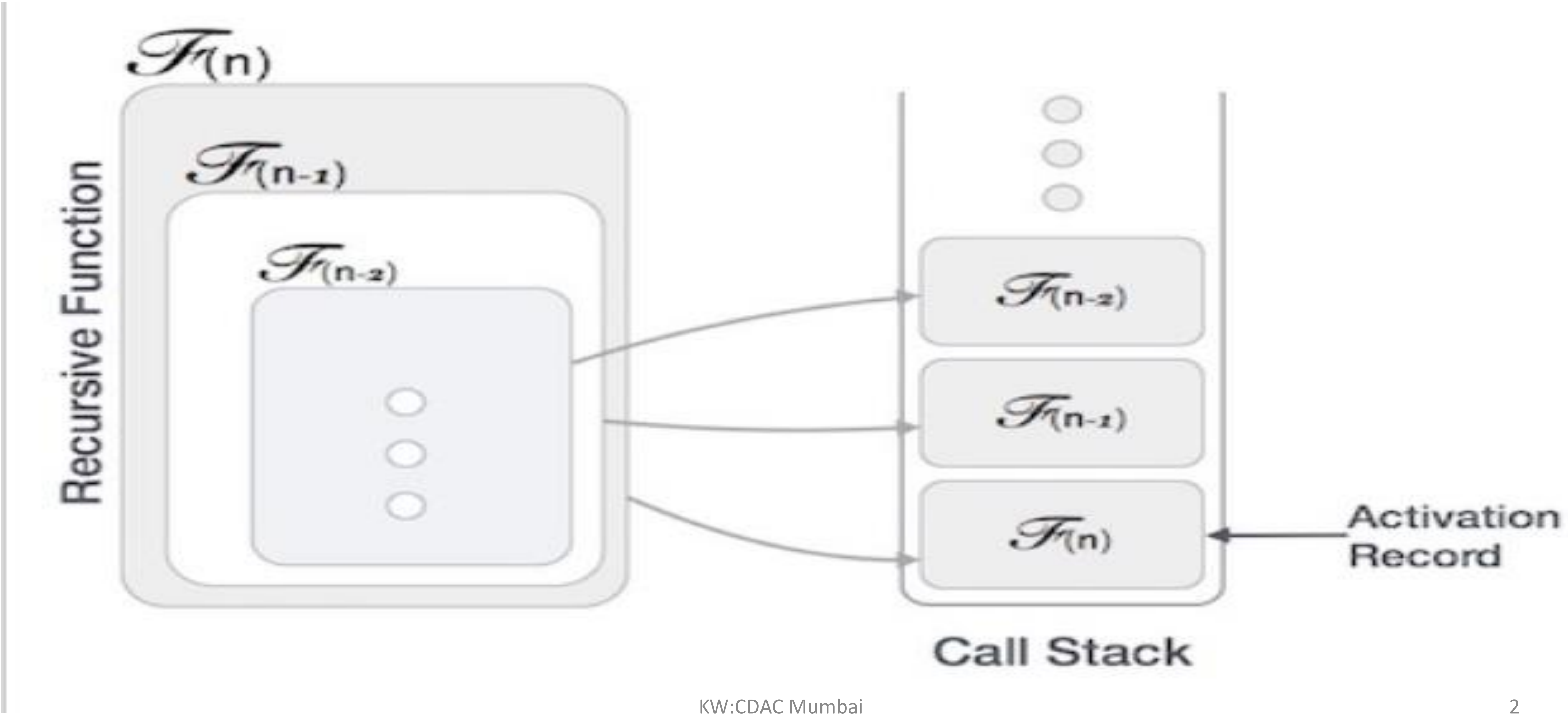
Data Structure

Sep23 : Day 2

Kiran Waghmare

CDAC Mumbai

How Data Structure Recursive function is implemented?

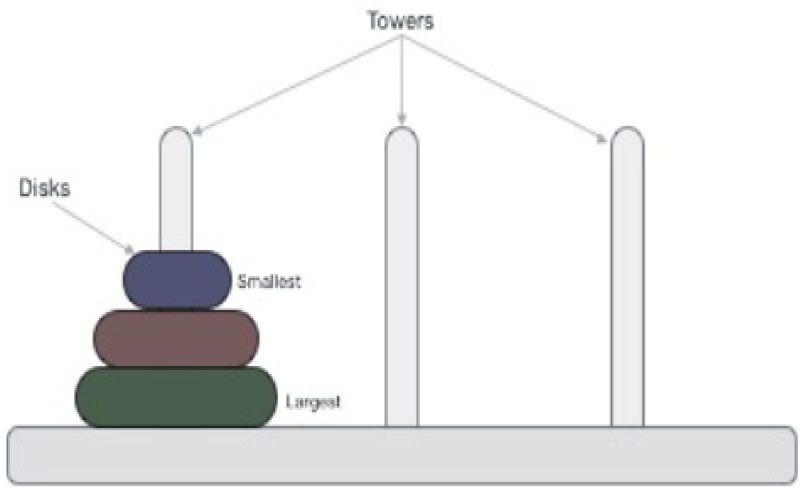


What is Tower of Hanoi?

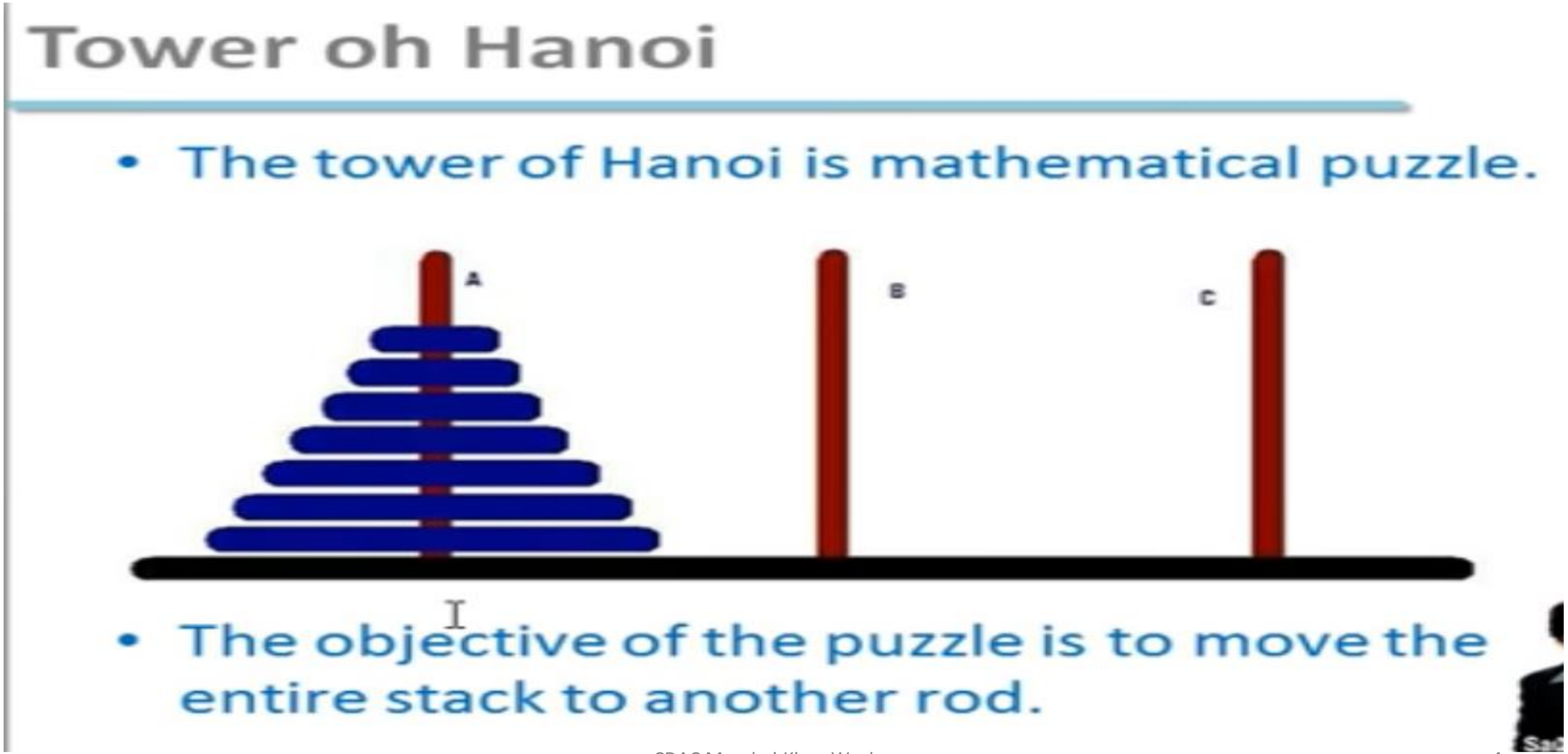
- A mathematical puzzle consisting of three towers and more than one ring is known as Tower of Hanoi.

