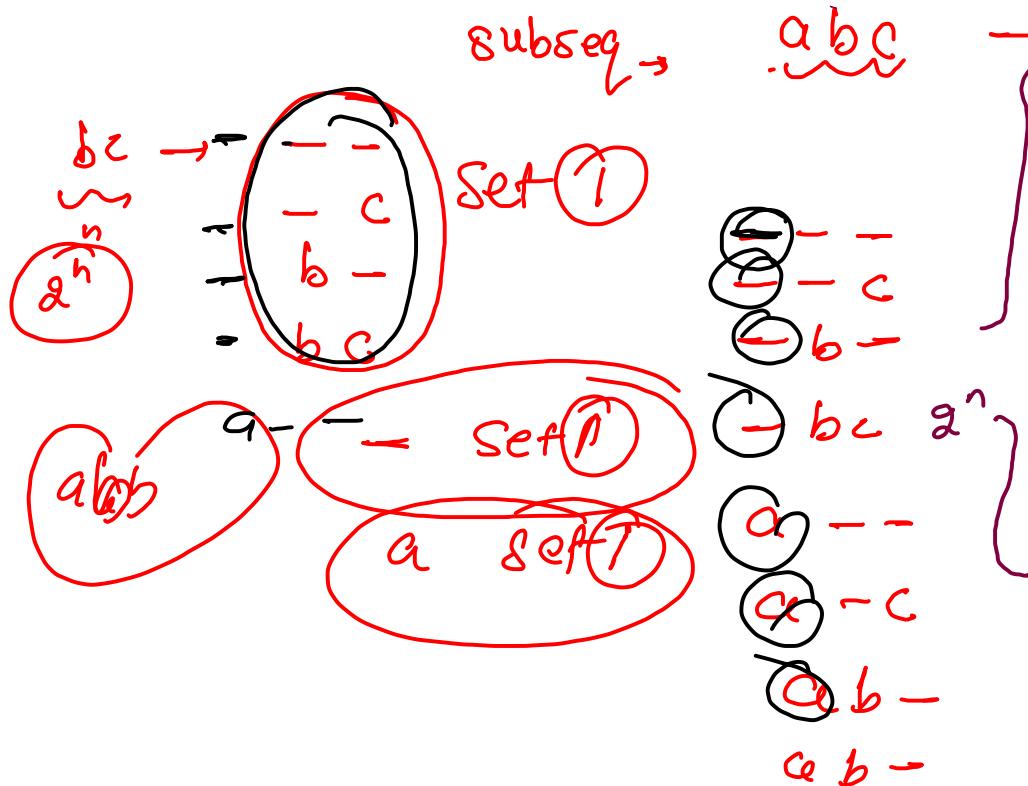


✓ Subseq

✓ Subset

✓ Substring

✓ Subarray.



0	\rightarrow	0	0	0	\rightarrow	-	-	-
1	\rightarrow	0	0	1	\rightarrow	-	-	c
2	\rightarrow	0	1	0	\rightarrow	-	b	-
3	\rightarrow	0	1	1	\rightarrow	-	b	c
4	\rightarrow	1	0	0	\rightarrow	a	-	-
5	\rightarrow	1	0	1	\rightarrow	a	-	c
6	\rightarrow	1	1	0	\rightarrow	a	b	-
7	\rightarrow	1	1	1	\rightarrow	a	b	c

getSubsequence

Expectation,

$\text{gss}(abc) \stackrel{\text{def}}{=} \{ \text{"---", "-c", "-b-", "-bc", } \\ \text{"a--", "a-c", "ab-", "abc"} \}$

faith \rightarrow

$\text{gss}(bc) \stackrel{\text{def}}{=} \{ \text{"--", "-c", "b-", "bc"} \}$

Merging of \rightarrow

faith &

Expectation

qstring \rightarrow abc, subString

rest of
qstring \rightarrow bc

$\text{gss}(abc) \stackrel{\text{def}}{=} \{ \text{...} \}$

Recursion
 $\text{res} = \text{gss}(bc); \{ \text{...} \}$

subset of bc

memo { ---, -c, -b-, -bc, }
 memo { a-- , a-c, ab-, abc }

qstring.substring(L);

