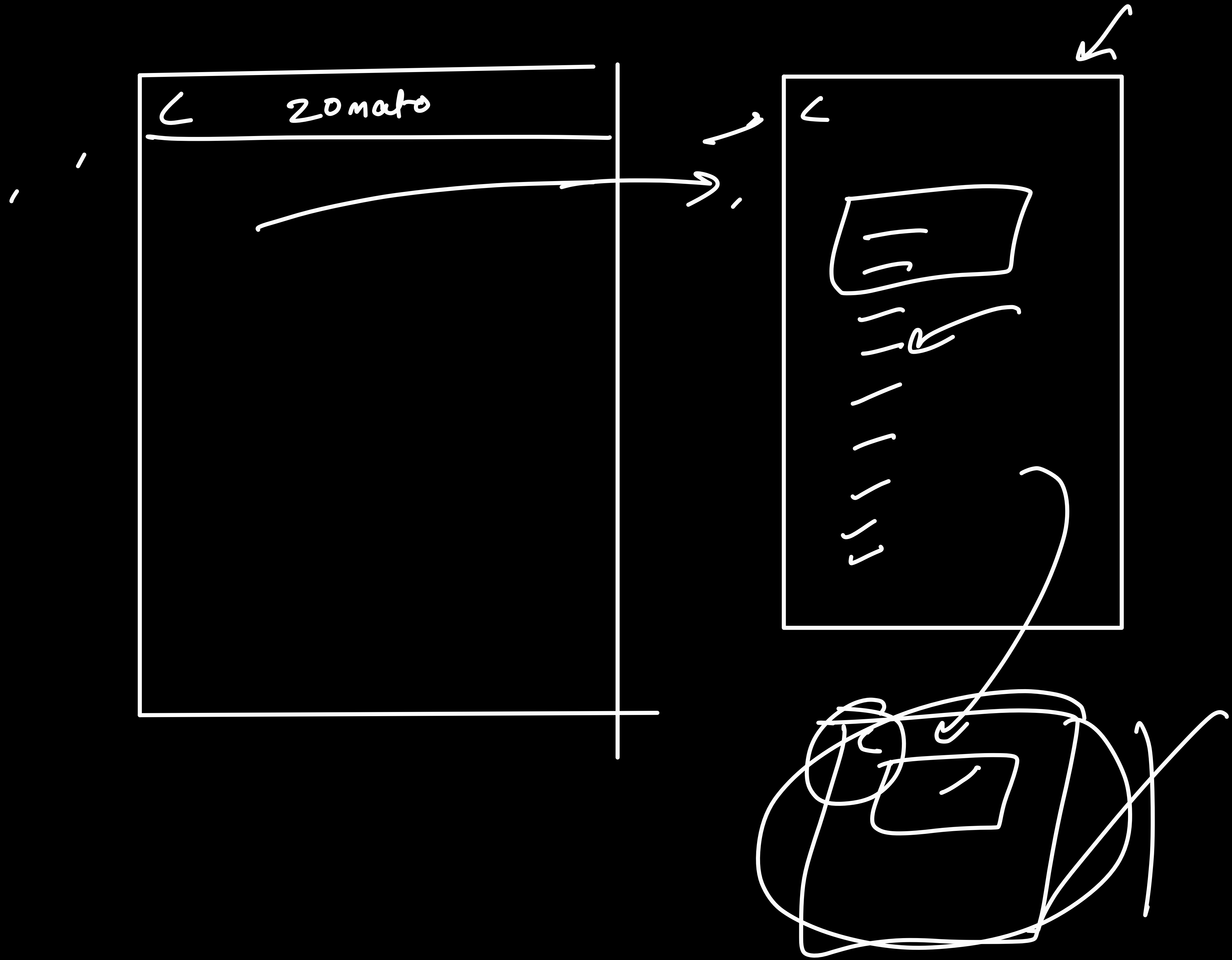
→ application

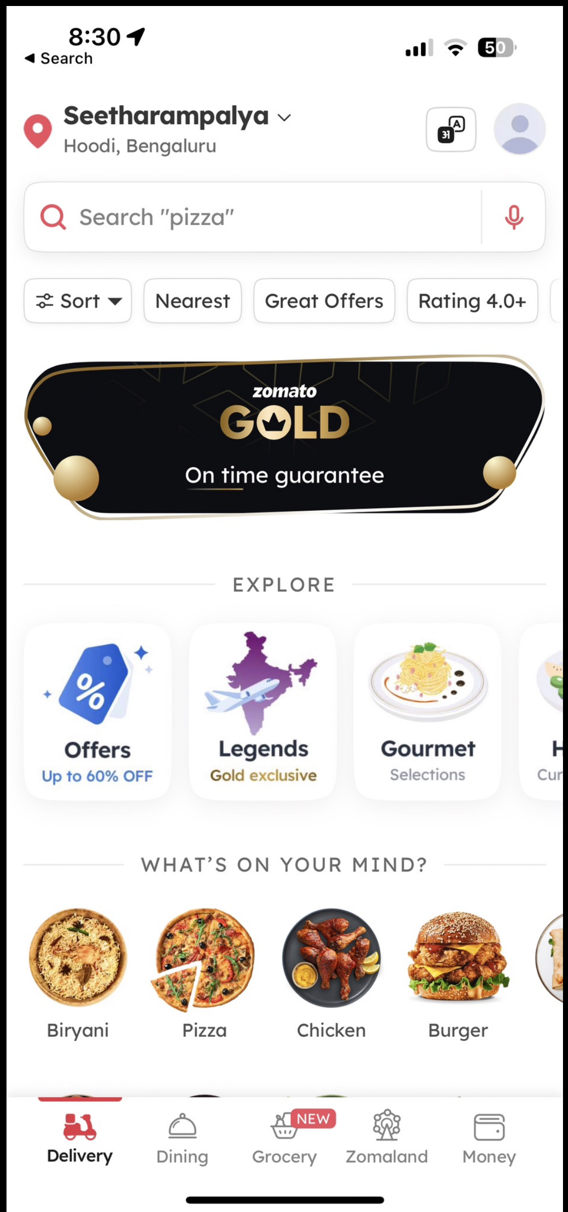
, , , , , 6



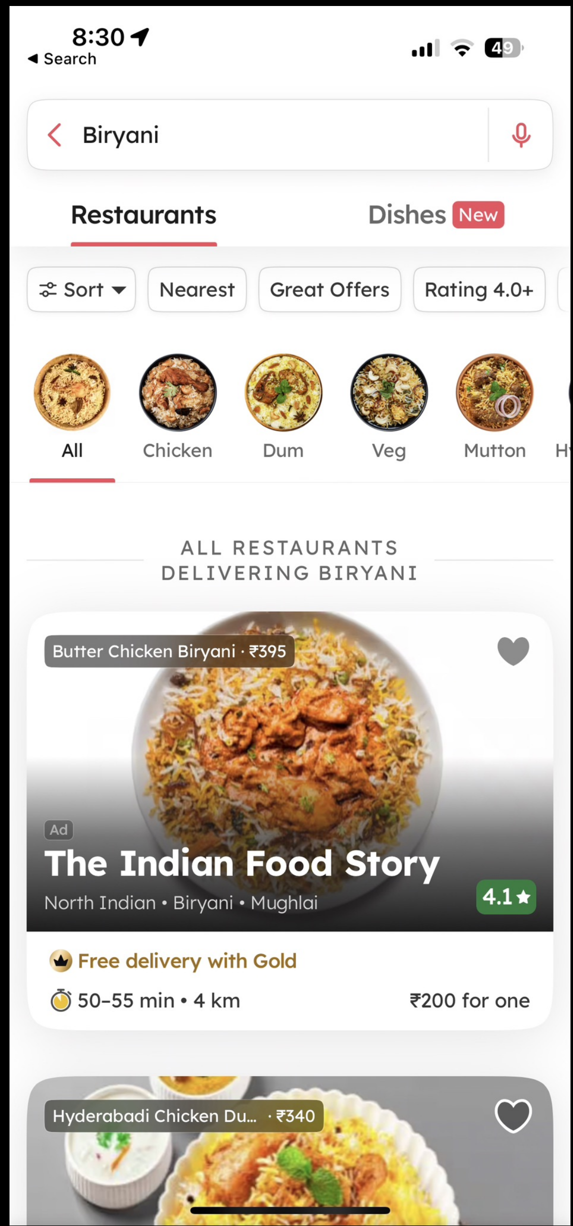
LIFO

19

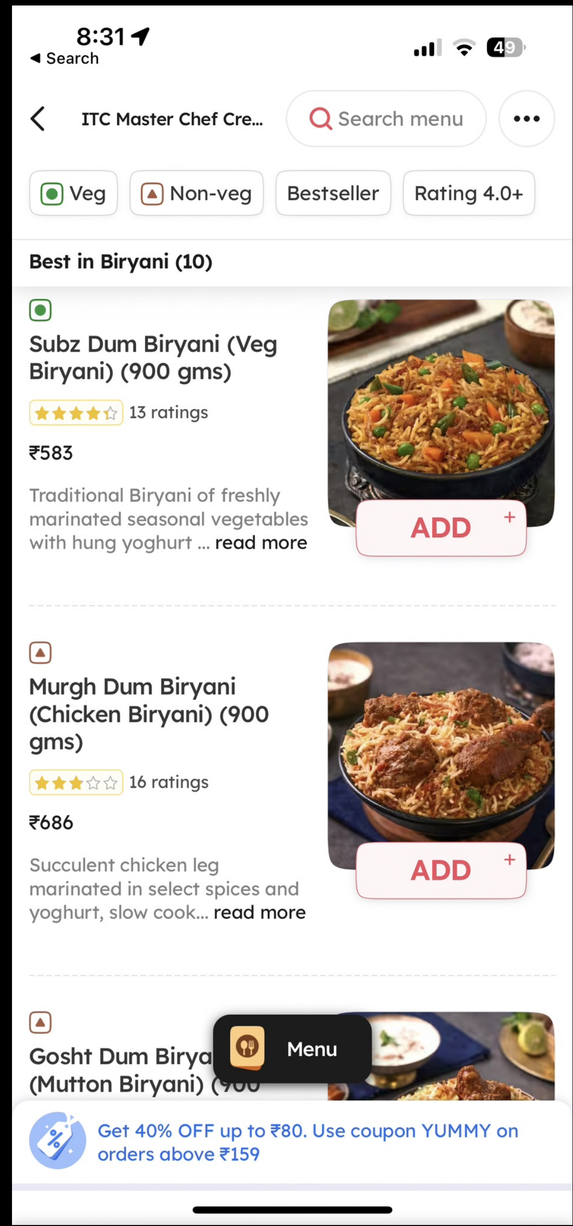

zomato


8:30
◂ Search
50
Seetharampalya
Hoodi, Bengaluru
Search "pizza"
Sort
Nearest
Great Offers
Rating 4.0+
zomato
GOLD
On time guarantee
EXPLORE
Offers
Up to 60% OFF
Legends
Gold exclusive
Gourmet
Selections
WHAT'S ON YOUR MIND?
Biryani
Pizza
Chicken
Burger
Delivery
Dining
NEW
Grocery
Zomaland
Money

1


8:30
◂ Search
49
Biryani
Restaurants
Dishes
New
Sort
Nearest
Great Offers
Rating 4.0+
All
Chicken
Dum
Veg
Mutton
ALL RESTAURANTS
DELIVERING BIRYANI
Butter Chicken Biryani · ₹395
Ad
The Indian Food Story
North Indian • Biryani • Mughlai
4.1★
Free delivery with Gold
50-55 min • 4 km
₹200 for one
Hyderabadi Chicken Du... · ₹340

