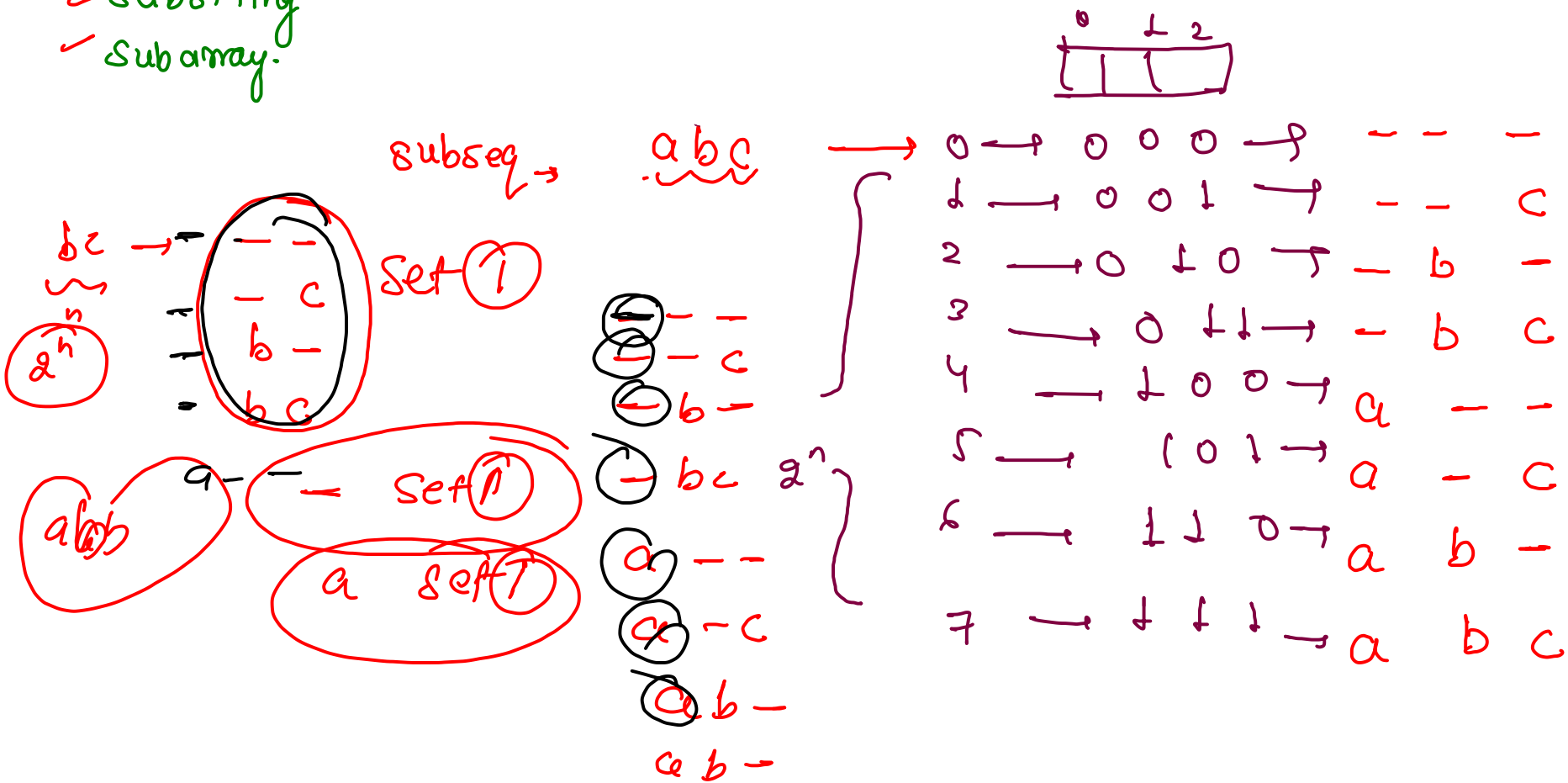
- Subseq
- Subset
- Substring
- Subarray.



getSubsequence

Expectation. gss (abc) => res: { "- - -", "- - c", "- b -", "- b c", "a - -", "a - c", "a b -", "a b c" }

faith → gss ("bc") => { "- -", "- c", "b -", "b c" }

Merging of faith & Expectation → gss (abc) => res = gss(bc); } Recursion

char ch = 'a';