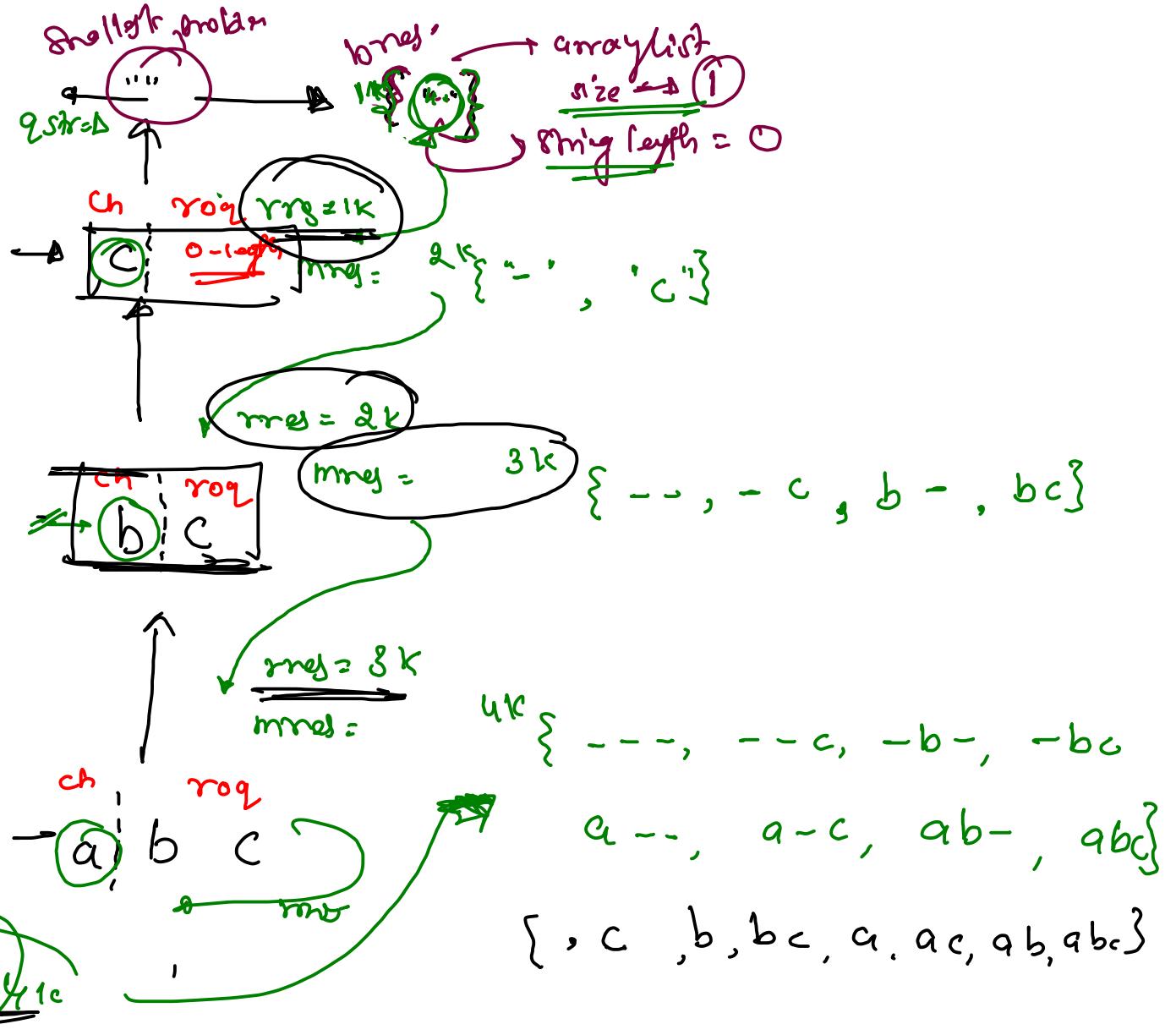
```

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;