- Tower of Hanoi

- The rings are of different sizes and are stacked in ascending order, i.e., the smaller one sits over the larger one. In some of the puzzles, the number of rings may increase, but the count of the tower remains the same.

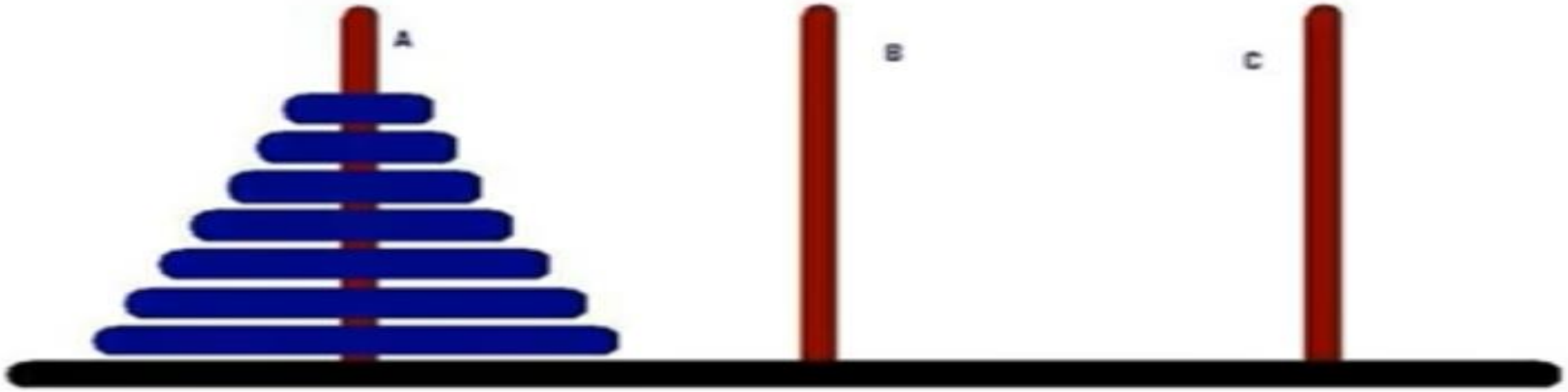


Tower of Hanoi



Tower oh Hanoi

- The tower of Hanoi is mathematical puzzle.



- The objective of the puzzle is to move the entire stack to another rod.

What are the rules to be followed by Tower of Hanoi?

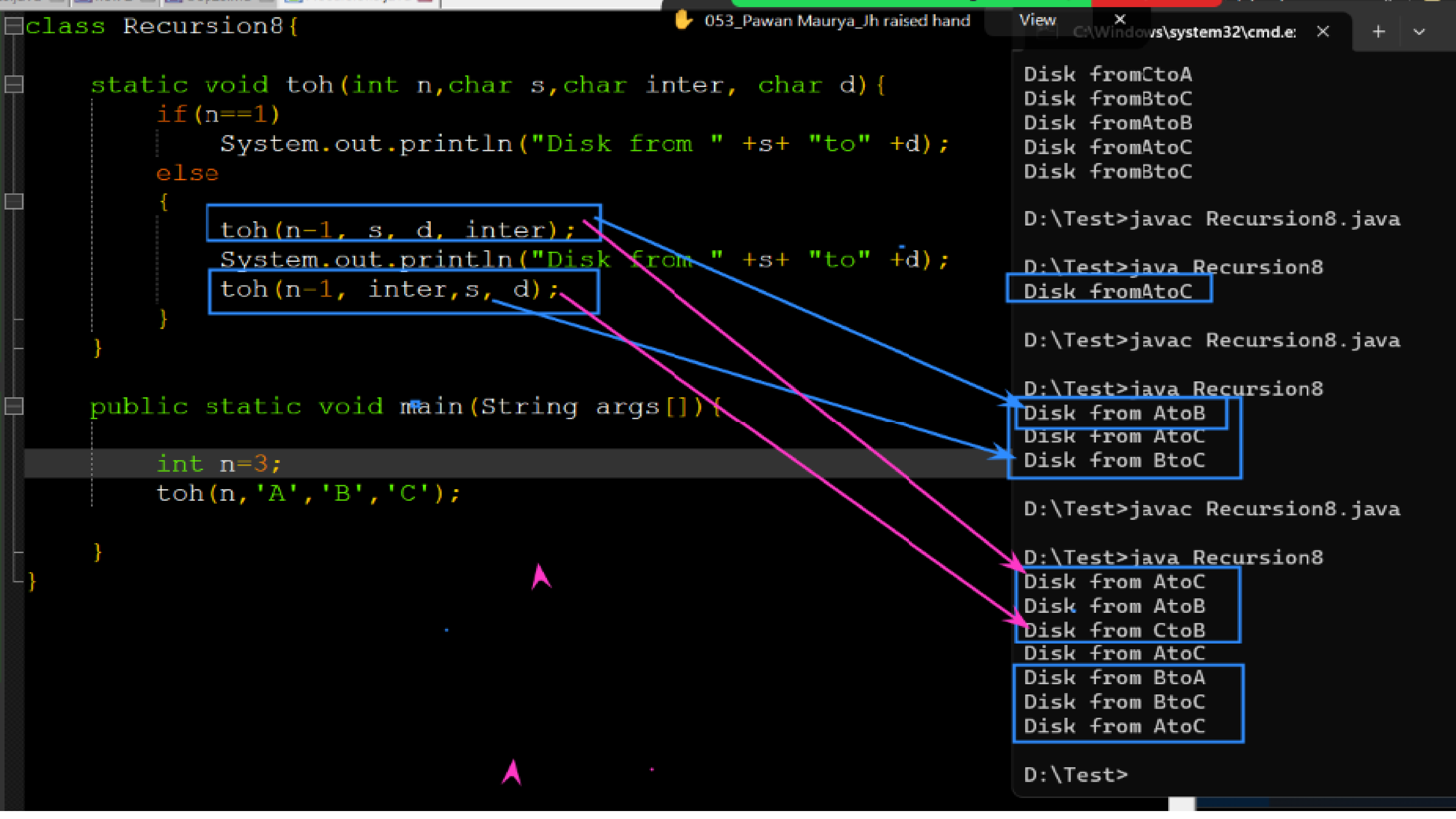
- **The Tower of Hanoi puzzle is solved by moving all the disks to another tower by not violating the sequence of the arrangements.**

The rules to be followed by the Tower of Hanoi are -

1. Only one disk can be moved among the towers at any given time.
2. Only the "top" disk can be removed.
3. No large disk can sit over a small disk.

