

⑤ RECURSION ON THE WAY UP

⑥ Print Subsequence

subsequences $\Rightarrow ["", c, b, bc, a, ac, cb, abc]$
of
abc

3 characters are there a, b and c

$$3 \rightarrow 2^3 \text{ items} = \frac{2^3}{2} \text{ pairs} = 2^2 \text{ pairs}$$

\therefore Har pair me
3 characters hai

$$\therefore 2^2 \times 3 \text{ characters} \\ \Rightarrow 12 \text{ characters}$$

⑦ length ki resultant arraylist ke andar total
 $2^{n-1} \cdot n$ characters hongi

$$\therefore [n \rightarrow 2^{n-1} \cdot n \text{ characters}]$$

For example

$$\therefore \text{let } n = 31$$

$$\therefore 31 \rightarrow 2^{30} \times 31 = (2^{10})^3 \cdot 31 = (1074)^3 \cdot 31 = (10^3)^3 \\ = 31 \times 10^9 \text{ characters} \\ = 31 \times 10^9 \text{ bytes} \\ = 31 \text{ GB}$$

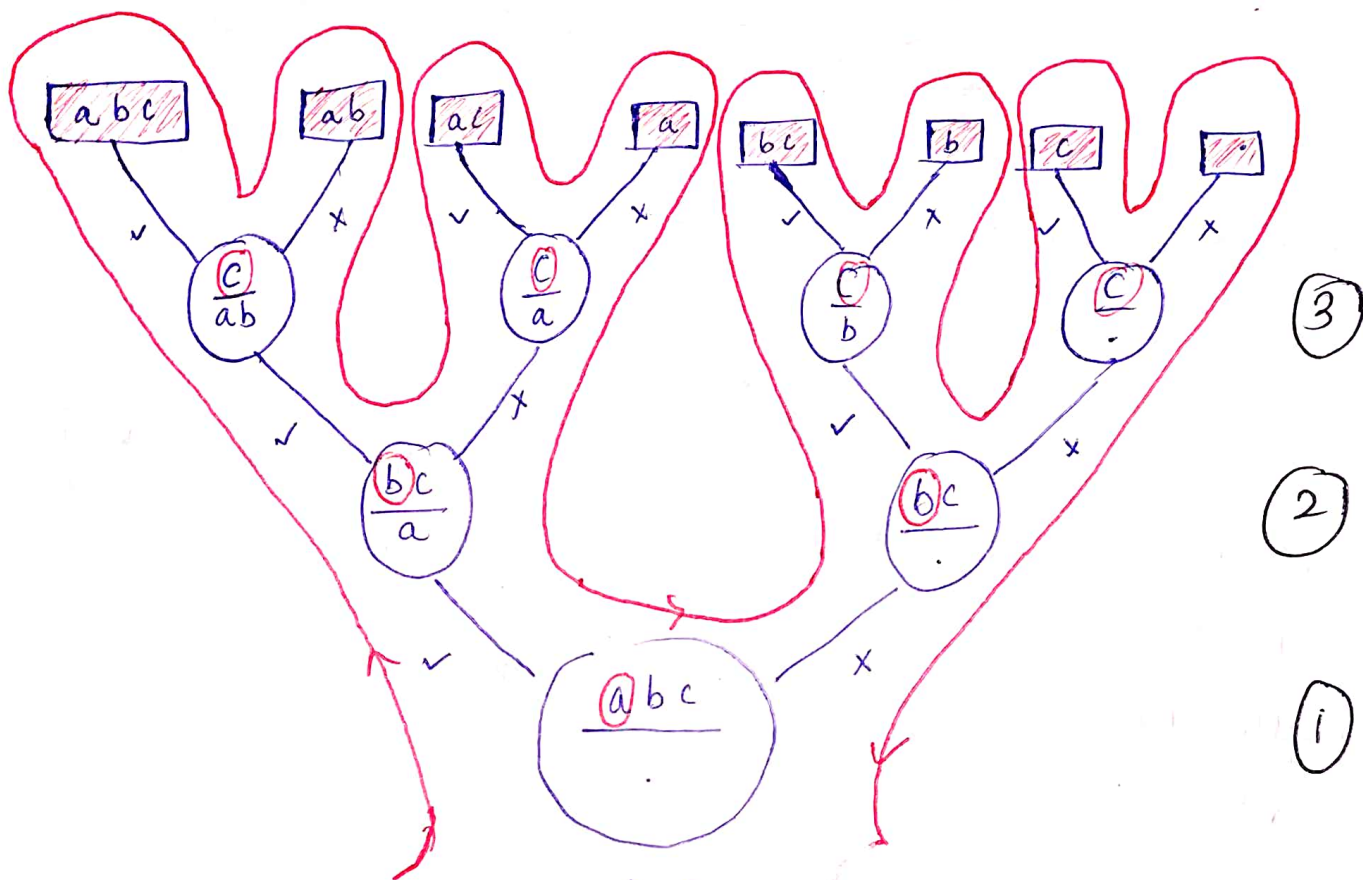
memory bahut
lag rahi hai

Hume sirf print
karne hai toh
store nahi krenay

itni toh RAM hi nahi
hai humare pass

Sirf print
hi krenge

Numerator = Question
Denominator = Answer



- Hence pe characters hai
- options me ek option Yes (✓) or No (X).
- Aab Yeh memory me kese better hoga?
- ↳ Yeh kashi memory me store nahi krega
- Yeh bulay chalega!