```

string of length

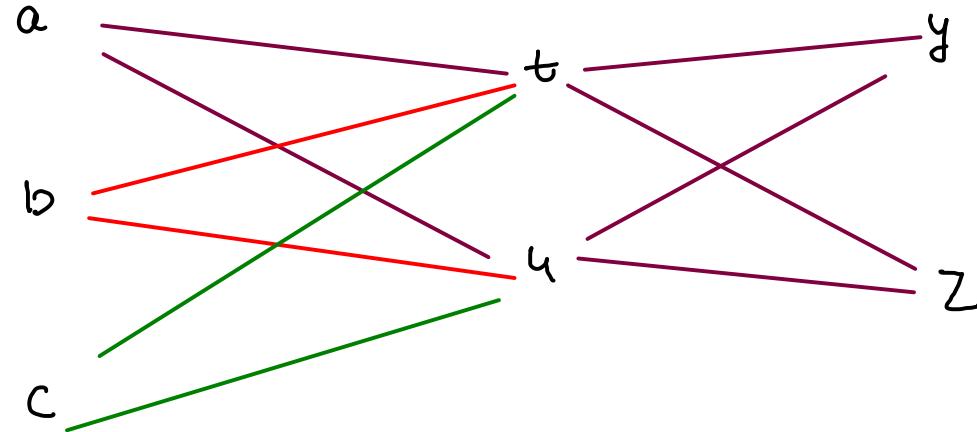


get kpc (Keypad Code)

get all
possibilities

array of String -

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



$$3 \times 2 \times 2 = 12$$

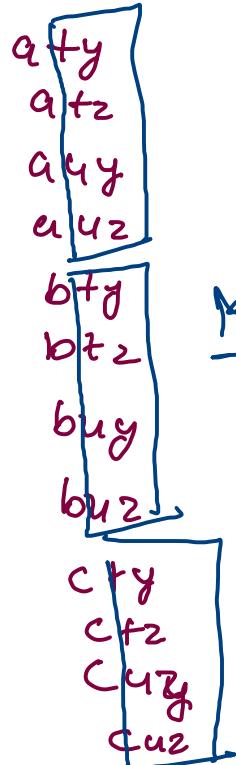
Ch : '1' → characters → tw → strings

$\left\{ \begin{array}{l} \text{aty} \\ \text{atz} \\ \text{auy} \\ \text{auz} \end{array} \right.$
 $\left\{ \begin{array}{l} \text{bt}y \\ \text{bt}z \\ \text{bu}y \\ \text{bu}z \end{array} \right.$
 $\left\{ \begin{array}{l} \text{ct}y \\ \text{ct}z \\ \text{cu}y \\ \text{cu}z \end{array} \right.$

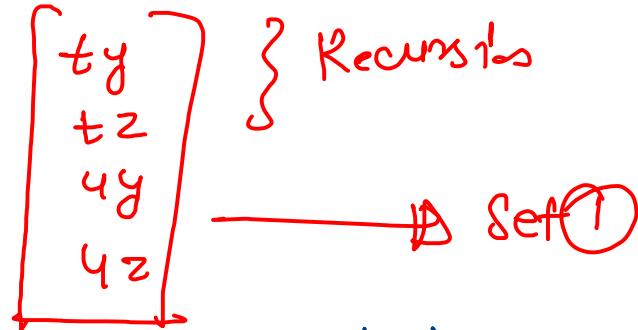
get kpc

$I \rightarrow abz$, $T \rightarrow t^4$, $S \rightarrow yz$.

