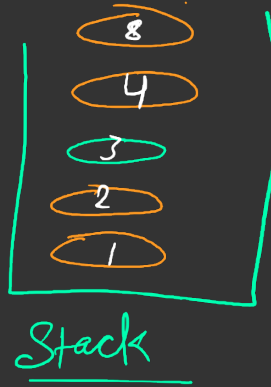


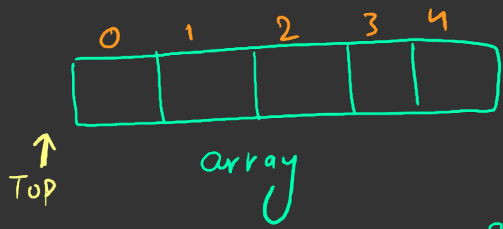
Introduction to Stack

- ✓ Push operation
- ✓ Pop operation - del
- ✓ Top operation
- ✓ Size operation
- ✓ Empty operation

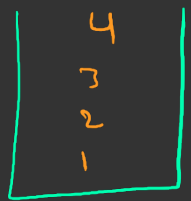


- ✓ LIFO
- ✓ FILO

Stack overflow
Stack underflow



- ✓ push
- ✓ pop
- ✓ Top
- ✓ size
- ✓ Empty



$$\text{Top} = 3 + 1 = 4$$

size

$$\text{size} = \text{Top} + 1$$
$$-1 + 1 = 0$$

8
2 4 6 1 3
2 4 6 1 3

$$\text{Top} = -1$$

```
class Stack {
```

```
    int * arr;
```

```
    int size;
```

```
    int top;
```

```
public:
```

```
    Stack(int s) {
```

```
        size = s;
```

```
        top = -1;
```

```
        arr = new int[s];
```

```
    }
```

```
    void Push (int value) {
```

```
        if (Top == size-1) {
```

```
            cout << "Stack overflow";
```

```
            return;
```

```
        }
```

```
        else { Top++;
```

```
            arr [top] = value;
```

```
        }
```

```
    void pop() {
```

```
        if (top == -1) {
```

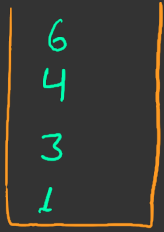
```
            cout << "Stack underflow"
```

```
        } return;
```

```
    } else {
```

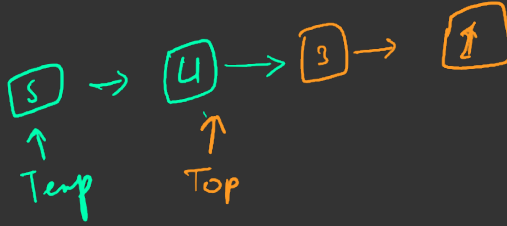
```
        Top--;
```

```
    }
```



Stack

1 3 4 6 8



Size

Size ~~4~~

pop
Size ~~4~~ 3

```
Class Node {
```

```
    public
```

```
        int data;
```

```
        Node *next;
```

```
        Node (int val) {
```

```
            data = val;
```

```
            next = NULL;
```

```
        }
```

```
    };
```

```
Class STACK2 {
```

```
    Node *TOP; ✓
```

```
    int size; ✓
```

```
    public:
```

```
Stack() {
```

```
    Top = NULL;
```

```
}
```

```
void Push (int val) {
```

```
    Node *temp = new Node (val);
```

```
    temp -> next = Top;
```

```
    Top = temp;
```

```
    size ++;
```

```
}
```

```
void Pop() {
```

```
    Node *temp = Top;
```

```
    Top = Top -> next;
```

```
    delete Temp;
```

```
}
```

{Stack overflow}

if (Top == NULL) {
 cout << "under flow"
 return
}

STL

✓ Stack (int) S;
S.push(5);
S.push(10);
S.pop();

heap

deque