Stack kabhi bhi n-level ke beyond nahi
jayegi! kuki humne kabhi kuch store nahi
kiya humne sirf Path store kiya hai

```

P s v m (s (J a) {
Scanner s = new Scanner (System.in);
String str = s.next();
Print SS (str, "");

```

3

p s v printSS (String ques, String ans) {

if (ques.length() == 0) {

System.out.println(ans);

return;

}

} BASIC
CASE

char ch = ques.charAt(0); a

String req = ques.substring(1); bc

printSS(bc, ans + ch); // Yes call

printSS(bc, ans + ""); // No call

}

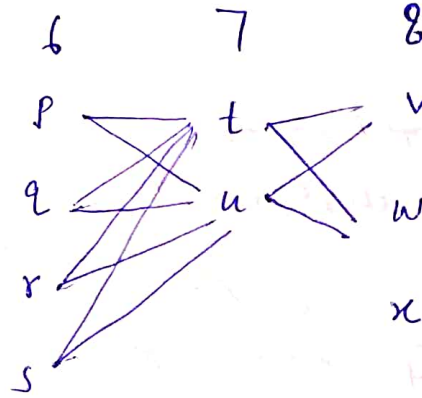
Time Complexity: $O(2^n)$ → Exponential becoz for each state, 2 recursive calls are made.

Space Complexity: $O(n)$

⑤ Print KPC → Ek-Ek word produce krke store krna hai taki saare words store na krne pade!

Signature : Ques-Answer Approach!

0 → .;
1 → abc
2 → def
3 → ghi
4 → jkl
5 → mno
6 → pqrs
7 → tu
8 → vwx
9 → yz

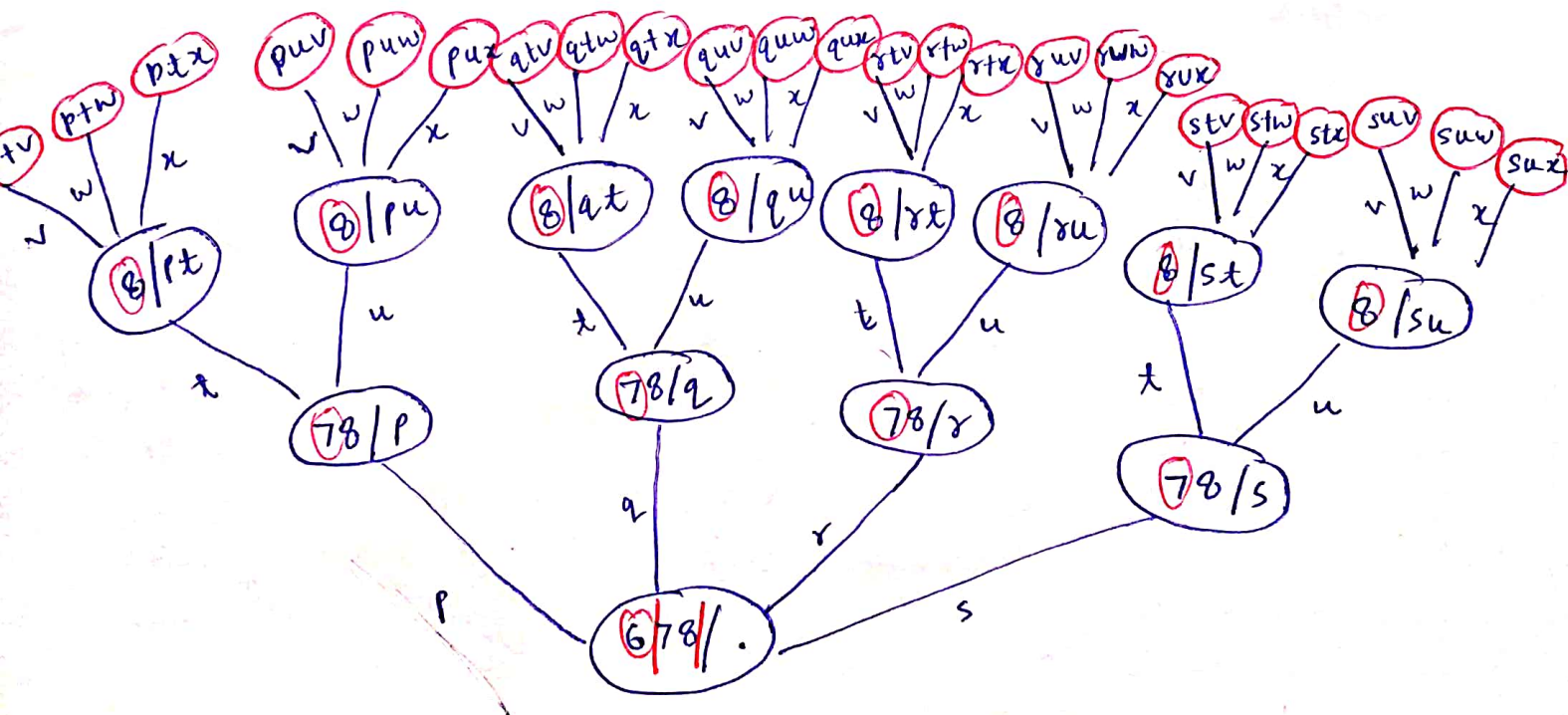


```

P s v m (S[] a) {
    Scanner s = new Scanner(System.in);
    String str = s.next();
    printKPC(str, "");
}

