Lecture-16

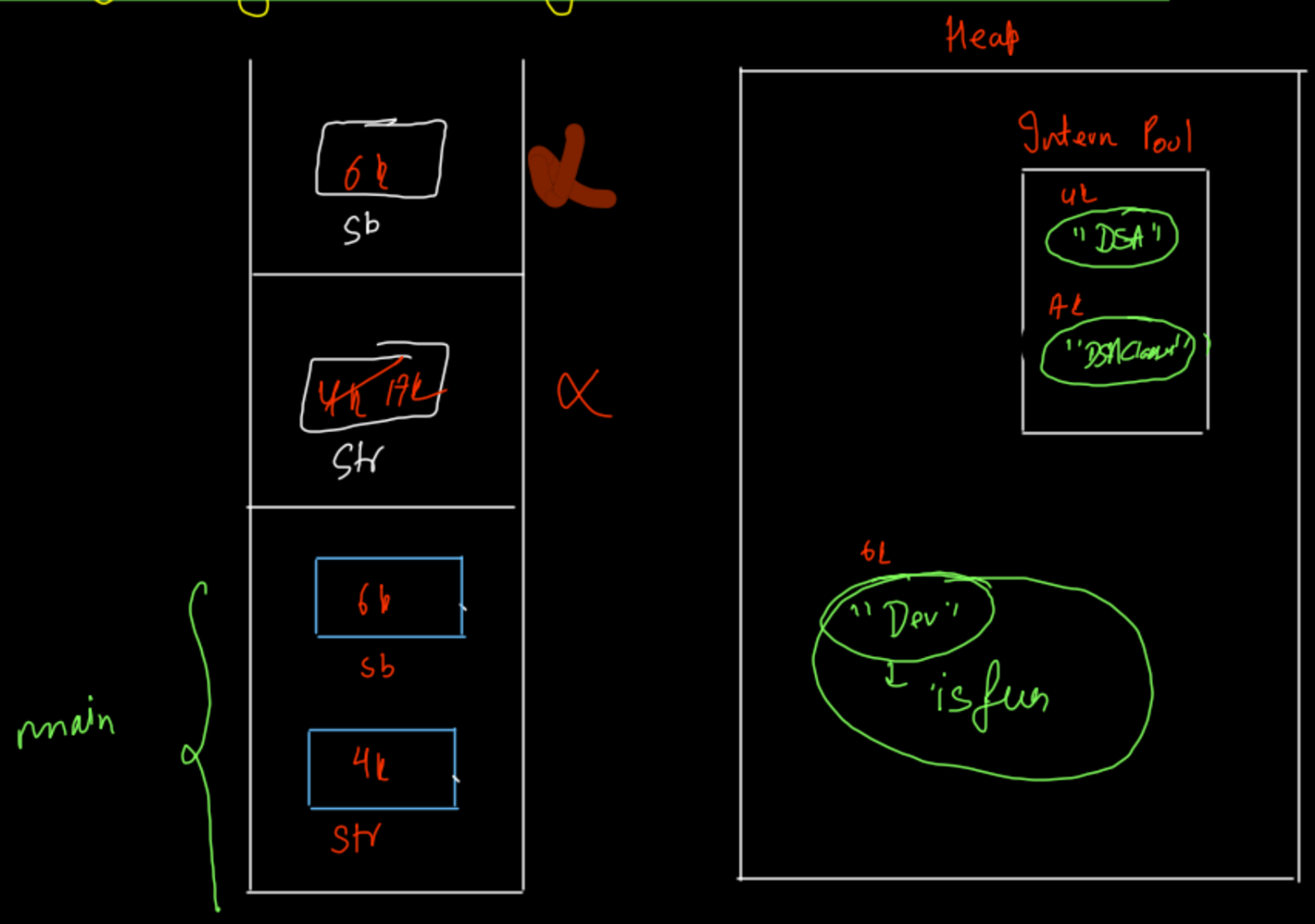
- → Passing String & StringBuilder into functions
- → Substring Method
- → Print All Palindromic Substrings
- → Reverse String
- → Change the String (Toggle Case)
- → String Compression
- → Reverse Words.

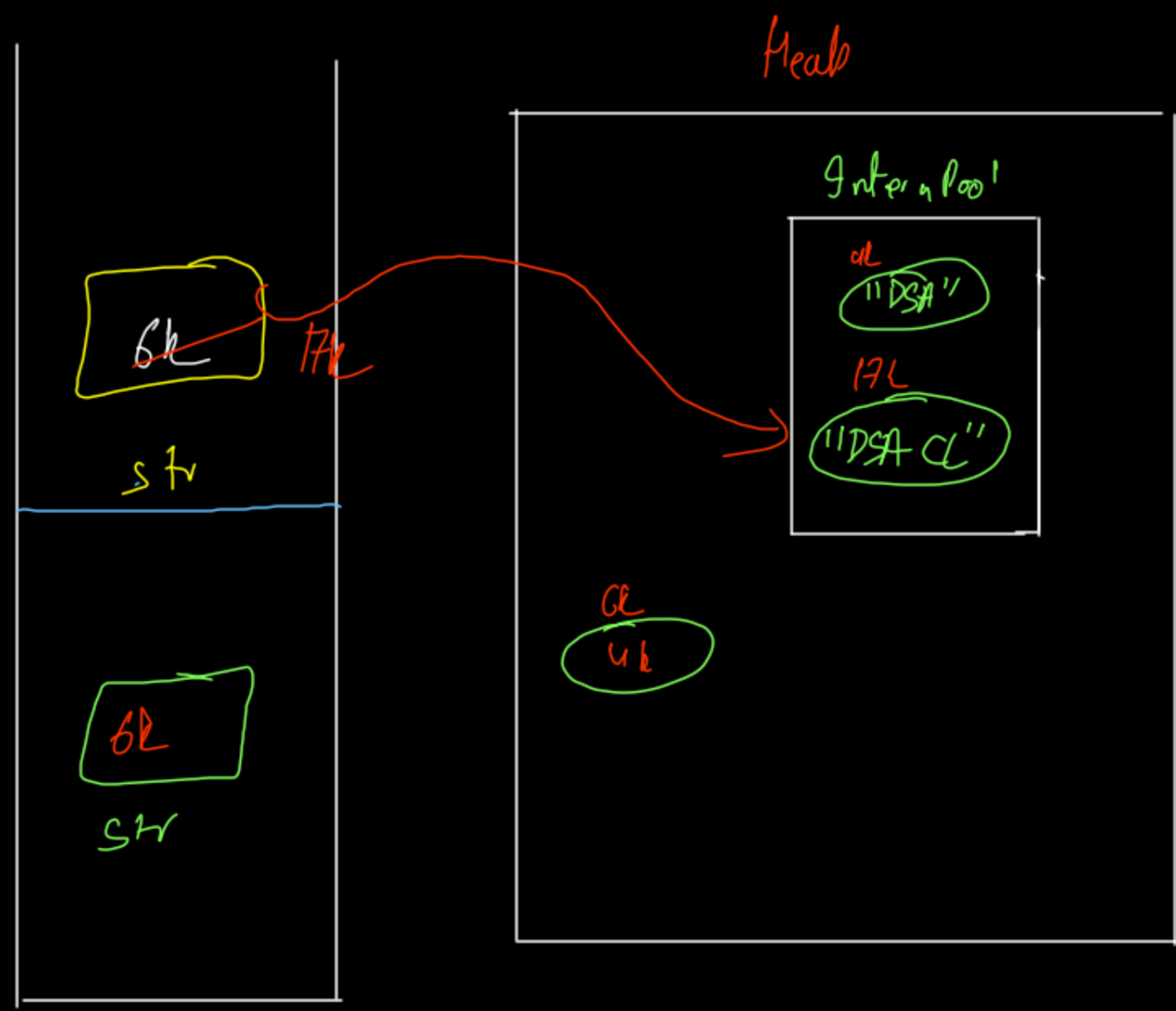
Passing String and StringBuilders into functions.

```java
import java.util.*;

public class Main {

    public static void modifyString(String str) {
        str = str + " Classes";
    }

    public static void modifyStringBuilder(StringBuilder sb) {
        sb.append(" is fun");
    }

    public static void main(String[] args) {
        String str = "DSA";
        StringBuilder sb = new StringBuilder("Dev");
        modifyString(str);
        modifyStringBuilder(sb);

        System.out.println(str);
        System.out.println(sb);
    }
}
```

Output ?

Passing String and StringBuilders into functions. (Memory Mapping)




Heap
Intern Pool
"DSA"
"DSA CL"
str
str

Subarrays

0 1 2

{1, 2, 3}

1 2

1,2 2,3

1,2,3 3

```
for(i=0; i<n ; i++)
{ for(j=i ; j<n ; j++)
  {
    for (k=i; k<=j ; k++)
    {
      }
```

String → substring() method

str = "abcdef" (0 1 2 3 4 5)

str.substring(2); → cdef

str.substring(start, end); → Returns a string (substring) which has the characters from our string str starting from index start till index end-1.

(1,2) → b

(1,4) → bcd

(0,5) → abcde

(0,6) → abcdef

Print all Substrings

```java
public static void main(String[] args) {
    String str = "abc";

    for(int i=0;i<str.length();i++) {
        for(int j=i;j<str.length();j++) {
            String subStr = str.substring(i,j + 1);
            System.out.println(subStr);
        }
    }
}
```

(6 1) → empty / blank string string

→ str.length() = 0

Print All Palindromic Substrings.

Print All Palindromic Substrings

Easy

Prev Next

1. You are given a string.
2. You have to print all palindromic substrings of the given string.

Input Format

A String

Output Format

All palindromic substrings(one in a line).
First, all palindromic substrings starting from first character of string will be printed, then from second character and so on.

"naan"

hi

~~lo~~ ~~lo~~ lo ~~hi~~ ~~hi~~

"naman"

0 1 2 3 4

while(lo< hi)

"~~lo~~ lo hi ~~hi~~"

nabn

hi lo

~~lo~~ ~~lo~~ ~~hi~~ ~~hi~~

"naan"

0 1 2 3

```
public static boolean isPalindrome(String str) {
    int lo = 0;
    int hi = str.length()-1;

    while(lo < hi) {
        if(str.charAt(lo) != str.charAt(hi))
        return false;
        lo++;
        hi--;
    }

    return true;
}

public static void solution(String str){
    //write your code here

    for(int i=0;i<str.length();i++) {
        for(int j=i;j<str.length();j++) {
            String subStr = str.substring(i,j+1);
            if(isPalindrome(subStr)) {
                System.out.println(subStr);
            }
        }
    }

}
```

Reverse String

= ["h", "e", "l", "l", "o"]

olleh

```java
class Solution {
    public void reverseString(char[] s) {
        int lo = 0;
        int hi = s.length -1;

        while(lo < hi) {
            char temp = s[lo];
            s[lo] = s[hi];
            s[hi] = temp;
            lo++;
            hi--;
        }
    }
}
```

Change the String

Change the string

Basic Accuracy: **61.52%** Submissions: **11111** Points: **1**

Given a string S, the task is to change the complete string to Uppercase or Lowercase depending upon the case for the first character.

Example 1:

Input:
S = "abCD"
Output: abcd
Explanation: The first letter (a) is lowercase. Hence, the complete string is made lowercase.

"abCD"

ch = 'a'

a = A+32

A = a-32

a - A = 32

b - B = 32

c - C = 32

long

int

```java
class Solution{
    String modify(String s){
        if(s.length() == 0 || s == null) return s;

        boolean isLowerCase = false;
        if(s.charAt(0) >= 'a' && s.charAt(0) <= 'z') isLowerCase = true;

        StringBuilder sb = new StringBuilder("");

        for(int i=0;i<s.length();i++) {
            char ch = s.charAt(i);
            if(ch >= 'a' && ch <= 'z' && isLowerCase == false) {
                ch = (char)(ch - 32);
            } else if(ch >= 'A' && ch <= 'Z' && isLowerCase == true) {
                ch = (char)(ch + 32);
            }
            sb.append(ch);
        }

        return sb.toString();
    }
```

0 1 2 3 4 5
a A B c D E

LC = true

ch = 'A'
a

aabc

Reverse words in a given string

Easy Accuracy: **50.0%** Submissions: **100k+** Points: **2**

This problem is part of GFG SDE Sheet. Click here to view more.

Given a String S, reverse the string without reversing its individual words. Words are separated by dots.

Example 1:

Input:
S = i.like.this.program.very.much
Output: much.very.program.this.like.i
Explanation: After reversing the whole string(not individual words), the input string becomes much.very.program.this.like.i

j = -1

i = -2

11

i.like.this.program

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

program.this.like.i

```java
String reverseWords(String S)
{
    // code here
    StringBuilder sb = new StringBuilder("");
    int i = S.length()-1;

    while(i >= 0) {
        int j = i-1;

        while(j >= 0 && S.charAt(j) != '.') {
            j--;
        }

        for(int x=j+1;x<=i;x++) {
            sb.append(S.charAt(x));
        }

        i = j-1;
        if(i >= 0) sb.append(".");
    }

    return sb.toString();

}
```

Lecture: 17

- → String Compression
- → Reverse String -2
- → Valid Palindrome -1
- → Valid Palindrome -2

443. String Compression

Medium 2398 4533 Add to List Share

Given an array of characters `chars`, compress it using the following algorithm:

Begin with an empty string `s`. For each group of **consecutive repeating characters** in `chars`:

- If the group's length is `1`, append the character to `s`.
- Otherwise, append the character followed by the group's length.

The compressed string `s` **should not be returned separately**, but instead, be stored **in the input character array** `chars`. Note that group lengths that are `10` or longer will be split into multiple characters in `chars`.

After you are done **modifying the input array,** return *the new length of the array*.

You must write an algorithm that uses only constant extra space.

j i i j

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

"aaaaaaaaaaaabbbbbbbbbcccccccccab"

j j j

```java
class Solution {
    public int compress(char[] chars) {
        int idx = 0;
        int i = 0;

        while(i < chars.length) {
            int j = i;

            while(j < chars.length && chars[j] == chars[i]) {
                j++;
            }

            chars[idx] = chars[i];
            idx++;

            String count = "" + (j-i);

            if(j-i > 1) {
               for(int x=0;x<count.length();x++) {
                   char ch = count.charAt(x);
                   chars[idx] = ch;
                   idx++;
               }
            }

            i = j;
        }

        return idx;
    }
}
```

541. Reverse String II

Easy 1222 2729 Add to List Share

Given a string `s` and an integer `k`, reverse the first `k` characters for every `2k` characters counting from the start of the string.

If there are fewer than `k` characters left, reverse all of them. If there are less than `2k` but greater than or equal to `k` characters, then reverse the first `k` characters and leave the other as original.

Example 1:

```
Input: s = "abcdefg", k = 2
Output: "bacdfeg"
```

Example 2:

```
Input: s = "abcd", k = 2
Output: "bacd"
```

i i

0 1 2 3 4 5 6 7 8 9 10

"abcdefghijk"

d c b a

lo hi

k = 4

i += 2 * k

2k

k k

k k

for current k elements that we have reversed

for next k elements that we don't have to reverse

lo = i

hi = min(length - 1, i + k - 1)

while(lo < hi) {

// swapping

}

```java
class Solution {
    public String reverseStr(String s, int k) {
        StringBuilder sb = new StringBuilder(s);

        for(int i=0;i<s.length(); i+= 2*k) {
            int lo = i;
            int hi = (int)Math.min(i+k,s.length()) - 1;

            while(lo < hi) {
                char temp = sb.charAt(lo);
                sb.setCharAt(lo,sb.charAt(hi));
                sb.setCharAt(hi,temp);
                lo++;
                hi--;
            }
        }

        return sb.toString();
    }
}
```

125. Valid Palindrome

Easy 4877 5989 Add to List Share

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string `s`, return `true` *if it is a* ***palindrome****, or* `false` *otherwise.*

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

```java
public boolean isLowerCase(char ch) {
    if(ch >='a' && ch <= 'z') return true;
    return false;
}

public boolean isUpperCase(char ch) {
    if(ch >='A' && ch <= 'Z') return true;
    return false;
}

public boolean isDigit(char ch) {
    if(ch >= '0' && ch <= '9') return true;
    return false;
}
```

```java
public String convert(String str) {

    StringBuilder sb = new StringBuilder("");

    for(int i=0;i<str.length();i++) {
        char ch = str.charAt(i);
        if(isLowerCase(ch) || isUpperCase(ch) || isDigit(ch)) {
            if(isUpperCase(ch)) {
                ch = (char)(ch + 32);
            }
            sb.append(ch);
        }
    }

    return sb.toString();
}
```

```java
public boolean isPalindrome(String s) {
    String str = convert(s);

    int lo = 0;
    int hi = str.length()-1;

    while(lo < hi) {
        if(str.charAt(lo) != str.charAt(hi)) return false;
        lo++;
        hi--;
    }

    return true;
}
```

680. Valid Palindrome II

Easy 6305 328 Add to List Share

Given a string `s`, return `true` *if the* `s` *can be palindrome after deleting* ***at most one*** *character from it.*

Example 1:

```
Input: s = "aba"
Output: true
```

Example 2:

```
Input: s = "abca"
Output: true
Explanation: You could delete the character 'c'.
```

"aba"

"abca"

lo lo lo lo lo lo hi hi hi hi hi hi

0 1 2 3 4 5 6 7 8 9 10 11 12 13

"abcdefghkedcba"

If string is a Palindrome,

1st part = 3rd part &

2nd part must be a palindrome.

```java
public boolean isPalindrome(String str) {

    int lo = 0;
    int hi = str.length()-1;

    while(lo < hi) {
        if(str.charAt(lo) != str.charAt(hi)) return false;
        lo++;
        hi--;
    }

    return true;
}
```

```java
  public boolean validPalindrome(String s) {
      int lo = 0;
      int hi = s.length()-1;

      while(lo < hi) {
          if(s.charAt(lo) == s.charAt(hi)) {
              lo++;
              hi--;
          } else {
              boolean ans1 = isPalindrome(s.substring(lo,hi));
              boolean ans2 = isPalindrome(s.substring(lo+1,hi+1));
              return ans1 || ans2;
          }
      }

      return true;
  }
}
```

Lecture - 18

→ Reverse Vowels

→ Reverse Words - 3

→ Reverse Words - 2

→ Arraylist (Syntax)

→ Wrapper Class → Autoboxing, Unboxing

Reverse Vowels

345. Reverse Vowels of a String

Easy 1910 2000 Add to List Share

Given a string `s`, reverse only all the vowels in the string and return it.

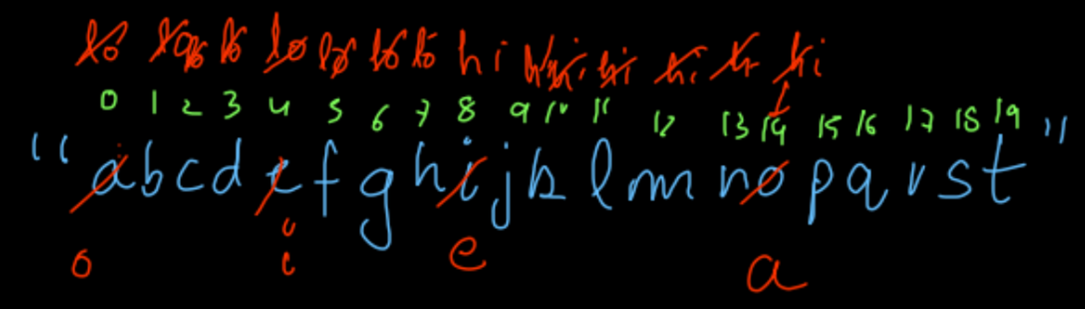
The vowels are `'a'`, `'e'`, `'i'`, `'o'`, and `'u'`, and they can appear in both cases.

Example 1:

```
Input: s = "hello"
Output: "holle"
```

Example 2:

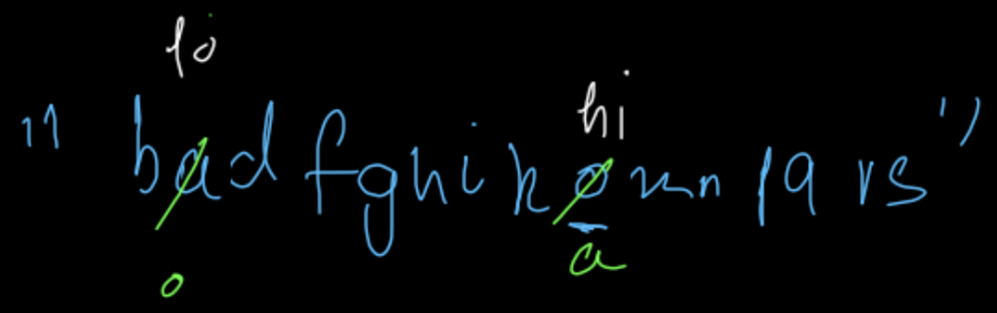
```
Input: s = "leetcode"
Output: "leotcede"
```



a, e, i, o & u

lo = 0

hi = sb.length() -1




```java
public boolean isVowel(char ch) {
    if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
        return true;
    } else if(ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
        return true;
    }

    return false;
}
```

```java
public String reverseVowels(String s) {
    int lo = 0;
    int hi = s.length() -1;

    StringBuilder sb = new StringBuilder(s);

    while(lo < s.length() && isVowel(s.charAt(lo)) == false) {
        lo++;
    }

    while(hi >= 0 && isVowel(s.charAt(hi)) == false) {
        hi--;
    }

    while(lo < hi) {
        char temp = sb.charAt(lo);
        sb.setCharAt(lo,sb.charAt(hi));
        sb.setCharAt(hi,temp);

        lo++;
        hi--;

          while(lo < s.length() && isVowel(s.charAt(lo)) == false) {
            lo++;
          }

          while(hi >= 0 && isVowel(s.charAt(hi)) == false) {
              hi--;
          }
    }

    return sb.toString();
}
```

557. Reverse Words in a String III

Easy 4103 209 Add to List Share

Given a string `s`, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

Example 1:

```
Input: s = "Let's take LeetCode contest"
Output: "s'teL ekat edoCteeL tsetnoc"
```

Example 2:

```
Input: s = "God Ding"
Output: "doG gniD"
```

"Hello world"

"olleH_ dlrow"

"Hello world"

"olleH_ dlrow"

$i = j + 1$

$i < s.length()$

```java
class Solution {
    public String reverseWords(String s) {
        int i = 0;
        StringBuilder sb = new StringBuilder("");

        while(i < s.length()) {
            int j = i + 1;

            while(j < s.length() && s.charAt(j) != ' ') {
                j++;
            }

            for(int x=j-1;x>=i;x--) {
                sb.append(s.charAt(x));
            }

            i = j + 1;
            if(i < s.length()) sb.append(' ');
        }

        return sb.toString();
    }
}
```

151. Reverse Words in a String

Medium 4002 4140 Add to List Share

Given an input string `s`, reverse the order of the **words**.