Subseq of bc

Recursion → { - -, - c, b -, b c }

res { - - -, - - c, - b -, - b c,
a - -, a - c, a b -, a b c }

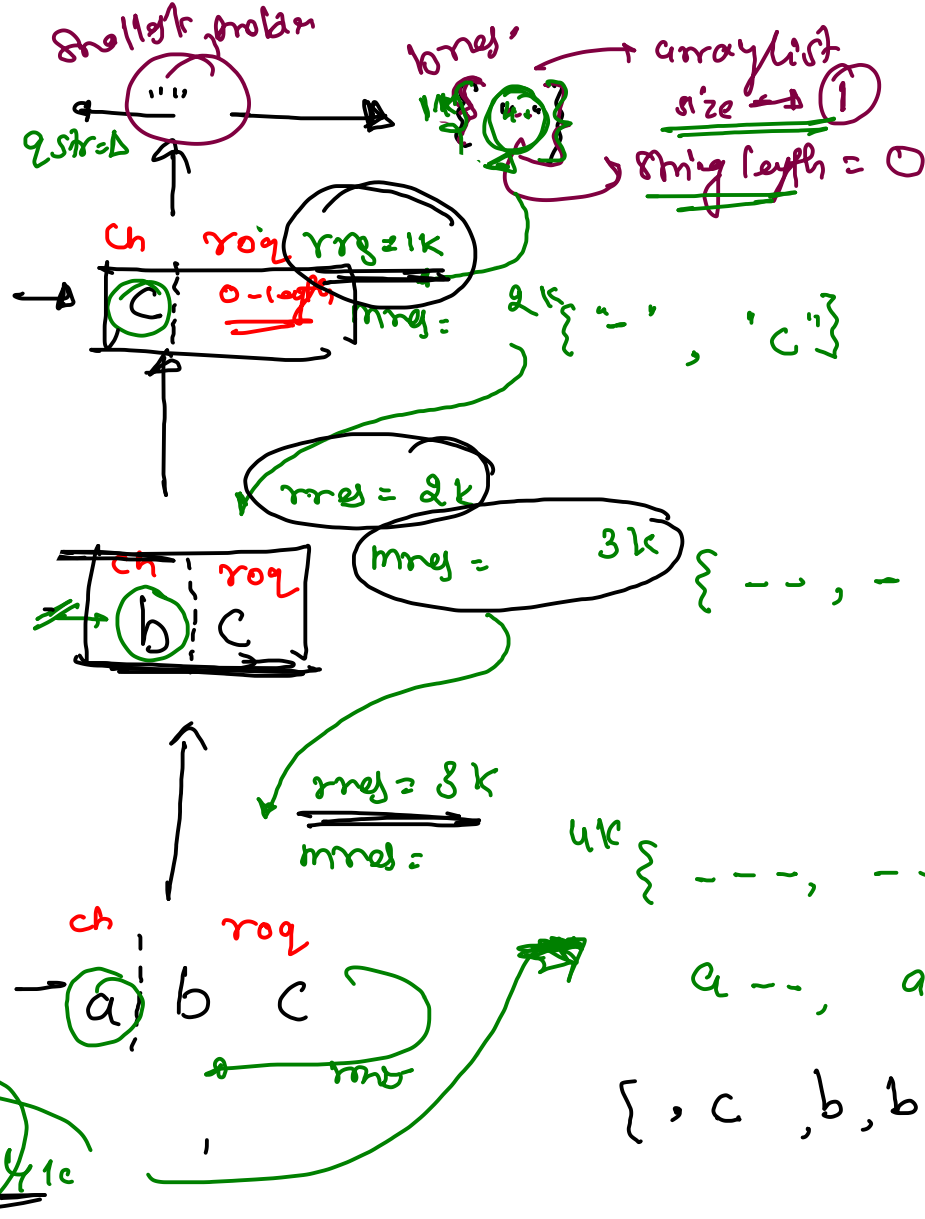
res.

qstring → abc, substring

rest of question → bc

q string . substring(1);