```

static String[] codes = {".;", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tu", "vwx", "yz"};



Levels and Options Style

```

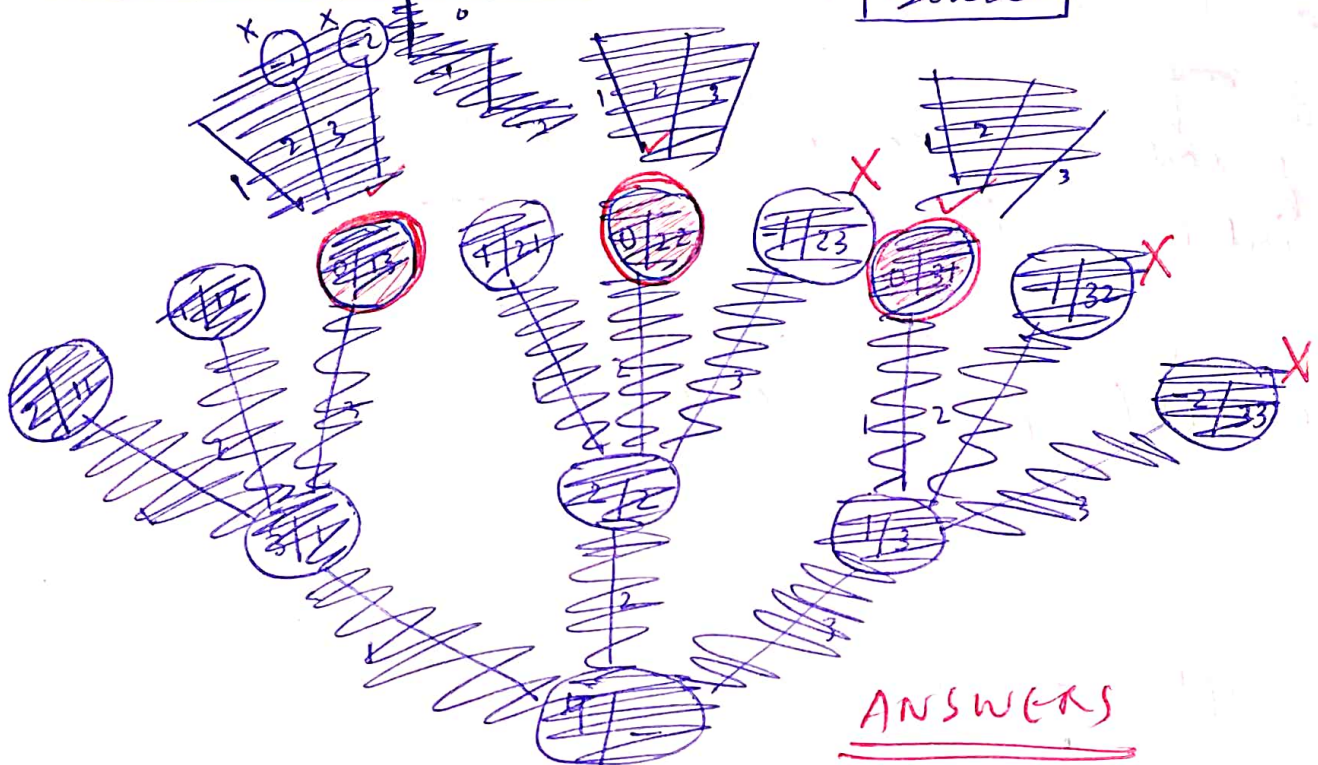
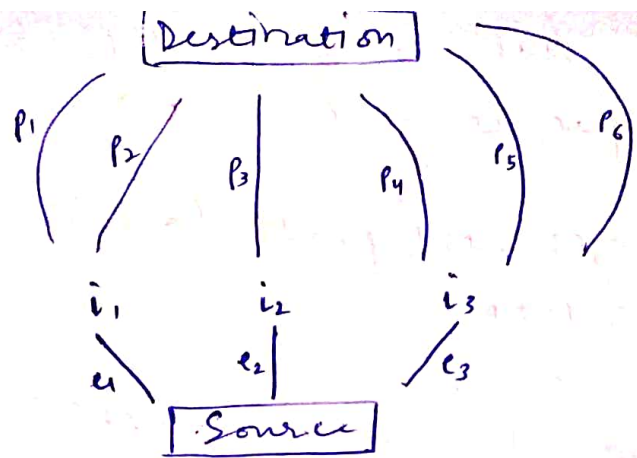
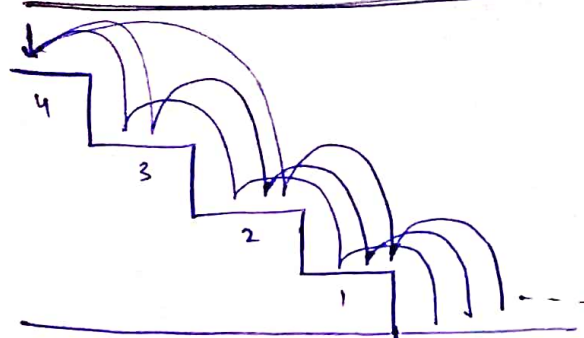
public static void printKPC (String ques, String ans) {
    if (ques.length() == 0)
    {
        System.out.println(ans);
        return;
    }
    //678
    char ch = ques.charAt(0); //6
    String req = ques.substring(1); //78
    String codeForch = codes[ch - "0"];
    for (int i = 0; i < codeForch.length(); i++)
    {
        char cho = codeForch.charAt(i);
        printKPC(req, ans + cho);
    }
}

