Data Science & AI
&
NIC - Param

Python-For Data Science

Stack and Queue

Lecture No.- 02

By- Pankaj Sharma Sir

Recap of Previous Lecture

Topic Stack and Queues Part-01

Topics to be Covered

Topic Stack and Queues Part-02

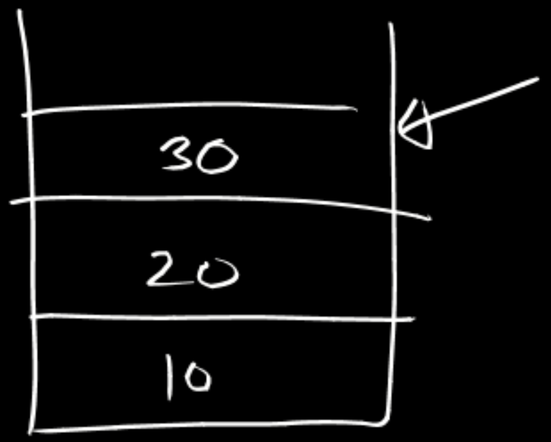
Topic : Stack and Queues

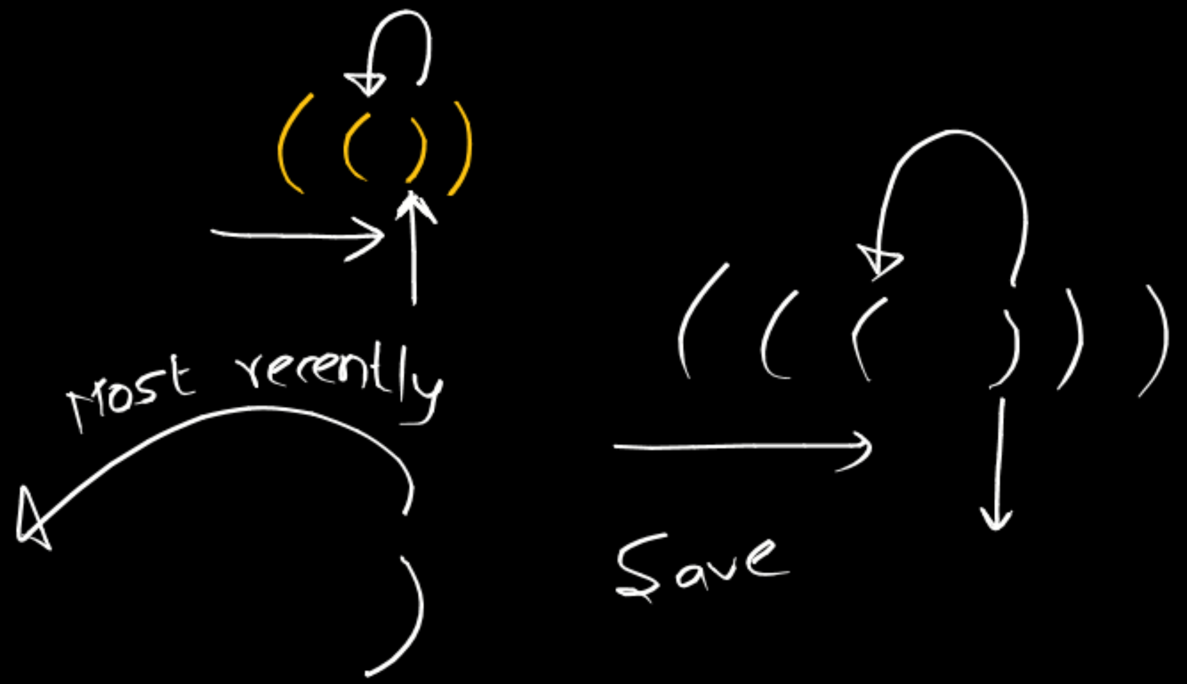
Balanced Paranthesis

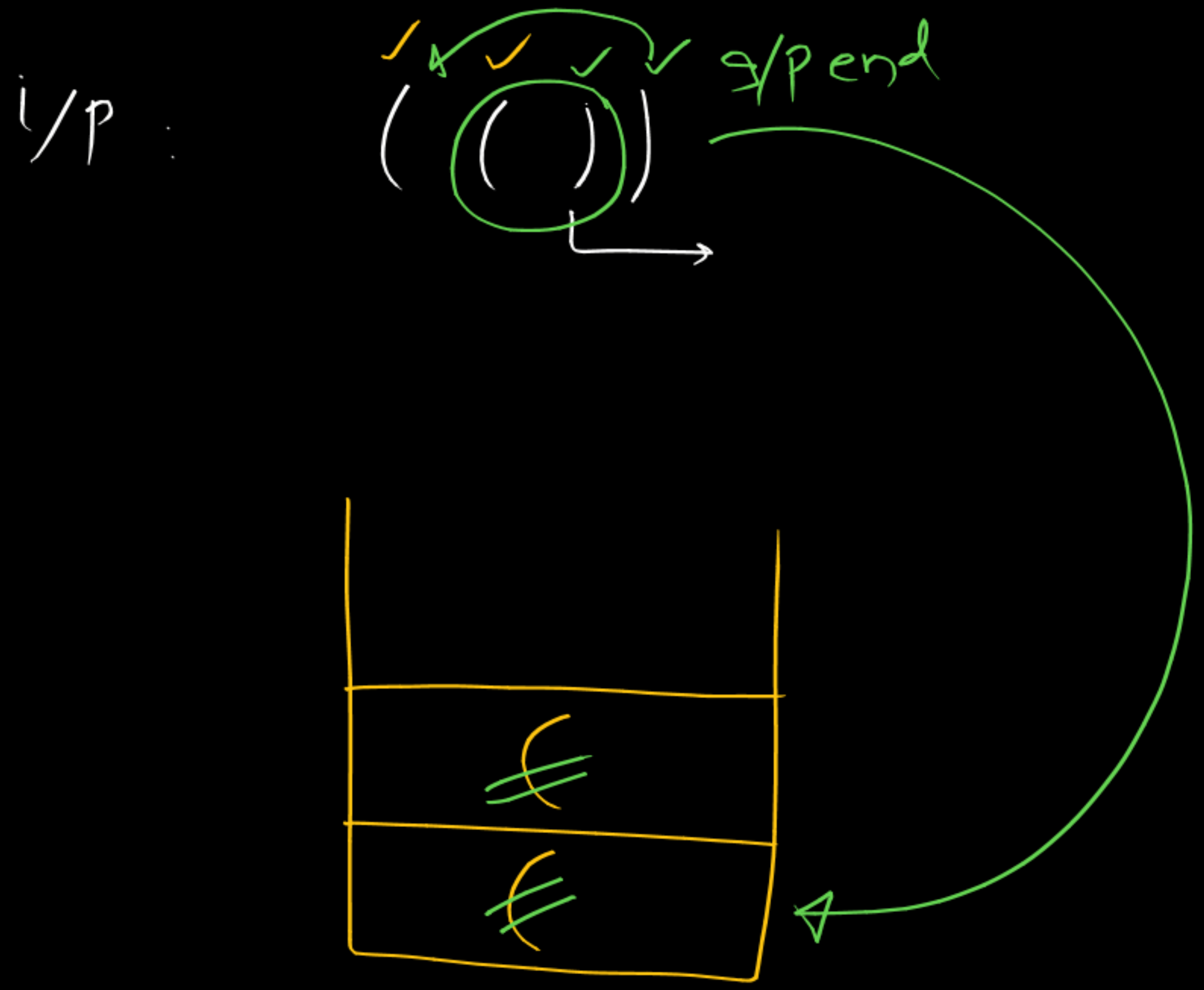
$(a+b) \Rightarrow ()$ ✓

$(())\Rightarrow$ ✓

$()(())$ ✓


30
20
10


(())
Most recently
((()))
Save


i/p :
(( ))
i/p end
(
(

```
str = input( )
l = [ ]
for char in str :

    if char in '({[' :
        l.append(char)

    elif char == ')' :                     Eval
                   True
        if (not l  or  l[-1] != '('):

                return False
        l.pop()
    elif char == '}' :
        if (not l or l[-1] != '{'):
                return False
        l.pop()
```