String of length



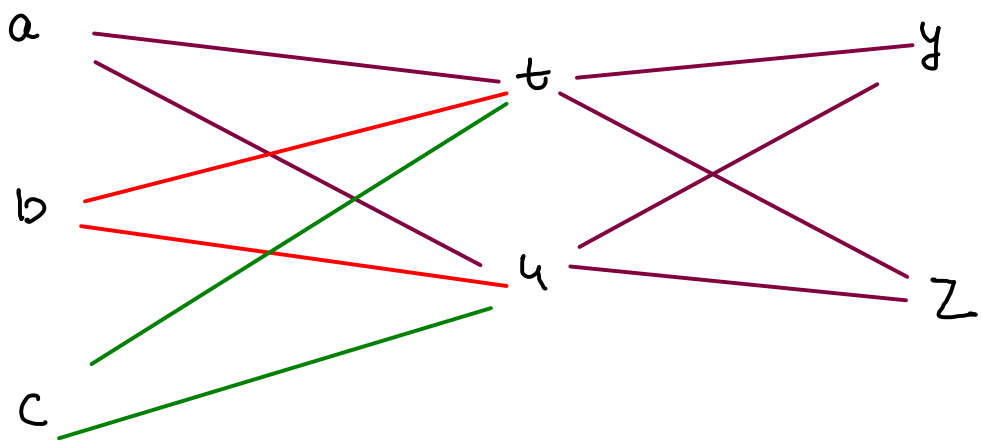

```
char ch = str.charAt(0);
// roq => rest of question
String roq = str.substring(1);
ArrayList<String> rres = getSubseq(roq);
ArrayList<String> mres = new ArrayList<>();
// character add with "no" option
for(String s : rres) {
    mres.add("-" + s);
}
// character add with "yes" option
for(String s : rres) {
    mres.add(ch + s);
}

return mres;
```

get kpc (keypad code)

get all possibilities

"179"

1 7 9



aty
atz
auy
auz

bty
btz
buy
buz

ctz
cuy
cuz

3 x 2 x 2 = 12

array of String

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

ch : '1' → character → int → integer

getkpc

1 → abz, 7 → t4, 9 → yz.

Expectation

getkpc (1 7 9)

faith (7 9) → {ty, tz, 4y, 4z} } Recursion

→ Set 1

aty
atz
a4y
a4z
bty
btz
b4y
b4z
cty
ctz
c4y
c4z

Merging — getkpc (179) — set 1 = getkpc (79);

code

a → {ty tz 4y 4z}

b → {ty tz 4s .