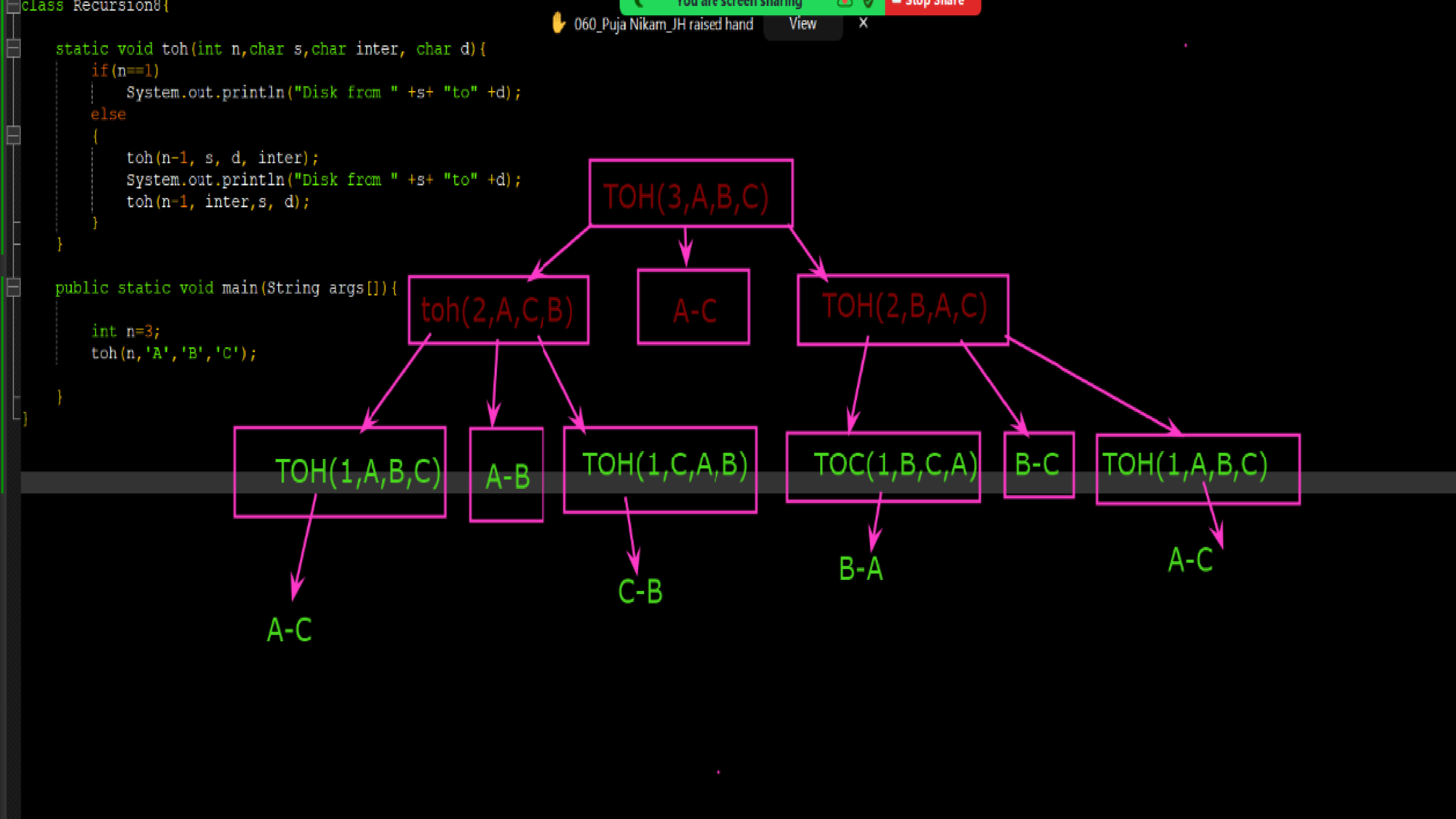
Algorithm 1: Recursive algorithm for solving Towers of Hanoi

```
function recursiveHanoi(n, s, a, d)
    if n == 1 then
        print(s + " to " + d);
        return;
    end
    recursiveHanoi(n − 1, s, d, a);
    print(s + " to " + d);
    recursiveHanoi(n − 1, a, s, d);
end
```

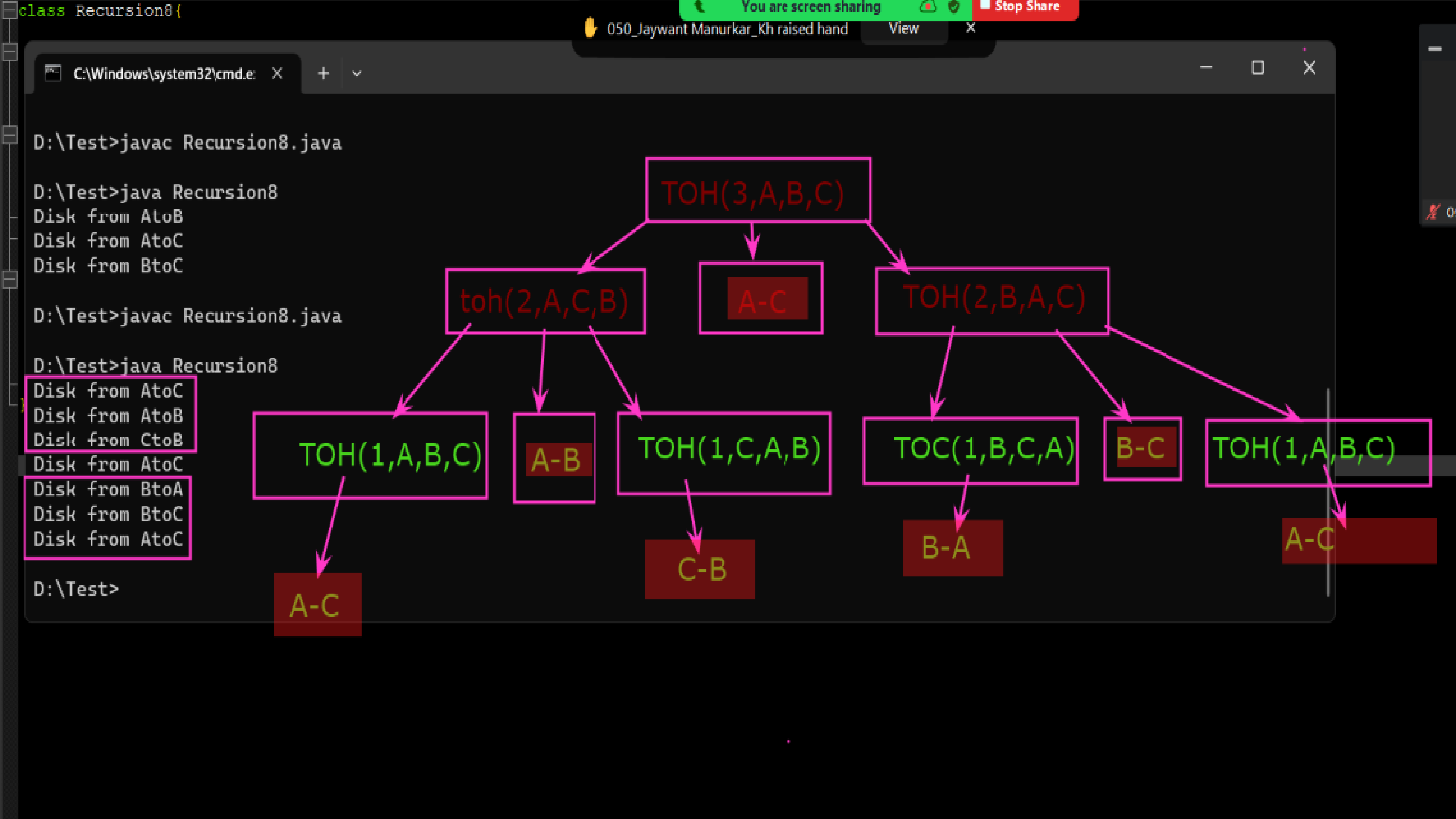


```java
class Recursion8{

    static void toh(int n,char s,char inter, char d){
        if(n==1)
            System.out.println("Disk from " +s+ "to" +d);
        else
        {
            toh(n-1, s, d, inter);
            System.out.println("Disk from " +s+ "to" +d);
            toh(n-1, inter,s, d);
        }
    }

    public static void main(String args[]){

        int n=3;
        toh(n,'A','B','C');

    }
}
```

```
Disk fromCtoA
Disk fromBtoC
Disk fromAtoB
Disk fromAtoC
Disk fromBtoC

D:\Test>javac Recursion8.java

D:\Test>java Recursion8
Disk fromAtoC

D:\Test>javac Recursion8.java

D:\Test>java Recursion8
Disk from AtoB
Disk from AtoC
Disk from BtoC

D:\Test>javac Recursion8.java

D:\Test>java Recursion8
Disk from AtoC
Disk from AtoB
Disk from CtoB
Disk from AtoC
Disk from BtoA
Disk from BtoC
Disk from AtoC

D:\Test>
```



```java
class Recursion8{

    static void toh(int n,char s,char inter, char d){
        if(n==1)
            System.out.println("Disk from " +s+ "to" +d);
        else
        {
            toh(n-1, s, d, inter);
            System.out.println("Disk from " +s+ "to" +d);
            toh(n-1, inter,s, d);
        }
    }

    public static void main(String args[]){

        int n=3;
        toh(n,'A','B','C');

    }
}
```