Inbuilt stack

↓

list.append( )

list.pop( )

elif ______________

Recursion

Assume ⇒ smaller size cnt

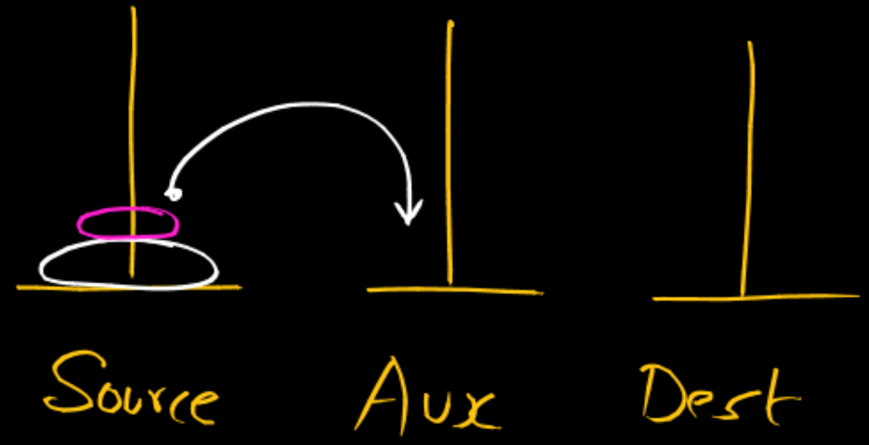
answer

↓

available

Tower of Hanoi

n no. of disc of diff. size

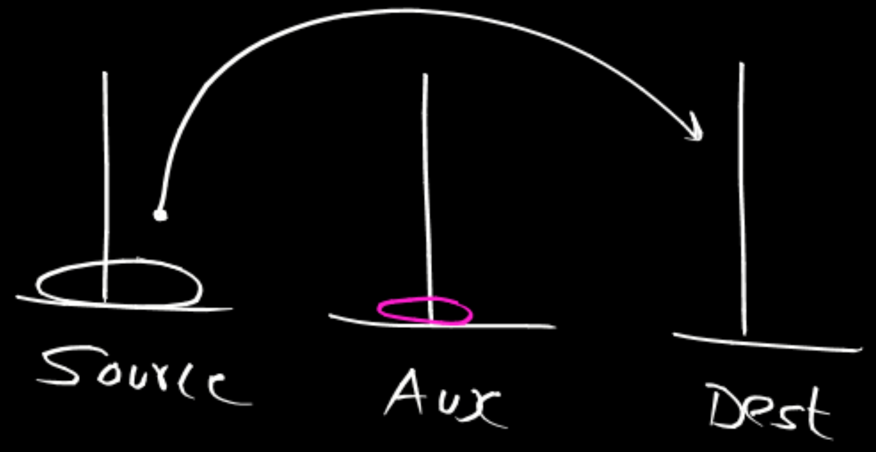


(i) move 1 disc at a time

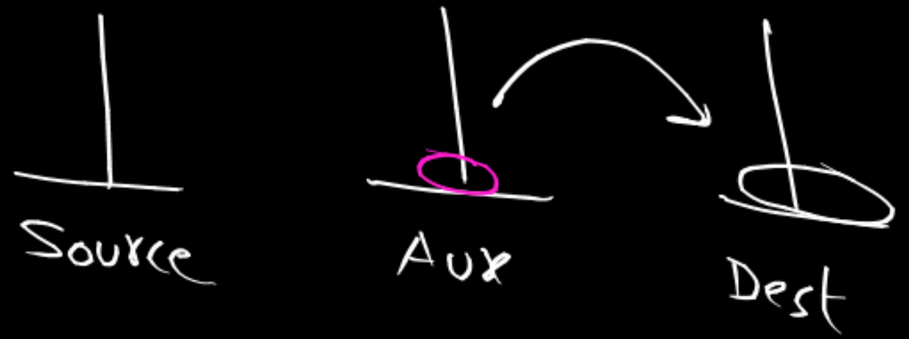
(ii) small on large

Source → destination

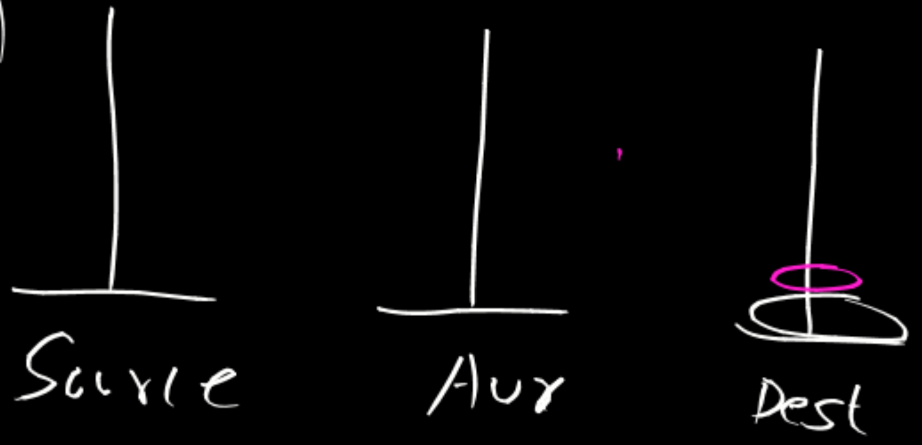
(i)