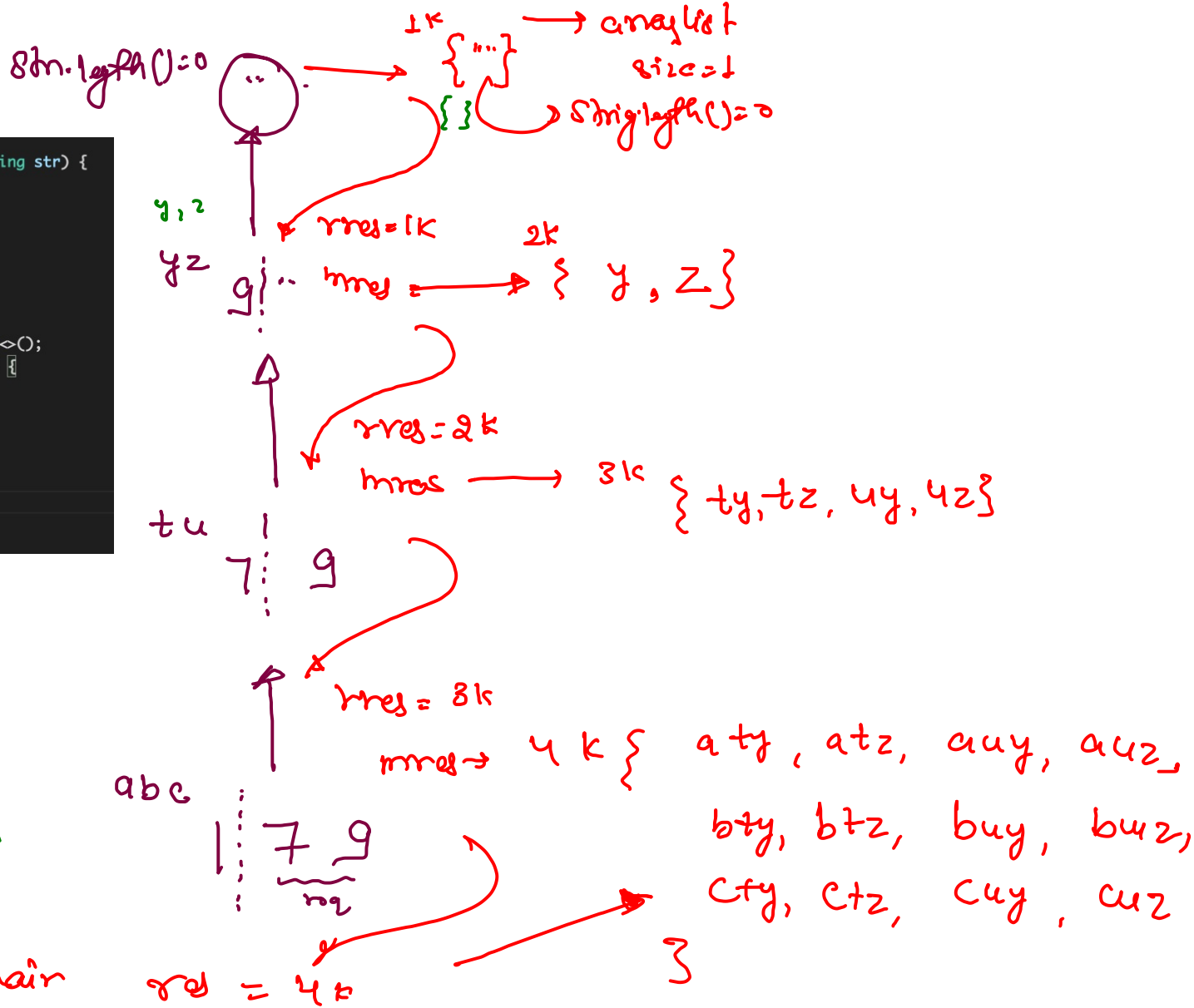
c →

mrg

1 → abc
a set 1
b set 1
c set 1

1 → abc
7 → tu
9 → yz

```
public static ArrayList<String> getKPC(String str) {

    int indx = str.charAt(0) - '0';
    String code = codes[indx];
    String roq = str.substring(1);

    ArrayList<String> rres = getKPC(roq);

    ArrayList<String> mres = new ArrayList<>();
    for(int i = 0; i < code.length(); i++) {
        char ch = code.charAt(i);

        for(String s : rres) {
            mres.add(ch + s);
        }
    }
    return mres;
}
```



'0' → 48

'7' → 55 - 48

55 - 48 = 7

index 7

int ('7' - 48)

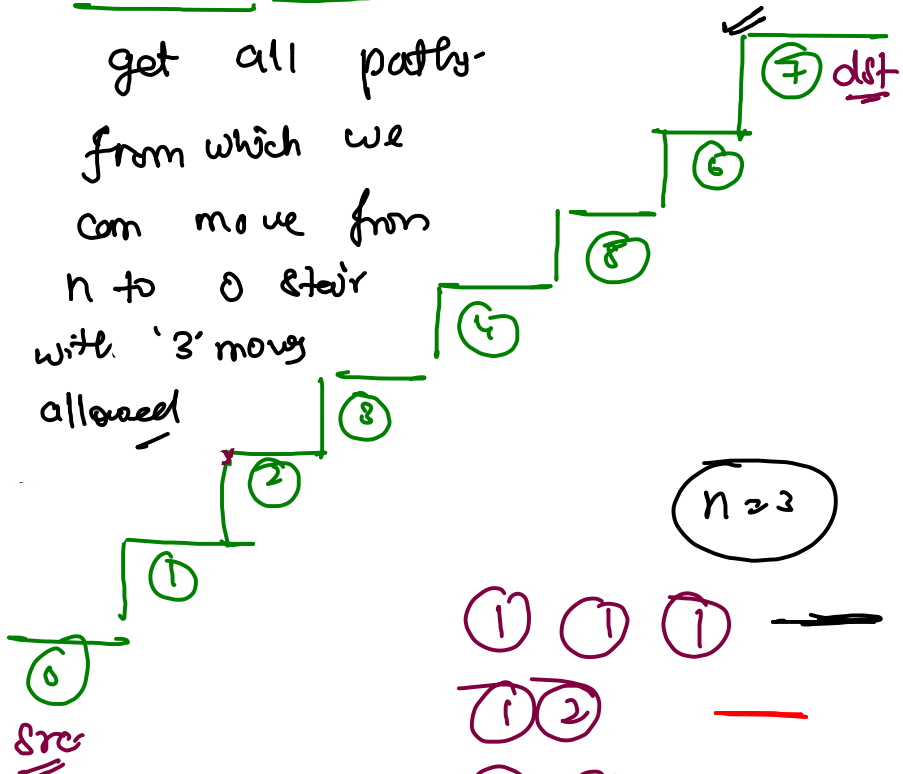
('7' - '0')

