

DAY
49

Recursion with ArrayList

1. Date → 25 October 2021.

2. Day → Monday.

3. Agenda →

1. Get Subsequence = CW
 2. Get KPC = CW
 3. Get stairs Path = CW
 4. Get Maze Path
 5. First Index
 6. Last Index
 7. All Indices of array
- Done in day 46 HW

Ques. 1 Get Subsequence

→ हमें एक string मिलता है जो इनput है, हमें उसे subsequence के subsequences return करने हैं जो output in a ArrayList है।

→ यहाँ परीक्षा question strings को Module में लिया गया है।
लिंकिन iteratively करेंगे हैं।

→ हमें अब यह question Recursively करना है और solve.
तो पहले जानते हैं What is Subsequence?

A subsequence is a part derived from a given string which is obtained by deleting some or no elements of the given string without changing the order of the remaining elements.

For a String of n length,
we have 2^n subsequences

eg → Steinig = abc.

Subsequences of Steing \Rightarrow

Step 1) → Take a blank steering

अब इसे भी लो (इस Steing को) और एक इसकी copy लगाओ तिथि c को उसकी correction position पर सरकार दो

Step 2)

→ Put last character
at correct position
same
copied
& put at its pos.

इस लो और Copy बनाओ तक (इन दोनों strings की) उनके code में b की उसकी correct position पर रखो

Step 3)

100

- Copy it
- Put last second character at correct position

b

~~→ copied same as~~

Take these

strong Copy all these

Copy all these & spelling and

position

position

Copy all these & spelling and

position

Step 4

— — —
— — c
— b —
— b c.

a — —
a — c
a b —
a b c.

Copied as Previous
Strings and
have put
a (last 3rd) character
at its correct position

These are the Subsequences
of abc, because

character in abc = 3

Subsequences of abc = $(2)^n = (2)^3 = 8$

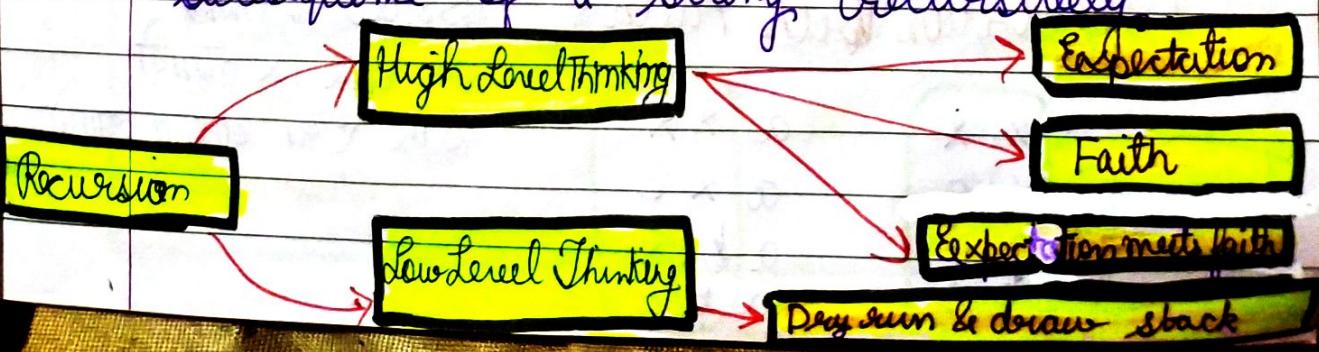
& these
are total

8 Subsequences.

Hence, we are done
with finding all
the subsequences of
abc

* अब हमने ये समझा कि एक Subsequence का एक ऐसी

Now we need its code to find all the
Subsequence of a String, recursively



1. High Level Thinking

a) Expectation

$$gss(abc) = \boxed{x \ x \ x}$$

get subsequence.

$$x \ x \ c.$$

$$x \ b \ x$$

$$x \ b \ c.$$

$$a \ x \ x.$$

$$a \ x \ c.$$

$$a \ b \ x$$

$$a \ b \ c.$$

Expectation

of gss(abc)

function

of 8 subsequences

देता है

$x \ x \ x.$	$a \ x \ x$
$x \ x \ c.$	$a \ x \ c.$
$x \ b \ x$	$a \ b \ x$
$x \ b \ c.$	$a \ b \ c.$

b) Faith → faith, expectation का दोनों part

Same.