Expectation



$\text{get kpc} (\sqcup \text{fg})$
faith (fg) \rightarrow Recurs¹s

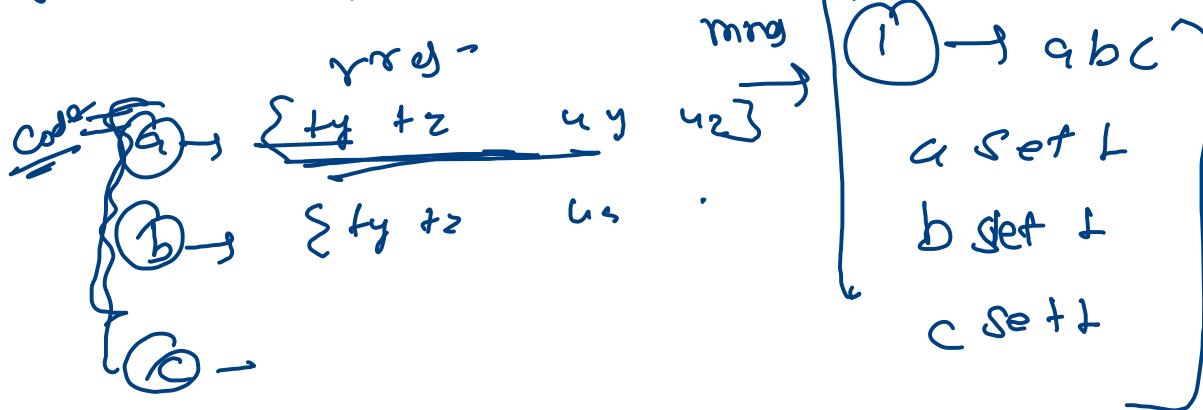


Set 1

Merging - - -

$\text{get kpc} (179)$

$\xrightarrow{\text{rel}}$ set 1 = $\text{get kpc} (\text{fg})$:



$1 \rightarrow abc$
 $7 \rightarrow tu$
 $9 \rightarrow yz$

```

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;
}

```

$8 \text{th.} \logPA() = 0$

 The diagram shows a state transition. An arrow points from an initial state (empty circle) to a state containing '...'. From this state, another arrow points to a string of size 1, labeled 'size=1'. A red annotation indicates 'String.length() = 0'.

The diagram illustrates the recursive generation of KPC for 'yz'. It starts with 'yz' at level 1, which branches into 'g' and '..'. 'g' leads to 'g|..', which then leads to 'g|.. mres'. '.. mres' leads to 'yz, z'. This is labeled 'rres=1k' and 'mres' at level 2. At level 3, 'yz, z' leads to 'yz, z|..', which then leads to 'yz, z|.. mres'. This is labeled 'rres=2k' and 'mres' at level 3. Finally, 'yz, z|.. mres' leads to 'yz, z|.. mres' at level 4, which is labeled 'rres=3k' and 'mres' at level 4.

'0' → 48

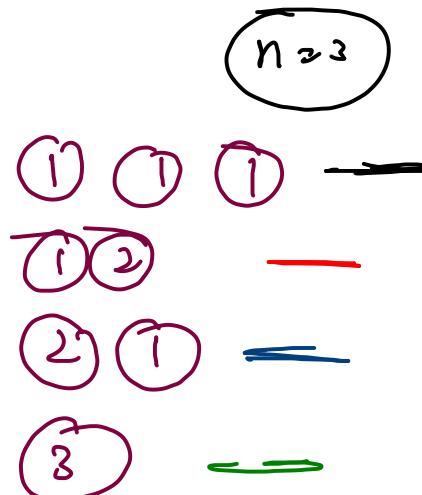
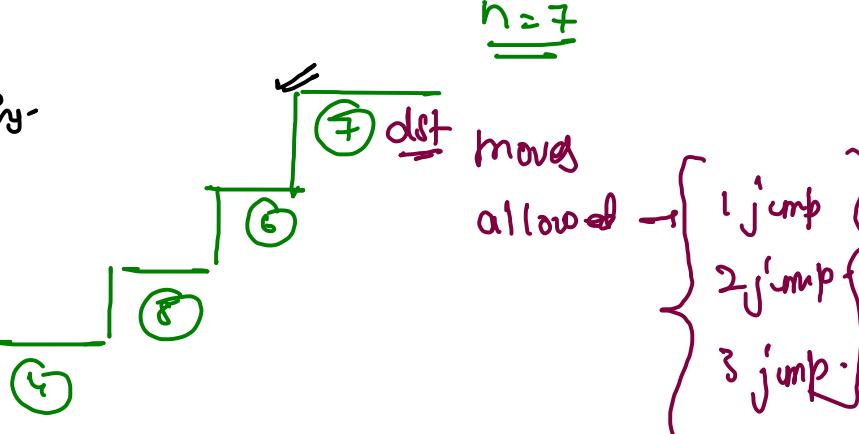
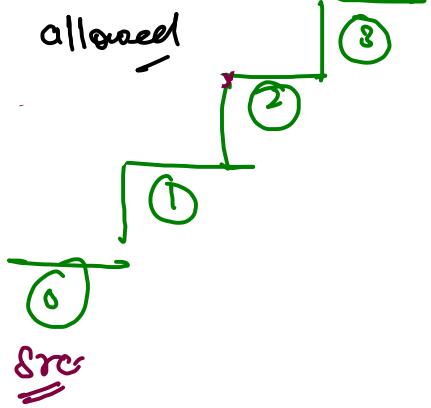
'7' → 55 - 48