A **word** is defined as a sequence of non-space characters. The **words** in `s` will be separated by at least one space.

Return *a string of the words in reverse order concatenated by a single space.*

Note that `s` may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

i

j

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

"the _ _ sky _ _ is _ _ _ blue _ ."

i = j

"blue _ is _ sky _ the"

j i

blue _ _

j i

_ _ the

the

```java
class Solution {
    public String reverseWords(String s) {
        int i = s.length()-1;
        StringBuilder sb = new StringBuilder("");

        while(i >= 0 && s.charAt(i) == ' ') {
            i--;
        }

        while(i >= 0) {
            int j = i-1;

            while(j >= 0 && s.charAt(j) != ' ') {
                j--;
            }

            for(int x=j+1;x<=i;x++) {
                sb.append(s.charAt(x));
            }

            while(j >= 0 && s.charAt(j) == ' ') {
                j--;
            }

            i = j;
            if(i >= 0) sb.append(' ');
        }

        return sb.toString();
    }
}
```

ArrayList

Arrays are static.

int [] arr =

0	1	2	3	4
1	2	3	4	5

6

Array Lists are dynamic arrays. (Size can be modified (automatically))

// Declare & Initialization

```java
import java.util.*;

public class Main {

    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        System.out.println(arr);


    }
}
```

This does not print any Hashcode. You can directly print AL in Java.

```
Finished in 87 ms
[]
```

//Add (Insert at End (Append))

```java
public static void main(String[] args) {

    //declaration
    ArrayList<Integer> arr;
    //initialization
    arr = new ArrayList<>();

    arr.add(10);
    System.out.println(arr);

    arr.add(20);
    System.out.println(arr);

    arr.add(30);
    System.out.println(arr);


}
```

```
Finished in 87 ms
[10]
[10, 20]
[10, 20, 30]
```

// Size → Returns the size of AL (current no of ele)

```java
public static void main(String[] args) {

    //declaration
    ArrayList<Integer> arr;
    //initialization
    arr = new ArrayList<>();

    arr.add(10);
    System.out.println(arr.size());
    System.out.println(arr);

    arr.add(20);
    System.out.println(arr.size());
    System.out.println(arr);

    arr.add(30);
    System.out.println(arr.size());
    System.out.println(arr);

}
```

```
Finished in 93 ms
1
[10]
2
[10, 20]
3
[10, 20, 30]
```

Traversal & get() method.

```java
public class Main {

    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        arr.add(10);
        arr.add(20);
        arr.add(30);

        for(int i=0;i<arr.size();i++) {
            System.out.println(arr.get(i));
        }

    }
}
```

Returns the value at a particular index.

for Each

```java
public class Main {

    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        arr.add(10);
        arr.add(20);
        arr.add(30);

//          for(int i=0;i<arr.size();i++) {
//              System.out.println(arr.get(i));
//          }

        for(int ele : arr) {
            System.out.println(ele);
        }
    }
```

0 1 2 3 4

[10, 20, 30, 40, 50]

Add (index, value)

```java
    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        arr.add(10);
        arr.add(20);
        arr.add(30);
        arr.add(1,100);

//          for(int i=0;i<arr.size();i++) {
//              System.out.println(arr.get(i));
//          }

        for(int ele : arr) {
            System.out.println(ele);
        }
    }
}
```

[ ]

[10]

[10, 20]

0 1 2
[10, 20, 30]

0 1 2 3
[10, 100, 20, 30]

Wrapper Class

byte → Byte
short → Short
int → Integer
long → Long
float → float
double → Double
char → Character
boolean → Boolean

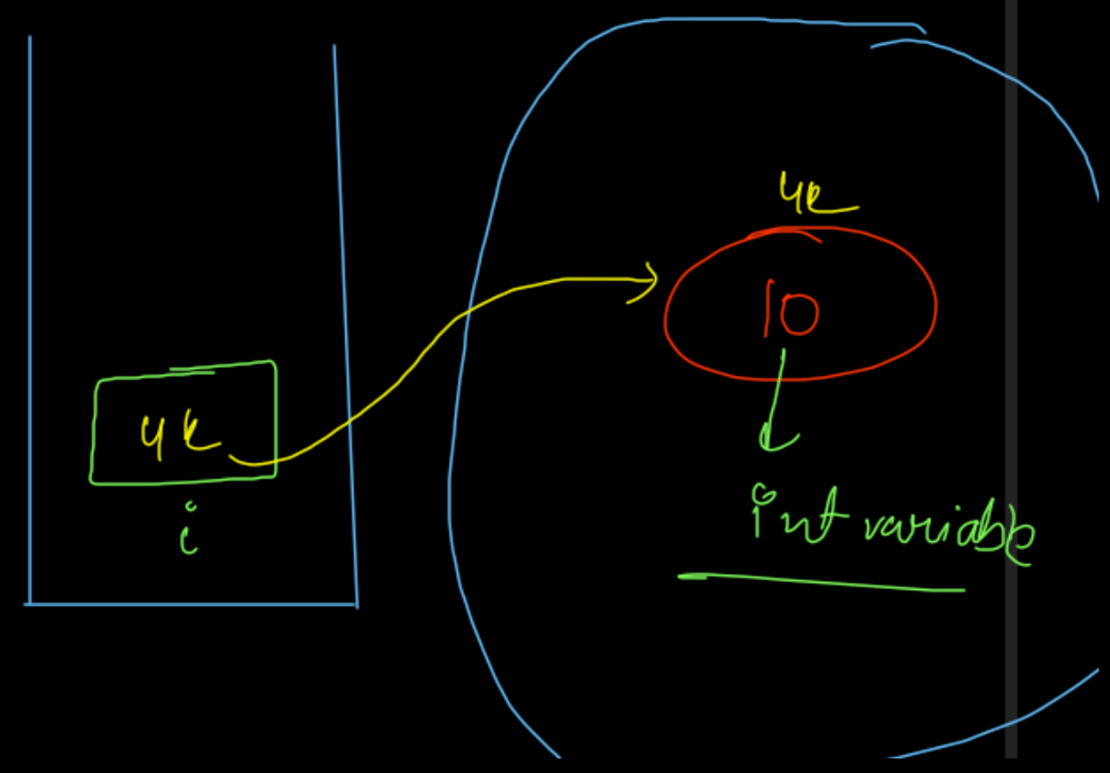
int x = 10;

Integer i = 10;

Integer i = new Integer(10);

↓

autoboxing



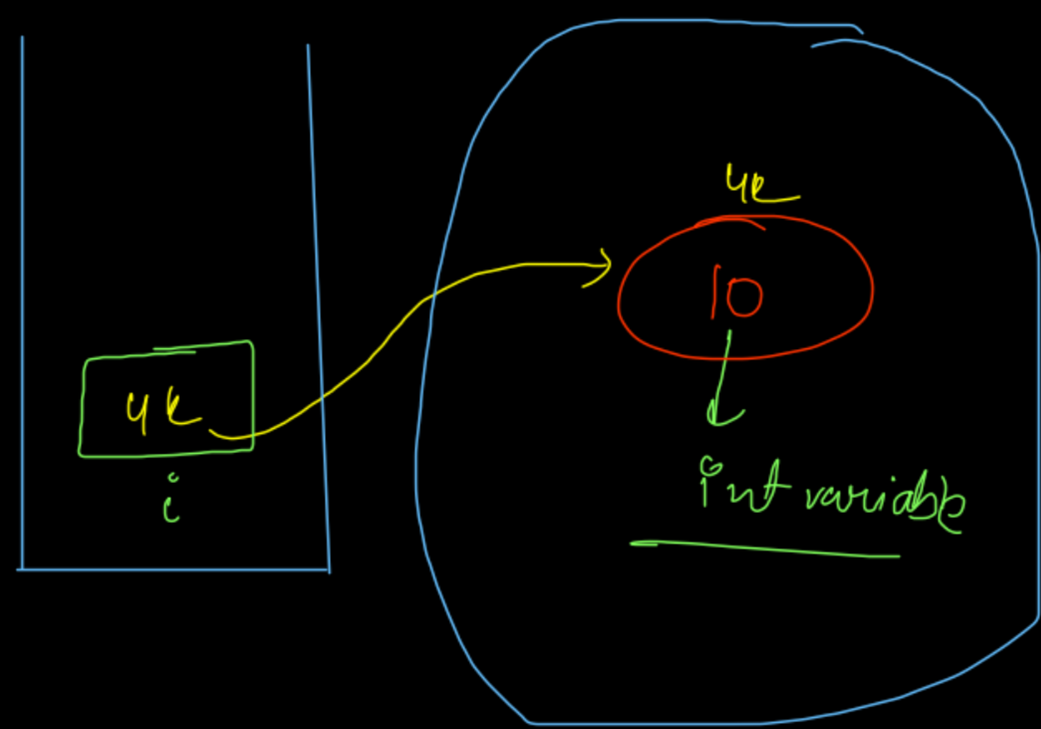
Integer i = 10;

int x = i;

↓

unboxing

(automatically, value is taken from the box)



```java
public static void main(String[] args) {
    Integer i = 10; //autoboxing
    int x = i; //unboxing

    System.out.println(i);
    System.out.println(x);
}
```

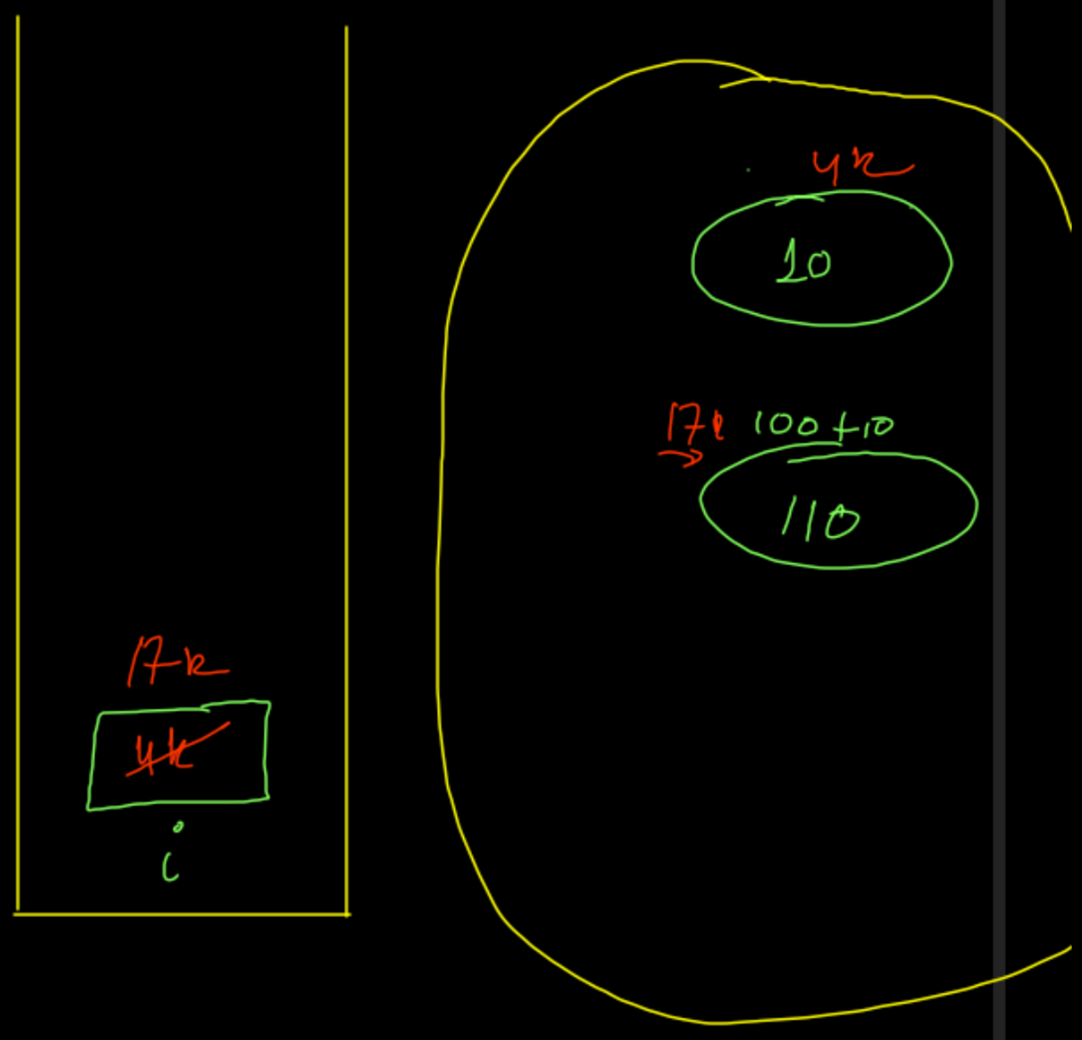
Lecture - 19

- Wrapper Classes (Immutability)
- ArrayList Memory Mapping
- Doubling
- ArrayList of ArrayLists (& its Memory Mapping)
- List vs ArrayList

```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        Integer i = 10; //autoboxing
        i += 100;

        System.out.println(i);
    }
}
```

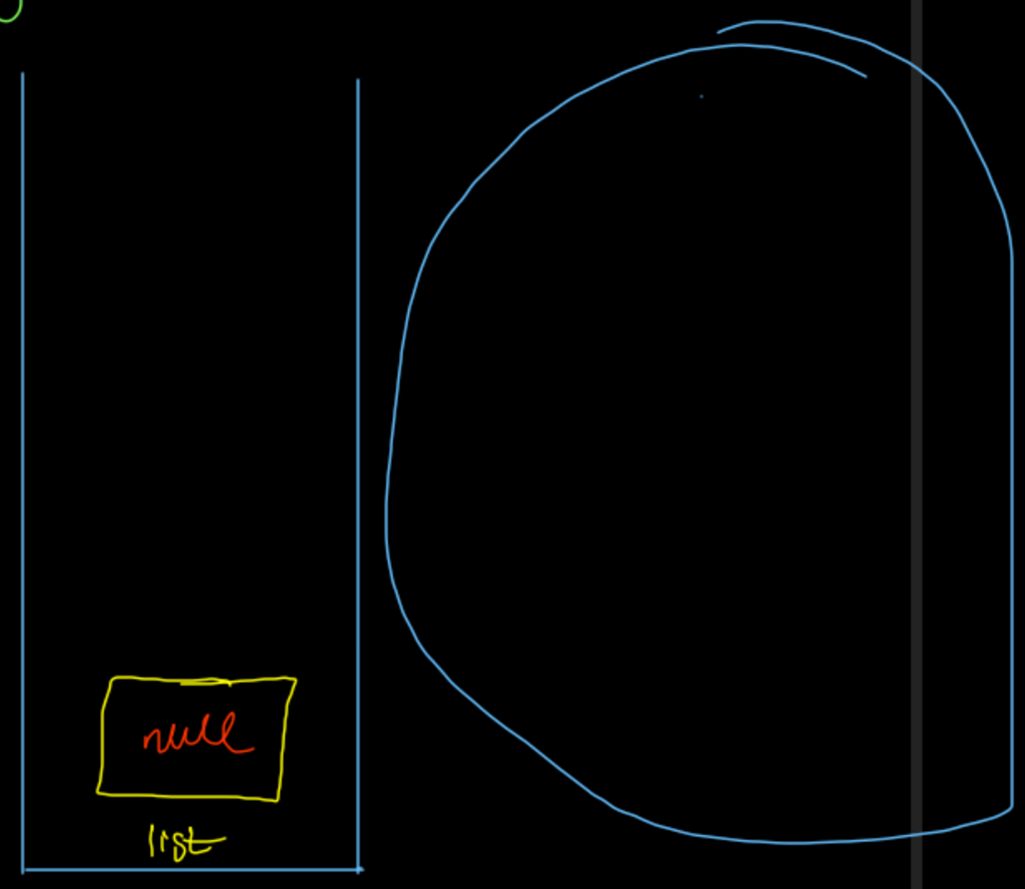
→ data membe

i = i + 100



Arraylist Declaration (Memory Mapping)

```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        ArrayList<Integer> list;
    }
}
```

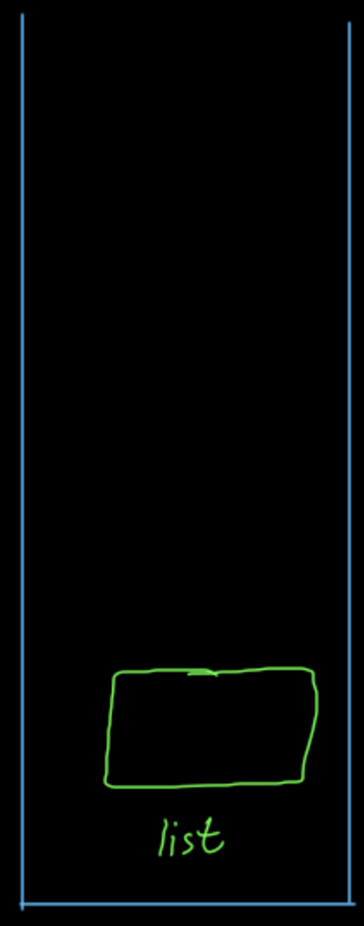


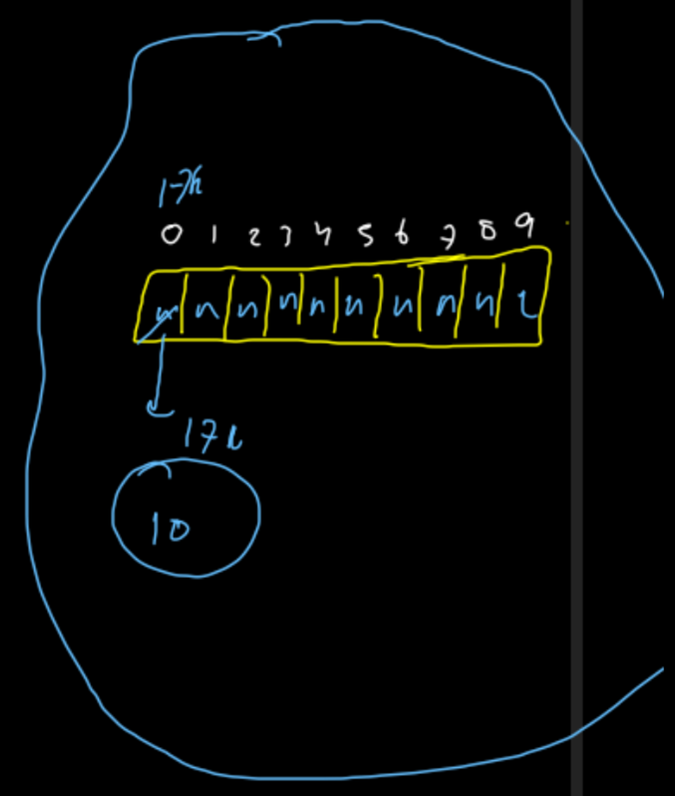
ArrayList Initialization

```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
    }
}
```

Initial size of AL in Java is 10.

list.add(10)