```

Time Complexity : $O(2^n)$

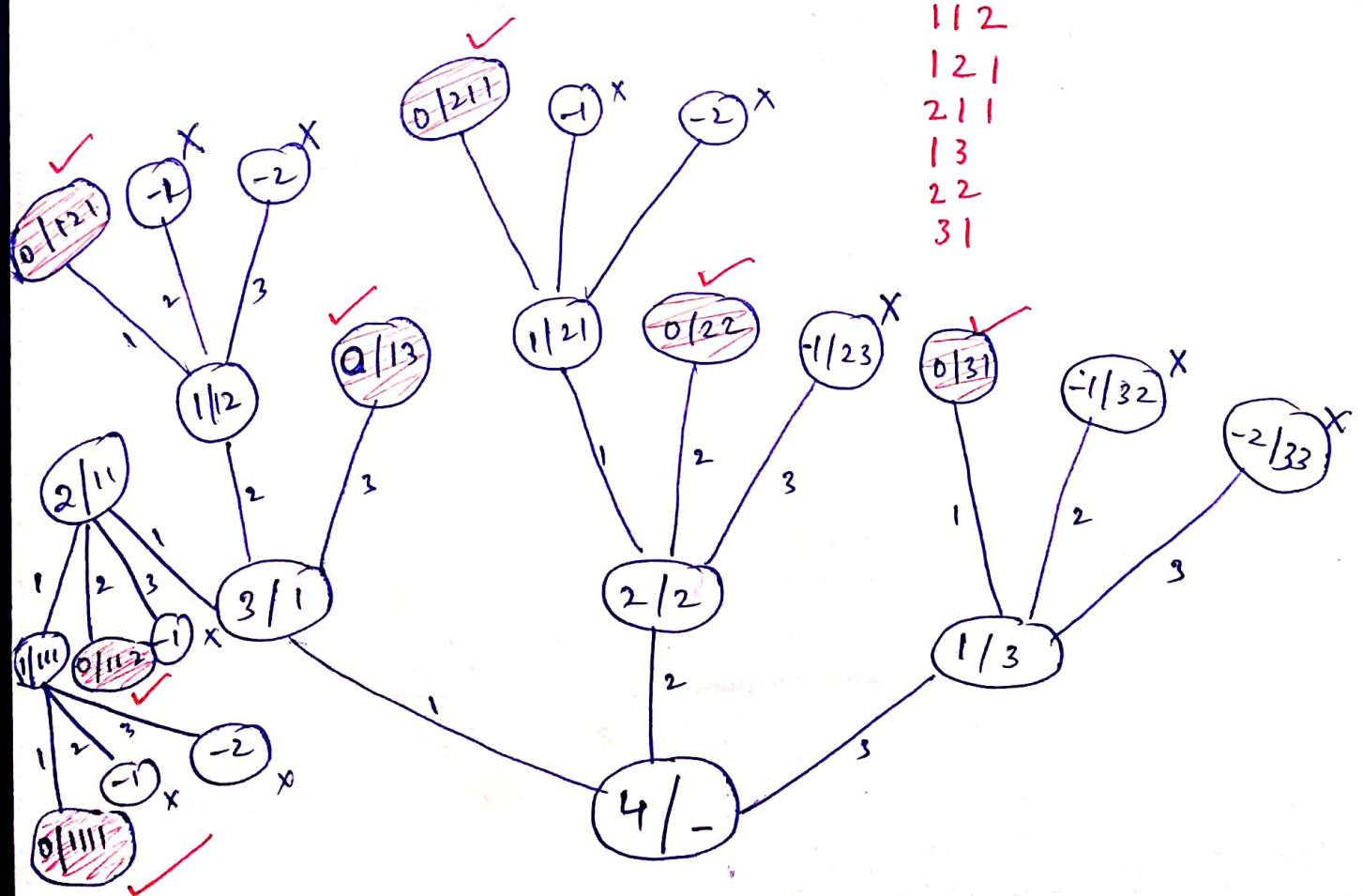
Space Complexity : $O(1)$

Print Stairs Paths



ANSWERS

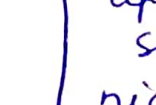
- 1111
- 112
- 121
- 211
- 13
- 22
- 31



```
p s v m (SCJA) {  
Scanner s = new Scanner(System.in);  
int n = s.nextInt();  
printStairPaths(n, "");  
}
```

```
print StairPaths (n-1, path+1); // ek kadam Rakhma
_____ (n-2, path+2); // do kadam Rakhma
_____ (n-3, path+3); // Teen kadam Rakhma
}
```

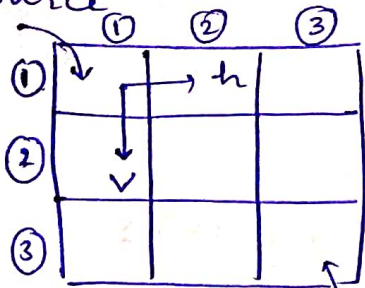
④ Level - Option Recursion


 upar
 se
 niche solve
 yaha par anke
 ke baad jawab
 barega!