$$\begin{matrix} x & x \\ x & c \\ b & x \\ b & c \end{matrix}$$

होता है

$$gss(bc) \rightarrow$$

) faith की पैरे output हो
अपने आप ही मिलेगा
अगर हमने bc pass
किया तो.

c) Expectation meets Faith → gss(bc) के सामने

$$\begin{matrix} x & x & x \\ x & x & c \\ x & b & x \\ x & b & c \end{matrix}$$

$$\begin{matrix} a & x & x \\ a & x & c \\ a & b & x \\ a & b & c \end{matrix}$$

एक बार x लगाए
उसके बार a लगाए

High Level Thinking about E; so now, we can write the basic code.

Get Subsequence <code>

Date			
------	--	--	--

```
→ import java.util.*;  
→ import java.io.*;  
→ public class Main {  
    → public static void main (String [] args)  
        {  
            → Scanner scan = new Scanner (System.in);  
            → String str = scan.nextLine();  
            → ArrayList <String> Subsequence = gss(str);  
            → System.out.println (Subsequence);  
        }  
    → }  
    → getsubsequence
```

Expectation // gss(abc) = [., .c, .b, .bc] + [a., a.c, ab, abc]

Truth // g(abc) = [., .c, b, bc]

Expectation meets truth // = . + gss(bc), + , a + gss(bc).

```
→ public static ArrayList <String> gss (String str)
```

This is
the
base
case
& it
is
explained
in Low
level
Thinking

```
{ if (str.length () == 0) → get subsequence  
    { ArrayList <String> bres = new ArrayList <> ();  
        → base result  
        String theSubsequence = "";  
        → Blank str is its subsequence  
        bres.add (theSubsequence);  
        return bres;  
    }
```

```
    → first character of given string  
    char ch = str.charAt(0);  
    → String rstr = str.substring(1); → recursion ↑  
    → ArrayList <String> rres = gss(rstr); → result  
    → ArrayList <String> mres = new ArrayList <> ();  
    → for (String rstr: rres)  
        { mres.add ("_" + rstr); } → my result  
    → for (String rstr: rres)  
        { mres.add (" " + ch + rstr); } → ↑ for each  
    → return mres; → loop till E.  
    → for each element  
        of rres (recursion  
        ↑ result)
```

2. low level thinking

→ Drysum & drawstack .

→ starting at
0th character
fixed/215

Page No. _____
Date _____

Line 1 → char ch = str.charAt(0); // abc → a
Line 2 → String res = str.substring(1); // abc → bc
Line 3 → ArrayList<String> res = gss(res); // gss(bc)
Line 4 → ArrayList<String> more = new ArrayList<>();
rest of the strings

→ for (string str : res) (Other character
at edit bar)

Line 5 { moves add ($x^y + \text{rest}$) ; }

3

~~// metre = [---, --c, -b-, -bc]~~

→ for (String state : rows)

Linef → f. mres · add (ch + rstr);

$$\text{II matrix} = [-c, -b, -bc]$$

$$\text{Line 7} \quad f_{a,b,c} = + [a-- , a-c , ab- , abc]$$

Return mstr;

→ This is the Base Case.

$\rightarrow x^0 = 1$ (पहली Multiplication
में Identity 1)

$\rightarrow x^2 = x * x$ (ईस्ट ऐसी)
(Sum of identity
0 एवं 1)

$\rightarrow x^1 = x$

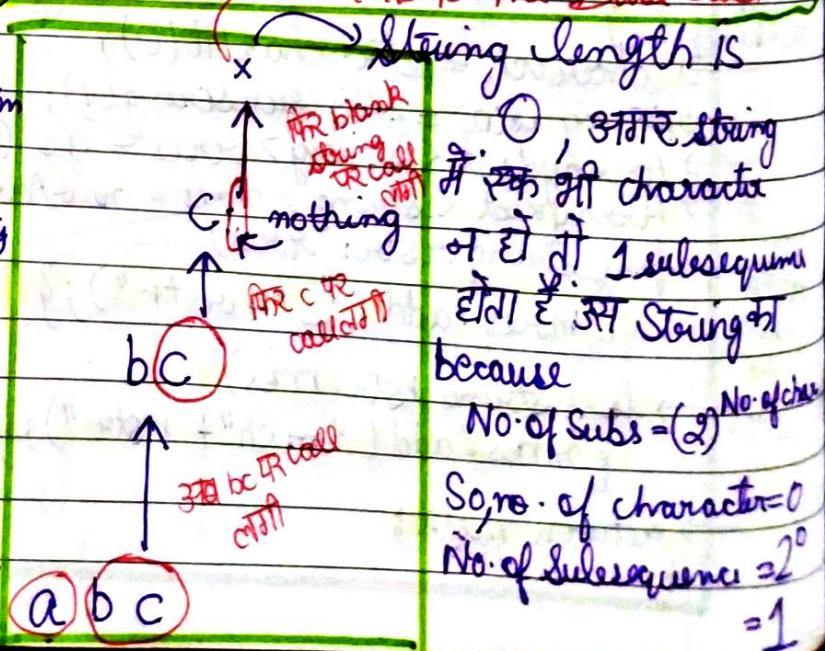
→ In how many ways can we select 0 apple out of 5 apples?