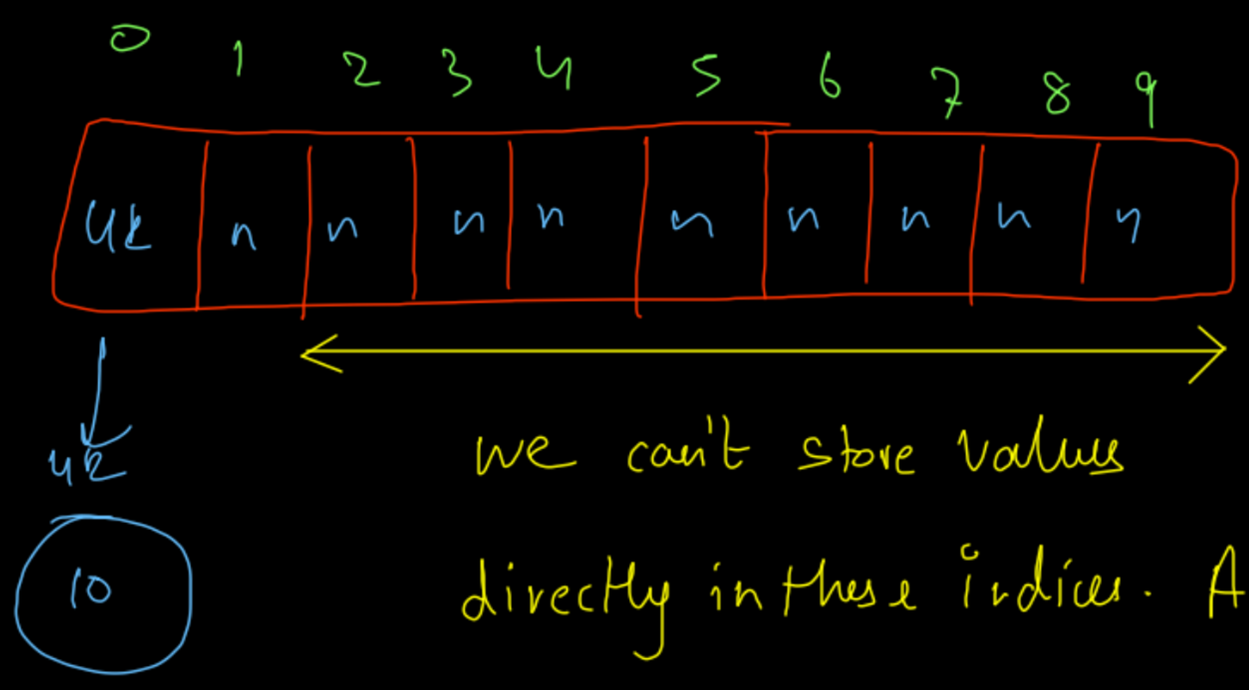




```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();

        for(int i=0;i<10;i++) {
            System.out.println(list.get(i));
        }
    }
}
```

```
Finished in N/A
java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0
  at line 64, java.base/jdk.internal.util.Preconditions.outOfBounds
  at line 70, java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex
  at line 266, java.base/jdk.internal.util.Preconditions.checkIndex
  at line 359, java.base/java.util.Objects.checkIndex
  at line 427, java.base/java.util.ArrayList.get
  at line 13, Main.main
```

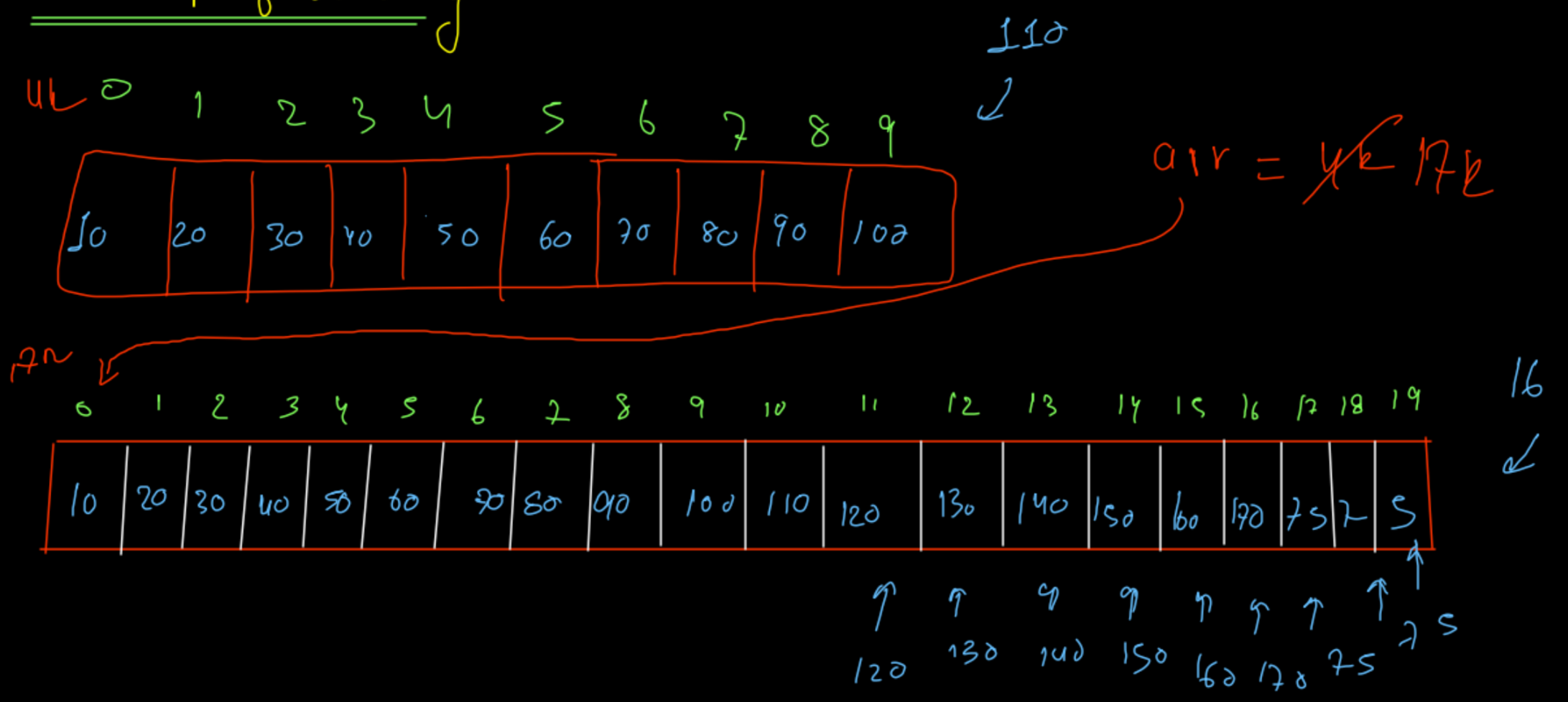
// Even though, there are 10 elements in an AL initially, Java gives us access to only those elements that we have inserted.



0	1	2
10	20	30

we can't store values directly in these indices. Appending a value in the list is allowed just at the index = list.size()

// Concept of Doubling



In Java, AL grows by 50% of its original size.

From the Javadoc, it says this is from Java 2 and on, so its a safe bet in the Sun JDK.

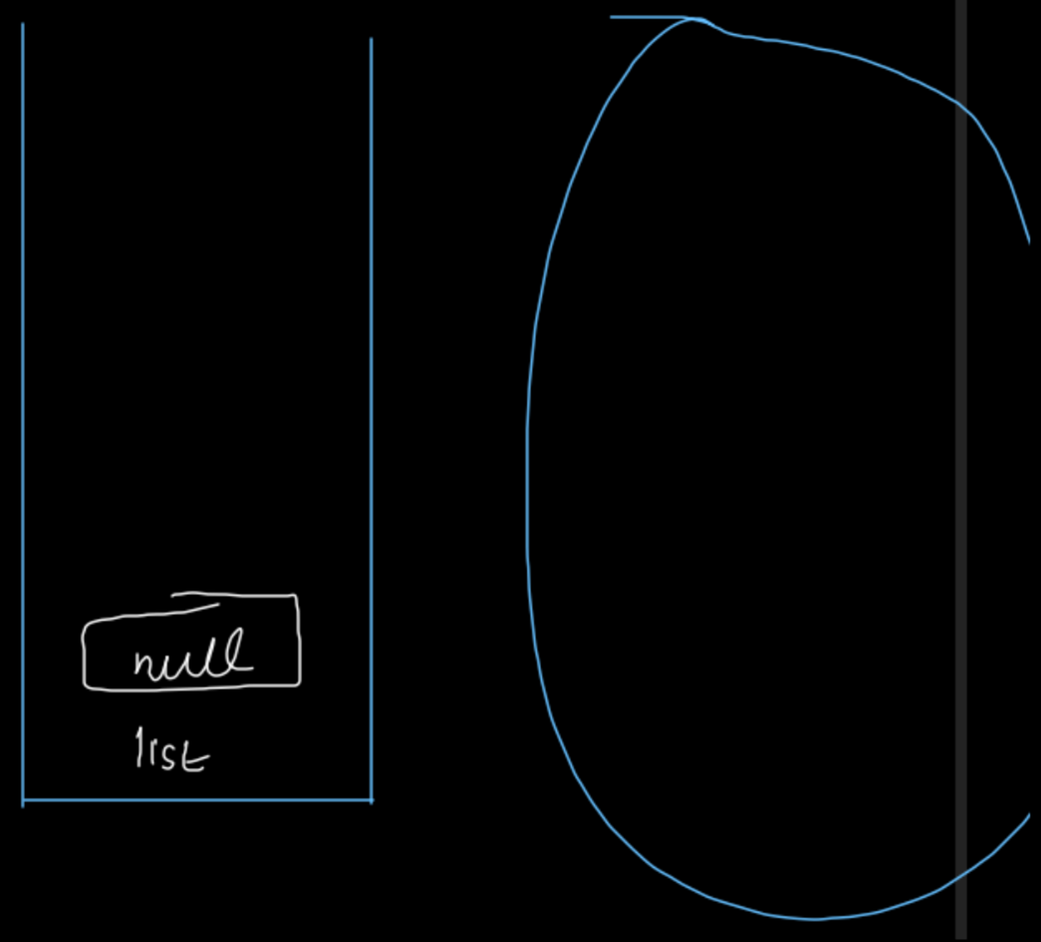
EDIT : for those who didn't get what's the connection between multiplying factor `1.5` and `int newCapacity = oldCapacity + (oldCapacity >> 1);`

`>>` is right shift operator which reduces a number to its half. Thus,

```
int newCapacity = oldCapacity + (oldCapacity >> 1);
=> int newCapacity = oldCapacity + 0.5*oldCapacity;
=> int newCapacity = 1.5*oldCapacity ;
```

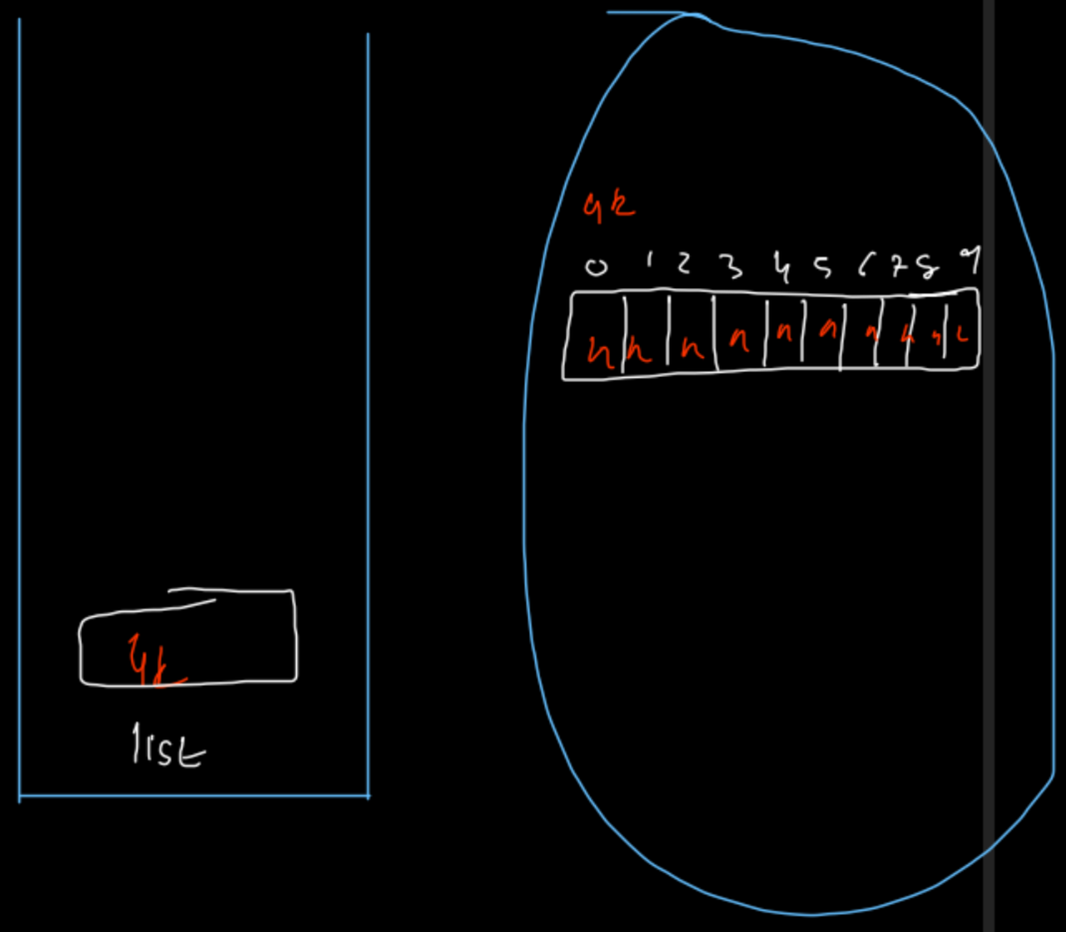
Array List of Array list (Declaration)

```java
import java.util.*;

public class Main {

    class MyIntegerArrayList {
        int[] arr;
    }

    public static void main(String[] args) {
        //declaration
        ArrayList<ArrayList<Integer>> list;
    }
}
```

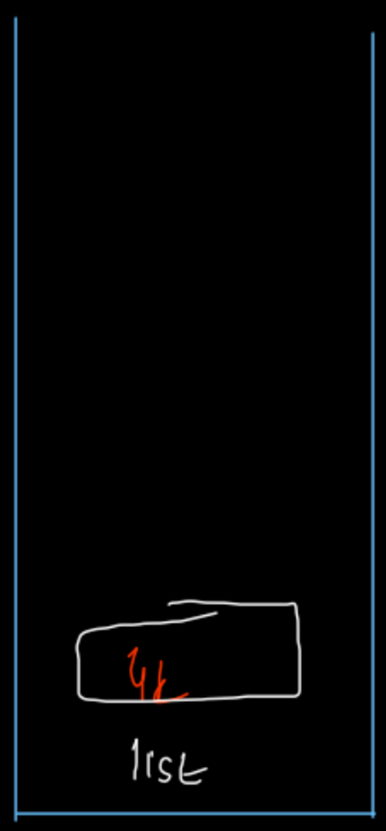


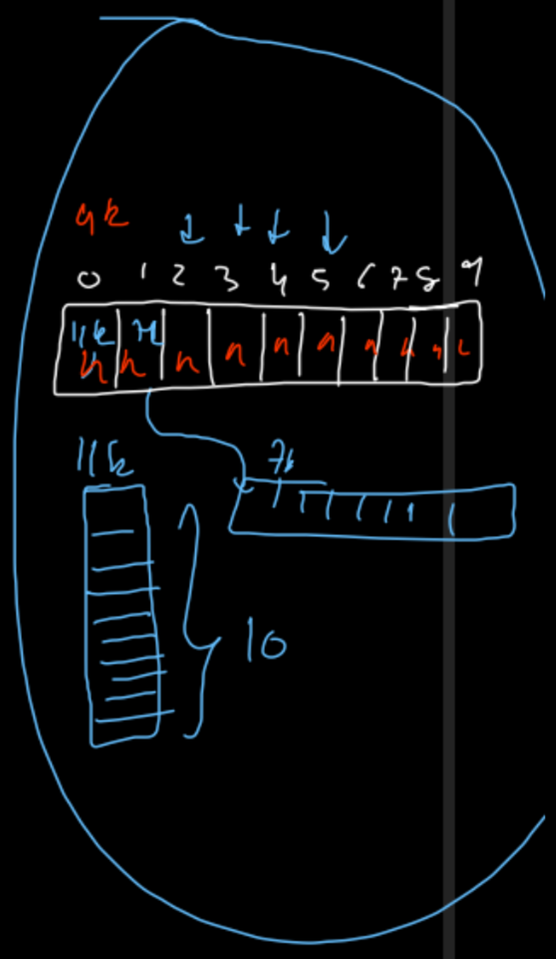
ArrayList of ArrayList (Initialization)

```java
import java.util.*;

public class Main {

    class MyIntegerArrayList {
        int[] arr;
    }

    public static void main(String[] args) {
        //declaration
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();
    }
}
```

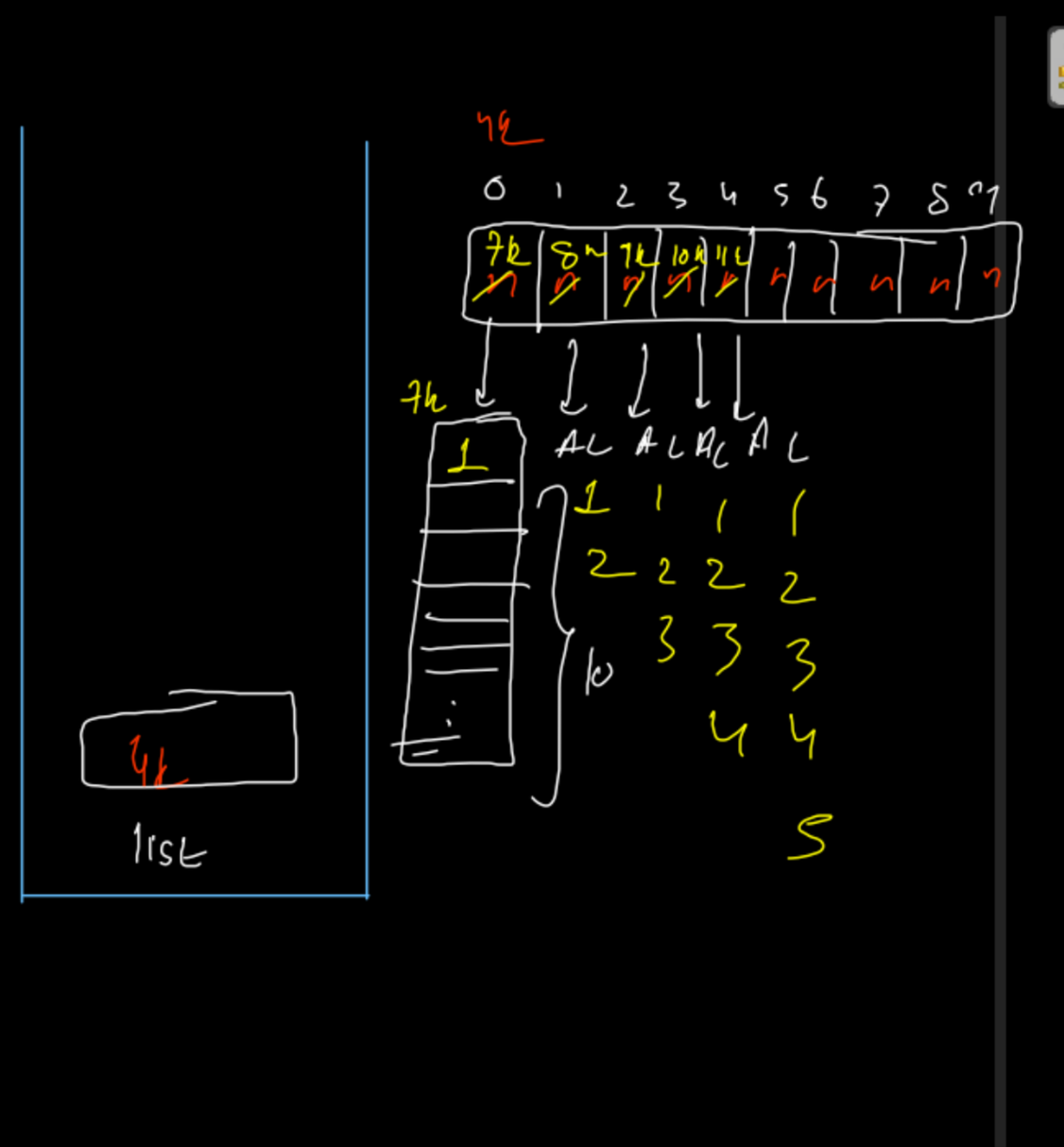


```java
import java.util.*;

public class Main {

    class MyIntegerArrayList {
        int[] arr;
    }

    public static void main(String[] args) {
        //declaration
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();

        for(int i=0;i<5;i++) {
            list.add(new ArrayList<>());
        }


    }
}
```