main

getStair path -

get all paths from which we can move from n to 0 stair with '3' moves allowed

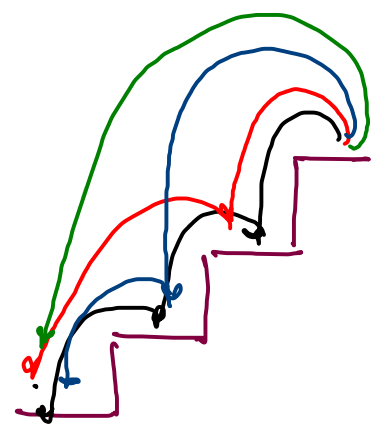
n = 7



moves allowed → { 1 jump, 2 jump, 3 jump }

n = 3

{"1 1 1", "1 2", "2 1", "3"}



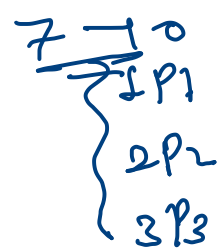
Expectation

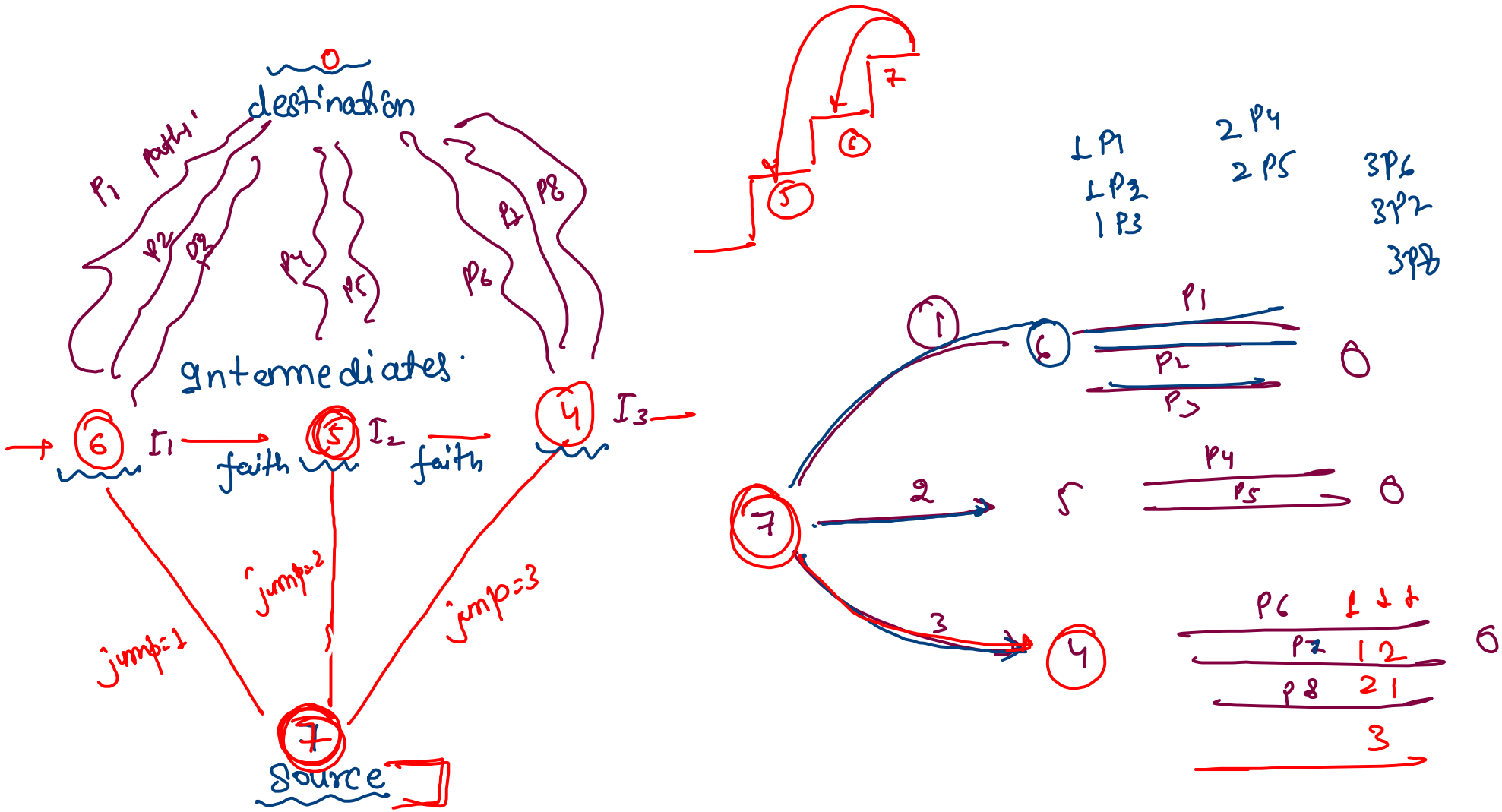
getStairPath (7) → all paths

faith ⇒ getStairPath(6) → p_1

p_2 " (5)

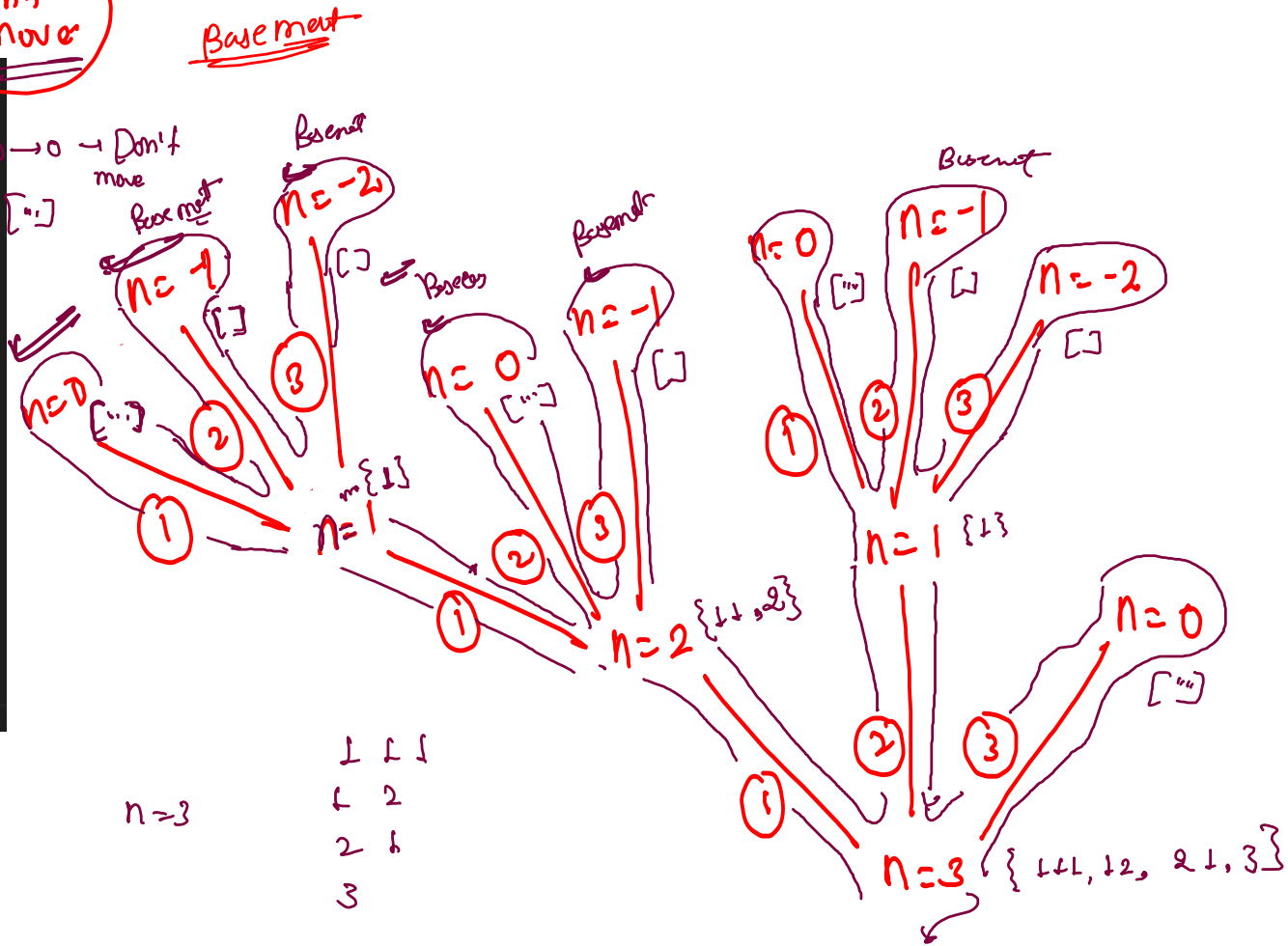
p_3 " (4)