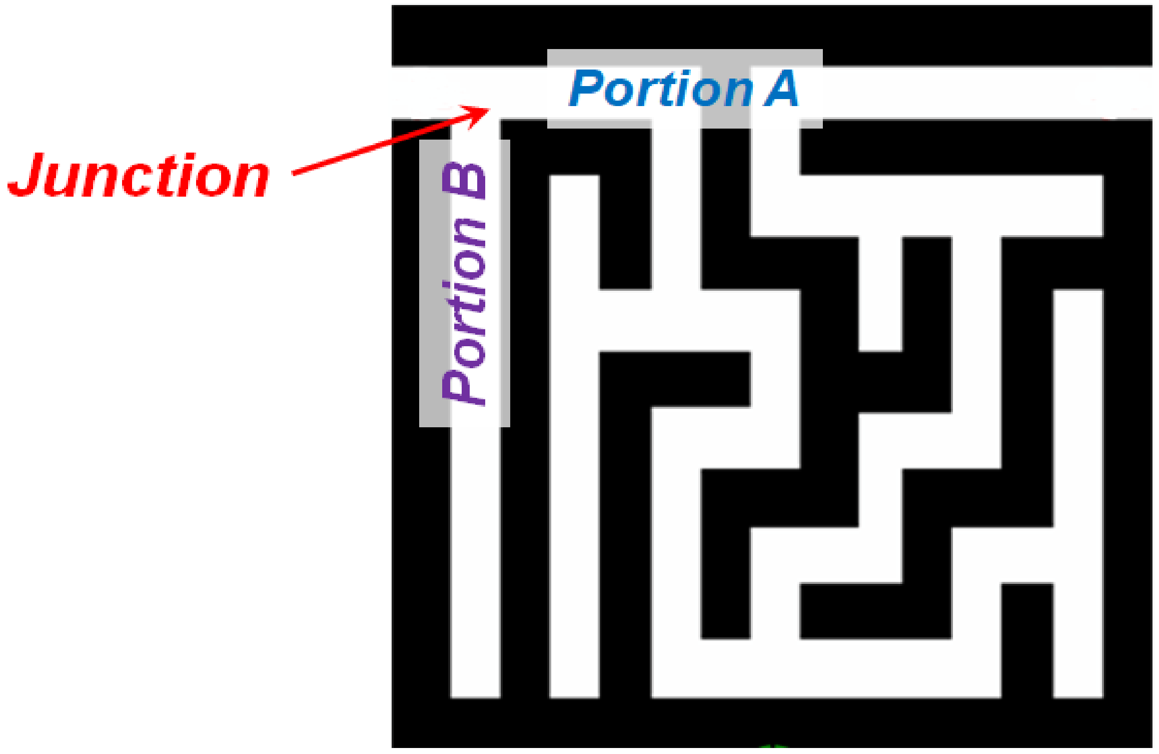
You are screen sharing
Stop Share
050_Jaywant Manurkar_Kh raised hand
View
C:\Windows\system32\cmd.e

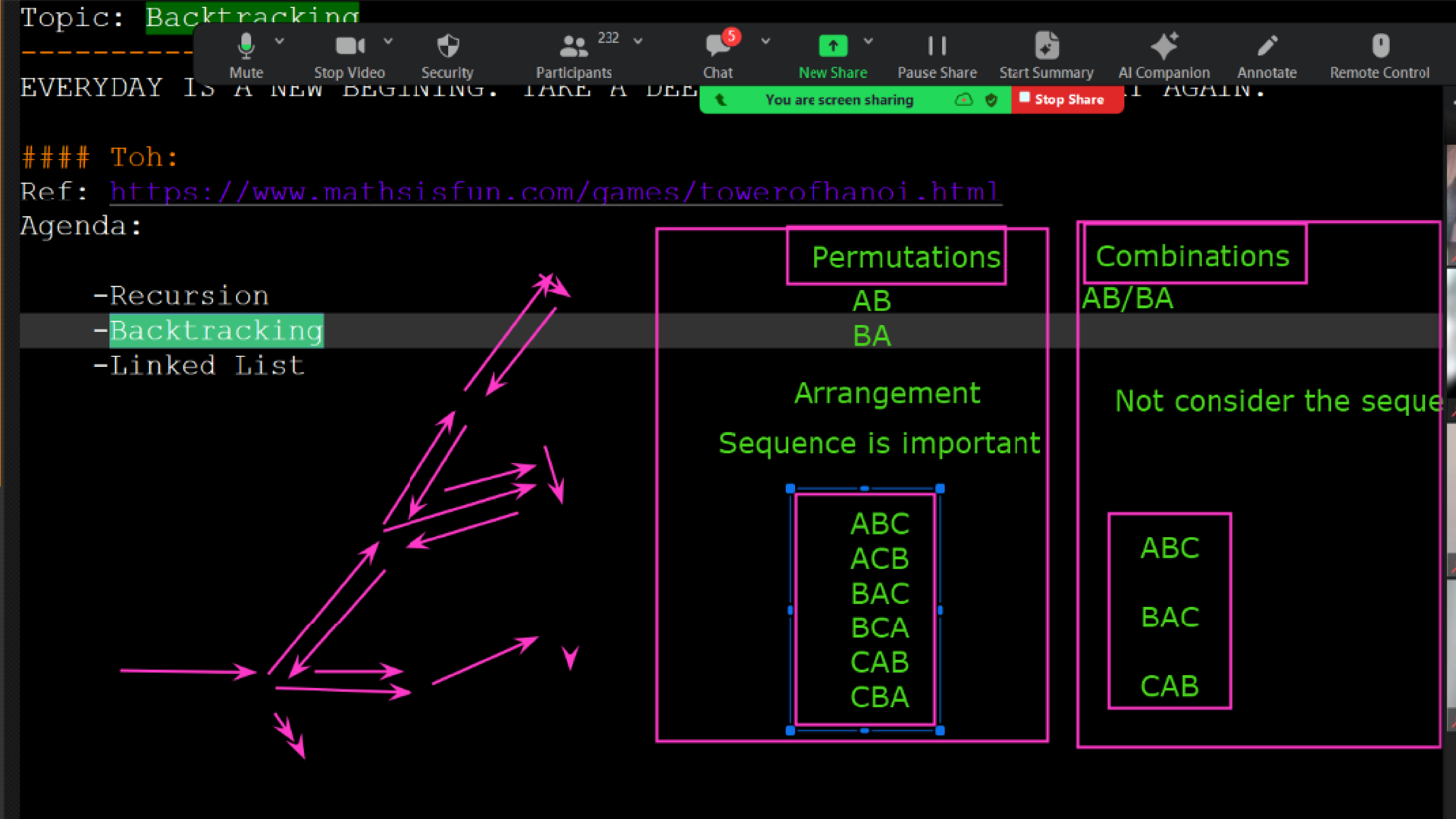
```
class Recursion8{
```

```
D:\Test>javac Recursion8.java

D:\Test>java Recursion8
Disk from AtoB
Disk from AtoC
Disk from BtoC

D:\Test>javac Recursion8.java

D:\Test>java Recursion8
Disk from AtoC
Disk from AtoB
Disk from CtoB
Disk from AtoC
Disk from BtoA
Disk from BtoC
Disk from AtoC

D:\Test>
```

Backtracking: Idea

- **Backtracking is a technique used to solve problems with a large search space, by systematically trying and eliminating possibilities.**
- **A standard example of backtracking would be going through a maze.**
  - At some point, you might have two options of which direction to go:





Topic: Backtracking

EVERYDAY IS A NEW BEGINING. TAKE A DEE AGAIN.