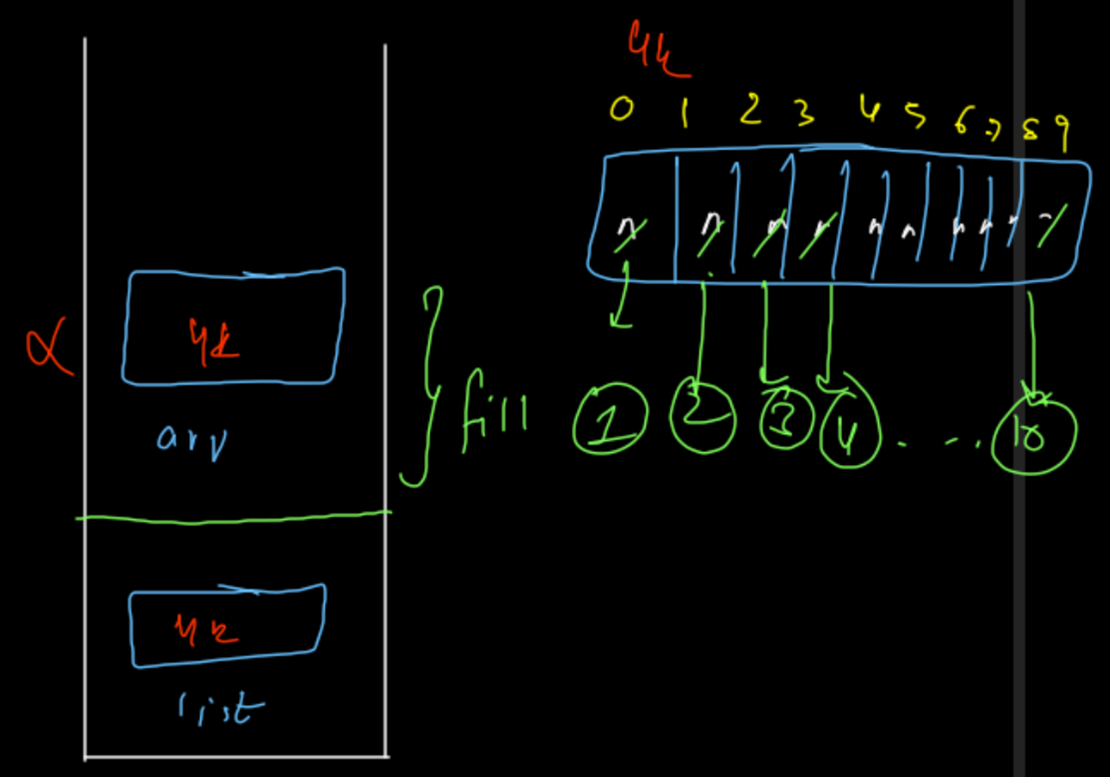


```java
public static void main(String[] args) {
    //declaration
    ArrayList<ArrayList<Integer>> list = new ArrayList<>();

    for(int i=0;i<5;i++) {
        list.add(new ArrayList<>());
    }

    for(int i=0;i<5;i++) {
        for(int j=0;j<=i;j++) {
            list.get(i).add(j+1);
        }
    }

    for(int i=0;i<5;i++) {
        System.out.println(list.get(i));
    }
}
```



ArrayList passed as a function Parameter

```java
import java.util.*;

public class Main {

    public static void fill(ArrayList<Integer> arr) {
        for(int i=1;i<=10;i++) {
            arr.add(i);
        }
    }

    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();

        System.out.println("Before calling the fill function " + list);

        fill(list);  //shallow copy.

        System.out.println("After calling the fill function" + list);
    }
}
```



List vs ArrayList

```
List<Integer> arr = new ArrayList<>();
                  or LinkedList<>();
```

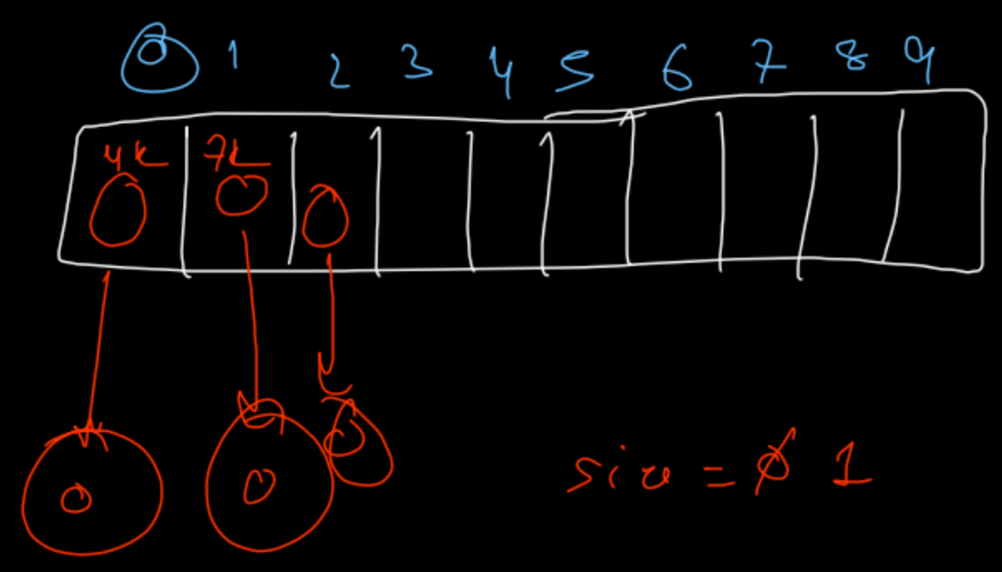
```
ArrayList<Integer> arr = new ArrayList<>();
```

```java
public static void main(String[] args) {
    //Object cannot be of type List
    //because List is an interface and
    //objects of interface cannot be created

    // List<Integer> arr = new List<>();

    List<Integer> arr = new ArrayList<>();
    arr = new LinkedList<>();

    //This is not allowed
    //If the reference is of type arraylist then the object should also
    //be of the type ArrayList
    ArrayList<Integer> list = new LinkedList<>();
    }
}
```

List<Integer> → LC
Arraylist → GfG
} returns

$i = \cancel{1}\ 2$

list.add(0);



size = $\cancel{0}$ 1

Java Collection | Set 1 (ArrayList) Part-1

Basic Accuracy: **50.38%** Submissions: **2157** Points: **1**

ArrayList are dynamic size arrays. Try this problem using ArrayList.

Given a ArrayList of **N** elements and a integer **Q** defining the type of query(which will be either 1 or 2) :
Q = 1 includes two integers **p** and **r**. Which means insert the value r at index p in the ArrayList and print the whole updated ArrayList.
Q = 2 includes one integer **p**. In this query print the index at which the value p is last found in the ArrayList. If the value p is not found in the ArrayList then print "**-1**".

NOTE: Assume 0 based indexing

p = 10

[10] →

[-1] –

0 1 2 3 4 5 6
{10, 20, 10, 30, 10, 40, 50}

```java
ArrayList<Integer> solve(int N, ArrayList<Integer> A, int Q, ArrayList<Integer> Que
{
    // code here
    if(Q == 1) {
        int p = Query.get(0);
        int r = Query.get(1);
        A.add(p,r);
        return A;
    } else {
        ArrayList<Integer> res = new ArrayList<>();
        int p = Query.get(0);
        for(int i=A.size()-1;i>=0;i--) {
            if(A.get(i) == p) {
                res.add(i);
                return res;
            }
        }

        res.add(-1);
        return res;
    }
}
```

Java Collection | Set 1 (ArrayList) Part-2

Easy Accuracy: **46.02%** Submissions: **6083** Points: **2**

Implement different operations on a ArrayList A .

Input:

The first line of input contains an integer **T** denoting the no of test cases . Then T test cases follow. The first line of input contains an integer **Q** denoting the no of queries . Then in the next line are **Q** space separated queries .

A query can be of five types

1. a x (Adds an element x to the ArrayList A at the end)
2. b (Sorts the ArrayList A in ascending order)
3. c (Reverses the ArrayList A)
4. d (prints the size of the ArrayList)
5. e (prints space separated values of the ArrayList)
5. f (Sorts the ArrayList A in descending order)

{9, 8, 6, 3, 4, 7, 2}

↓

{2, 3, 4, 6, 7, 8, 9}

```java
/*sort the ArrayList A in ascending order*/
void sort_ArrayList_Asc(ArrayList<Integer> A)
{
    // Your code here
    Collections.sort(A);
}
```

```java
/*reverses the ArrayList A*/
void reverse_ArrayList(ArrayList<Integer> A)
{
    // Your code here
    Collections.reverse(A);
}
```

{1, 2, 3, 4, 5}

↓ reverse

{5, 4, 3, 2, 1}

{9, 8, 6, 3, 4, 7, 2}

↓ sort (descending order)

→ ① Sort {2, 3, 4, 6, 7, 8, 9}

→ ② Reverse {9, 8, 7, 6, 4, 3, 2}

{9, 8, 7, 6, 4, 3, 2}

Remove Primes

Easy

Prev Next

1. You are given an ArrayList of positive integers.
2. You have to remove prime numbers from the given ArrayList and return the updated ArrayList.

Note -> The order of elements should remain same.

Input Format

A number N
arr1
arr2.. N numbers

$\hat{i}$ i

$[\overset{0}{10}, \overset{1}{\cancel{13}}, \overset{2}{11}, \overset{3}{17}, \overset{4}{20}, \overset{5}{22}, \overset{6}{23}]$

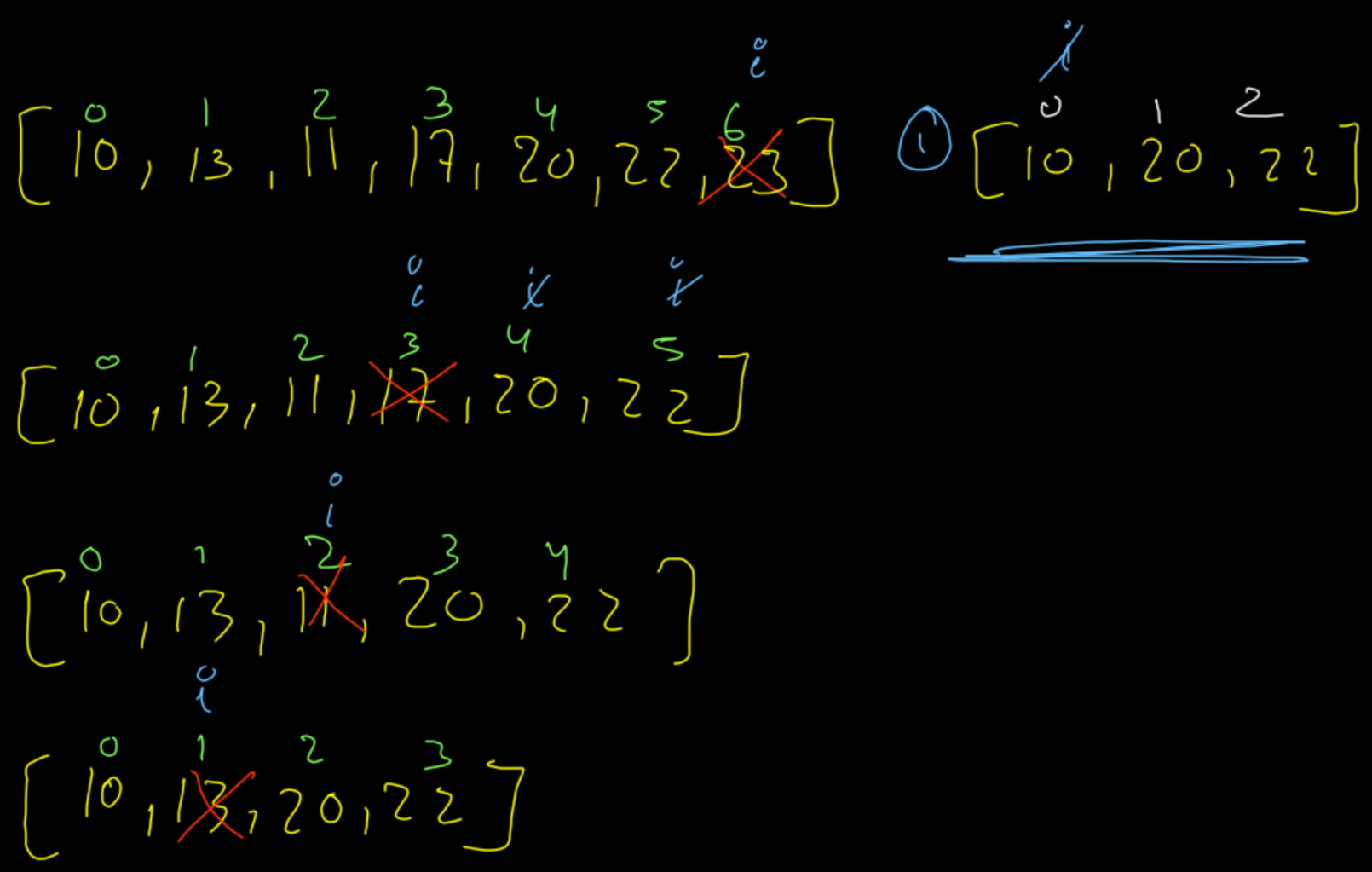
i

$[\overset{0}{10}, \overset{1}{11}, \overset{2}{17}, \overset{3}{20}, \overset{4}{22}, \overset{5}{23}]$

} Concurrent access & modification anomaly.

Increment i only when we encounter a non-prime no.

```java
public static void solution(ArrayList<Integer> al){
    // write your code here

    int i = 0;
    while(i < al.size()) {
        if(isPrime(al.get(i)) == true) {
            al.remove(i);
        } else {
            i++;
        }
    }

}
```




```java
public static void solution(ArrayList<Integer> al){
    // write your code here

    for(int i=al.size() -1;i>=0;i--) {
        if(isPrime(al.get(i)) == true) {
            al.remove(i);
        }
    }

}
```

// Traverse in Reverse

Recursion

PMI (Principle of Mathematical Induction)

PT $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$ } → Expectation

① Trivial Case (Small Problem) → Base Case

for $i = 1$

$LHS = \sum_{i=1}^{1} i = 1$

$RHS = \frac{1(\cancel{1+1})}{\cancel{2}} = 1$

for $i=1$, $LHS = RHS$

② Assume that LHS = RHS for $i = k$ → faith

$$LHS = \sum_{i=1}^{k} i = 1+2+3+4 \dots + k$$

$$RHS = \frac{k(k+1)}{2}$$

Assume that they are equal

③ for $i = k+1$ → Meeting expectation with faith.

$$LHS = \sum_{i=1}^{k+1} i = 1+2+3+4 \dots + k + (k+1)$$

$$= \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}$$

$$RHS = \frac{N(N+1)}{2} \rightarrow \text{Put } N = k+1$$

$$= \frac{(k+1)(k+1+1)}{2} = \frac{(k+1)(k+2)}{2}$$

LHS = RHS for $i = k+1$

By PMI, this formula holds true.

$\sum^{N} i$

1 → true

if it is true for k, it will be true for k+1

k k+1

1 2 3 4 . . . . N

k+1

Recursion (Divide & Conquer)

Print Decreasing

① Expectation : printDecreasing(N) → N, N-1, N-2, N-3, ⋮, 1 for N=5: 5, 4, 3, 2, 1

② faith : pd(5) → 5, 4, 3, 2, 1 } pd(4) pd(4) → 4, 3, 2, 1 pd(N) → N, N-1, N-2, N-3, ⋮, 1 } pd(N-1)

// We have faith that pd(N-1) will do its work.

③ Meeting Expectation with faith:

pd(N) → N ①
pd(N-1) ②

for (N=5)
5 → Print N
4, 3, 2, 1 } pd(4) pd(N-1)

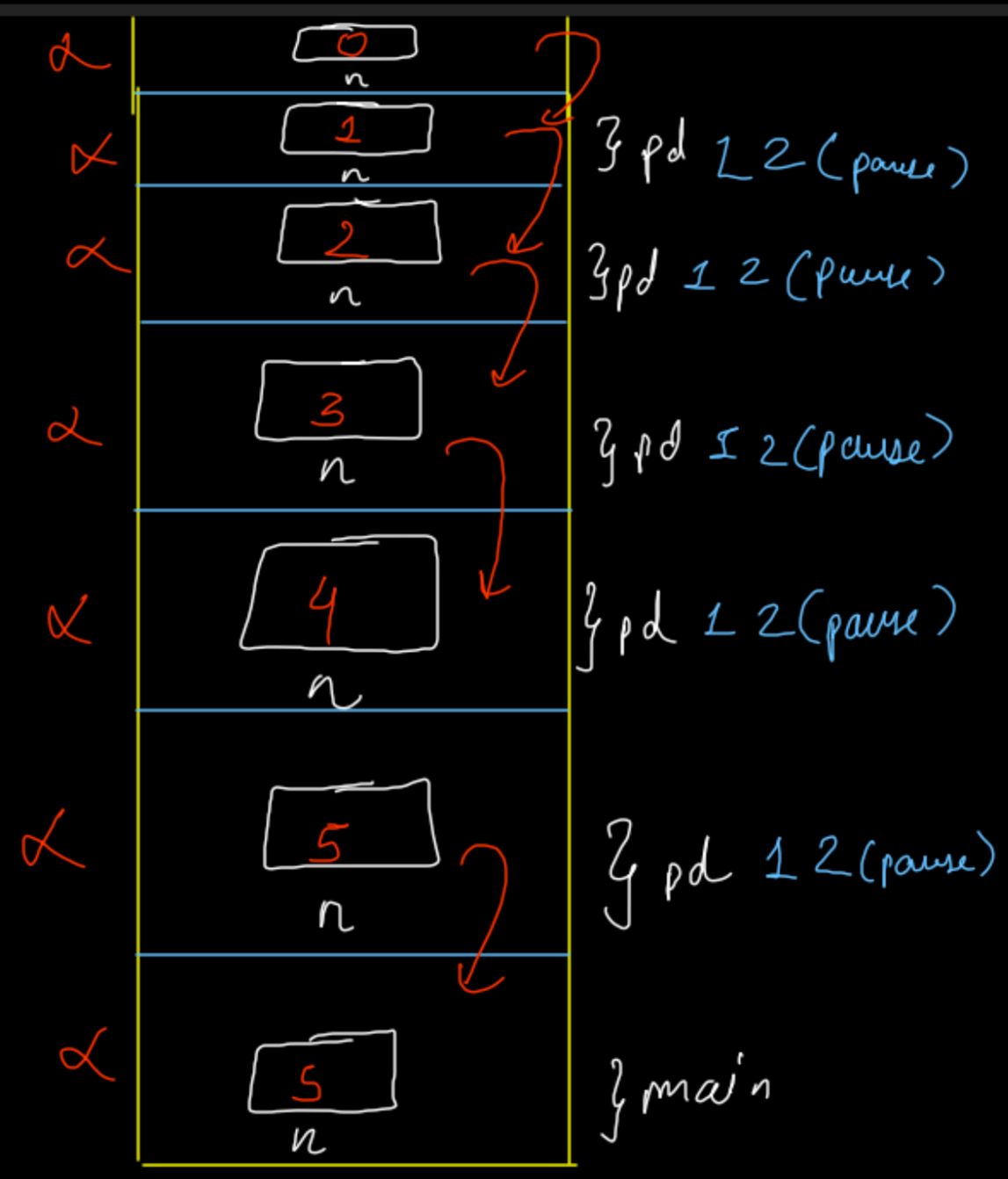
High Level Thinking

```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    printDecreasing(n);
}

public static void printDecreasing(int n){

    System.out.println(n);
    printDecreasing(n-1);
}

}
```

5
4
3
2
1

if (n == 0) → Base Case
return;

Low-level thinking



```java
import java.io.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        printDecreasing(n);
    }

    public static void printDecreasing(int n){

        if(n == 0) return; //base case

        System.out.println(n);
        printDecreasing(n-1); //recursive call
    }

}
```

Tail Recursion

(when recursive call is the last statement of a recursive function)

Print 1 To N Without Loop

School Accuracy: **51.9%** Submissions: **74255** Points: **0**

Print numbers from **1 to N** without the help of loops.