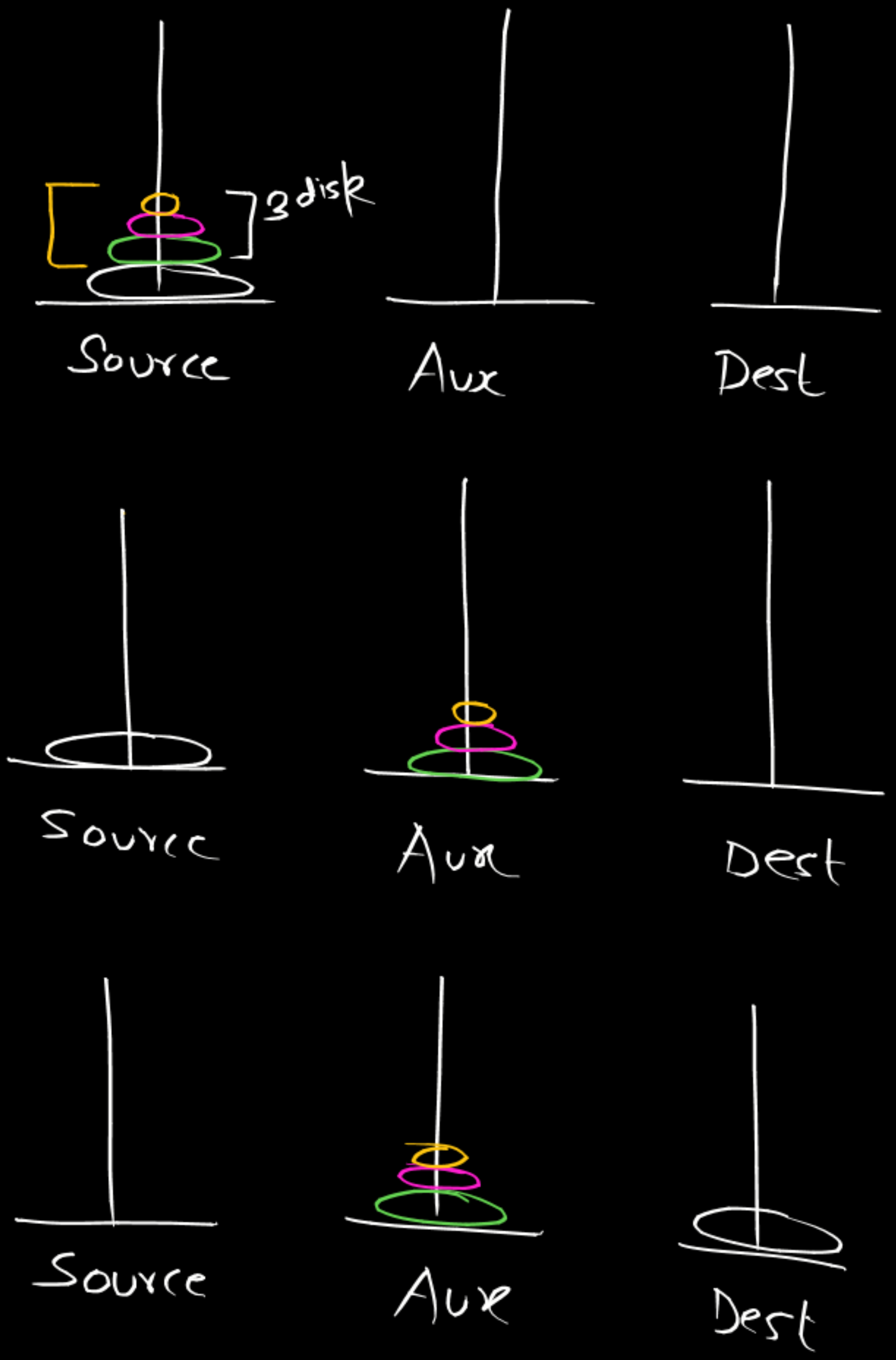


(ii)



(iii)



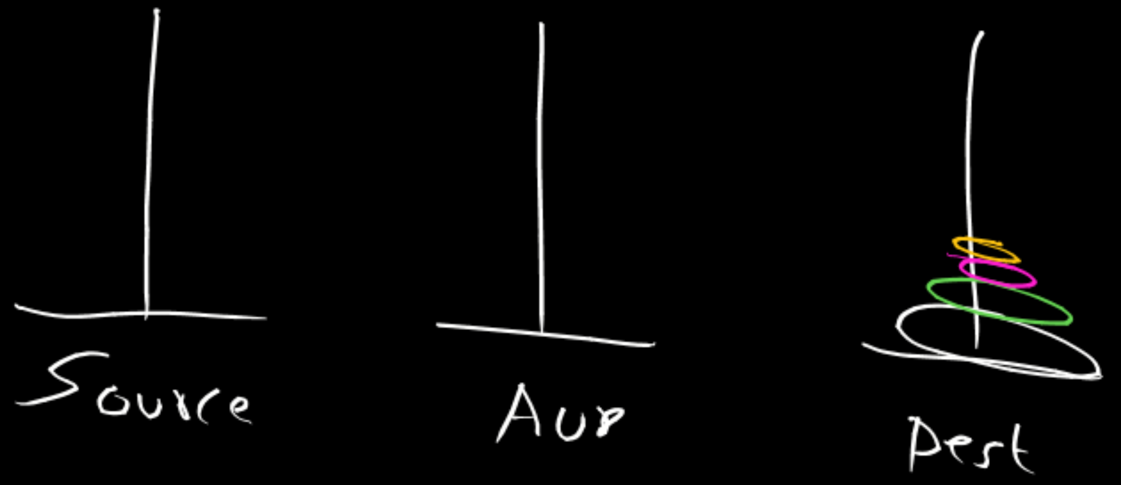


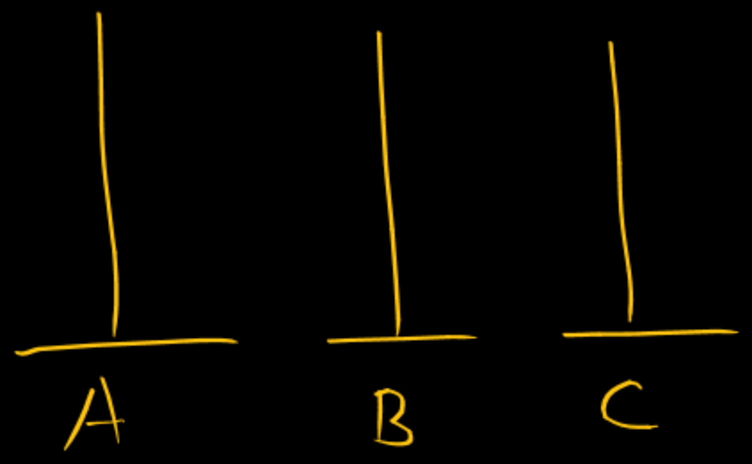
Recursion

Recur 1.) Move (n-1) disk from Source to Aux

2.) Source to dest.