int ^{int} (7) - 48 = ①
 $(7 - 48)$
 $\leftarrow 7 - 0$

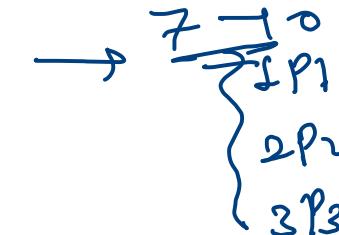
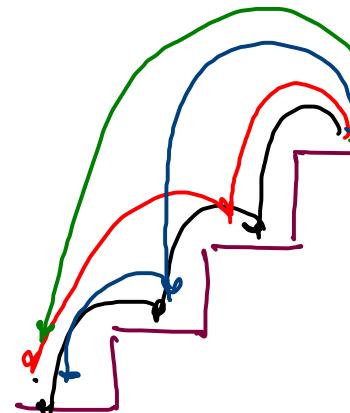
The diagram illustrates the recursive generation of KPC for 'abc'. It starts with 'abc' at level 1, which branches into 'a', 'b', and 'c'. 'a' leads to 'a|bc', which then leads to 'a|bc|..'. 'b' leads to 'b|bc', which then leads to 'b|bc|..'. 'c' leads to 'c|bc', which then leads to 'c|bc|..'. This is labeled 'rres=1k' and 'mres' at level 2. At level 3, each path leads to a string of length 2, such as 'atc', 'atc|..', 'actc', etc. This is labeled 'rres=2k' and 'mres' at level 3. Finally, at level 4, each path leads to a string of length 3, such as 'atc', 'atc|..', 'actc', etc. This is labeled 'rres=3k' and 'mres' at level 4.