Example 1:

Input:
N = 10
Output: 1 2 3 4 5 6 7 8 9 10

① Expectation : pi(N) → 1, 2, 3, 4, ⋮, N

pi(5) → 1, 2, 3, 4, 5

② faith : pi(N-1) → 1, 2, 3, 4, ⋮, N-1

pi(4) → 1, 2, 3, 4

③ Meeting expectation with faith

pi(n-1);
Print(n);

pi(4) → 1, 2, 3, 4

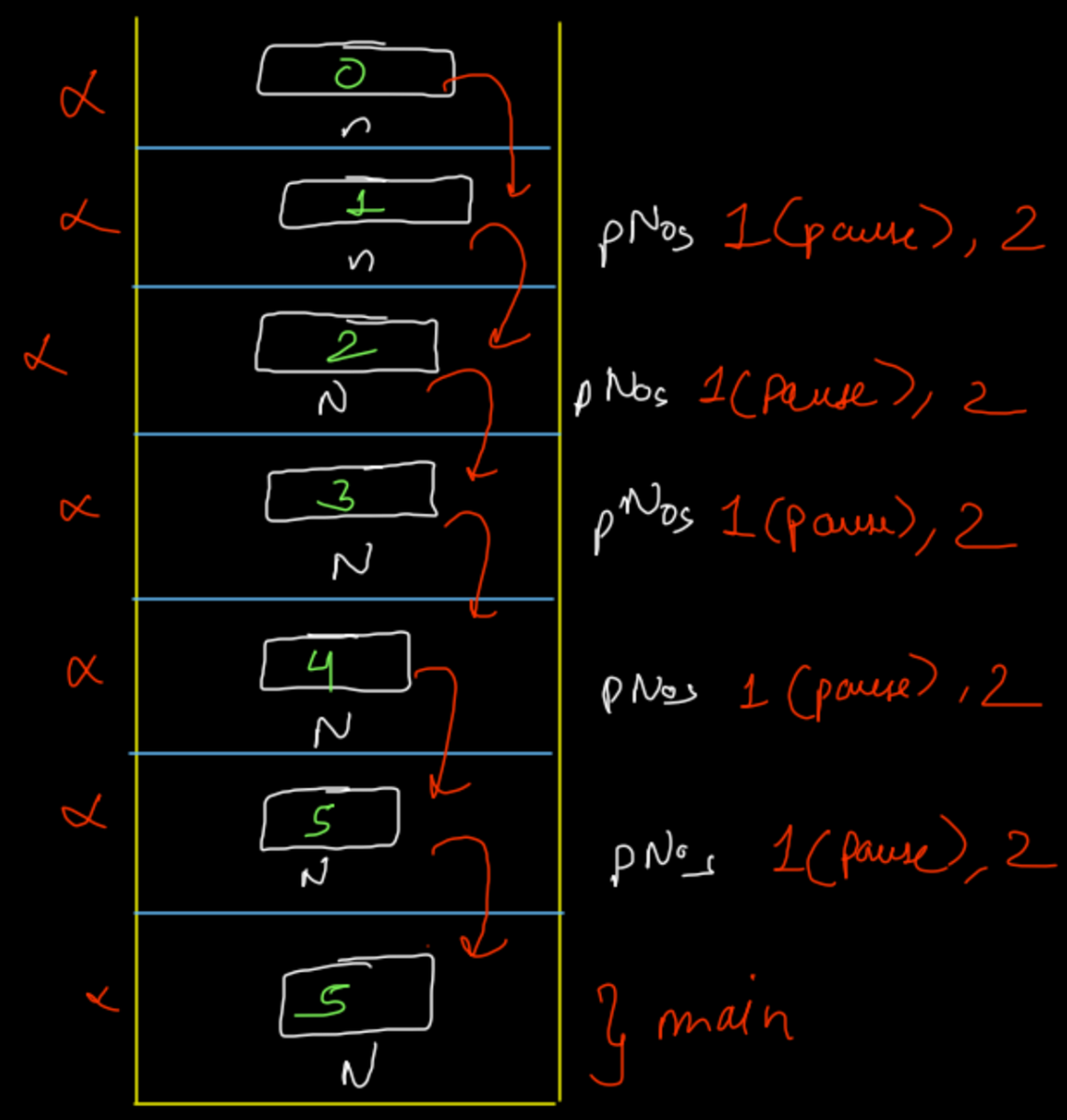
print(5) → 5

```java
class Solution
{

  public void printNos(int N
  {
      //Your code here

    1 printNos(N-1);
    2 System.out.println(N);
  }
}
```

if (n==0) return;

Base Case

1
2
3
4
5



```java
class Solution
{

  public void printNos(int N)
  {
      //Your code here

      if(N == 0) return;

      printNos(N-1);
      System.out.print(N + " ");
  }
}
```

Head Recursion

{ when recursion call is the first statement of a recursive function }

Print Increasing Decreasing

Easy

Prev Next

1. You are given a positive number n.
2. You are required to print the counting from n to 1 and back to n again.
3. You are required to not use any loops. Complete the body of pdi function to achieve it. Don't change the signature of the function.

Note -> The online judge can't force you to write the function recursively but that is what the spirit of question is.Write recursive and not iterative logic. The purpose of the question is to aid learning recursion and not test you.

$N = 3$

3
2
1
1
2
3

① Expectation : pdi(N)

N
N-1
⋮
1
1
2
⋮
N

pdi(3) → 3
2
1
1
2
3

② faith : pdi(N-1) → N-1
N-2
⋮
1
1
2
⋮
N-1

pdi(2) → 2
1
1
2

---

③ Meeting expectation with faith:

Print(N)

pdi(N-1);

Print(N);

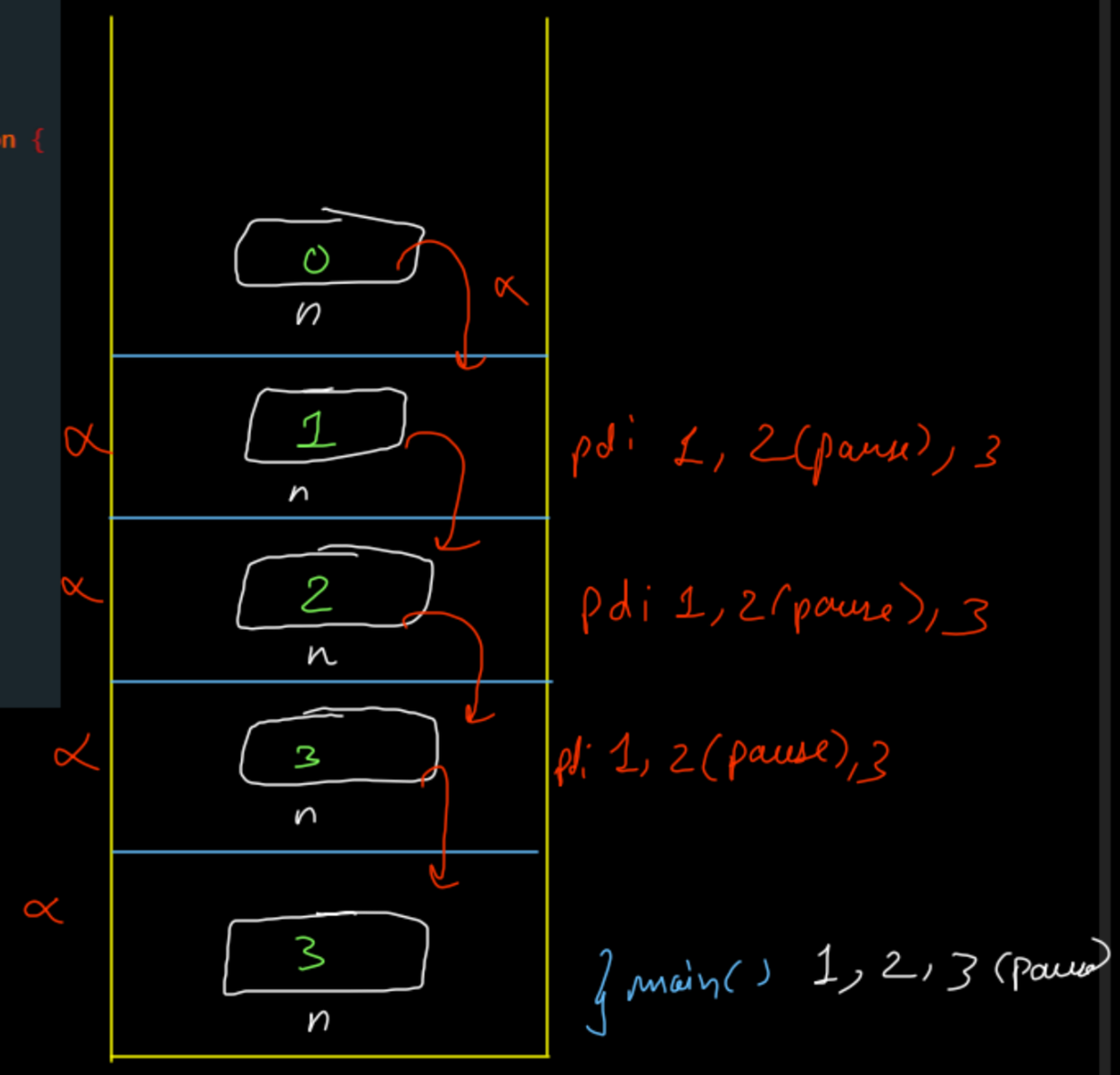
pdi(3) → Print(3)

pdi(2) → 2
1
1
2

Print(3)

```java
import java.io.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        pdi(n);
    }

    public static void pdi(int n){
        if (n == 0) return;
        System.out.println(n); 1

        //recursion call
        pdi(n-1); 2

        System.out.println(n); 3
    }

}
```

3
2
1
1
2
3



```java
import java.io.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        pdi(n);
    }

    public static void pdi(int n){

        if(n == 0) return;

        System.out.println(n);

        //recursion call
        pdi(n-1);

        System.out.println(n);
    }

}
```

Factorial

Basic Accuracy: **53.69%** Submissions: **35842** Points: **1**

Given a positive integer, **N**. Find the factorial of **N**.

$5! = 1 \times 2 \times 3 \times 4 \times 5$

or

$5 \times 4 \times 3 \times 2 \times 1$

$N! = N * (N-1) * (N-2) \ldots\ldots 1$

① Expectation : factorial (N) → N * (N-1) * (N-2) . . . 1

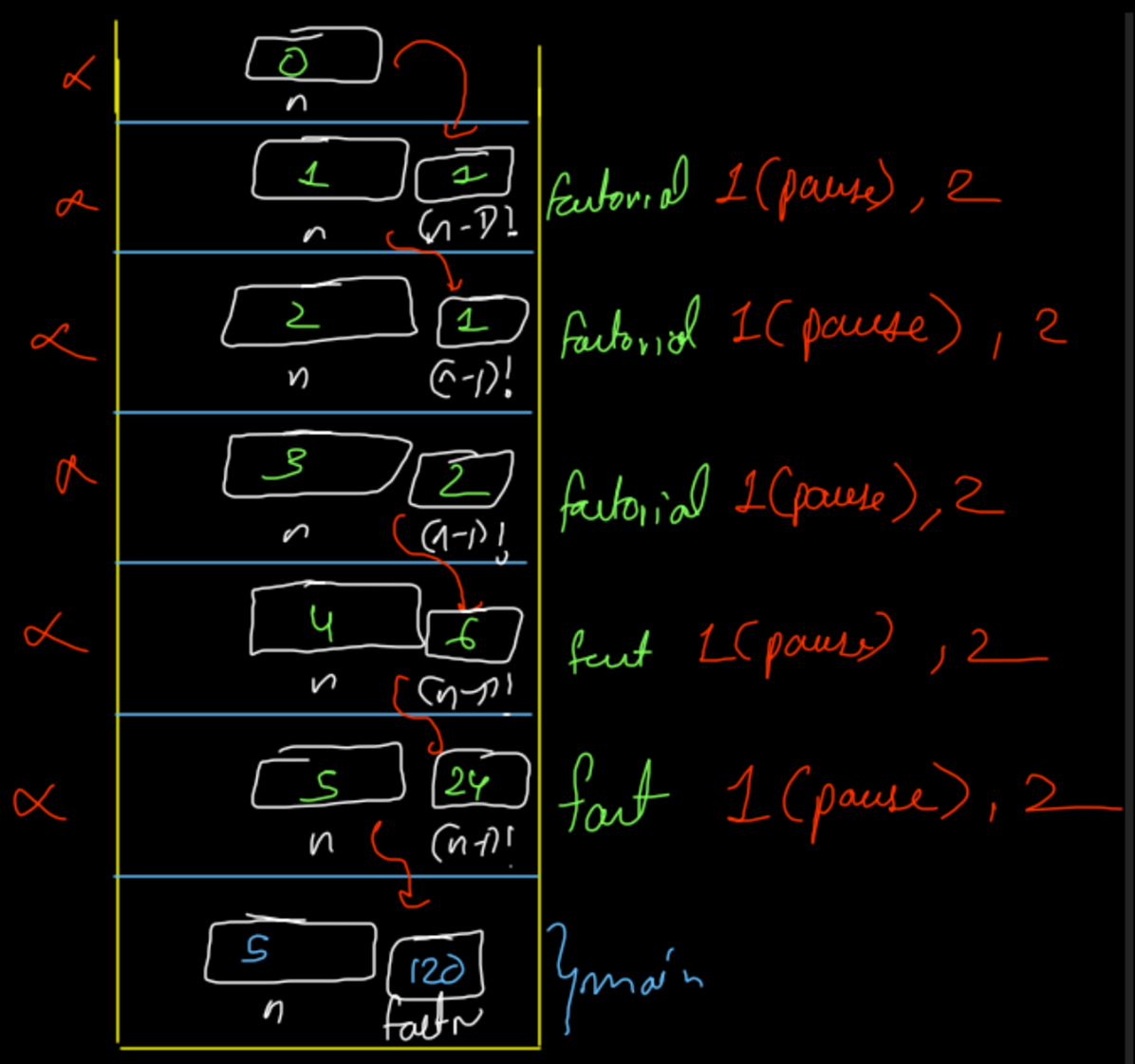
② faith : factorial (N-1) → (N-1) * (N-2) * (N-3) 1

③ Meeting expectation with faith : N * factorial (N-1)

```
class Solution{
    static long factorial(int N){
        long nm1Factorial = factorial(N-1);
        return N * nm1Factorial;
    }
}
```

(1) 1 2

if (n==0) return 1;



```java
class Solution{
    static long factorial(int N){
        // code here
        if(N == 0) return 1L;

        long nm1Factorial = factorial(N-1);
        return N * nm1Factorial;
    }
}
```

Print Zigzag

Easy

Prev Next

1. Here are a few sets of inputs and outputs for your reference
Input1 -> 1
Output1 -> 1 1 1

Input2 -> 2
Output2 -> 2 1 1 1 2 1 1 1 2

Input2 -> 3
Output3 -> 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

2. Figure out the pattern and complete the recursive function pzz to achieve the above for any positive number n.

Note -> The online judge can't force you to write the function recursively but that is what the spirit of question is.Write recursive and not iterative logic. The purpose of the question is to aid learning recursion and not test you.

Expectation

3211121112321112111123 → for N = 3

faith :

2111121112 → for N = 2

Meeting Expec : With faith

3211121112 3211121112 3

① Print N

② pzz(N-1)

③ Print(N)

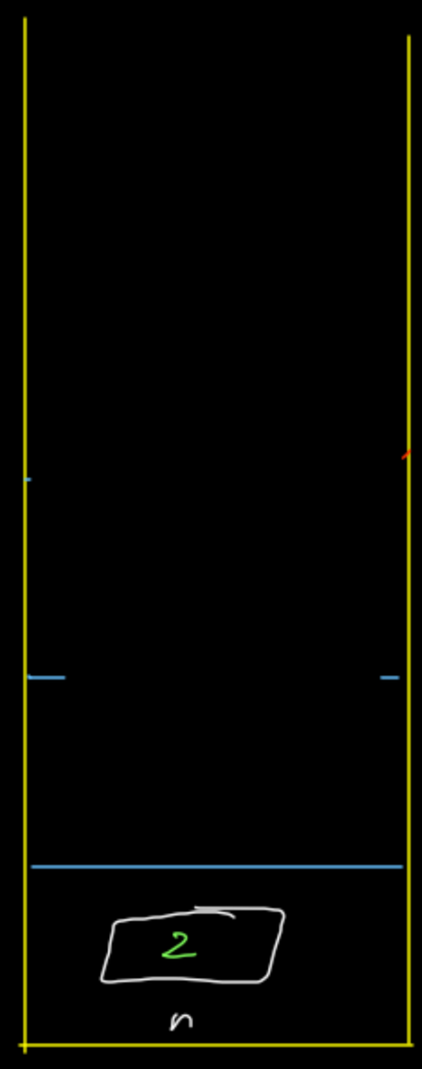
④ pzz(N-1)

⑤ Print(N)

```java
import java.io.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
    }

    public static void pzz(int n){

        System.out.print(n + " "); 1
        pzz(n-1); 2
        System.out.print(n + " "); 3
        pzz(n-1); 4
        System.out.print(n + " ");5
    }

}
```

2
1
1
1
1
2
1

1
1
2



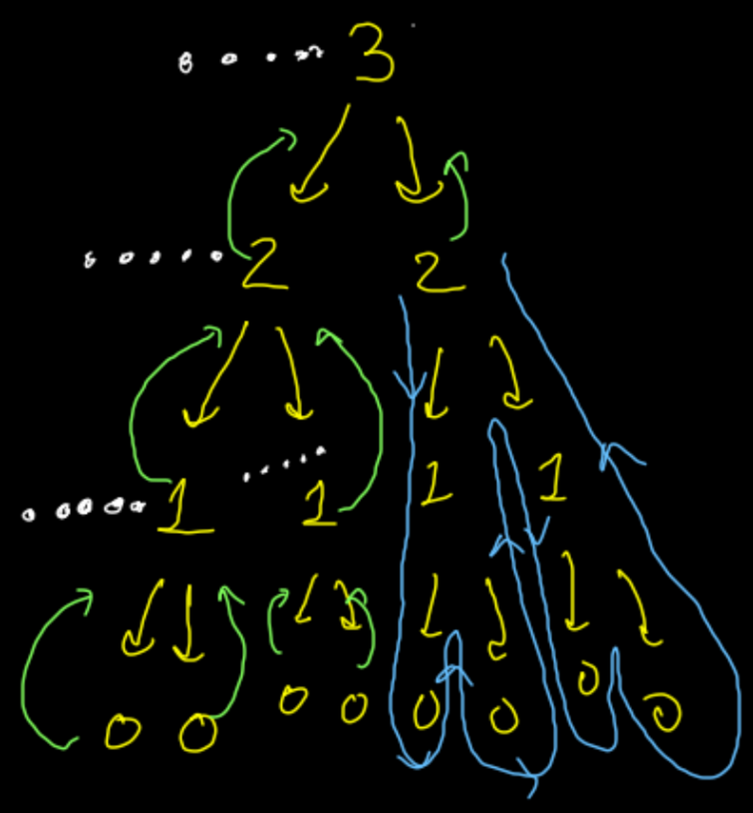
pzz 1, 2(p), 3, 4, 5

& main

Recursion Tree (Euler Tree)

3 2 1 1 1 2 1 1 1 2 3 [2 1 1 1 2 1 1 1 2] 3

```java
import java.io.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
    }

    public static void pzz(int n){

        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
    }

}
```




```java
import java.io.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        pzz(n);
    }

    public static void pzz(int n){

        if(n == 0) return;

        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
    }

}
```

factorial (Recursion Tree)

5! (120)

↓

(24) 4! * 5

↓

(6) 3! * 4

↓

(2) 2! * 3

↓

(1) 1! * 2

↓

0! * 1

(1)

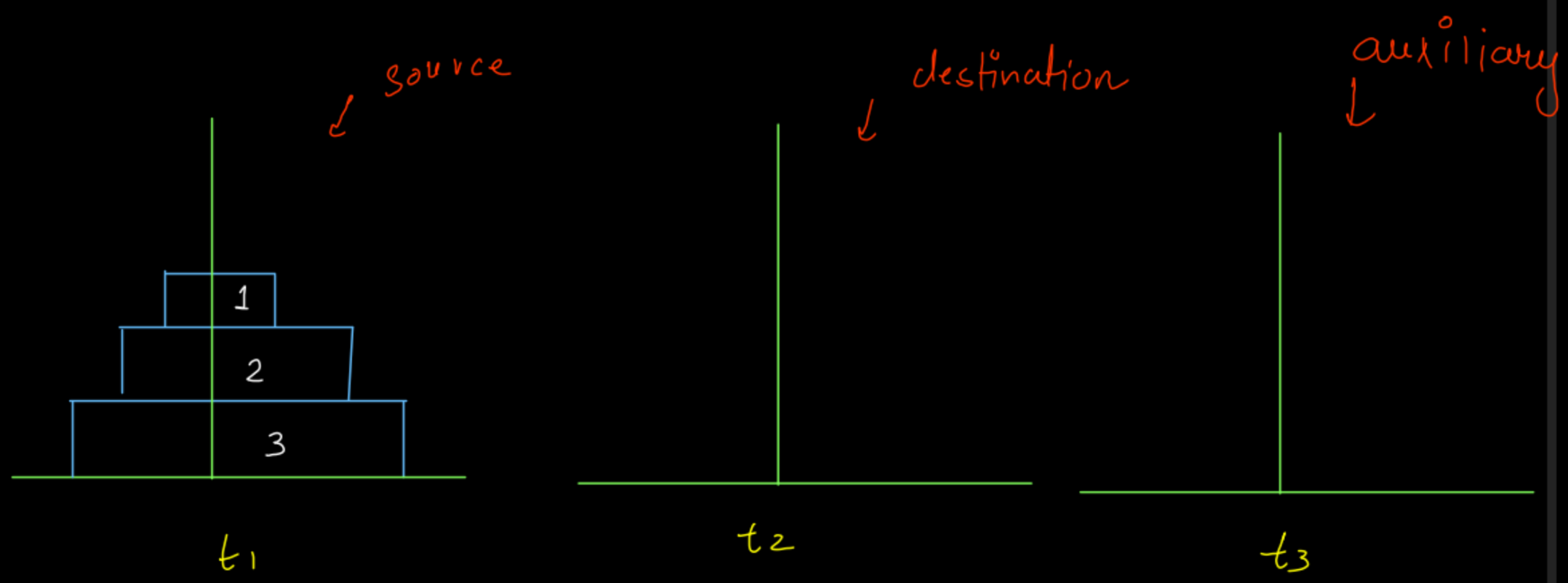
Lecture - 21

Tower of Hanoi Problem

- Understanding the Problem Statement
- High Level Thinking
- Program (With Base Case)
- Low level thinking → Using Call Stack
- Low level thinking → Using Euler Tree
- Conclusion