Space Complexity: $O(1)$

5) PRINT MAZE PATH

Source



Destination

⇒ Coding kerty huay, yeh gaad rakhna hai ki! → Every level pe humarepe

2 options had

- ① Horizontally right
→ ② vertically down

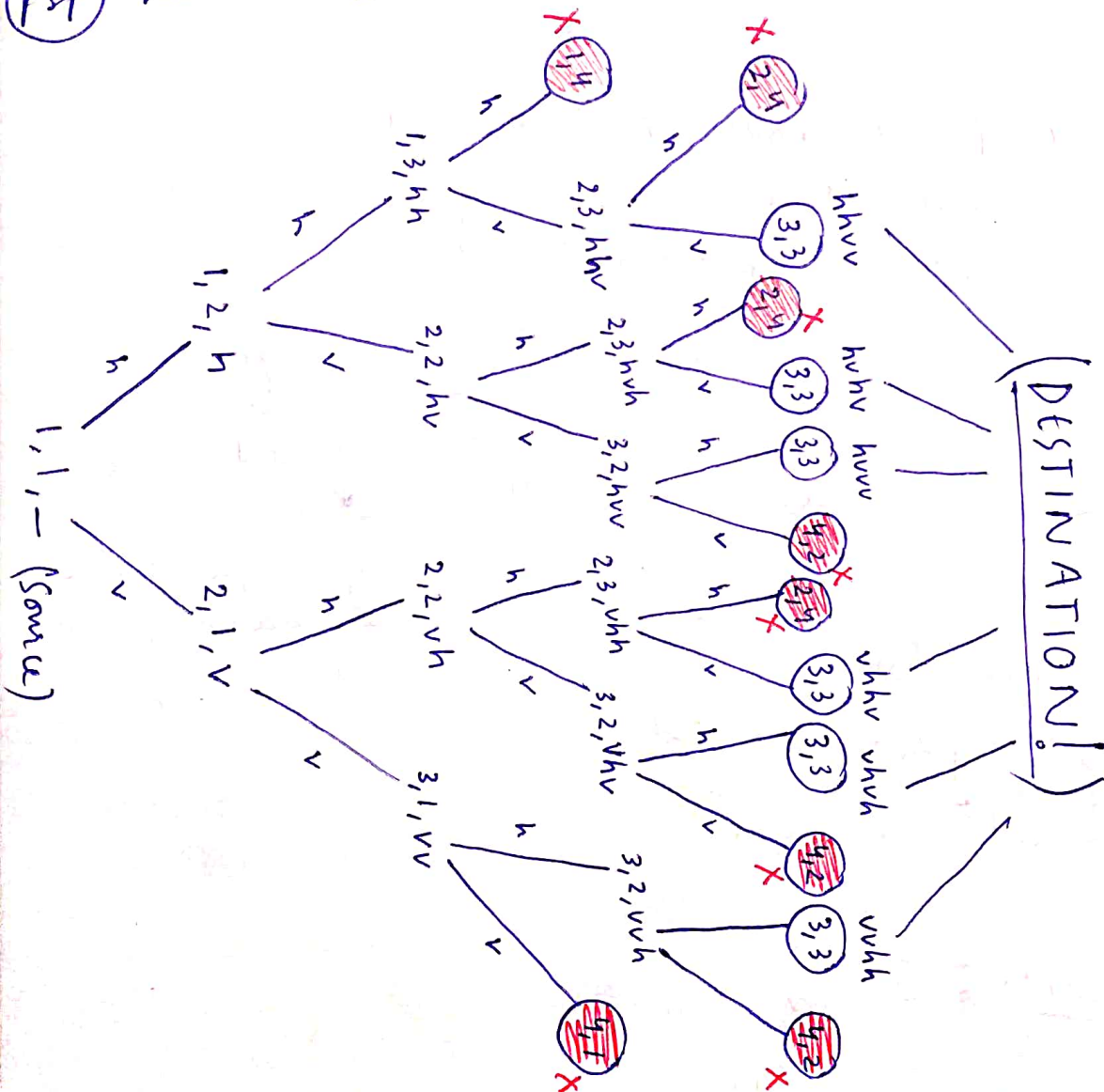
✓ Horizontally right se (SC)

change hota hai $(sc+1)$ me
aur (psf) update hoga

(psf) ke sath (h) lagega!
(updating by concatenating "h")

≡ Vertically down se (sr)
change hoga $(sr+1)$ me
aur (psf) bhi update
hoga.

hoga. PSP ke sath (V) lagega!



~~///~~ jab bhi hum destination
 reach karengy
 means $\{ SC = d < ? \}$
 $\{ sr = dx \}$
 tab hum (psf) ko print
 karna hae!