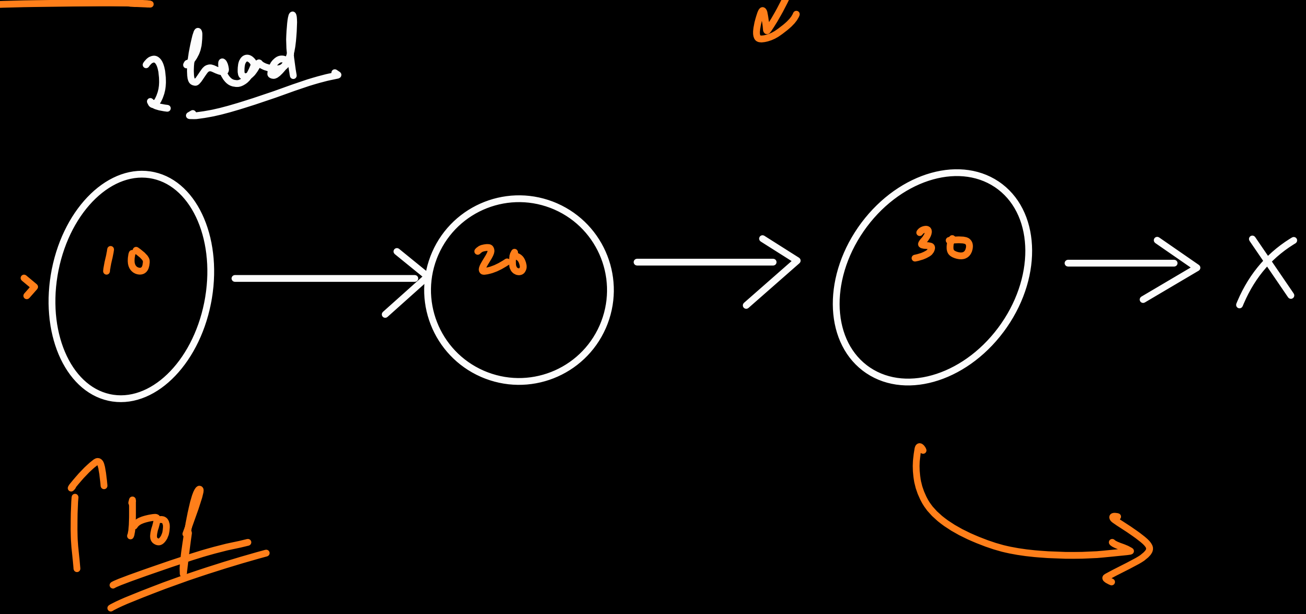
2

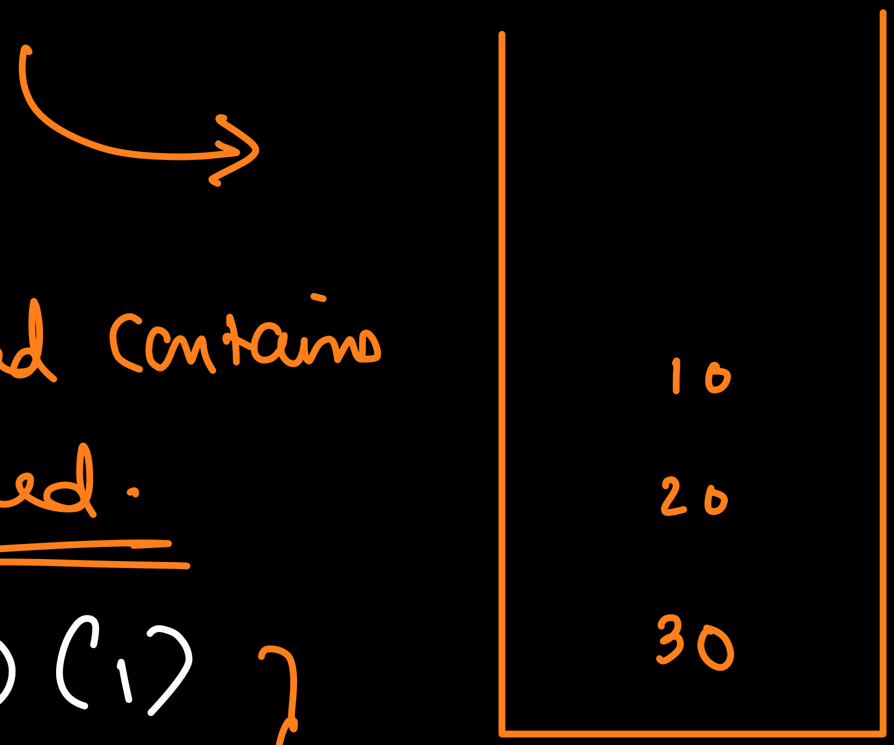

8:31
◂ Search
49
ITC Master Chef Cre...
Search menu
Veg
Non-veg
Bestseller
Rating 4.0+
Best in Biryani (10)
Subz Dum Biryani (Veg Biryani) (900 gms)
13 ratings
₹583
Traditional Biryani of freshly marinated seasonal vegetables with hung yoghurt ... read more
ADD
Murgh Dum Biryani (Chicken Biryani) (900 gms)
16 ratings
₹686
Succulent chicken leg marinated in select spices and yoghurt, slow cook... read more
ADD
Gosht Dum Birya
(Mutton Biryani) (900
Menu
Get 40% OFF up to ₹80. Use coupon YUMMY on orders above ₹159