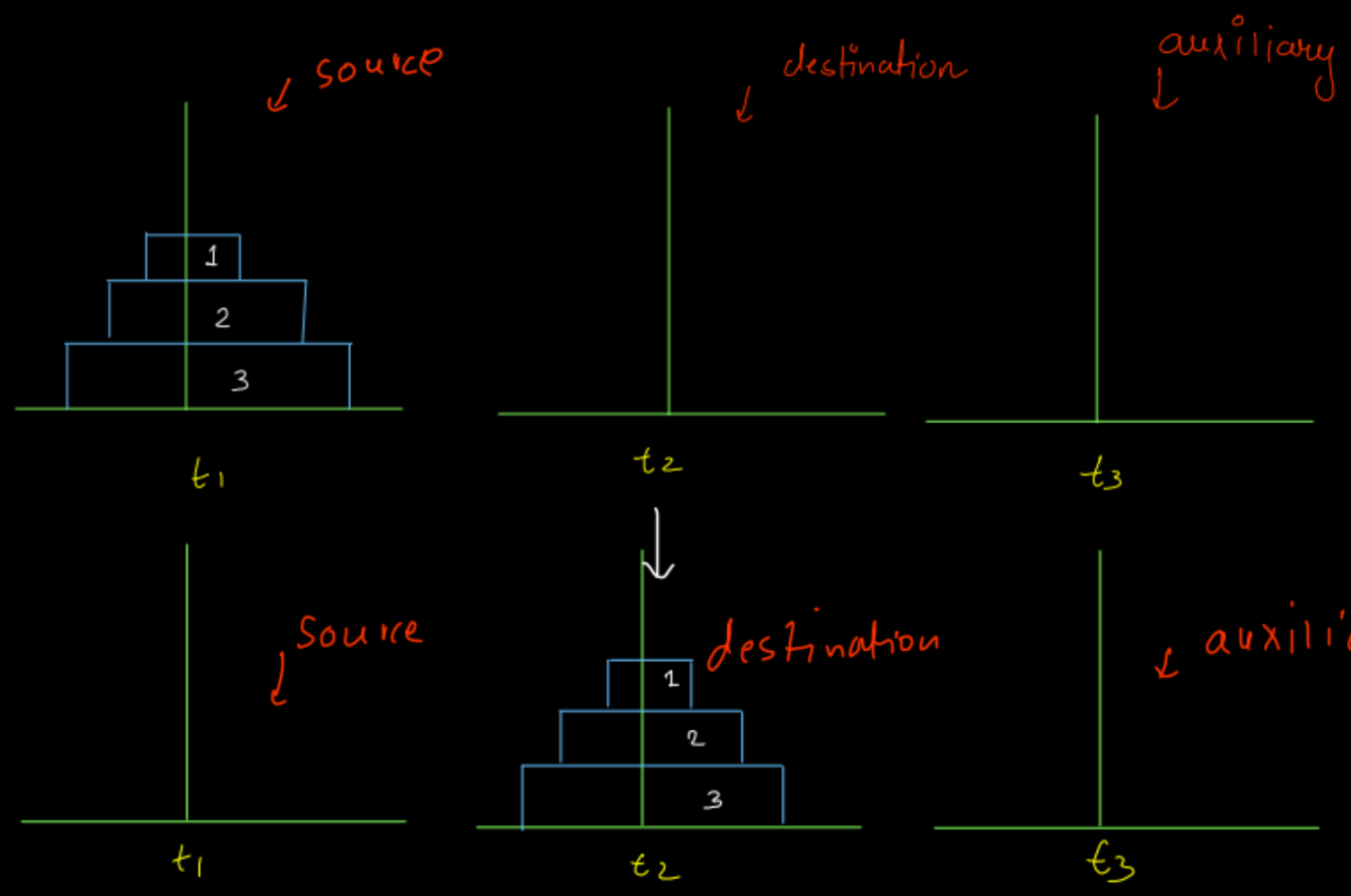
Toh Problem

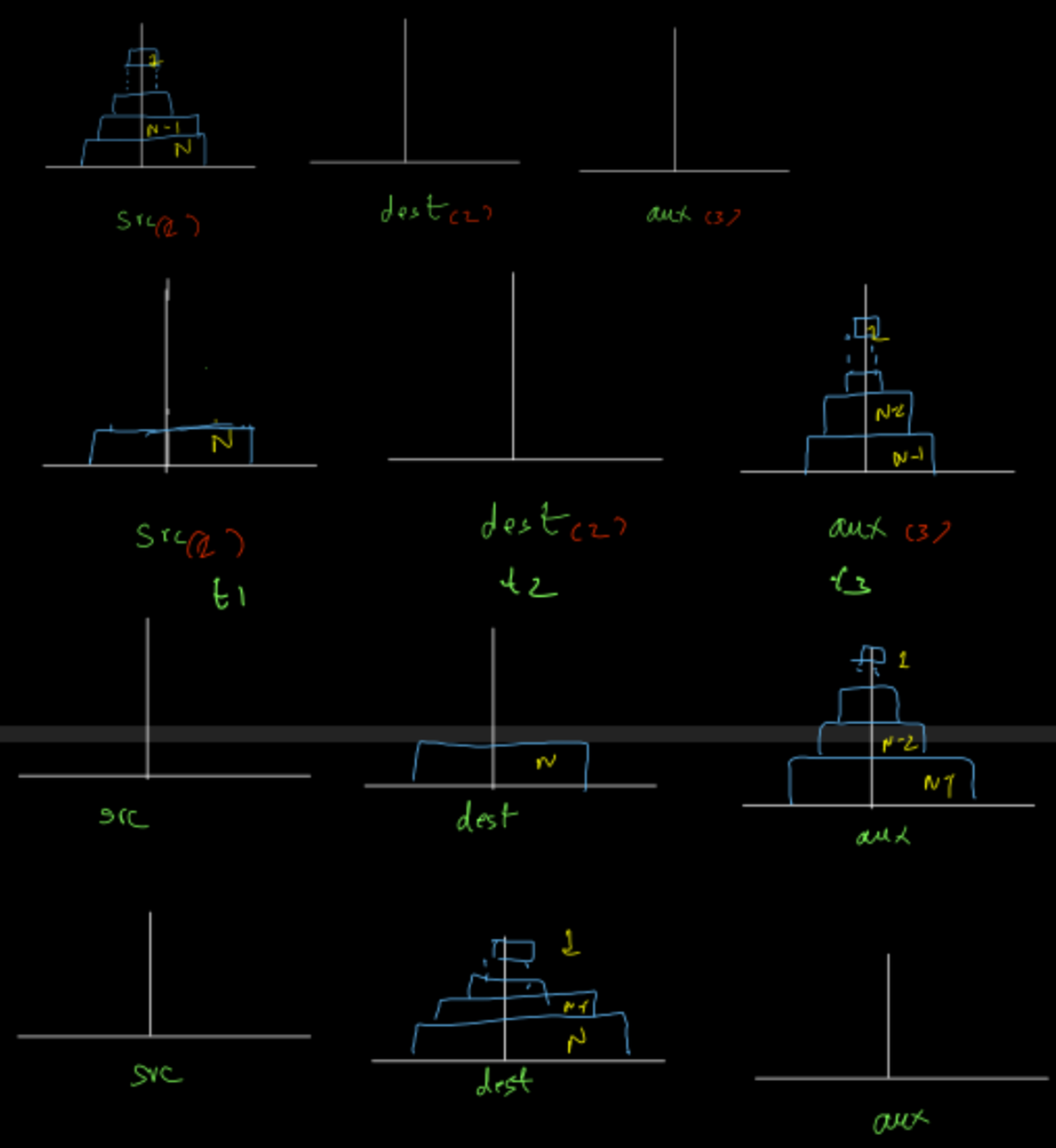
3.3. following the rules
- 3.3.1 move 1 disk at a time.
- 3.3.2 never place a smaller disk under a larger disk.
- 3.3.3 you can only move a disk at the top.



Expectation (for N=3) toh(N, t_1, t_2, t_3)

** following the rules





src dest aux

$toh(N-1, t_1, t_3, t_2)$ {faith 1}

$x \rightarrow$ no of moves

src dest aux

$toh(1, t_1, t_2, t_3)$ 1

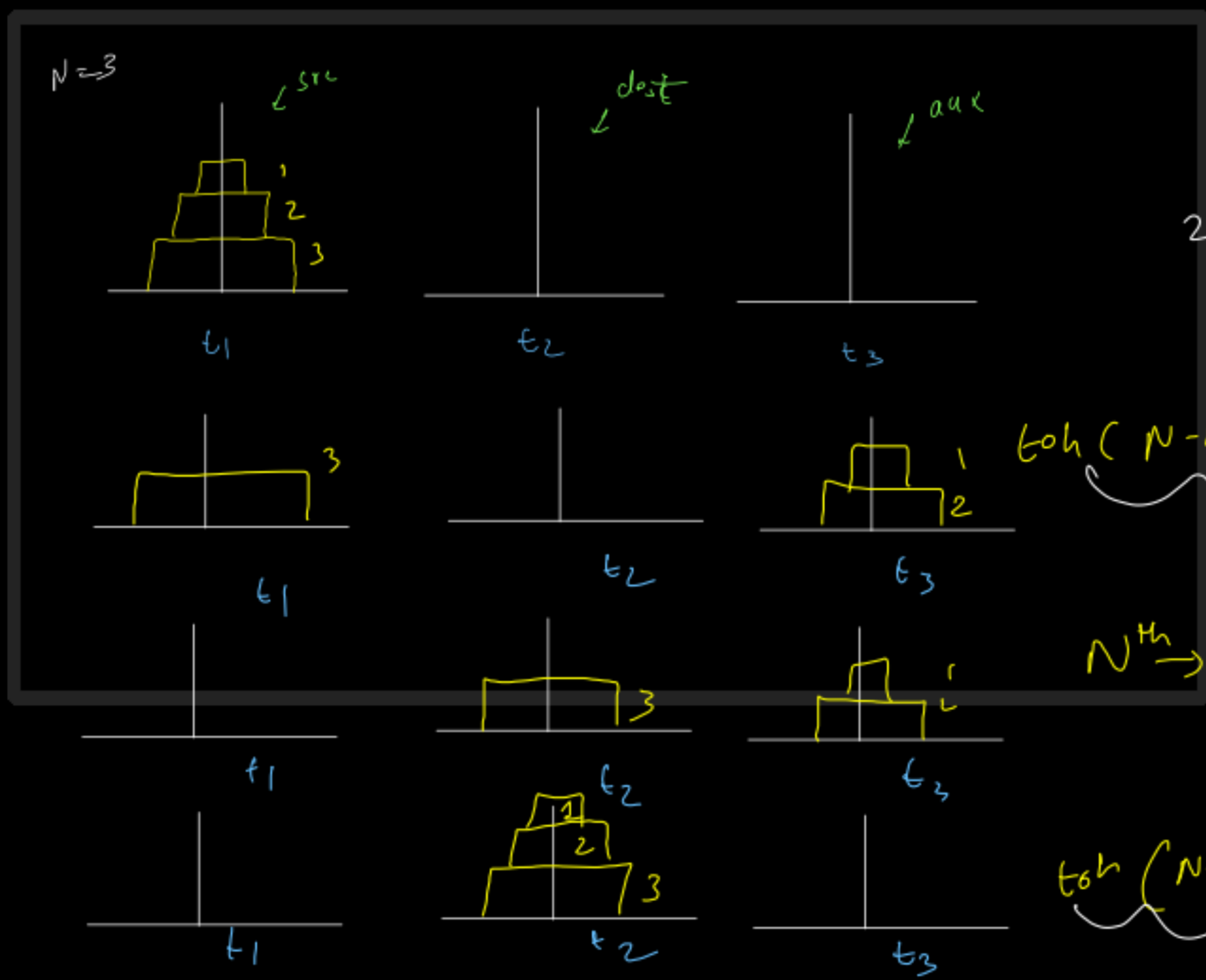
$x + y + 1$

& $x = y$

src dest aux

$toh(N-1, t_3, t_2, t_1)$ {faith 2}

$y \rightarrow$ No of moves.



$2 * x + 1$

src dest aux

$\underbrace{toh(N-1, t_1, t_3, t_2)}_{x}$

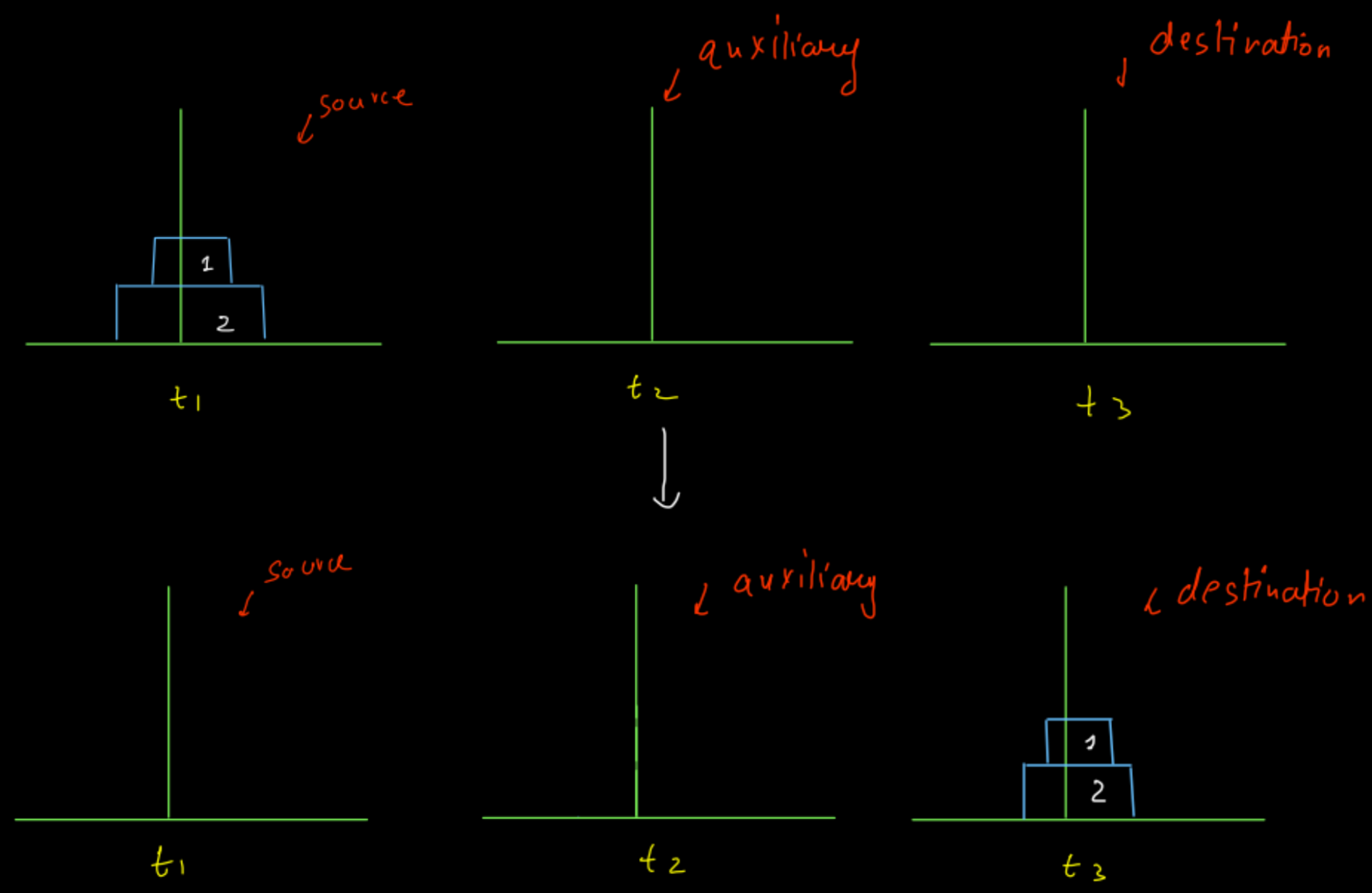
$N^{th} \rightarrow t_1$ to t_2

1

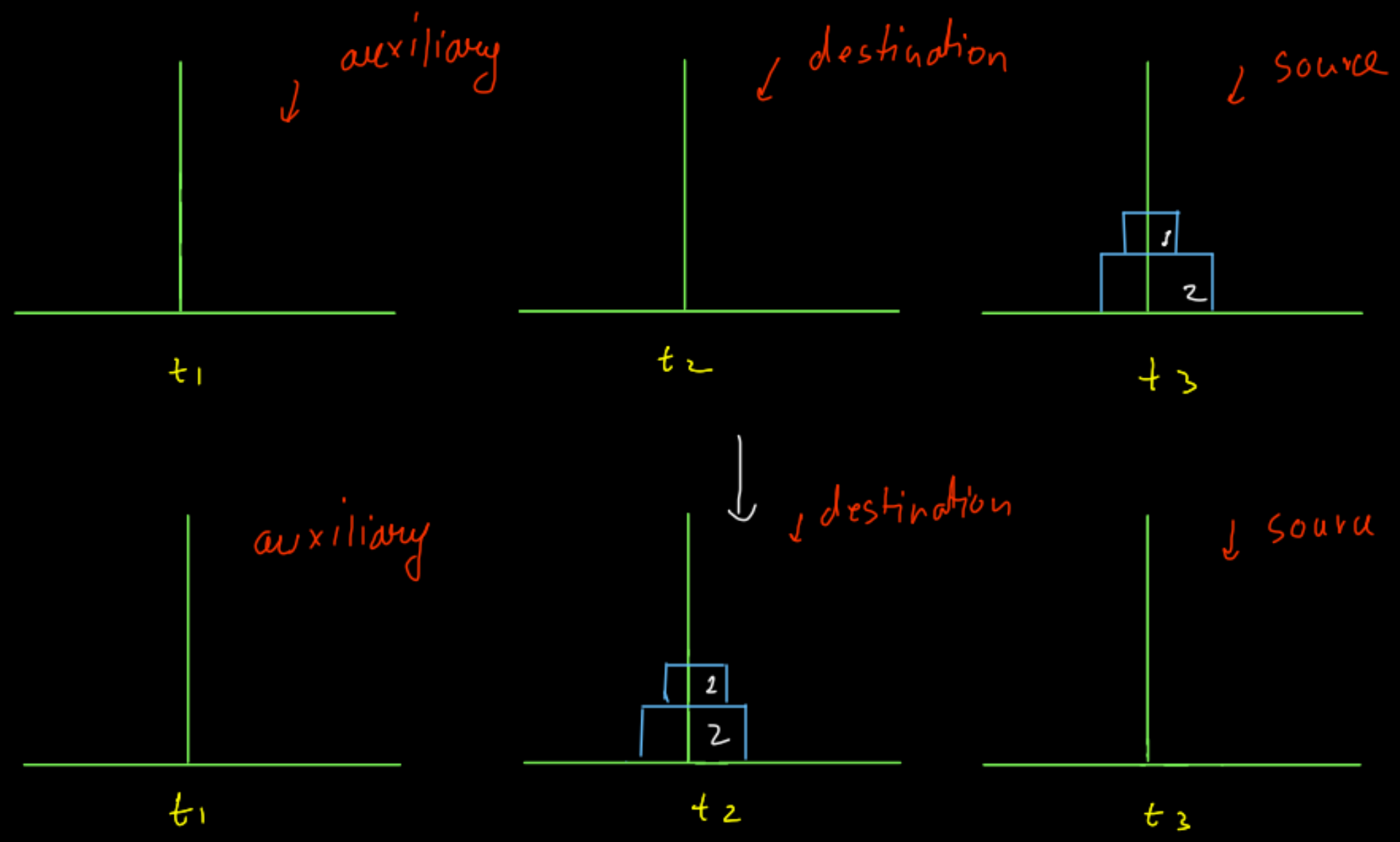
src dest aux

$\underbrace{toh(N-1, t_3, t_2, t_1)}_{x}$

faith1 (N = 2) toh(N-1, t1, t3, t2)

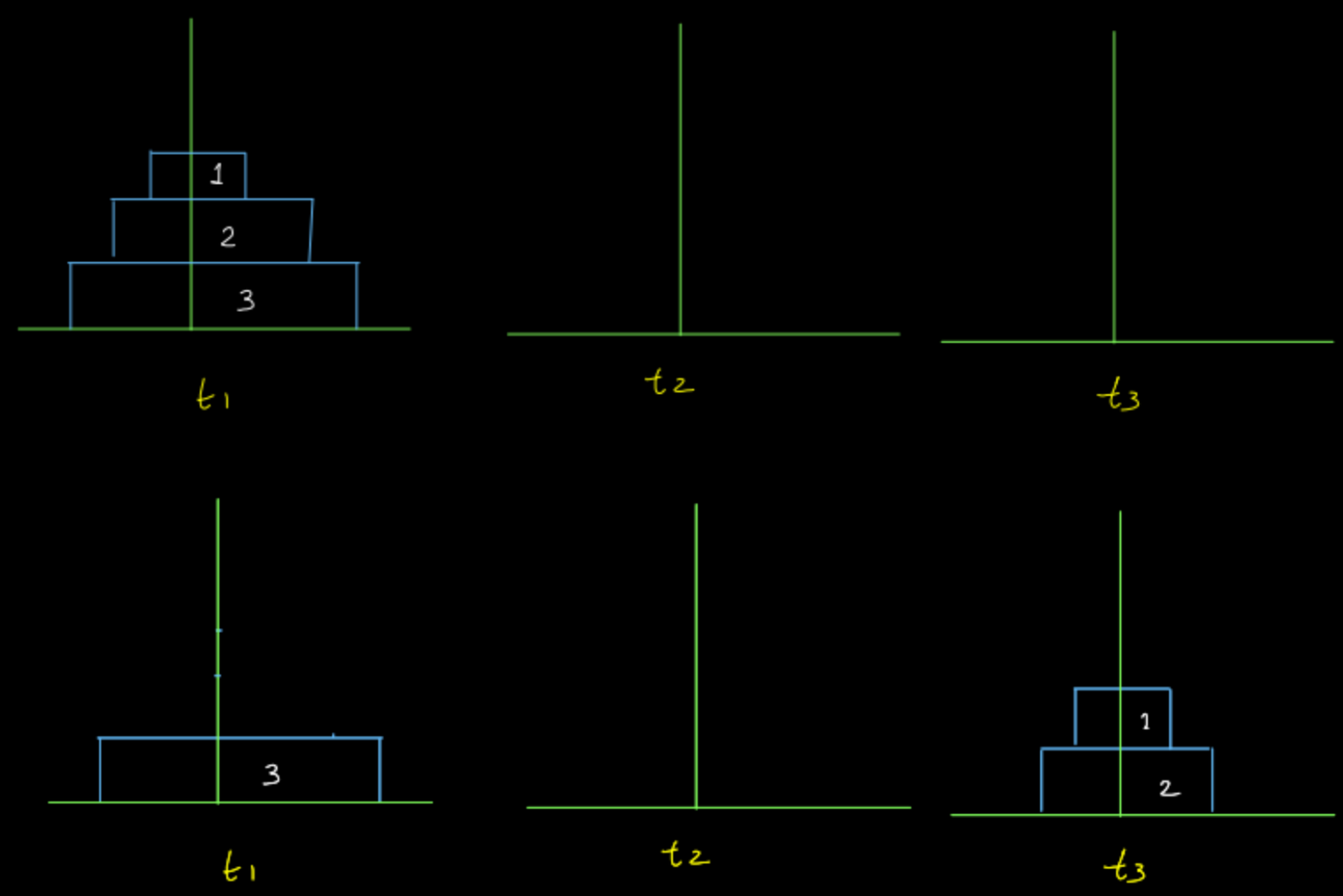


faith 2 (N = 2) toh (N-1, t3, t2, t1)