Horizontal travelling me $sr > dr$ hua aur $sc > dc$ hua toh kuch return nahi hoga!
Yeh possible hai nahi


```

public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int n = s.nextInt();
    int m = s.nextInt();
    printMazePaths(0, 0, n-1, m-1, "");
}

```

$\left\{ \begin{array}{l} sr = \text{Source row} \\ sc = \text{Source column} \\ dc = \text{destination column} \\ dr = \text{destination row} \end{array} \right.$

```

public static void printMazePaths(int sr, int sc, int dr, int dc, String psf) {
    if (sr > dr || sc > dc) {
        return;
    }
    if (sr == dr && sc == dc) {
        System.out.println(psf);
        return;
    }
    printMazePaths(sr, sc+1, dr, dc, psf + "h");
    printMazePaths(sr+1, sc, dr, dc, psf + "v");
}

```

Time Complexity : ~~1000~~

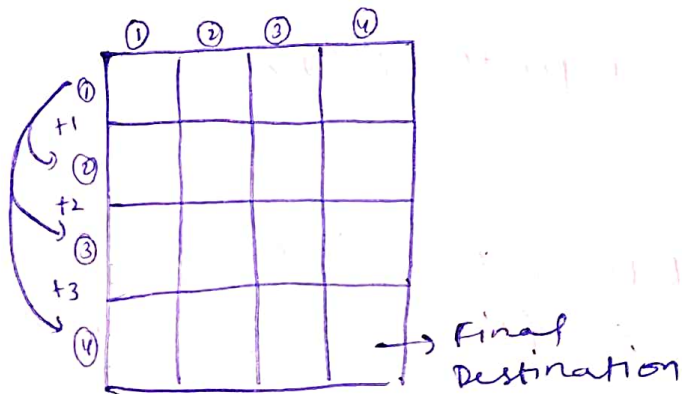
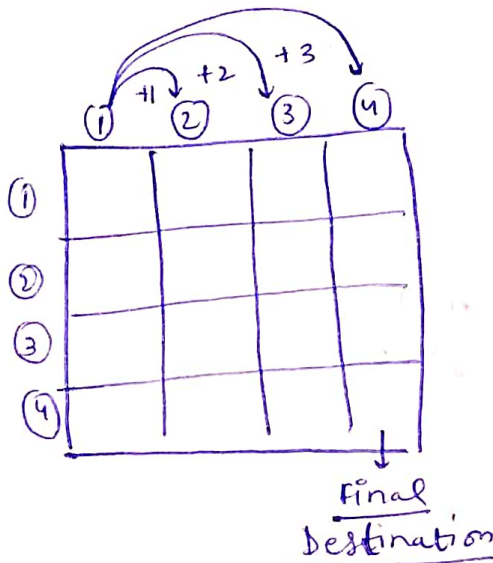
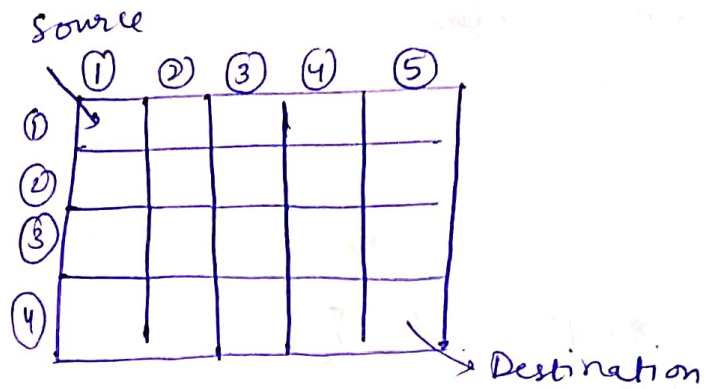
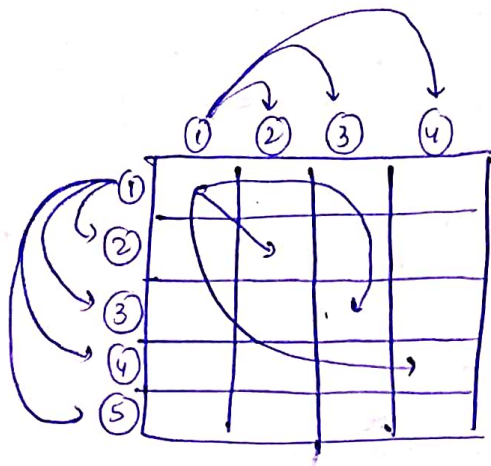
↳ Time complexity of this method is $O(2^{n \times m})$

$n \Rightarrow$ no. of rows
 $m \Rightarrow$ columns

Space Complexity :