Toh:

Ref: https://www.mathsisfun.com/games/towerofhanoi.html

Agenda:

-Recursion
-Backtracking
-Linked List

Permutations

AB
BA

Arrangement

Sequence is important

ABC
ACB
BAC
BCA
CAB
CBA

Combinations

AB/BA

Not consider the seque

ABC

BAC

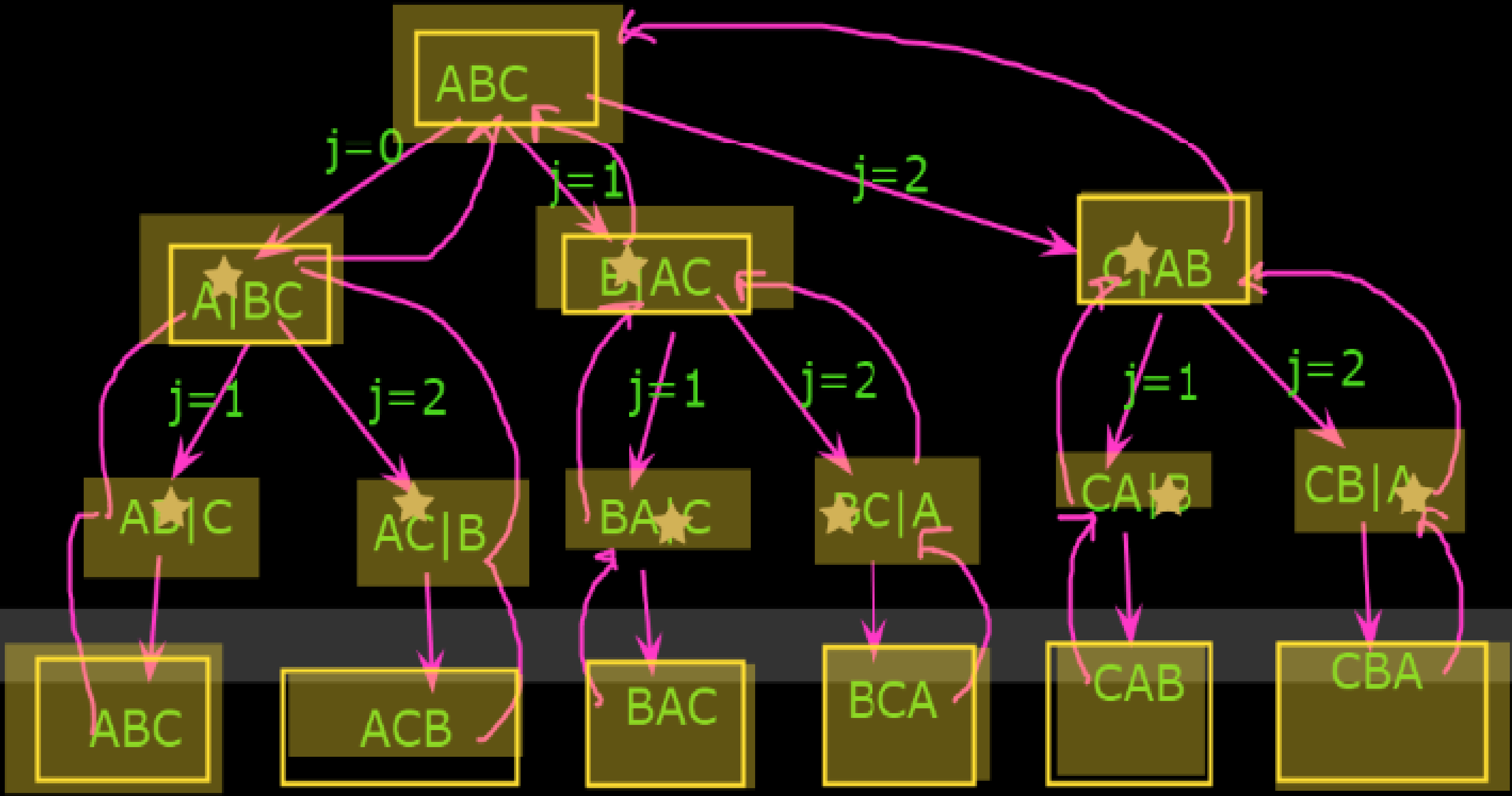
CAB

-Li

ABC
ACB

BAC
BCA

CAB
CBA



```java
class Recursion9{

    static void display(String str,String res)
    {
        if(str.length() == 0){
            System.out.println(res+ "");
            return;
        }

        for(int i=0;i<str.length();i++){
            char ch = str.charAt(i);
            String ros=str.substring(0,i)+ str.substring(i+1);
            display(ros, res+ch);
        }
    }


    public static void main(String args[]){

        String s="ABCD";
        display(s,"");


    }
}
```

ABC

A|BC

AB|C

ABC|

Problem 1

Recursive program to find the Sum of the series 1 – 1/2 + 1/3 – 1/4 … 1/N
Given a positive integer N, the task is to find the sum of the series 1 – (1/2) + (1/3) – (1/4) +…. (1/N) using recursion.

Examples:

Input: N = 3
Output: 0.8333333333333333
Explanation:
1 – (1/2) + (1/3) = 0.8333333333333333

Input: N = 4
Output: 0.5833333333333333
Explanation:
1- (1/2) + (1/3) – (1/4) = 0.5833333333333333

Problem 2

Recursive Program to print multiplication table of a number
Given a number N, the task is to print its multiplication table using recursion.
Examples

Input: N = 5
Output:
```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Input: N = 8
Output:
```
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
```

Problem 3

Recursive program to print formula for GCD of n integers

Given a function gcd(a, b) to find GCD (Greatest Common Divisor) of two number. It is also known that GCD of three elements can be found by gcd(a, gcd(b, c)), similarly for four element it can find the GCD by gcd(a, gcd(b, gcd(c, d))). Given a positive integer n. The task is to print the formula to find the GCD of n integer using given gcd() function.
Examples:

Input : n = 3
Output : gcd(int, gcd(int, int))

Input : n = 5
Output : gcd(int, gcd(int, gcd(int, gcd(int, int))))

Problem 4

Java Program to Reverse a Sentence Using Recursion

A sentence is a sequence of characters separated by some delimiter. This sequence of characters starts at the 0th index and the last index is at len(string)-1. By reversing the string, we interchange the characters starting at 0th index and place them from the end. The first character becomes the last, the second becomes the second last, and so on.

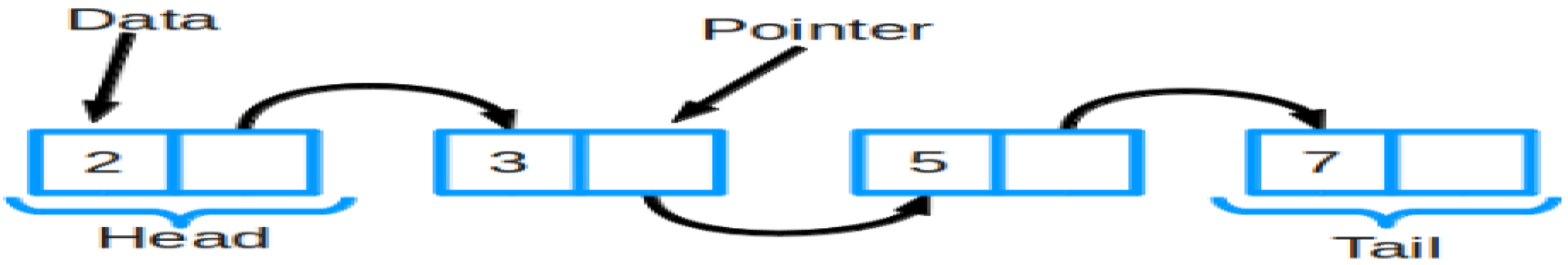
Example:

Input : CDAC Mumbai.
Output: .iabmuM CADC

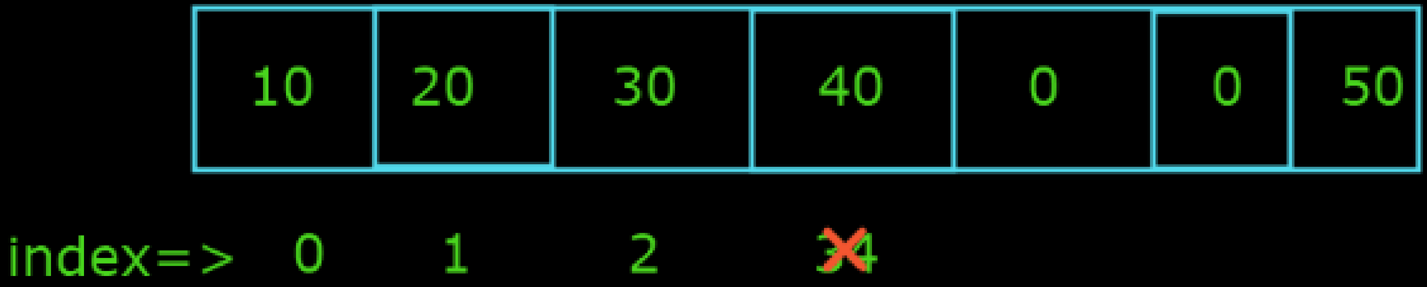
Input : Alice in wonderland.
Output: .dnalrednow ni ecilA
Approach:

Check if the string is empty or not, return null if String is empty.
If the string is empty then return the null string.
Else return the concatenation of sub-string part of the string from index 1 to string length with the first character of a string. e.g. return substring(1)+str.charAt(0); which is for string "Mayur" return will be "ayur" + "M".

Linked list

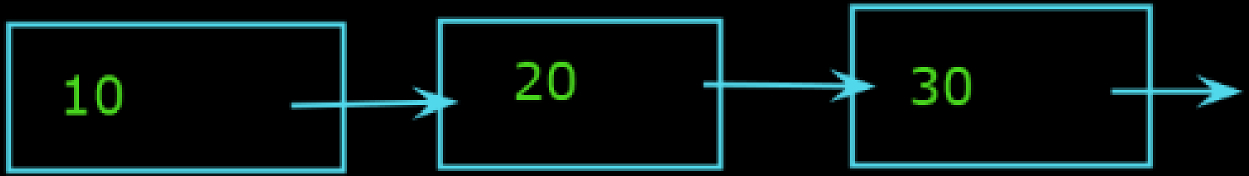