Implementation of Stacks ←

↳ add → push

→ Linked Lists





whenever we do addAtHead, head contains the last element that we have added.

push → add → addAtHead → O(1)

pop → remove → removeAtHead → O(1)

arrays

queue

4

st = [ ]

st.push(70)

st.pop()

( ) (X

) ( )   X

{ [( ) [( )] (( ))] }

$S_1 + S_2 \longrightarrow \underline{S_1 S_2}$ ✓

$(S_1) \longrightarrow$ ✓

any opening bracket

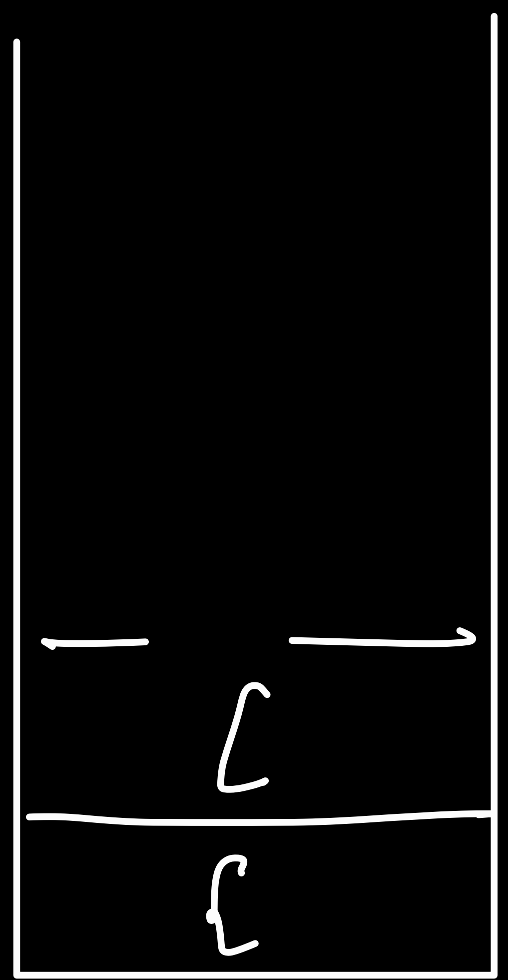
push it in the stack.

{ [( ) [( )] (( ))] }

closing bracket

O(N)