get Stair Path -

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



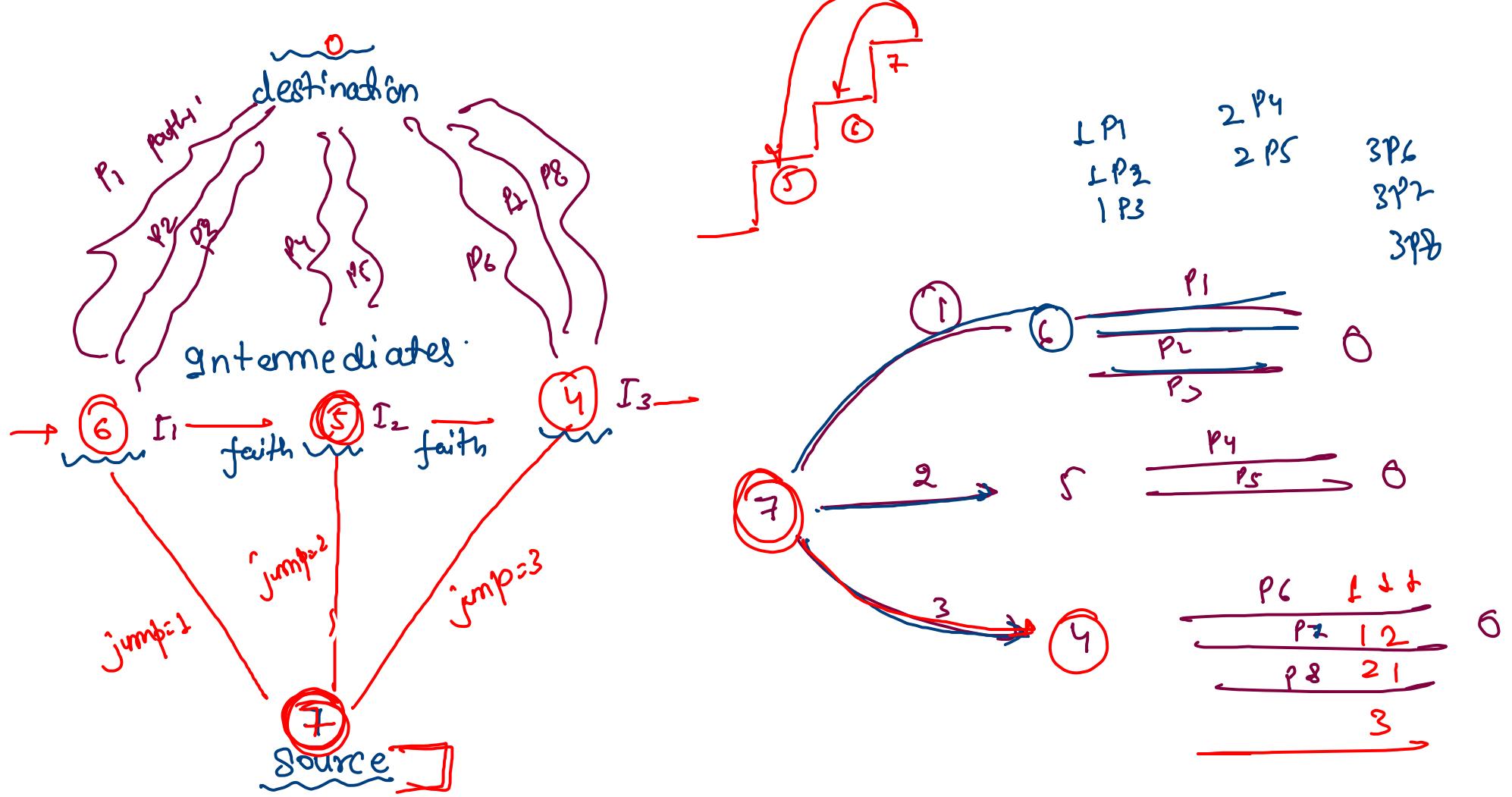
{↑↑↑, ↑↓, ↓↑, ↓↓}



Expectation

getstairpath(7)
call path

father P_1 getstairpath(6)
 P_2 " (5)
 P_3 " (4)



```

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);

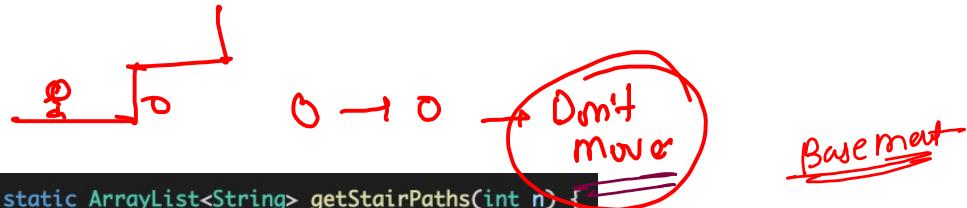