Recu 3.) Move (n-1) disk from Aux to dest.



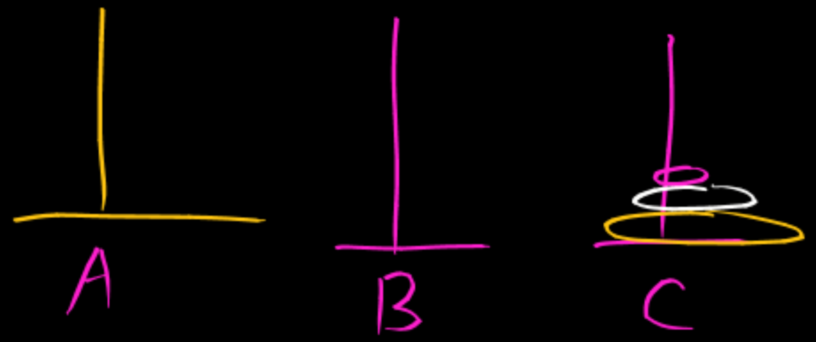


```
def   TOH(n, Source, Aux, Dest):

        if (n == 1):
            print( Source, "->", Dest) ✓

        else:
            ① TOH(n-1, Source, Dest , Aux)
            ② print( Source, "->", Dest)
            ③ TOH(n-1, Aux, Source, Dest)

n = int(input())
TOH(n, 'A', 'B', 'C')
```