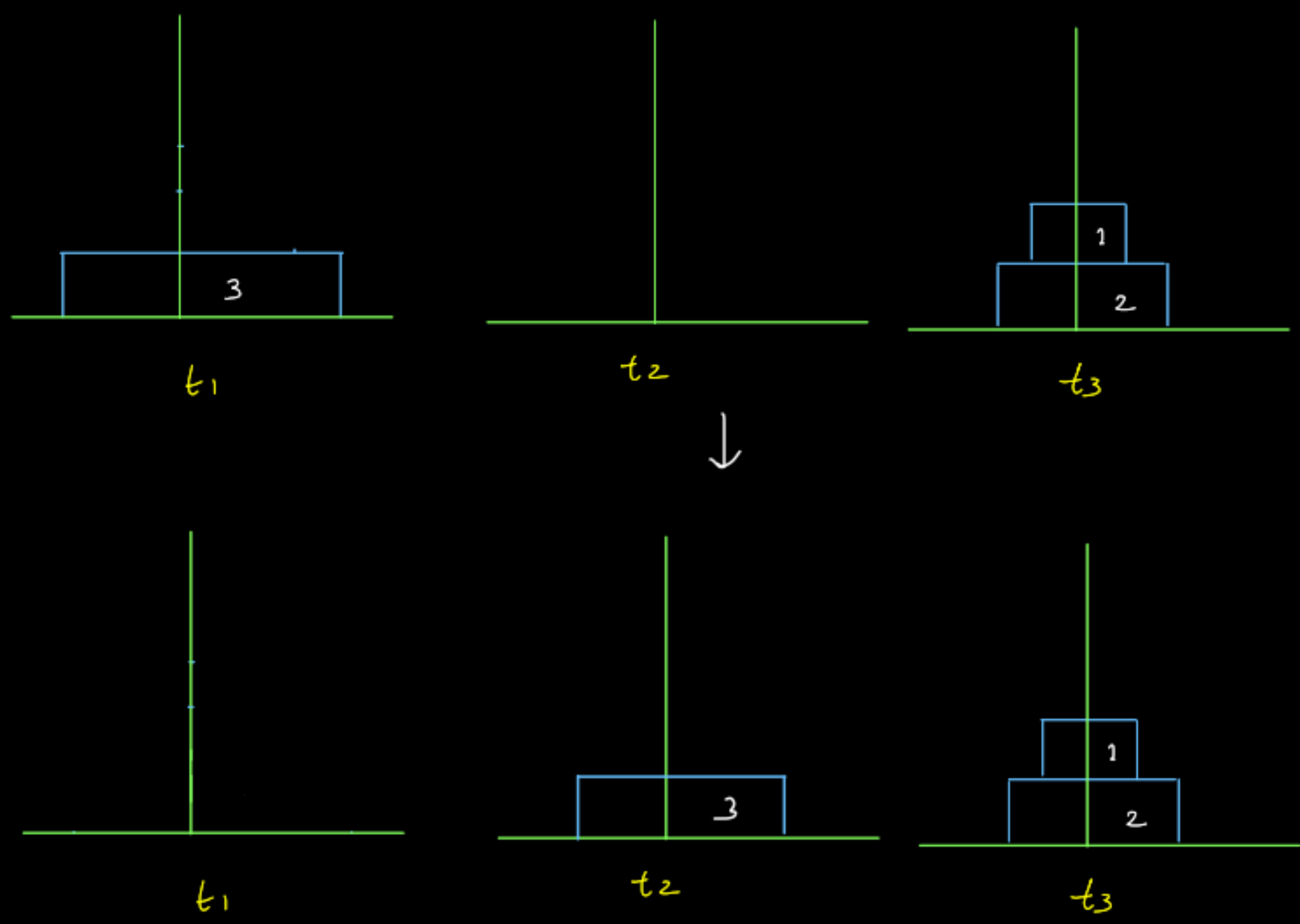
Meeting Expectation with faith

① Move (N-1) disks from t_1 to t_3 using t_2. (faith 1)



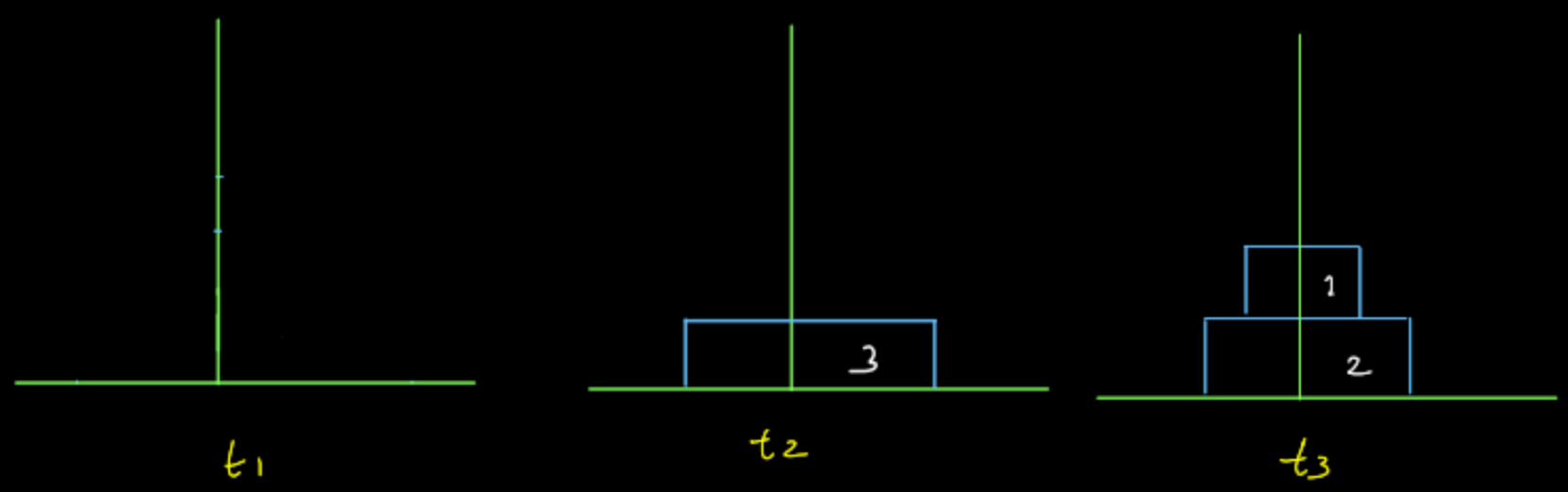
Meeting Expectation with faith

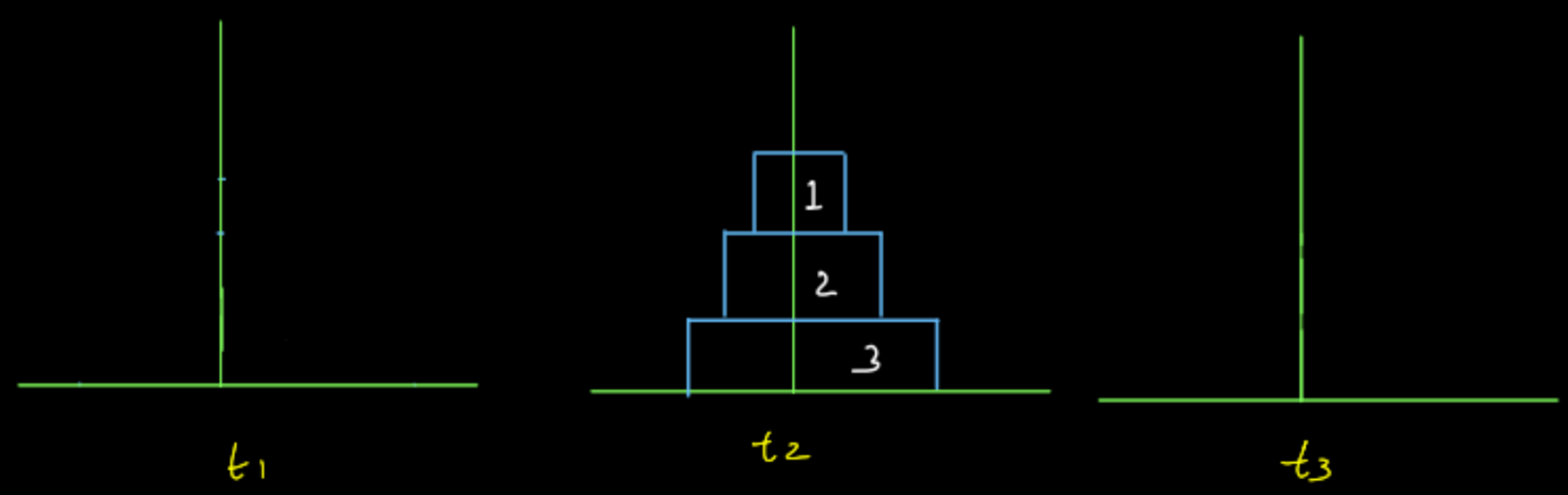
② Move N^{th} disk from t_1 to t_2.



Meeting Expectation with faith

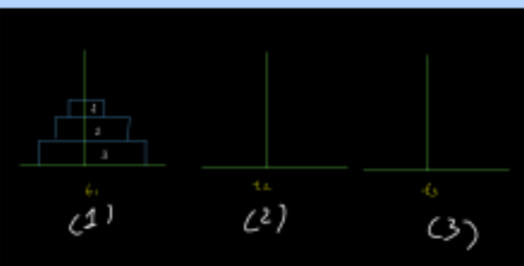
③ Move (N-1) disks from t3 to t2 using t1. (faith 2)





N = 3

```
// t1 -> src t2-> dest t3-> aux
public long toh(int N, int t1, int t2, int t3) {
    // Your code here

    if(N == 0) return 0;

    long ans1 = toh(N-1,t1,t3,t2);   1
    System.out.println("move disk " + N + " from rod " + t1 + " to rod " + t2);   2
    long ans2 = toh(N-1,t3,t2,t1);   3

    return ans1 + ans2 + 1;   4
}
```



Check recording to view call stack dry run

1, 2, 3, 4

1. from 1 to 2
2. from 1 to 3
1. from 2 to 3

ans 1 = 3

ans 2 = 3

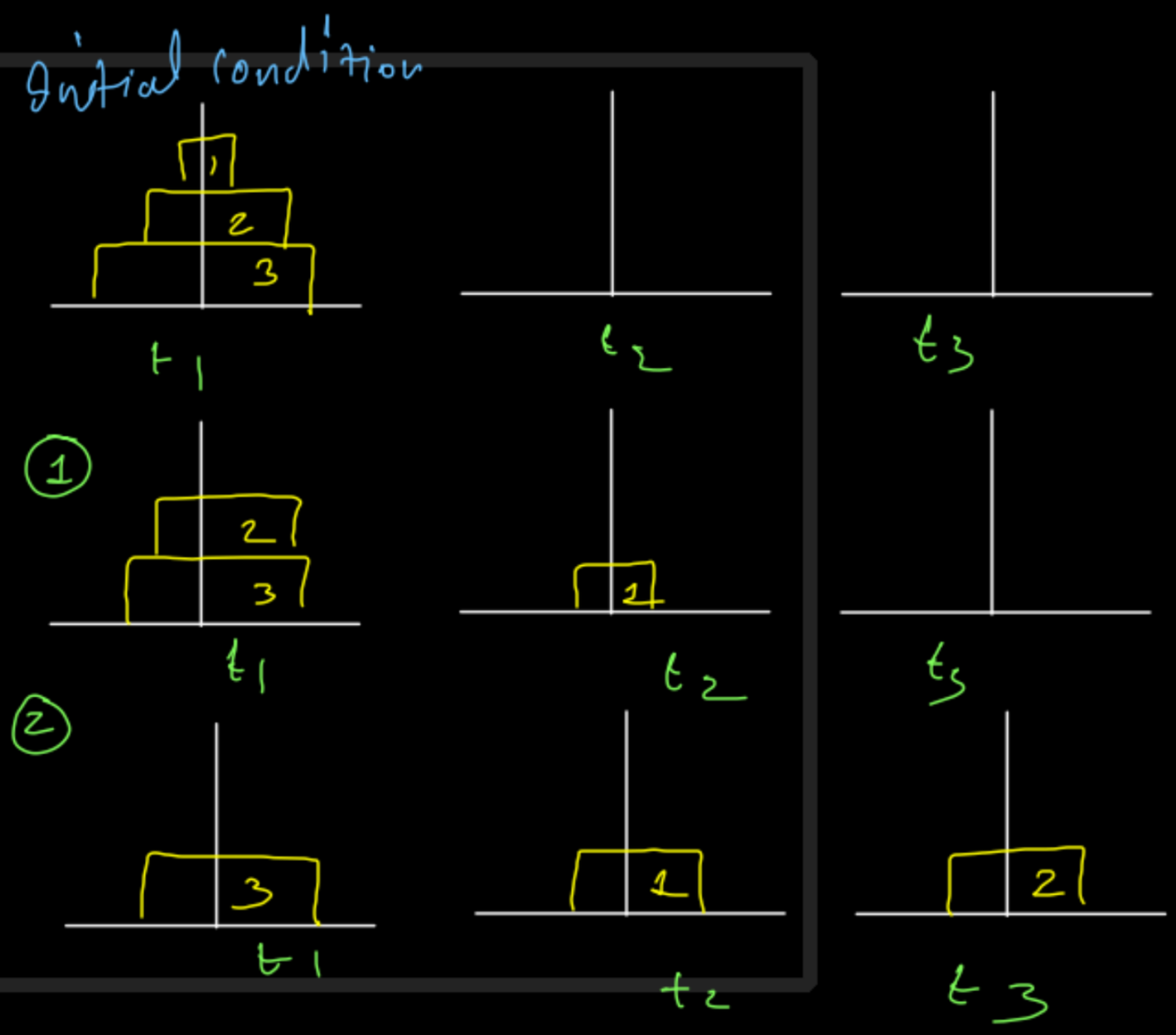
+1 = 7

3. from 1 to 2

1. from 3 to 1
2. from 3 to 2
1. from 1 to 2

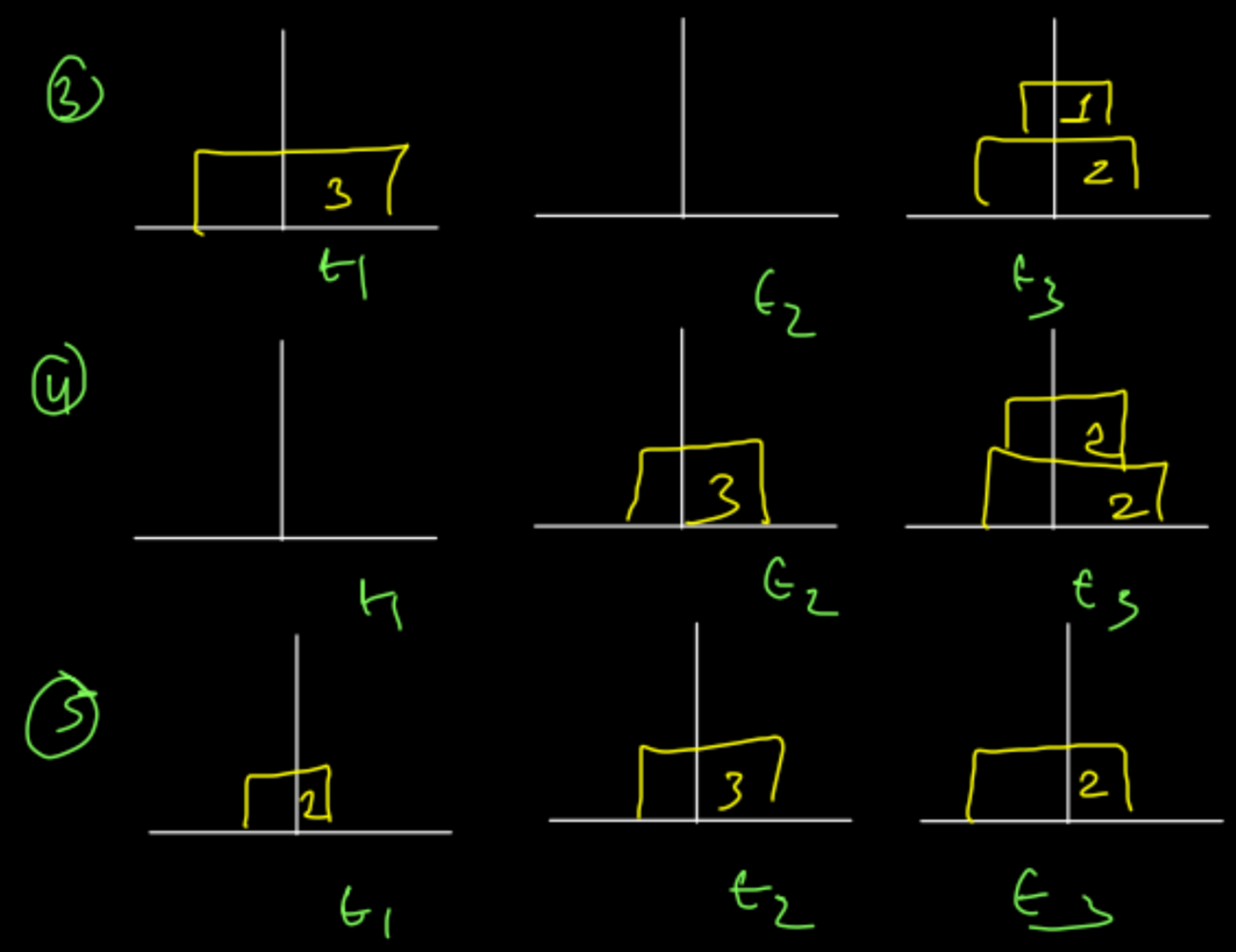
```
// avoid space at the starting of the string in "move disk....."
class Hanoi {

    // t1 -> src t2-> dest t3-> aux
    public long toh(int N, int t1, int t2, int t3) {
        // Your code here

        if(N == 0) return 0;

        long ans1 = toh(N-1,t1,t3,t2);
        System.out.println("move disk " + N + " from rod " + t1 + " to rod " + t2);
        long ans2 = toh(N-1,t3,t2,t1);

        return ans1 + ans2 + 1;
    }
}
```

1. ~~from 1 to 2~~
2. ~~from 1 to 3~~
3. from 2 to 3
4. from 1 to 2
1. from 3 to 1
2. from 3 to 2
3. from 1 to 2