0
destination
P1 path1
P2
P3
P4
P5
P7
P8
P6
intermediates
6
I1
faith
5
I2
faith
4
I3
jump=1
jump=2
jump=3
7
source
7
6
5
1 P1
1 P2
1 P3
2 P4
2 P5
3 P6
3 P7
3 P8
1
6
P1
P2
P3
0
7
2
5
P4
P5
0
3
4
P6 1 1 1
P7 1 2
P8 2 1
3
0


```java
public static ArrayList<String> getStairPaths(int n) {

    // jump allowed = 3
    ArrayList<String> rres1 = getStairPaths(n - 1);
    ArrayList<String> rres2 = getStairPaths(n - 2);
    ArrayList<String> rres3 = getStairPaths(n - 3);

    ArrayList<String> mres = new ArrayList<>();
    for(String s : rres1) {
        mres.add("1" + s);
    }

    for(String s : rres2) {
        mres.add("2" + s);
    }

    for(String s : rres3) {
        mres.add("3" + s);
    }
    return mres;
}
```

Recursion

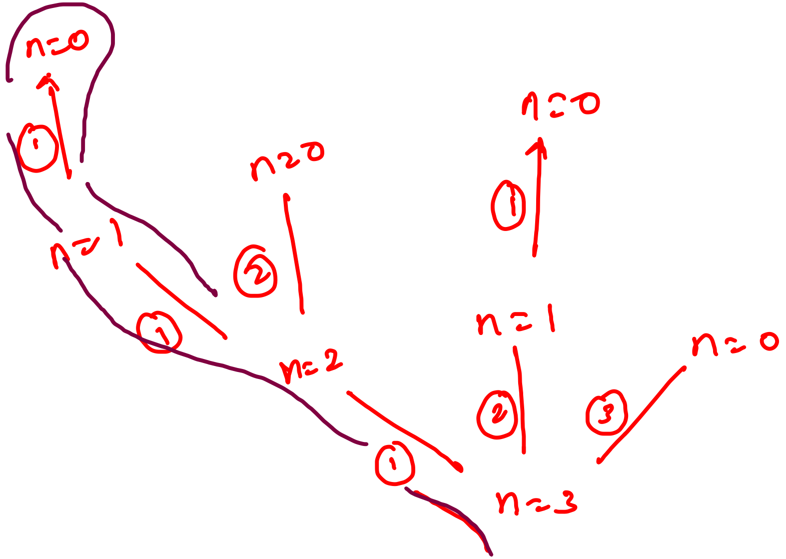


Calls → Base case.

✓ Smart → stupid

Stupid → smart

tree -



Base case - smart

if(n <= 0) {
if(n == 0)
br.add(n);
}

call(n-1)
call(n-2)
call(n-3)

} calls stupid