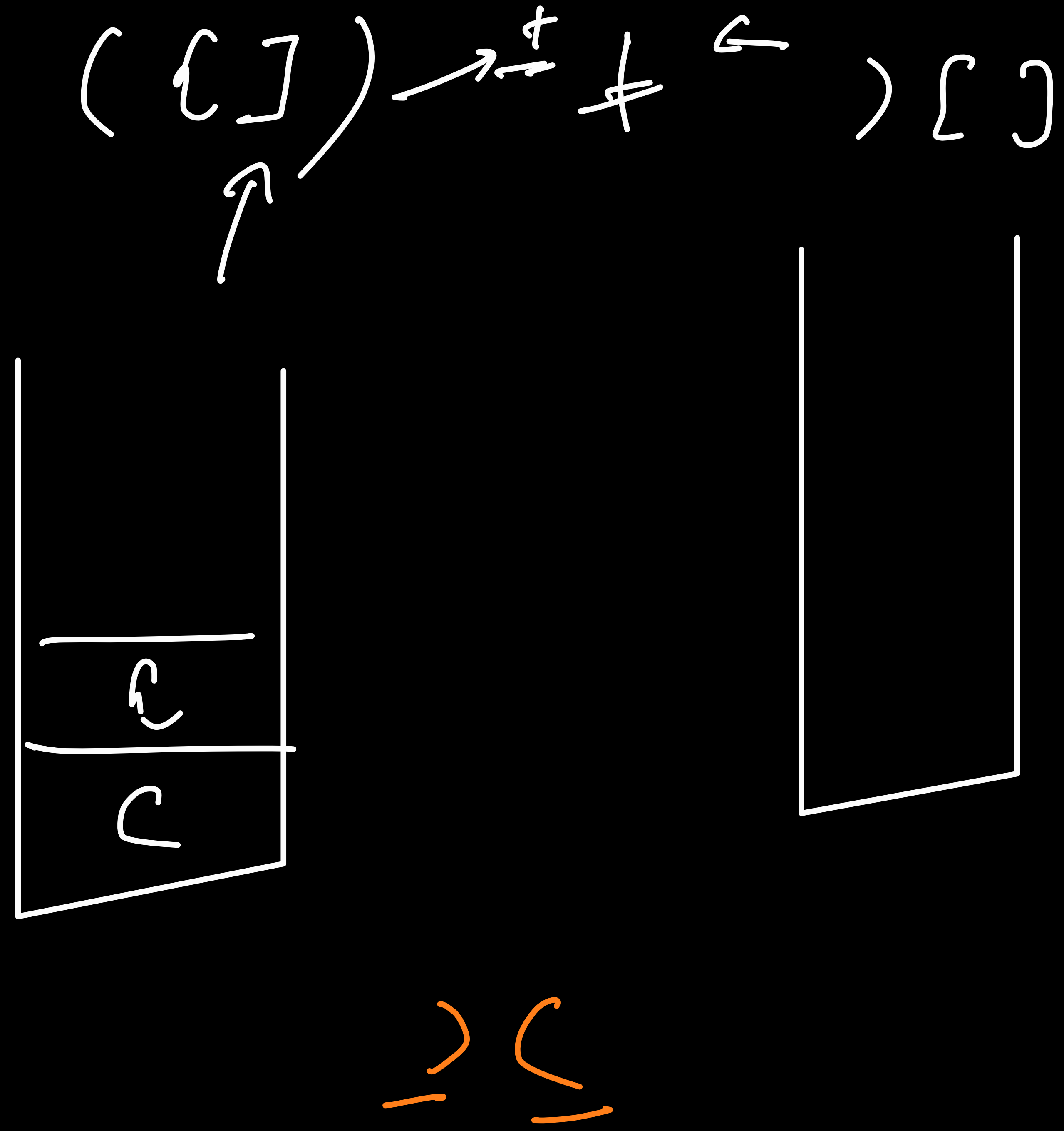
space → O(N)



→ at last stack is empty

balanced

↳ ST

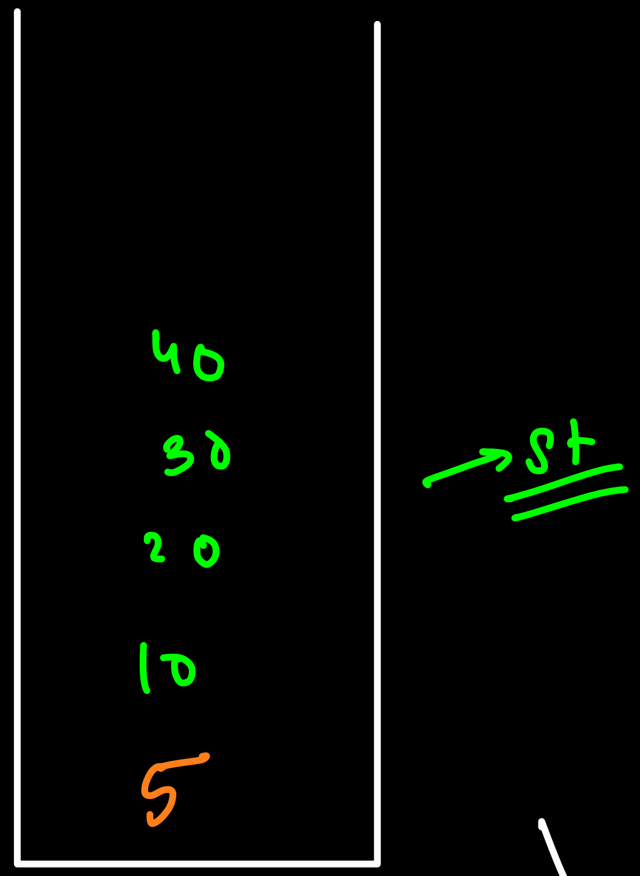


Q: Write a method that takes a stack as an input, and an element x, and then instead of inserting x at top, it inserts x at bottom of stack.