```
Linked List:
------------
```



homogeneous
continuous
sequential

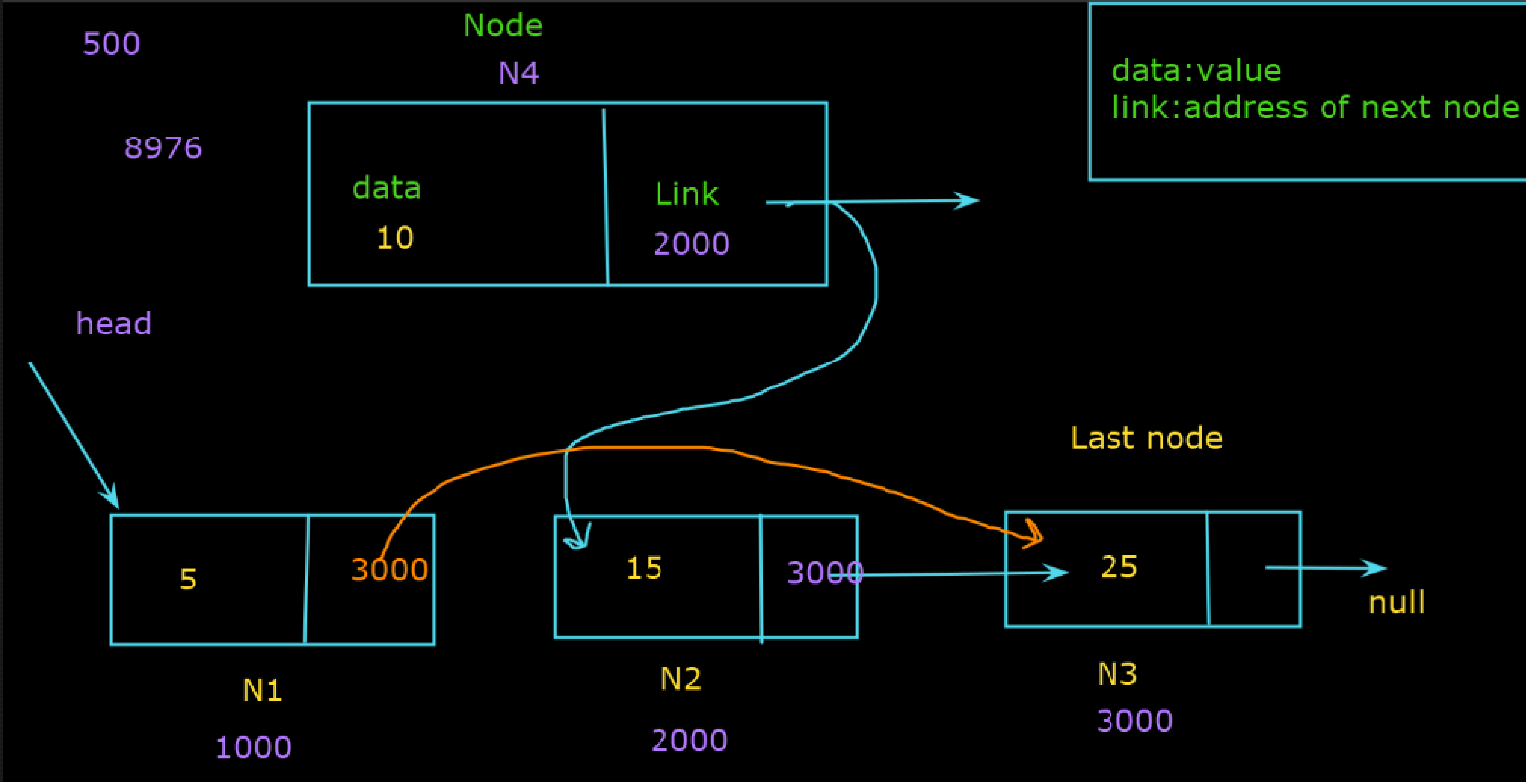
static => size=100000
fixed



flexibility: no size binding

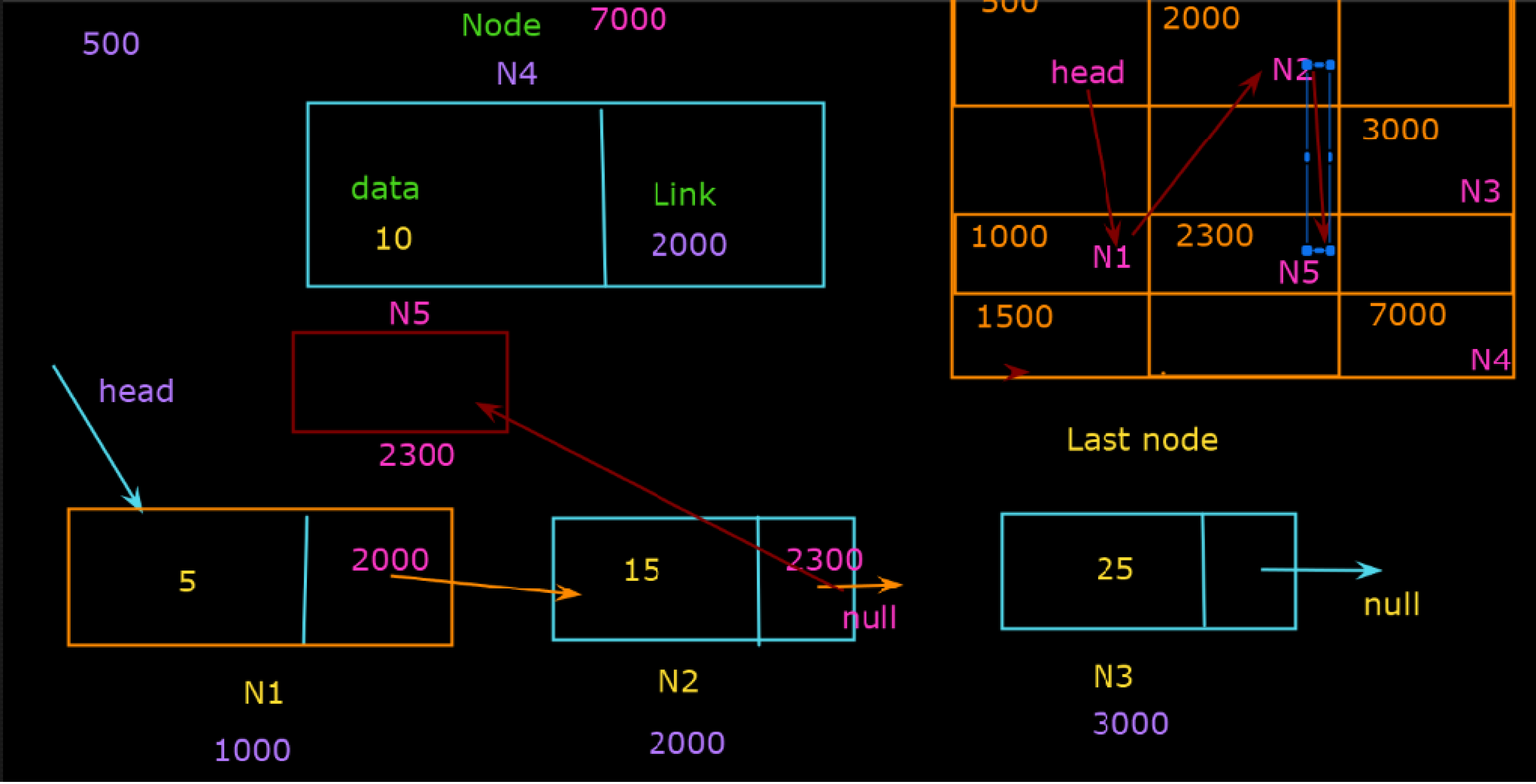
```
Linked List:
-------------
```

Structure of the node

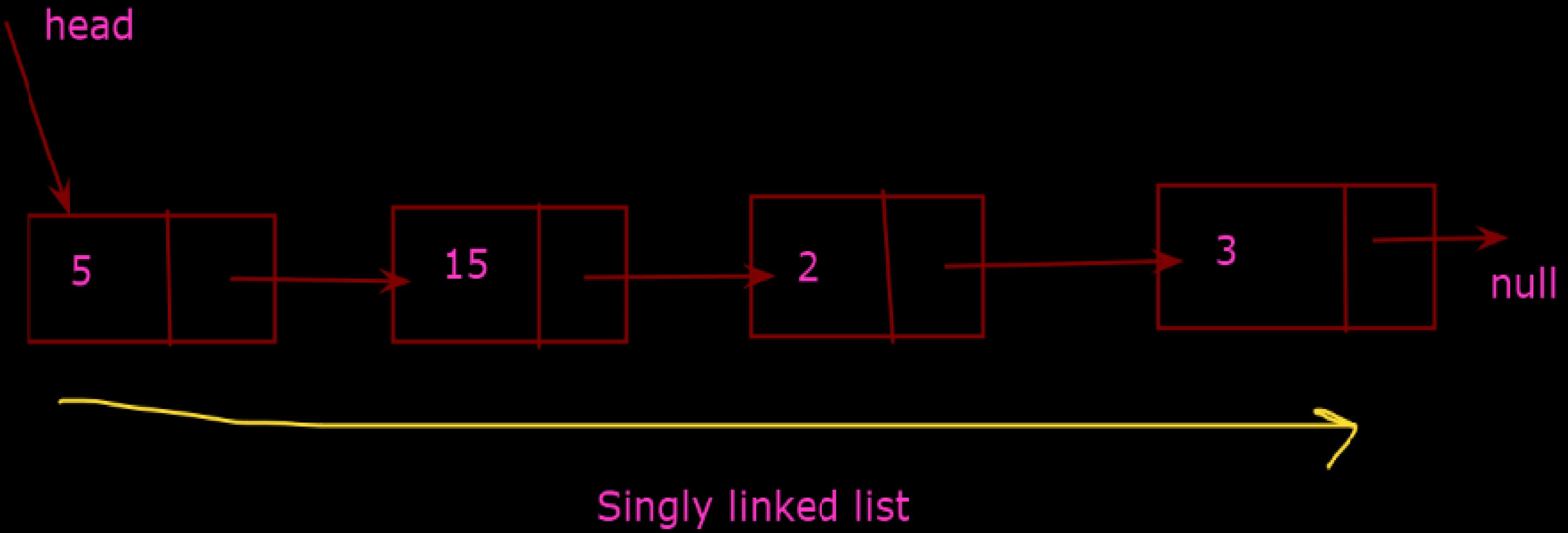