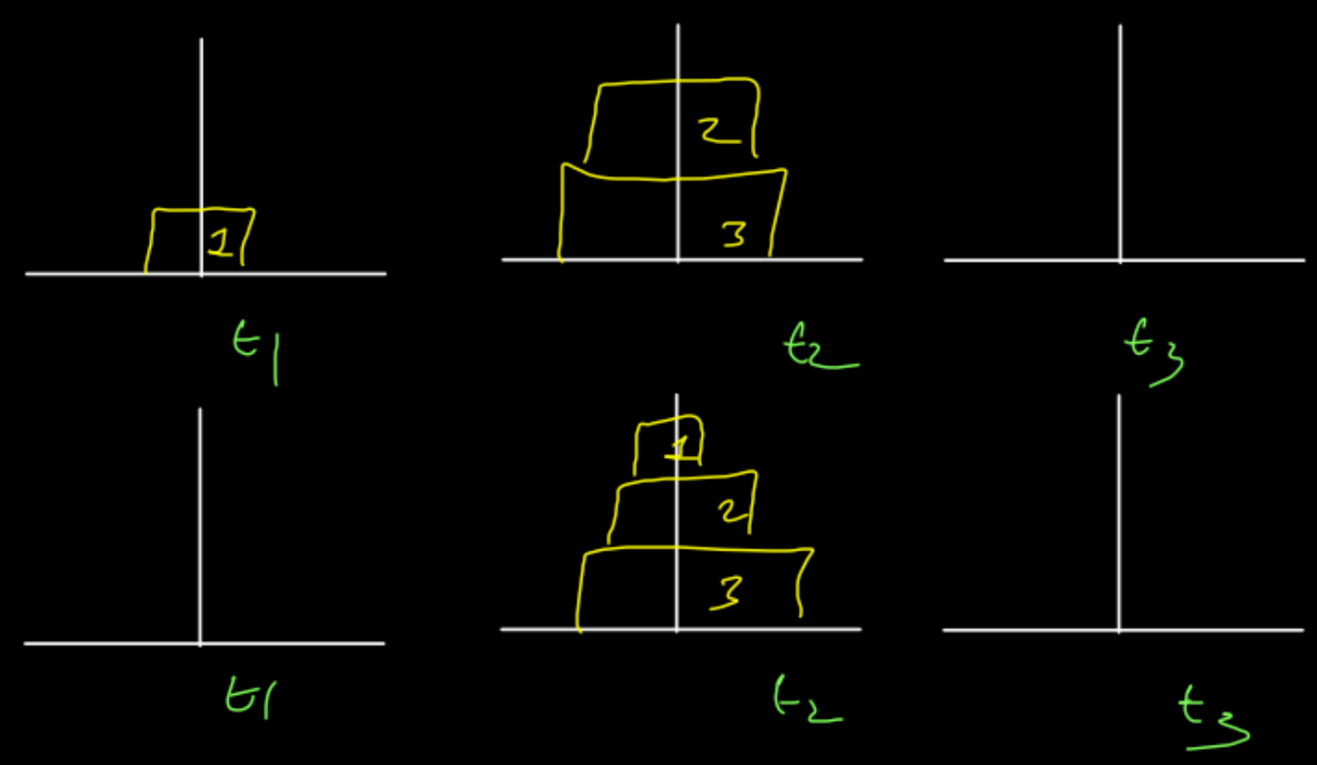


faith(4)

1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2



1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2



$t_1 = 1 \quad t_2 = 2 \quad t_3 = 3$

```java
// t1 -> src t2-> dest t3-> aux
public long toh(int N, int t1, int t2, int t3) {
    // Your code here

    if(N == 0) return 0;

    long ans1 = toh(N-1,t1,t3,t2); 1
    System.out.println("move disk " + N + " from rod " + t1 + " to ro
    long ans2 = toh(N-1,t3,t2,t1); 3

    return ans1 + ans2 + 1; 4
}
```

1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2

src dest aux

$toh(N, t_1, t_2, t_3)$

$\ldots toh(3, t_1, t_2, t_3)$ → 7

3

$\ldots toh(2, t_1, t_3, t_2)$ | $\ldots toh(2, t_3, t_2, t_1)$

1 | 1

$\ldots toh(1, t_1, t_2, t_3)$: $toh(1, t_2, t_3, t_1)$: $toh(1, t_3, t_1, t_2)$: $toh(1, t_1, t_2, t_3)$

0 | 0 | 0 | 0

$toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$

Tree

1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2

Stack

1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2

Lecture - 21

→ Display Array

→ Reverse Array

→ Master formula Introduction (Preorder, Inorder & Postorder)

→ Power **

- → Linear 1
- → Linear 2
- → Logarithmic

Print Elements of Array

School Accuracy: **52.5%** Submissions: **31753** Points: **0**

Given an array **Arr** of size **N**, print all its elements.

Example 1:

Input:
N = 5
Arr[] = {1, 2, 3, 4, 5}
Output: 1 2 3 4 5

① Expectation:

0 1 2 3 4 5 6
{9, 8, 6, 3, 4, 7, 2}

display(arr, 0) → print space seperated elements. (N)
idx

9 8 6 3 4 7 2

② faith : display(arr, 1) → print space seperated elements (N-1)
idx+1

③ Meeting expec with faith :

idx 2 rdx

0 1 2 3 4 5 6
{9, 8, 6, 3, 4, 7, 2}

Print yourself + " "
↳ 9 _

display(arr, idx+1)
↳ 8 6 3 4 7 2

```java
void display(int[] arr, int idx) {

    |

    System.out.print(arr[idx] + " ");   1
    display(arr,idx + 1);               2
}
```

idx idx idx idx idx idx idx idx

0 1 2 3 4 5 6

{ 9, 8, 6, 3, 4, 7, 2 }

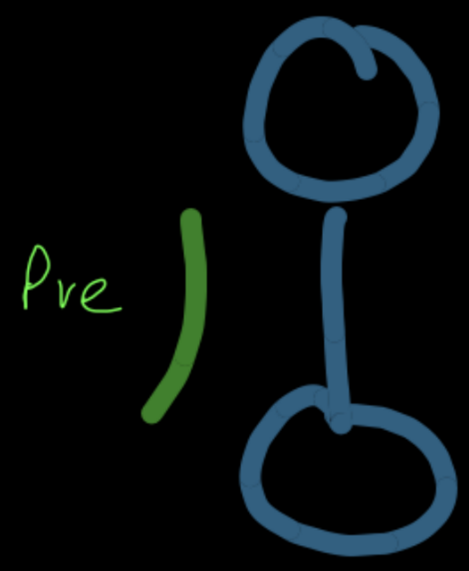
9_ 8_ 6_ 3_ 4_ 7_ 2_

```java
if(idx == arr.length)
    return;
```

→ Base Case

.. display(arr, 0)
↓
.. display(arr, 1)
↓
.. display(arr, 2)
↓
.. display(arr, 3)
↓
.. display(arr, 4)
↓
.. display(arr, 5)
↓
.. display(arr, 6)
↓
display(arr, 7)

```java
public static void displayArr(int[] arr, int idx){
    if(idx == arr.length) return;

    System.out.println(arr[idx]);
    displayArr(arr,idx + 1);
}
```



Reverse an Array

School Accuracy: **40.0%** Submissions: **100k+** Points: **0**

Given an array **A** of size **N**, print the reverse of it.

Example:

Input:
1
4
1 2 3 4
Output:
4 3 2 1

Expectation :

lo hi

0 1 2 3 4

{ 1, 2, 3, 4, 5 }

↓

lo hi

0 1 2 3 4

{ 5, 4, 3, 2, 1 }

reverse (arr, lo, hi);

lo → 0, hi → arr.length - 1

faith :

lo hi

0 1 2 3 4

{ 1, 2, 3, 4, 5 }

↓

lo hi

0 1 2 3 4

{ 1, 4, 3, 2, 5 }

reverse(arr, lo+1, hi-1);

Meeting expectation with faith :

1. Swap lo & hi
2. reverse (arr, lo+1, hi-1)

lo hi

0 1 2 3 4

$\{ \cancel{5}, 2, 3, 4, \cancel{8} \}$

s l

① 0 1 2 3 4

$\{ 5, 2, 3, 4, 1 \}$

lo hi

② 0 1 2 3 4

$\{ 5, 2, 3, 4, 1 \} \longrightarrow$ 0 1 2 3 4 $\{ 5, 4, 3, 2, 1 \}$

```java
public static void reverse(int[] arr, int lo, int hi) {

    //swap yourself
    int temp = arr[lo];
    arr[lo] = arr[hi];
    arr[hi] = temp;

    reverse(arr,lo+1,hi-1);
}
```

① ②

lo lo lo hi hi hi

0 1 2 3 4

{ 1, 2, 3, 4, 5 }

5 4 2 1

..reverse (arr, 0, 4)

↓

..reverse (arr, 1, 3)

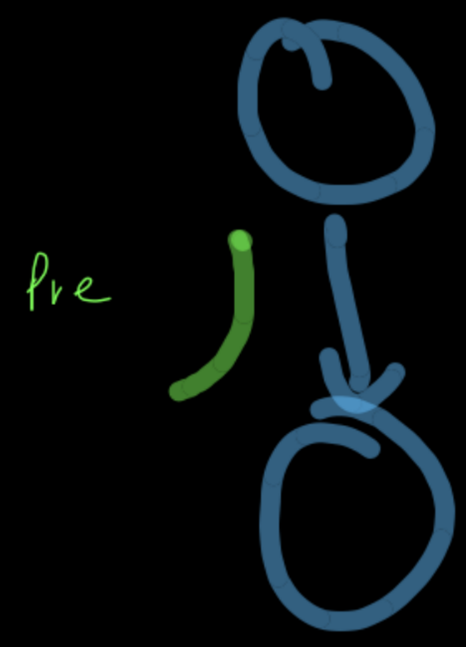
↓

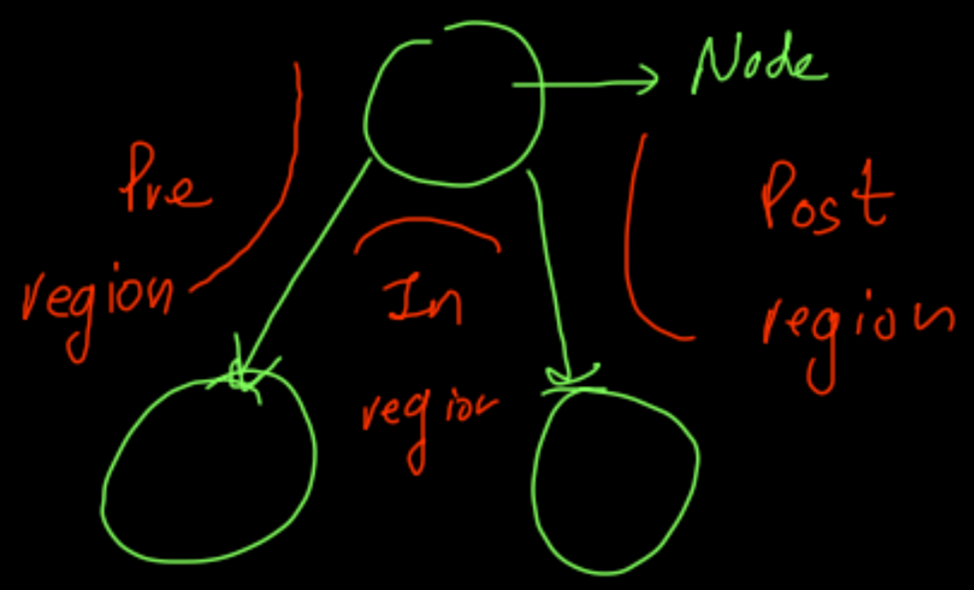
reverse (arr, 2, 2)

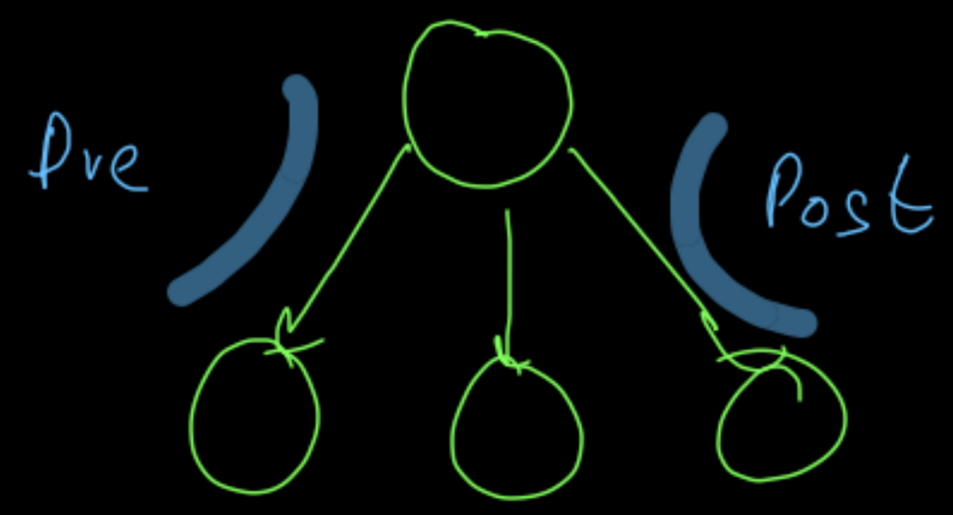
if (lo >= hi) return;

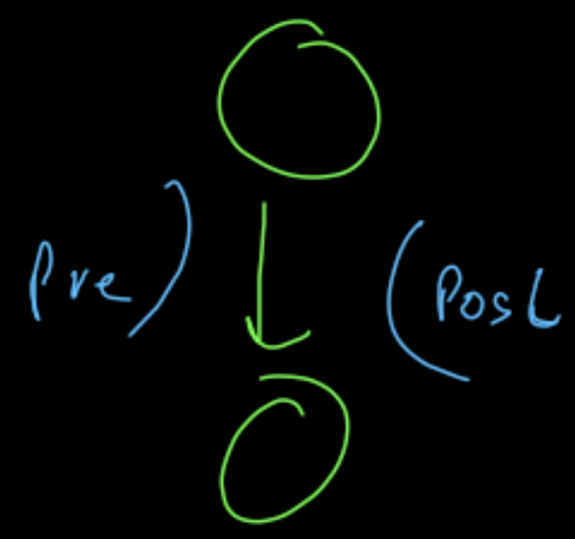
Base case

```java
public static void reverse(int[] arr, int lo, int hi) {

    if(lo >= hi) return;

    //swap yourself
    int temp = arr[lo];
    arr[lo] = arr[hi];
    arr[hi] = temp;

    reverse(arr,lo+1,hi-1);
}
```




Node
Pre
region
In
region
Post
region


Pre
Post


Pre
Post

Recursion Master formula

$$\text{Time Complexity} = (\text{calls})^{\text{height}} + (\text{Pre} + \text{Post}) * \text{height}$$

This is max no of calls from a node in the tree.

50. Pow(x, n)

Medium 5801 6346 Add to List Share

Implement pow(x, n), which calculates `x` raised to the power `n` (i.e., x^n).

Example 1:

```
Input: x = 2.00000, n = 10
Output: 1024.00000
```

① Expectation : power(x, n) → $\underbrace{x * x * x \cdots x}_{n \text{ times}}$

power(2, 7) → $2 * 2 * 2 * 2 * 2 * 2 * 2$

② faith : power(x, n-1) → $\underbrace{x * x * x \cdots x}_{(n-1) \text{ times}}$

power(2, 6) → $2 * 2 * 2 * 2 * 2 * 2$

③ Meeting expectation with faith : power(x, n) = power(x, n-1) * x

$$x^n = x * x^{n-1}$$

x^N

$\downarrow$

x^{N-1}

$\downarrow$

x^{N-2}

$\vdots$

$0 \longrightarrow$ if (n == 0) return 1

$(num)^0 \rightarrow 1$

$$x^{-N} = \frac{1}{x^N}$$

$$2^{-6} = \frac{1}{2^6}$$

$$2^{-6} = \frac{1}{2^{-(-6)}}$$

```java
class Solution {

    public double power(double x, int n) {
        if(n == 0) return 1;

        return x * power(x,n-1);

    }

    public double myPow(double x, int n) {
        if(n < 0) {
            return 1.0 / power(x,-n);
        }

        return power(x,n);
    }
}
```

This gives stack overflow

```java
class Solution {

    public double power(double x, int n) {
        if(n == 0) return 1;

        return x * power(x,n-1);

    }

    public double myPow(double x, int n) {
        if(n < 0) {
            return 1.0 / power(x,-n);
        }

        return power(x,n);
    }
}
```

$(\text{calls})^{h} + (\text{pre} + \text{post}) *$

$(1)^{N} + (k1 + k2) * N = 1 + k^{N} \approx O(N)$

pow(2,5) ↑ 2*16 = 32

↓

pow(2,4) ↑ 2*8 = 16

↓

pow(2,3) ↑ 2*4 = 8

↓

pow(2,2) ↑ 2*2 = 4

↓

pow(2,1) ↑ 2*1 = 2

↓

pow(2,0) ↑ 1

power(2,6) → power(2,3) * power(2,3)

power(2,7) → 2* pow(2,3) * pow(2,3)

x^N

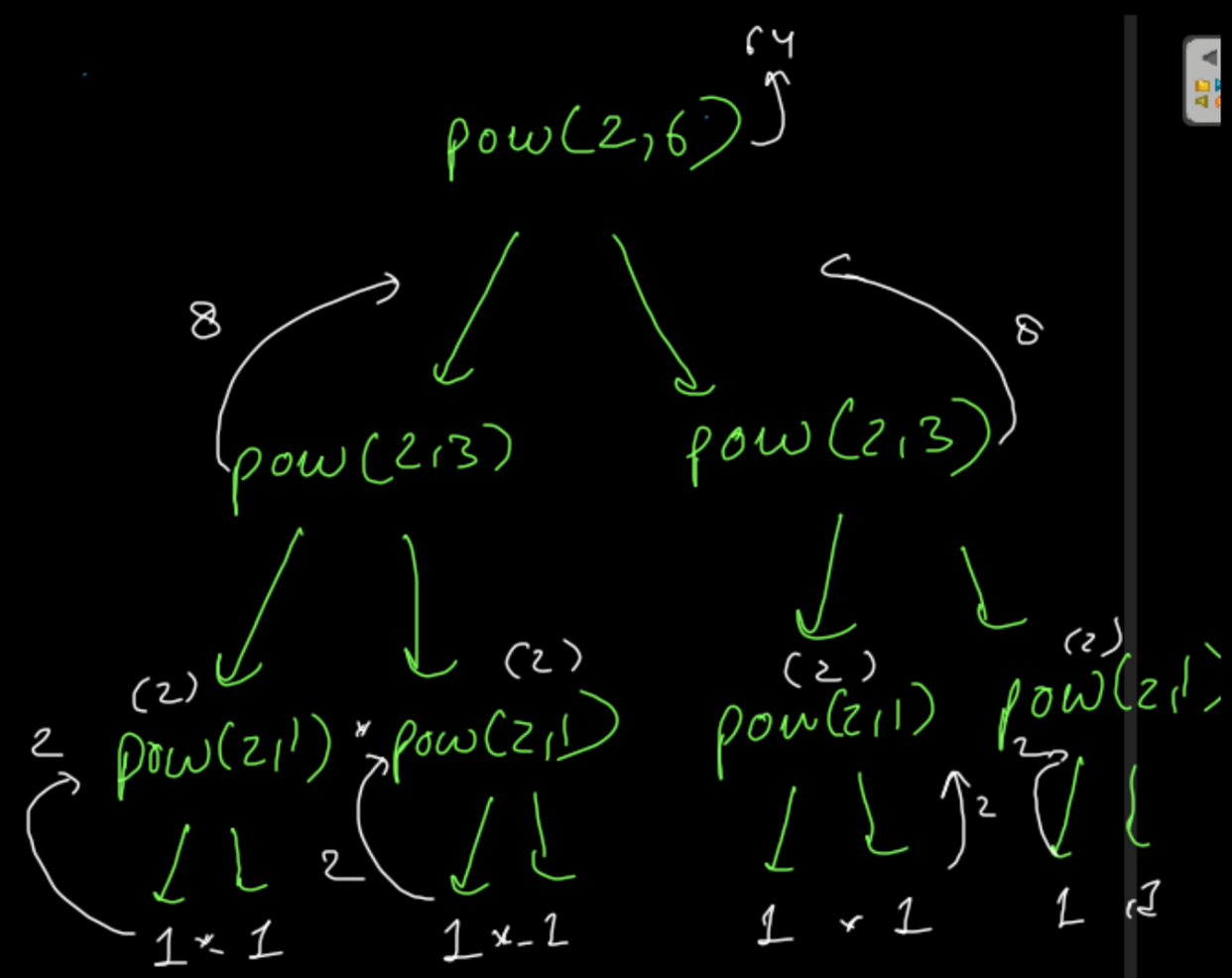
if (N % 2 == 0)

pow(x,n/2) * pow(x,n/2)

else

x * pow(x,n/2) * pow(x,n/2)

```java
class Solution {

    public double power(double x, int n) {
        if(n == 0) return 1;

        if(n % 2 == 0)
            return power(x,n/2) * power(x,n/2);

        return x * power(x,n/2) * power(x,n/2);

    }

    public double myPow(double x, int n) {
        if(n < 0) {
            return 1.0 / power(x,-n);
        }

        return power(x,n);
    }
}
```



(calls)^h + (pre+post) * h

$(2)^h$ + (k) * h =

$(2)^{\log_2 N}$ + (k) * $\log_2 N$

N + $\log_2 N$ = O(N)

pow(8)

↓

4

↓

2

↓

1

4

$\log_2 8 + 1$

$\approx O(\log_2 N)$

```
public double power(double x, int n) {
    if(n == 0) return 1;

    double smallAns = power(x,n/2);

    if(n % 2 == 0)
        return smallAns * smallAns;

    return x * smallAns * smallAns;

}

public double myPow(double x, int n) {
    if(n < 0) {
        return 1.0 / power(x,-n);
    }

    return power(x,n);
}
```

pow(2,6)

↓

pow(2,3)

↓

pow(2,1)

↓

a

$(\text{calls})^h + (\text{pre} + \text{post}) * h$

$(1)^{\log_2 N} + K * \log_2 N$

$1 + \log_2 N \approx O(\log_2 N)$