\hookrightarrow apple combi
selected in 1 way
i.e don't select any apple

He won't select any apple

apple
1 2 3 right side
Call

a b c



If old String $x = [""]$, एक blank string with 0 characters
 String $\in x$, तो उसके Subsequence.
 हमें 1

Page No.	
Date	

$$\begin{aligned} \text{no. of character} &= 0 \\ \text{no. of subsequence} &= (2)^{\text{no. of character}} \\ &= (2)^0 \\ &= 1 \end{aligned}$$

Time Complexity of get Subsequence
 $O(n) \rightarrow$ Linear bcz
 a recursion call is made along with using for loop.
 Space Complexity of get Subsequence $\rightarrow O(1) \rightarrow$ No extra space required.: constant.

अब क्या हमें तो पता चलगया कि 1 Subsequence होगा लेकिन उस Subsequence में value क्या होगा?

[]

कोई भी value नहीं होगी

Blank Subsequence

तो Base Case Code कैसे लिखें?

if (str.length() == 0)

base result

ArrayList<String> bres = new ArrayList<>();
 bres.add("");
 return bres;

प्रारंभिक blank subsequence add करे arraylist में, even without a space

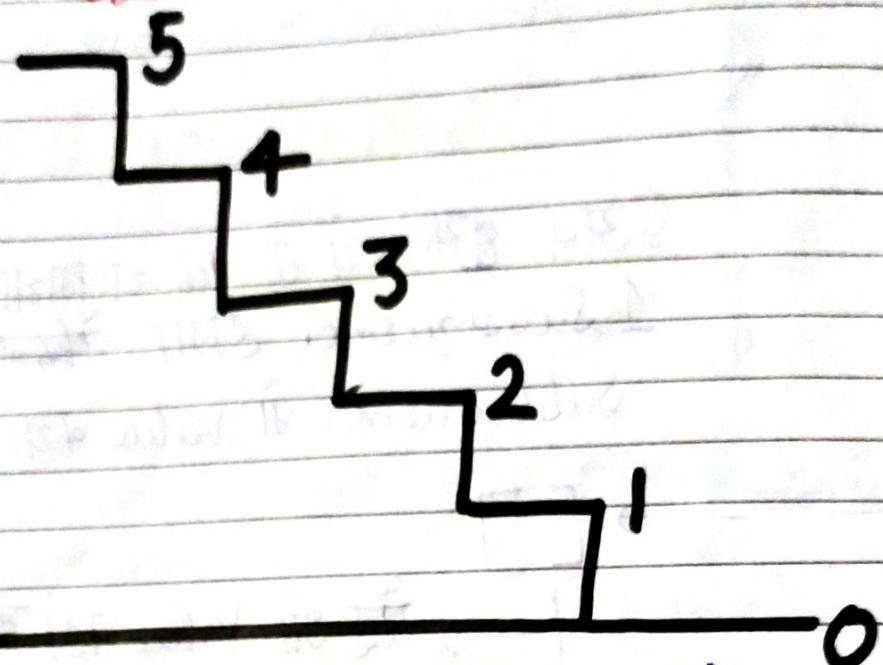
[" "]