(Time & Space need not to be O(1))

insertAtBottom(st, x) → push, pop, top

Time → O(N)
Space → O(N)



→ Remove all elements one by one and back them up.

→ add x to the empty stack

→ insert the backed up elements in the same order.

insertAtBottom(st, 5)

Pseudocode

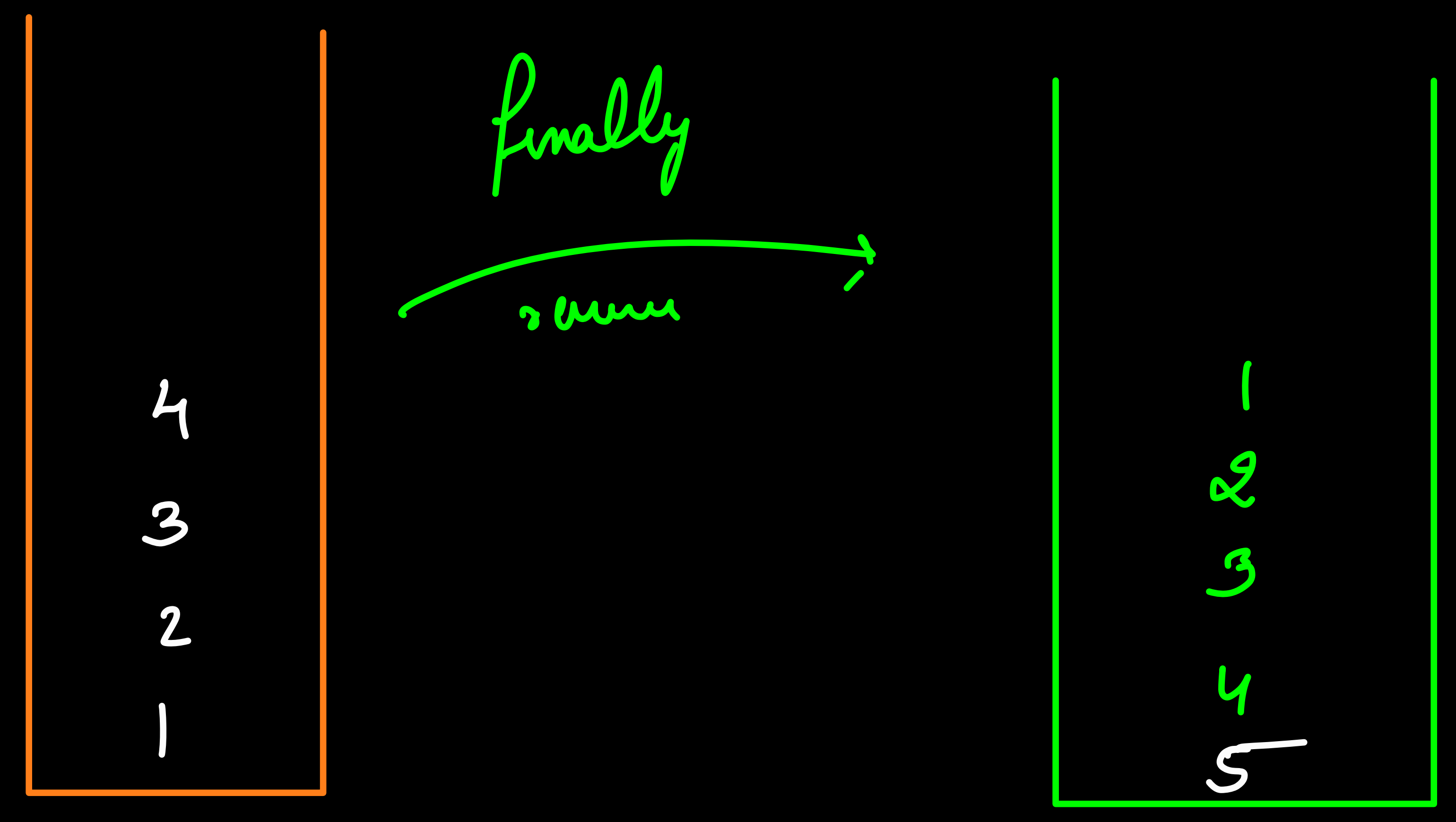
O(N)