```
Linked List:
------------
```

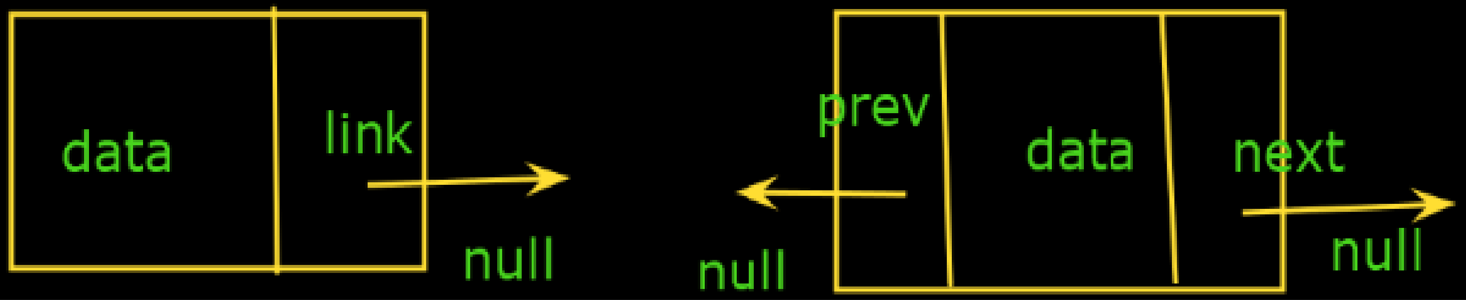
Structure of the node

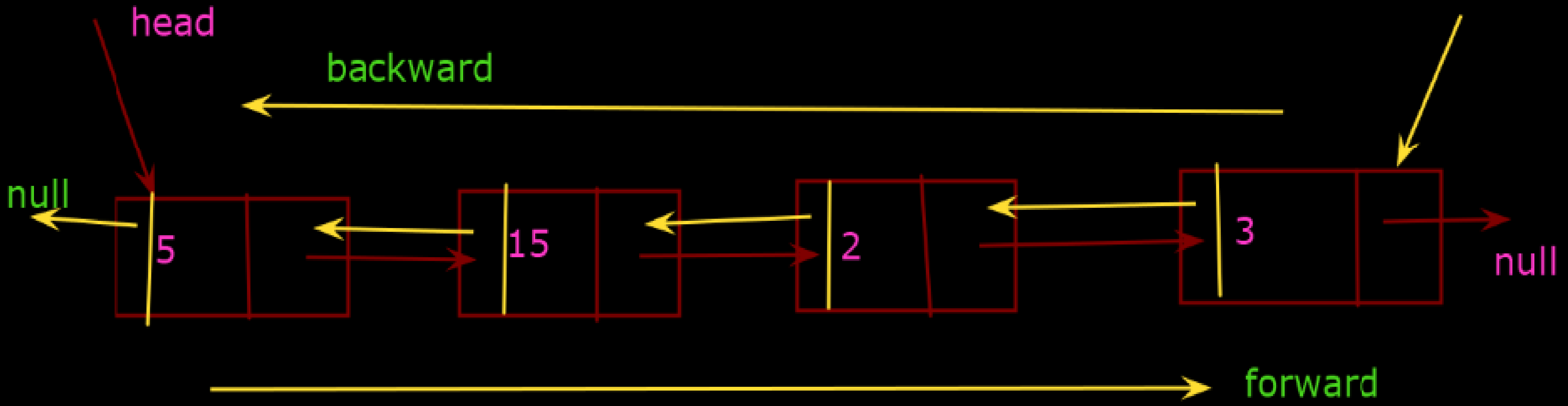


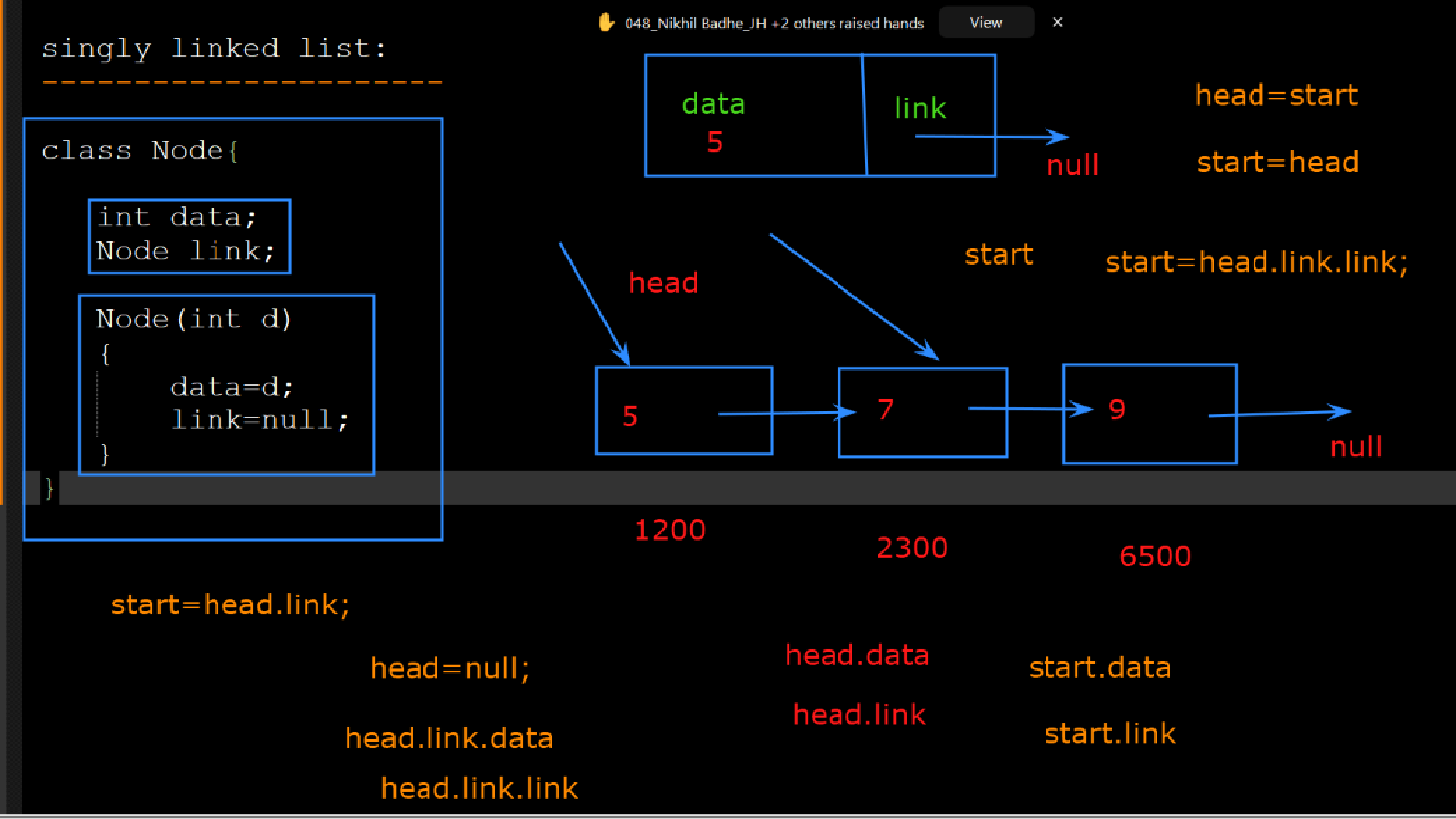
1. singly linked list
2. Doubly linked list
3. Circular linked list
4. Doubly circular linked list.



1. singly linked list
2. Doubly linked list
3. Circular linked list
4. Doubly circular linked list.