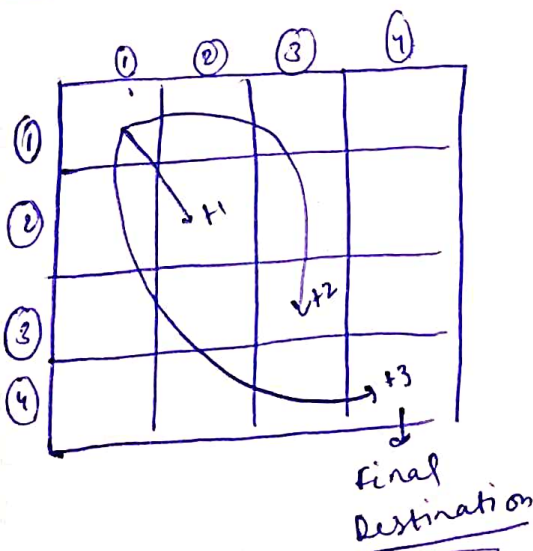
$O(1)$

Print Maze Paths with Jumps



Max. Size of Jump = $dc - sc$ along a row

Max. size of jumps = $dr - sr$ along a column



Base Case

```
if (sr == dr && sc == dc) {
    Sys.println(psf);
    return;
}
```

Max-Size of jumps along the rows and columns
= $dc - sc$ & $dr - sr$


```

public static void main (String[] args) throws Exception {
    BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
    int n = Integer.parseInt (br.readLine());
    int m = Integer.parseInt (br.readLine());;
    printMazePaths (0, 0, n-1, m-1, "");
}

private static void printMazePaths (int sr, int sc, int dr, int dc, String psf) {
    if (sc == dc && sr == dr) {
        System.out.println (psf);
        return;
    }

    for (int move = 1; move <= dc - sc; move++) {
        printMazePaths (sr, sc + move, dr, dc, psf + "h" + move);
    }

    for (int move = 1; move <= dr - sr; move++) {
        printMazePaths (sr + move, sc, dr, dc, psf + "v" + move);
    }

    for (int move = 1; move <= dc - sc && move <= dr - sr; move++) {
        printMazePaths (sr + move, sc + move, dr, dc, psf + "d" + move);
    }
}

```

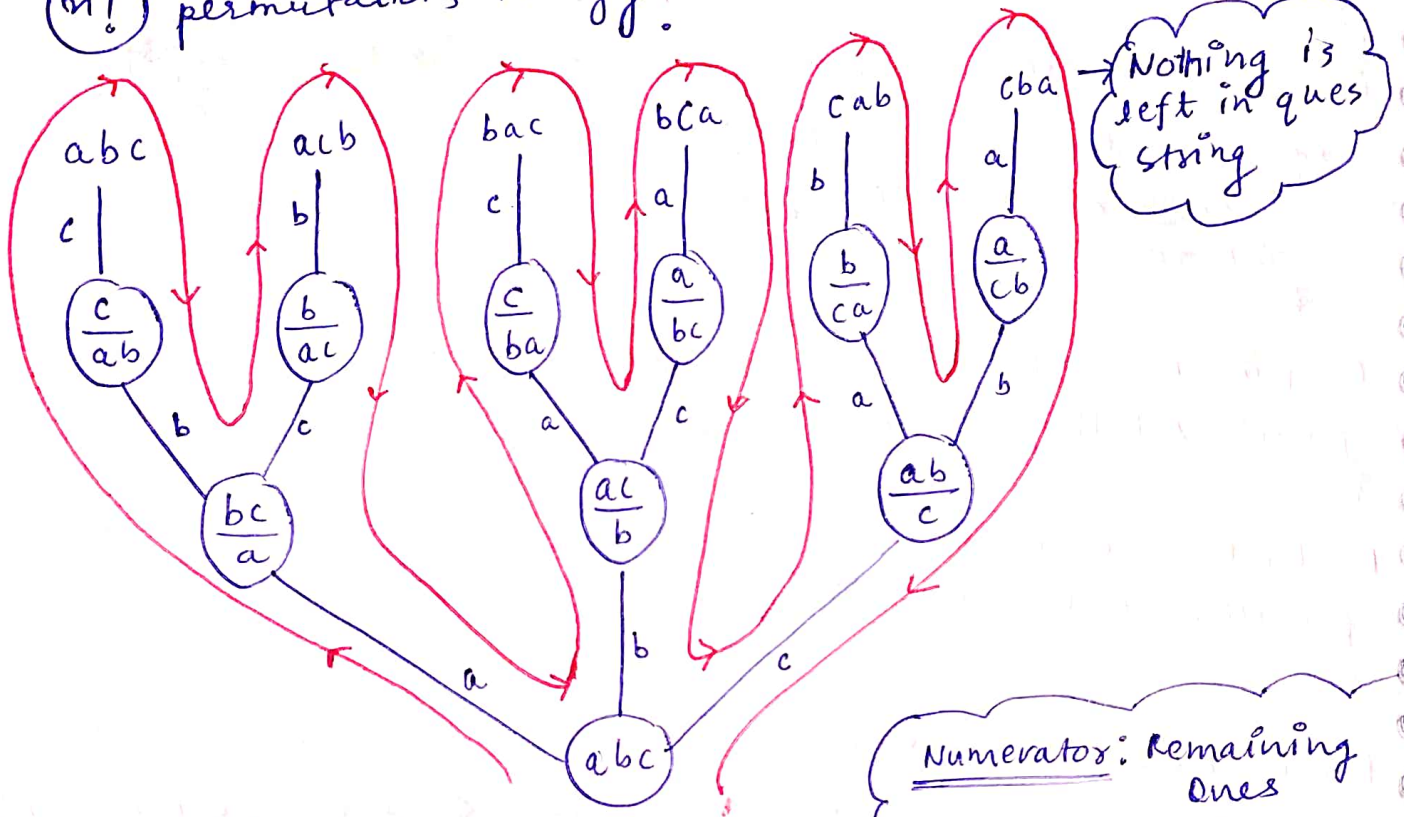
Time Complexity: $O(3m+n)$

Space Complexity: $O(1)$

③ Permutation (Permt)