✓ A → C

✓ A → B

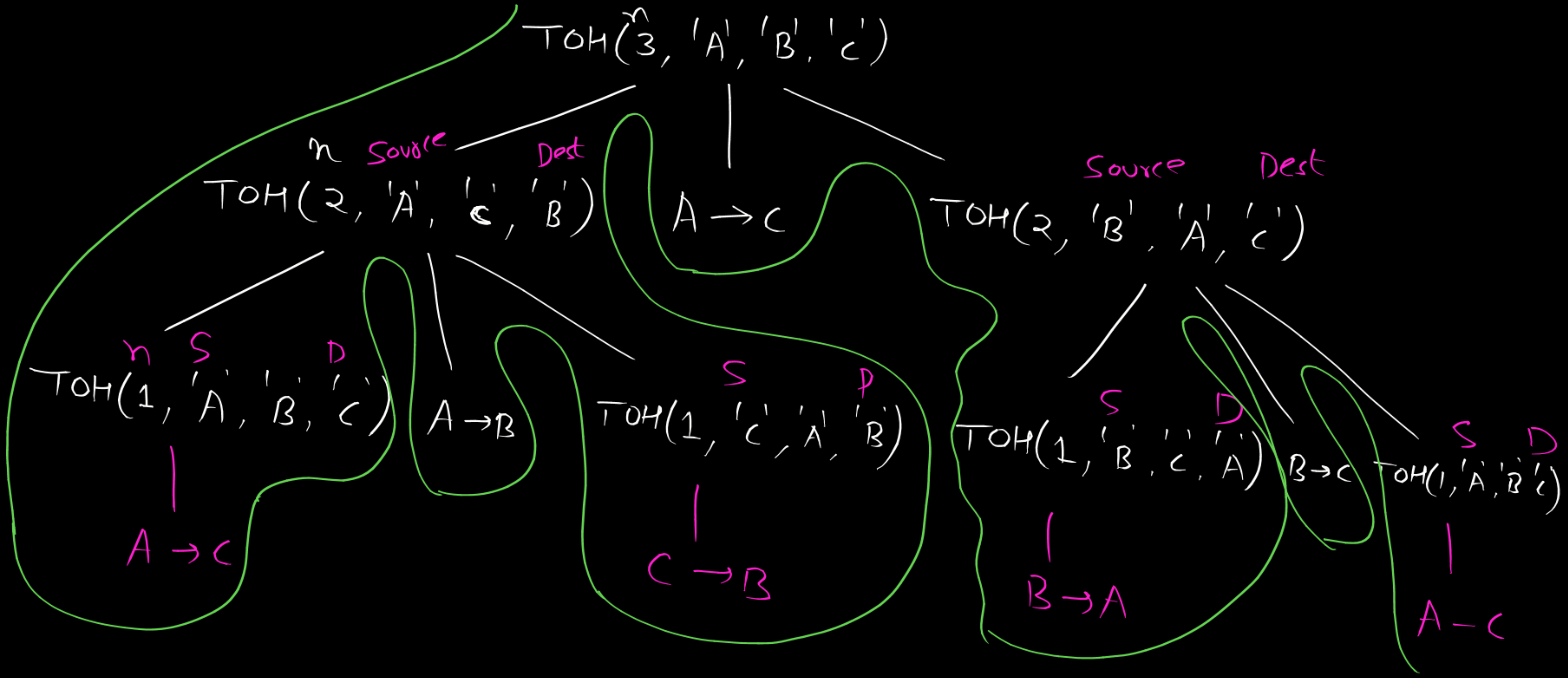
✓ C → B ✓

✓ A → C ✓

✓ B → A ✓

✓ B → C ✓

✓ A → C



n
TOH(3, 'A', 'B', 'C')
n Source Dest
TOH(2, 'A', 'C', 'B')
A → C
Source Dest
TOH(2, 'B', 'A', 'C')
n S D
TOH(1, 'A', 'B', 'C')
A → C
A → B
S D
TOH(1, 'C', 'A', 'B')
C → B
S D
TOH(1, 'B', 'C', 'A')
B → A
B → C
S D
TOH(1, 'A', 'B', 'C')
A – C

True False

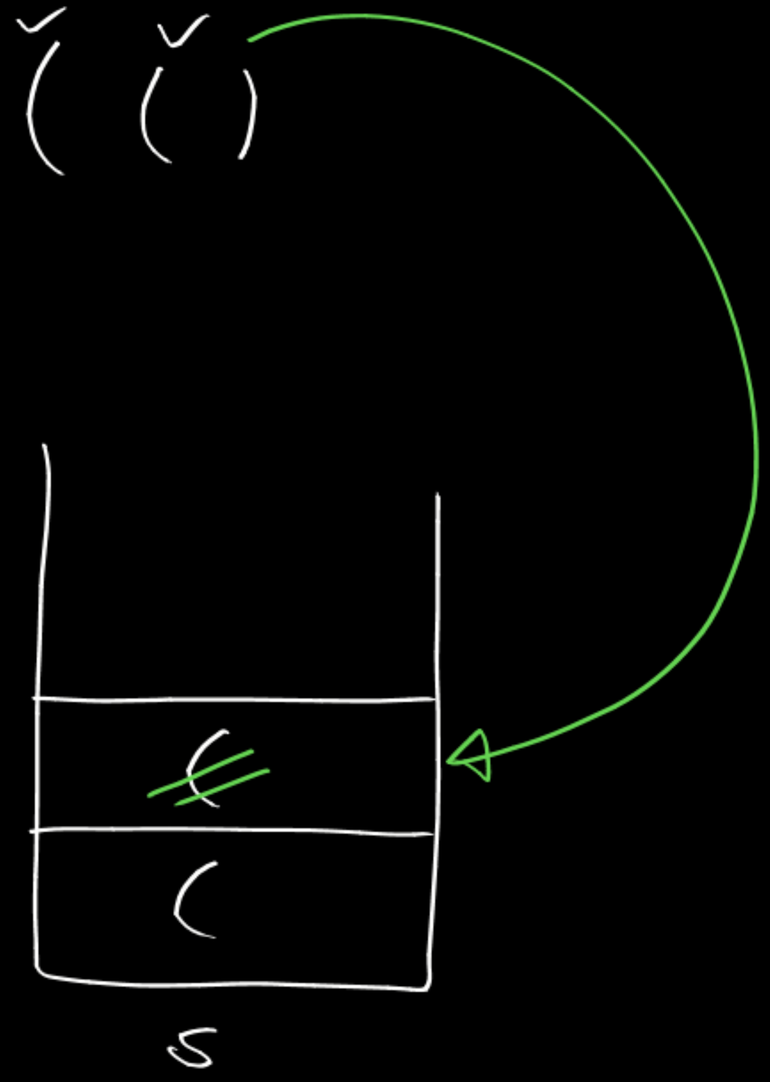
Empty — False

[ ]