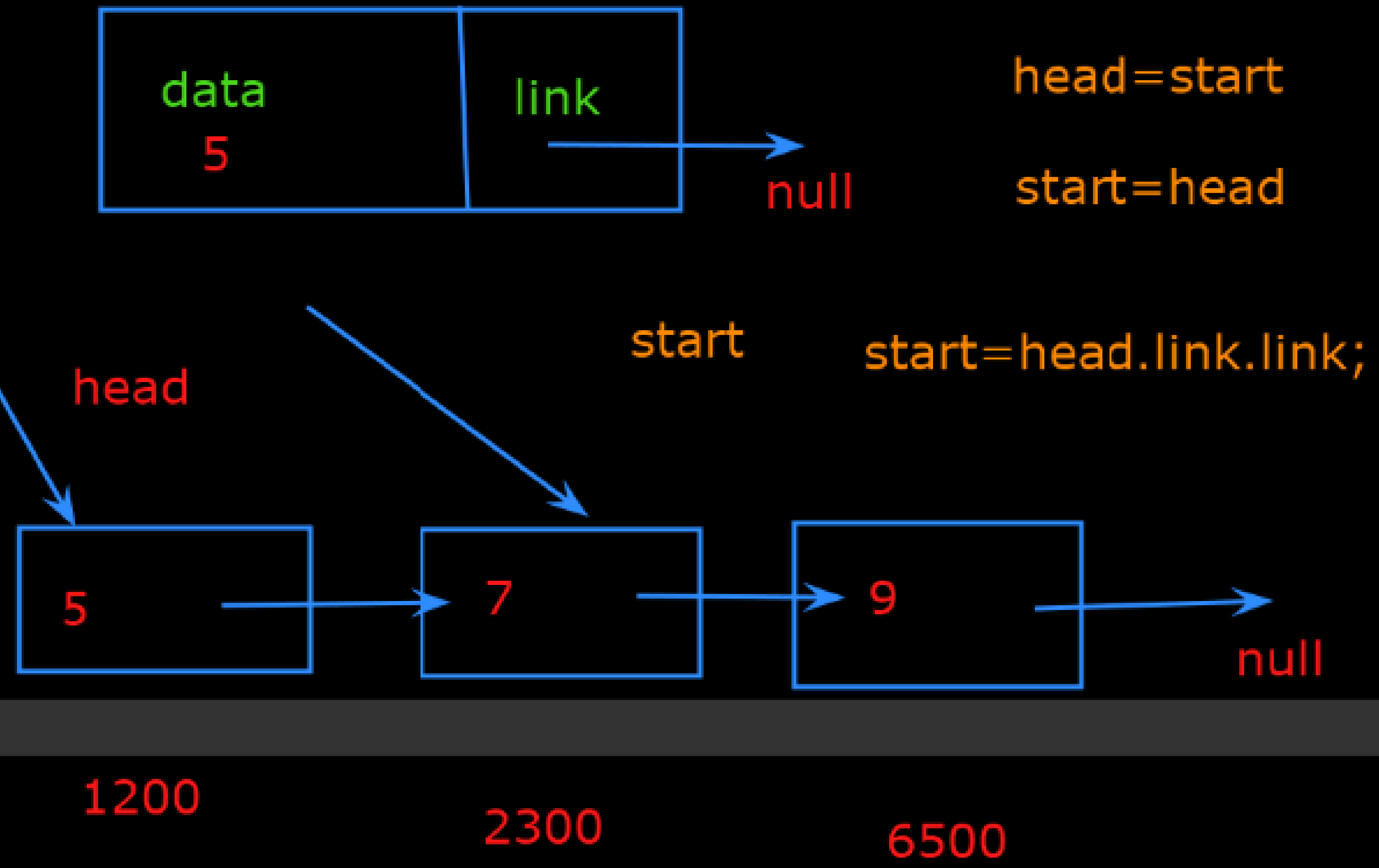




048_Nikhil Badhe_JH +2 others raised hands View ×

singly linked list:

```
class Node{

    int data;
    Node link;

    Node(int d)
    {
        data=d;
        link=null;
    }
}
```



head=start

start=head

start=head.link.link;

start=head.link;

head=null;

head.link.data

head.link.link

head.data

head.link

start.data

start.link