Permutation of "abc" : abc, acb, bac, bca, cab, cba

\therefore ek string agar (n) length ki hai toh uske $(n!)$ permutations hongi!



```

public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    String str = s.next();
    printPermutations(str, "");
}

```

Numerator: Remaining ones

Denominator : Answer
so-far

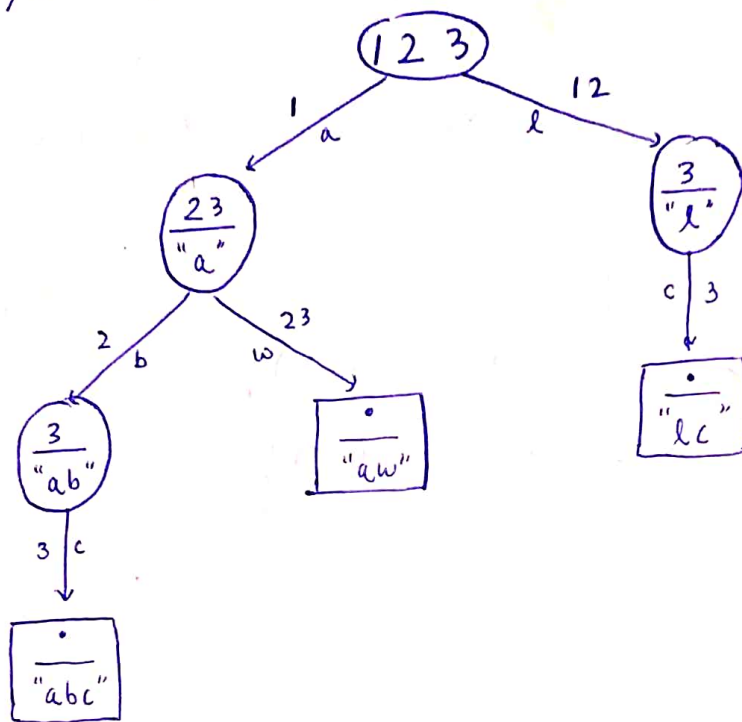
```

}
P 5
    v
    printPermutations (String ques, String asf) {
    if (ques.length() == 0) {
        Base Case : Question string is empty
        so print the answer now and
        return
        Syso. (asf);
        return;
    }
    for (int i = 0; i < ques.length(); i++) {
        char ch = ques.charAt(i);
        String qleft = ques.substring(0, i); // left to ch (substring from 0 to i-1)
        String qright = ques.substring(i+1, ques.length()); // right to ch (substring from i+1 to end)
        String roq = qleft + qright; // remaining string
        printPermutations(roq, asf + ch);
    }
}

```


Print Encodings

1 → a, 2 → b, 3 → c, ..., 25 → y and 26 → z



```

p s v m (SCJA) {
Scanner s = new Scanner(System.in);
int str = s.nextInt();
printEncodings(str, "");
}
p s v printEncoding(String ques, String asf) {
if (ques.length() == 0) {
    Syso(asf);
    return;
}
else if (ques.length() == 1) {
    char ch = ques.charAt(0);
    if (ch == '0') {
        return;
    }
    else {
        int chr = ch - '0';
        char char code = (char) ('a' + chr - 1);
        asf = asf + code;
        System.out.println(asf);
    }
}
}

```

```
else {
```

```
    char ch = ques.charAt(0);  
    String roq = ques.substring(1);
```

```
    if (ch == '0') {  
        return;
```

```
    } else {
```

```
        int chv = ch - '0';
```

```
        char code = (char)('a' + chv - 1);
```

```
        printEncodings(roq, asf + code);
```

```
    }
```

```
    String ch12 = ques.substring(0, 2);
```

```
    String roq12 = ques.substring(2);
```

```
    int ch12v = Integer.parseInt(ch12);
```

```
    if (ch12v <= 26) {
```

```
        char code = (char)('a' + ch12v - 1);
```

```
        printEncodings(roq12, asf + code);
```

```
    }
```

```
}
```

```
}
```

ques ki length

2 ya 2 se lambey

Ek character nikala aur rest of string nikali

Agar character 0 hai toh return warna, phle character value fir uska code niklega, fir printEncoding call lagegi

Phle do character nikaley aur uske baad ki string nikali.

Phle do ~~integer~~ character ko integer me convert kiya! Agar vo 26 chota hai → toh use code me convert krey aur call lagaye!

① → 0 → hai toh asf return krdey!

② → 1 → hai aur character 0 hai toh return krdo

hai aur character non-zero hai toh,

→ phle uss character ki value nikalo.

→ fir uska code nikalo

→ asf me code ko add kro

→ asf print krdo.

Time Complexity

Hum atmax 2 decision bana skty hai !

(1-length no. (ya) 2-length no.)

$$\therefore \underline{O(2^n)}$$

Space Complexity

$$\underline{O(n)}$$

X: not possible!
No mapping