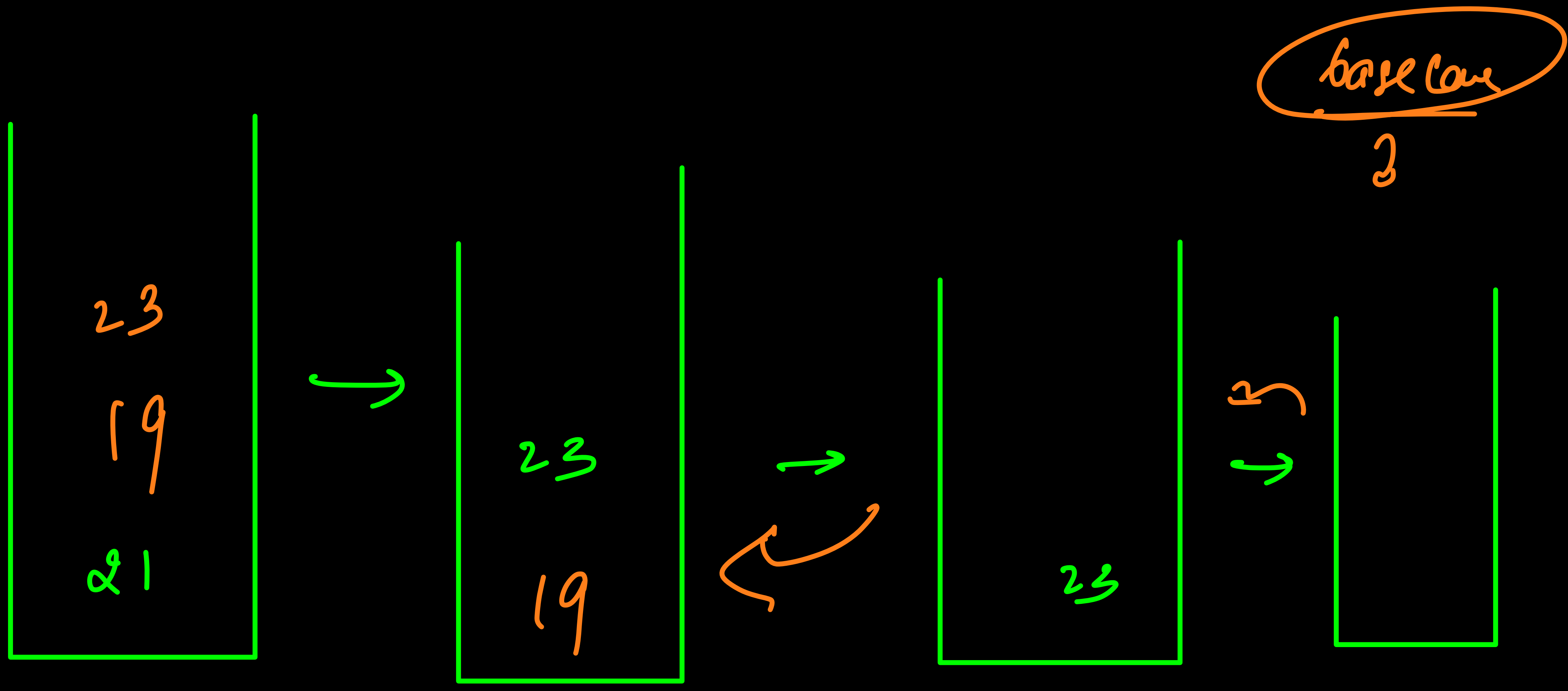
O(N)

```
insertAtBottom(st, x) {
    temp <- temp stack
    while (not st.empty()) {        st.length==0
        temp.push(st.top)           st[st.length-1]
        st.pop()
    }
    st.push(x);
    while (not temp.empty()) {
        st.push(temp.top)
        temp.pop()
    }
    return st;
}
```

→ st → reverse → insertAtBottom → O(N)

Recursively





19

insertAtBottom → $O(N)$

for every element, we are calling insertAtBottom.

$$O(N \times N) \rightarrow O(N^2)$$

$$\text{Space} \rightarrow O(N)$$

```
reverse (st) {
    if (st.empty())
        return;

    el = st.top
    st.pop()
    reverse (st);
    insertAtBottom(st, el);

}
```