( )

{ }

i/p : (( )



Not balanced

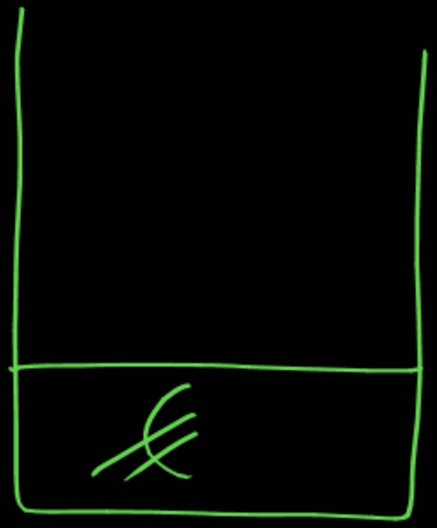
i/p → End

i/p : $\overset{\checkmark}{(}\ \overset{\checkmark}{)}\ )$

s ⇒ Empty

)

s is empty → X



(,) , { , } , [,]

(, { , [⇒ Push

) → Top of stack ⇒ '(' ⇒ ✓

} → ⇒ '{' ⇒

] → ⇒ '[' ⇒

In [2]:

```python
def kyabalancedhai(str1):
    l=[]
    for ele in str1:
        if ele in '{({[':
            l.append(ele)
        elif ele ==')':
            if (not l or l[-1]!='(' ):
                return False
            l.pop()
        elif ele =='}':
            if (not l or l[-1]!='{' ):
                return False
            l.pop()
        elif ele ==']':
            if (not l or l[-1]!='[' ):
                return False
            l.pop()
    if (not l):
        return True
    else:
        return False
```

In [3]:

```python
s=input("enter the string")
a=kyabalancedhai(s)
print(a)
```

```
enter the string()()
True
```

In [4]:

```python
s=input("enter the string")
a=kyabalancedhai(s)
print(a)
```

```
enter the string()(
False
```

In [5]:

```python
s=input("enter the string")
a=kyabalancedhai(s)
print(a)
```

```
enter the string(a+b)(c+d)
True
```

In [6]:

```python
s=input("enter the string")
a=kyabalancedhai(s)
print(a)
```

```
enter the string({})[()]
True
```

In [1]:

```python
def TOH(n,Source,Aux,Dest):
    if n==1:
        print(Source,"-->",Dest)
    else:
        TOH(n-1,Source,Dest,Aux)
        print(Source,"-->",Dest)
        TOH(n-1,Aux,Source ,Dest)
```

In [2]:

```
n=int(input("Enter number of disc:  "))
TOH(n,'A','B','C')
```

```
Enter number of disc:  3
A --> C
A --> B
C --> B
A --> C
B --> A
B --> C
A --> C
```

In [3]:

```
n=int(input("Enter number of disc:  "))
TOH(n,'A','B','C')
```

```
Enter number of disc:  4
A --> B
A --> C
B --> C
A --> B
C --> A
C --> B
A --> B
A --> C
B --> C
B --> A
C --> A
B --> C
A --> B
A --> C
B --> C
```

In [4]:

```
n=int(input("Enter number of disc:  "))
TOH(n,'A','B','C')
```

```
Enter number of disc:  6
A --> B
A --> C
B --> C
A --> B
C --> A
C --> B
A --> B
A --> C
B --> C
B --> A
C --> A
B --> C
A --> B
A --> C
B --> C
A --> B
C --> A
C --> B
A --> B
C --> A
B --> C
B --> A
C --> A
C --> B
A --> B
A --> C
B --> C
A --> B
C --> A
C --> B
A --> B
A --> C
B --> C
B --> A
C --> A
B --> C
A --> B
A --> C
B --> C
B --> A
C --> A
C --> B
A --> B
C --> A
B --> C
B --> A
C --> A
B --> C
A --> B
A --> C
B --> C
A --> B
C --> A
C --> B
A --> B
A --> C
B --> C
B --> A
C --> A
```

```
B --> C
A --> B
A --> C
B --> C
```

In [ ]:

Queue

THANK - YOU