ArrayList का Size 1 है अब

bres.get(0).length = 0 लेकिन इस ArrayList में भी String है उसका Size 0 है।

Get Stair Paths

L एवं एक Staircase (स्टेप्स) दिया गया है।



इस स्टेप्स पर एक कदम रख सकते हैं।

या दो कदम रख सकते हैं।

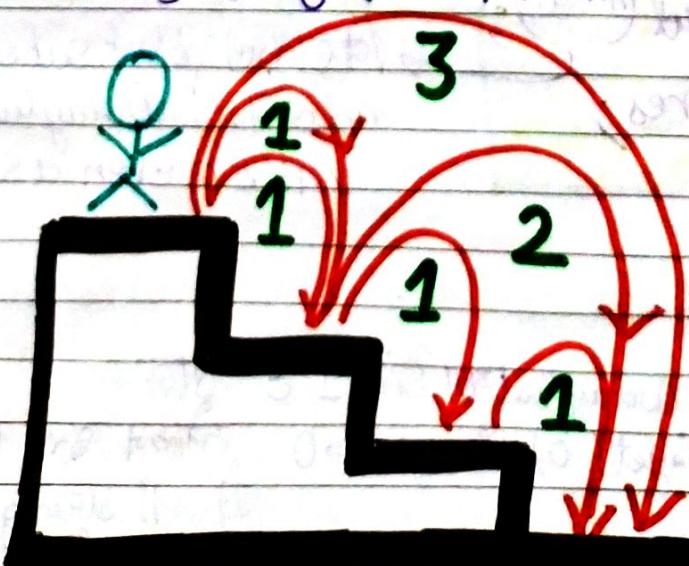
या तीन कदम रख सकते हैं।

स्टेप्स में 0 से 5 तक जाने के सारे तरीके कानूने हैं।

0 से 5 तक पहुँचने के 16 तरीके हो सकते हैं।

या फिर 5 से 0 तक जाने के सारे तरीके बहुबल हैं।

5 से 0 तक जाने के भी 16 तरीके हो सकते हैं।

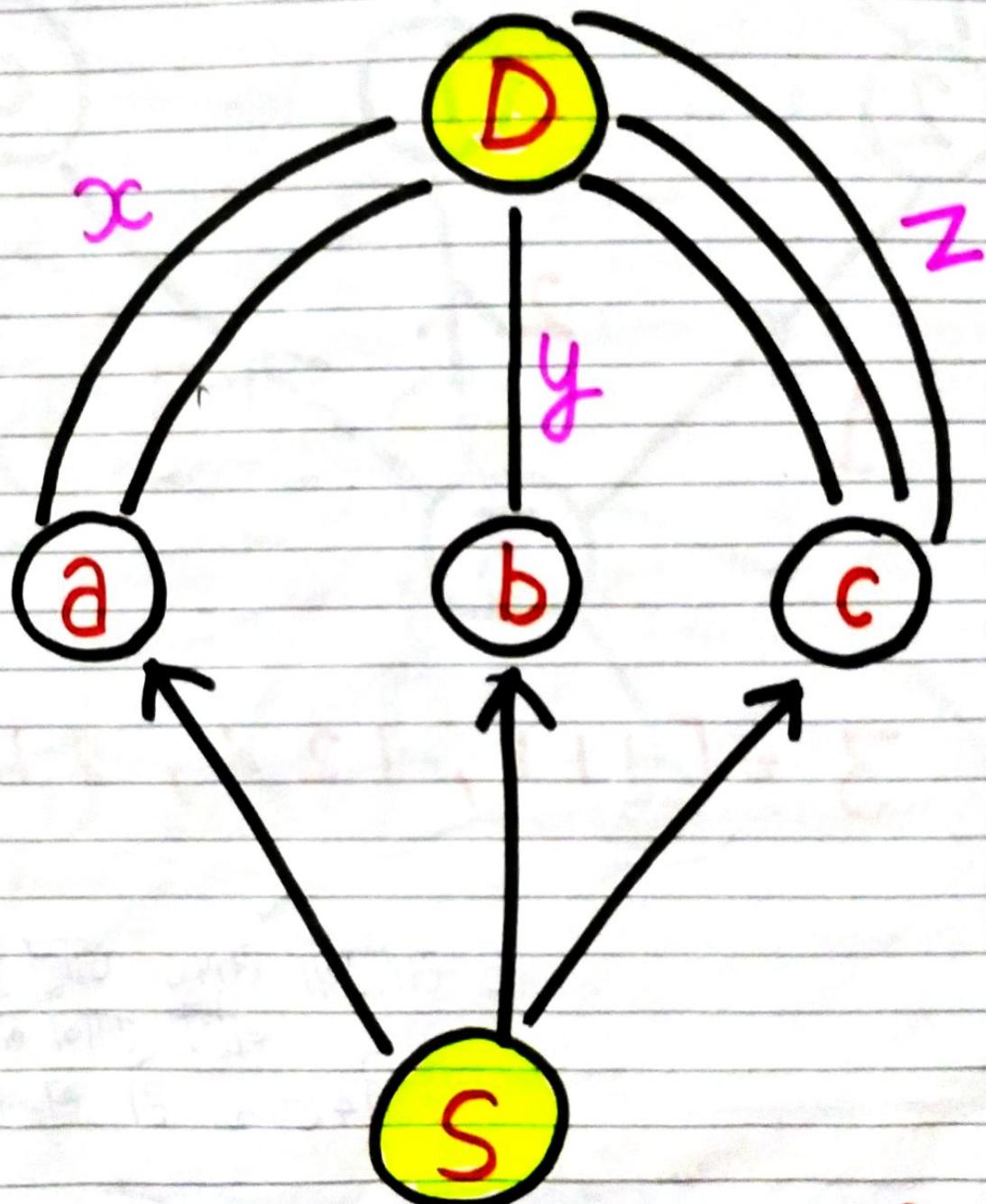


S=Source

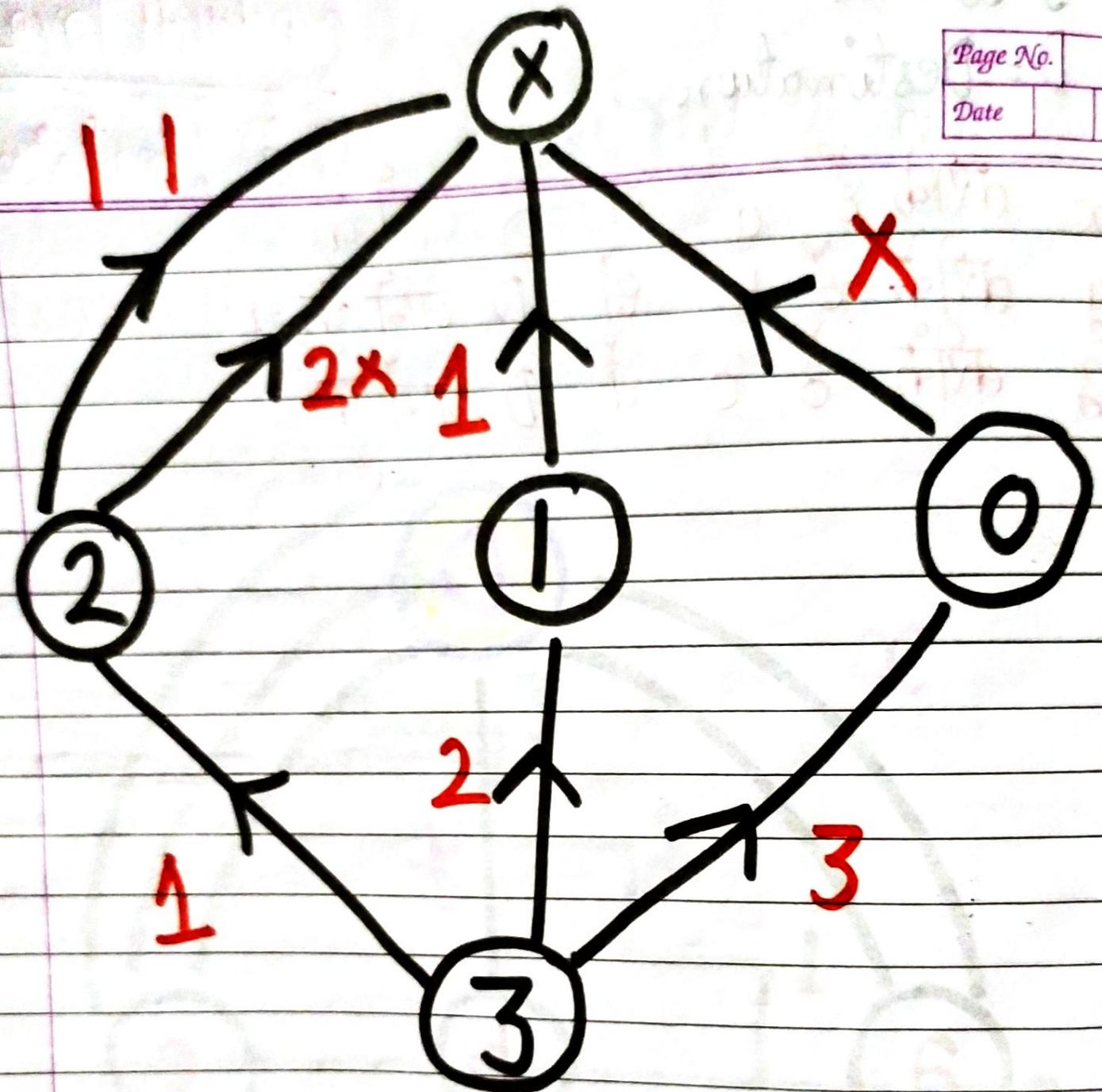
D= Destination

Page No.	
Date	

x तरीके से a से D जाने के
y तरीके से b से D जाने के
z तरीके से c से D जाने के

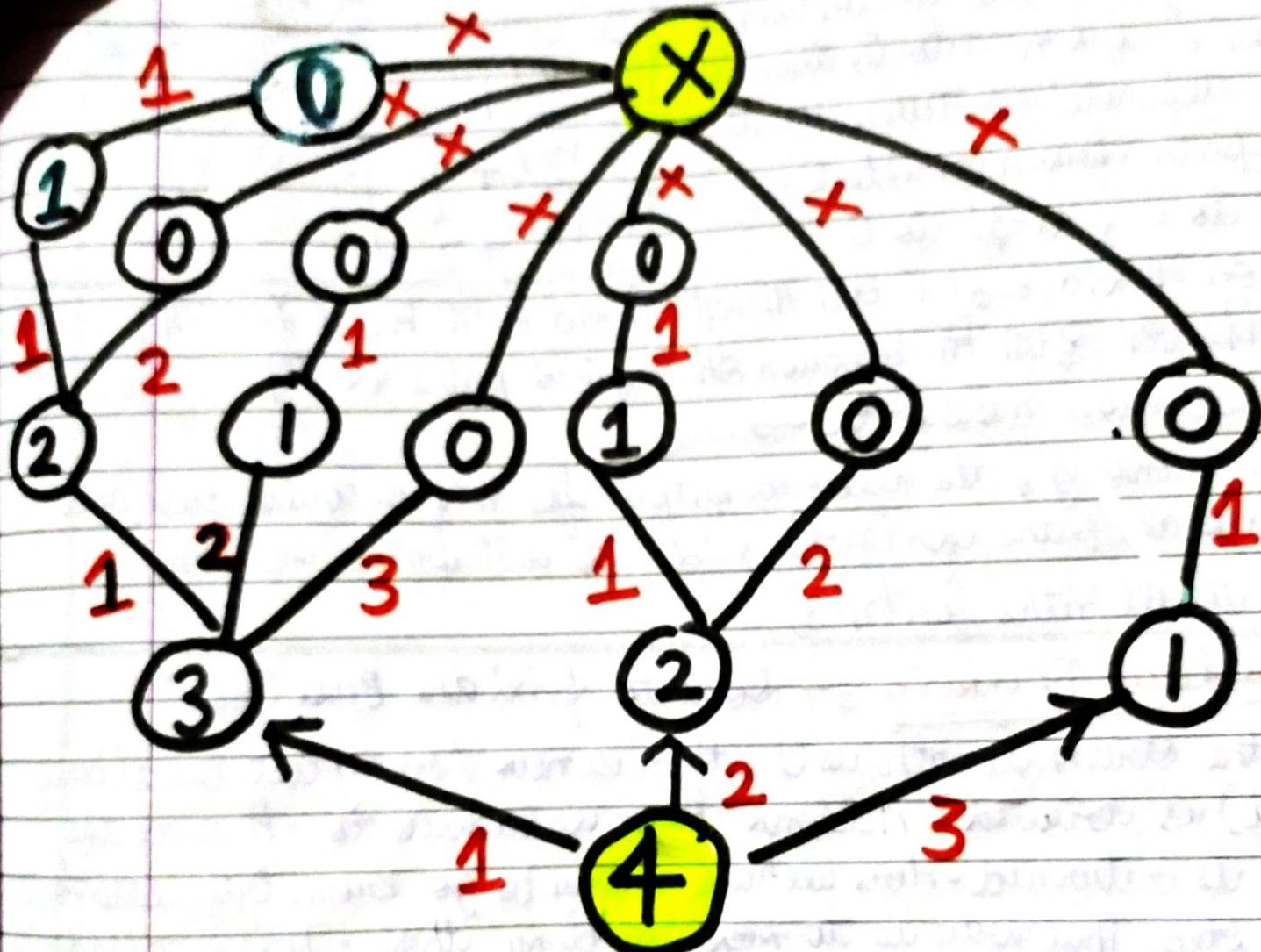


तो S से D तक जाने के
 $x + y + z$ तरीके होंगे।



$$3 = [111, 12x, 21, 3x]$$

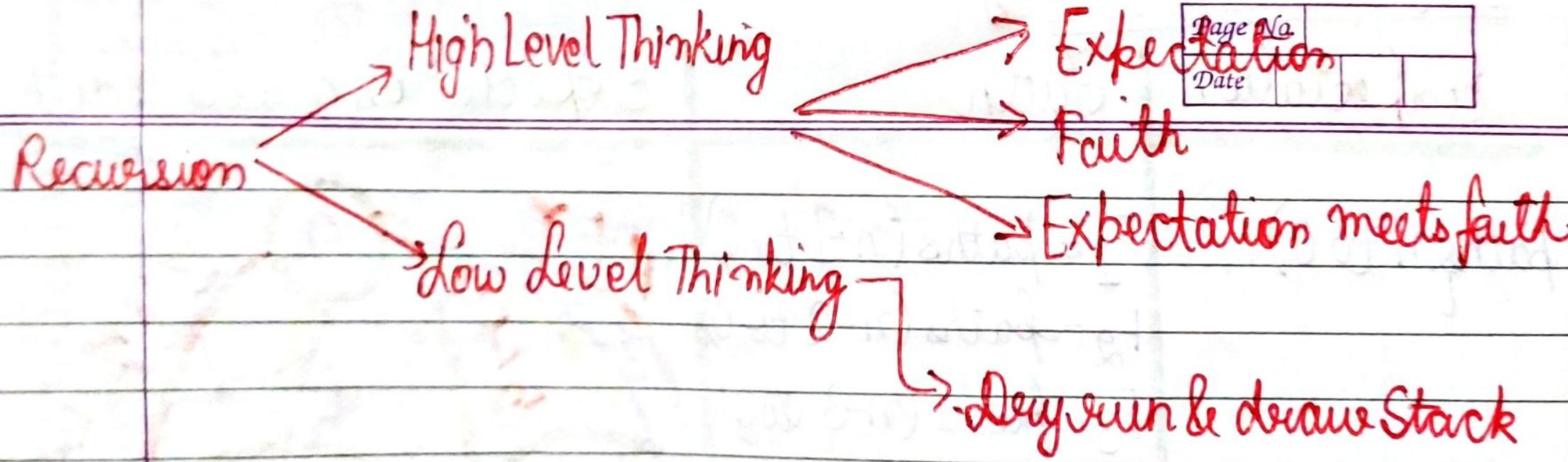
3 सीढ़ियाँ ऊपर जाने के/
नीचे जाने के/
पैर 4 Steps हो सकते हैं।



4. [1111, 112, 121, 13, 211, 22, 31]

4 सीटियों ऊपर जाने के / नीचे आने के थे 7
तरीके हो सकते हैं।

We need to solve this Problem Recursively



1. High Level Thinking :-

Expectation | Faith

paths(n to 0).

$$F_1 = \text{paths}(n-1 \text{ to } 0)$$

$$F_2 = \text{paths}(n-2 \text{ to } 0)$$

$$F_3 = \text{paths}(n-3 \text{ to } 0)$$

1. Set Expectation → We expect that we will get all the paths from n to 0.

2. Build Faith → यह faith है कि हमारा code सब path return कर सकता है. from n to 0, then it can definitely return us all the path from (n-1) to 0, (n-2) to 0, (n-3) to 0.

ऐसा हम मानलेते हैं, ये बस मानलो, ये मत सोचो कि ये कैसे होगा।
बस ऐसा दर्शक कि function द्वारा required paths होंगा।

3. Expectation meets faith →

→ for printing the desired output for n, we could just add "1" to all paths for (n-1), "2" to all paths for (n-2), "3" to all paths for (n-3).

2. Low Level Thinking :- Done to find the Base Case

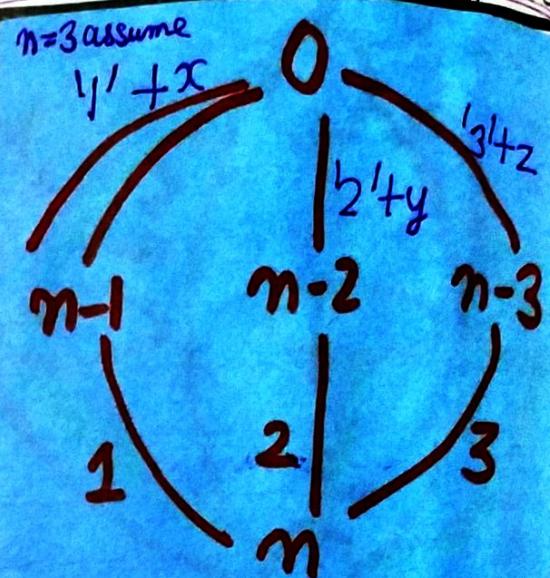
a) If the string length is 0, then a new ArrayList (base result) is created. A blank path is added to it and the list is returned. Here we have 1 way to go from 0th step to 0th step. That way is to keep standing there. This is called the Positive Base Case.

b) When n < 0, it means that there are no stairs left and the person will be going into the basement which isn't practically possible. Here, we don't add any path to the ArrayList.

This is called the Negative Base Case.

Expectation meets faith

Page No.



$$\left\{ \begin{array}{l} 1' + x \\ 2' + y \\ 3' + z \end{array} \right\}$$

Time Complexity:- $O(3^n)$

This time

complexity is exponential because for each state , 3 recursion calls are made .

Space Complexity: $O(1)$

~~As no extra space is required . So Space complexity is constant . If we include the space in recursive stack , then space complexity is $O(n)$.~~

get & Stair Path <code>

Page No.

```

→ import java.util.*;
→ import java.io.*;
→ public class Main {
    → public static void main (String [] args) {
        → Scanner scan = new Scanner (System.in);
        → int n = scan.nextInt();
        → ArrayList < String > paths = getStairPaths(n);
        → System.out.println(paths);
    }
}

→ public static ArrayList < String > getStairPath (int n) {
    → if (n == 0) {
        → ArrayList < String > leres = new ArrayList <> ();
        → leres.add ("");
        → return leres;
    } → else if (n < 0) {
        → ArrayList < String > leres = new ArrayList <> ();
        → return leres;
    } → else {
        → ArrayList < String > leres = new ArrayList <> ();
        → leres.add ("");
        → leres.addAll (getStairPath (n-1));
        → return leres;
    }
}

```

Annotations:

- Yellow highlights: `import`, `java.util.*`, `java.io.*`, `public class Main`, `public static void main (String [] args)`, `Scanner`, `System.in`, `int n`, `ArrayList < String > paths`, `System.out.println(paths)`, `getStairPaths(n)`, `if (n == 0)`, `ArrayList < String > leres`, `leres.add ("")`, `return leres`, `else if (n < 0)`, `ArrayList < String > leres`, `leres.add ("")`, `leres.addAll (getStairPath (n-1))`, `return leres`.
- Red annotations: `ArrayList < String >` is an array list of type String.
- Handwritten notes: "n is no. of stairs.", "function name", "base result", "element is there but no value", "no element.", "total path from n means 1".

These are `ArrayList<String> pathsFromNm1 = getStairPath(n-1);`
Recurrence `ArrayList<String> pathsFromNm2 = getStairPath(n-2);`
Calls `ArrayList<String> pathsFromNm3 = getStairPath(n-3);`

String
path from n to m
for each string path in ArrayList paths from nm1

Page No.
Date:

→ for (String pathfromnm1 : pathsfromnm1)

{

String pathfromn = 1 + pathfromnm1;
pathsfromn.add(pathfromn);

}

→ for (String pathfromnm2 : pathsfromnm2)

{

String pathfromn = 2 + pathfromnm2;
pathsfromn.add(pathfromn);

}

→ for (String pathfromnm3 : pathsfromnm3)

{

String pathfromn = 3 + pathfromnm2;
pathsfromn.add(pathfromn);

}

return pathsfromn;

}