    ArrayList<String> mres = new ArrayList<>();
    for(String s : rres1) {
        mres.add("1" + s);
    }

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

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

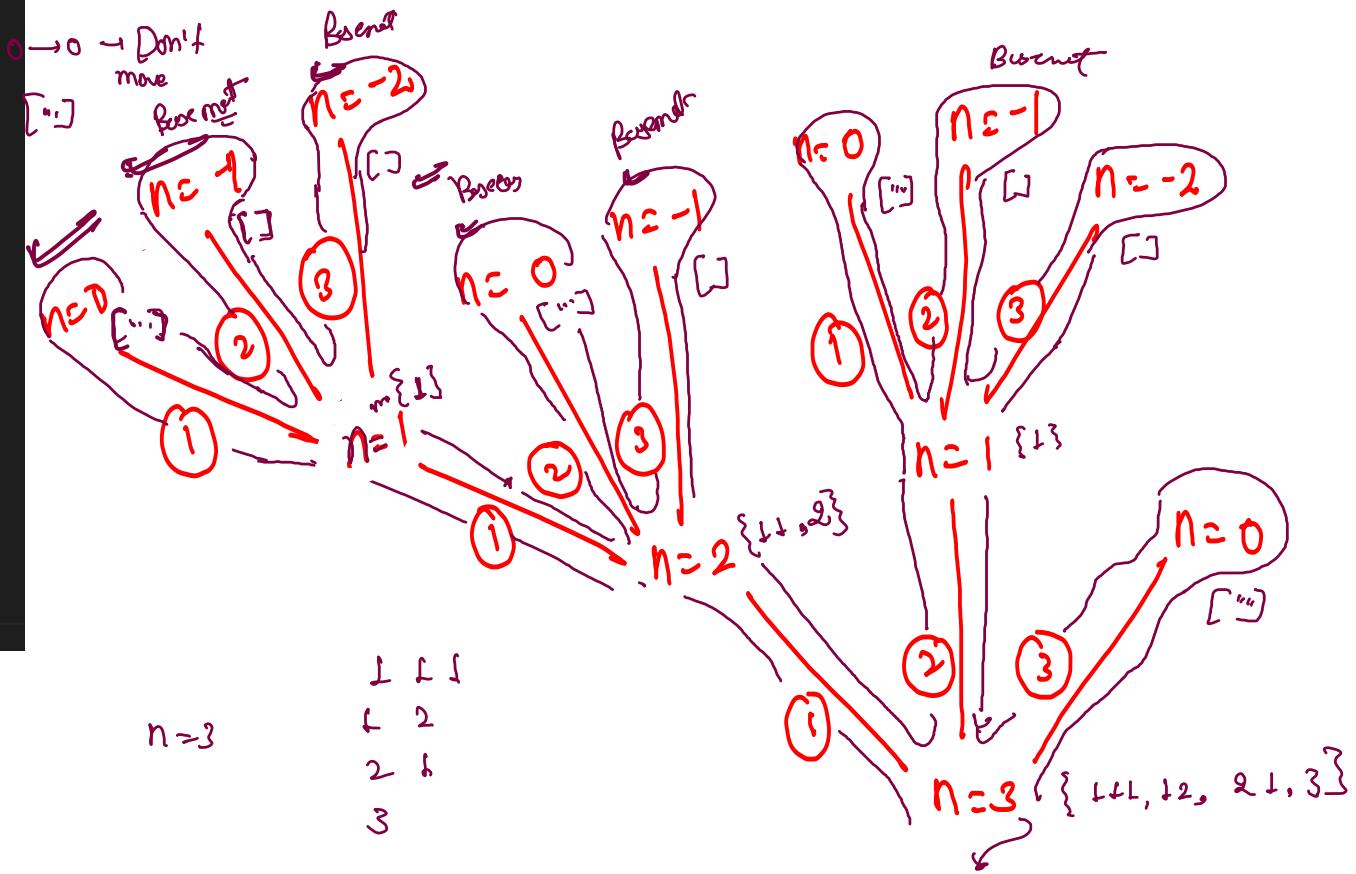
```

Recursion



$n=3$

$\begin{matrix} L & L & L \\ L & & 2 \\ 2 & & 1 \\ 3 & & \end{matrix}$



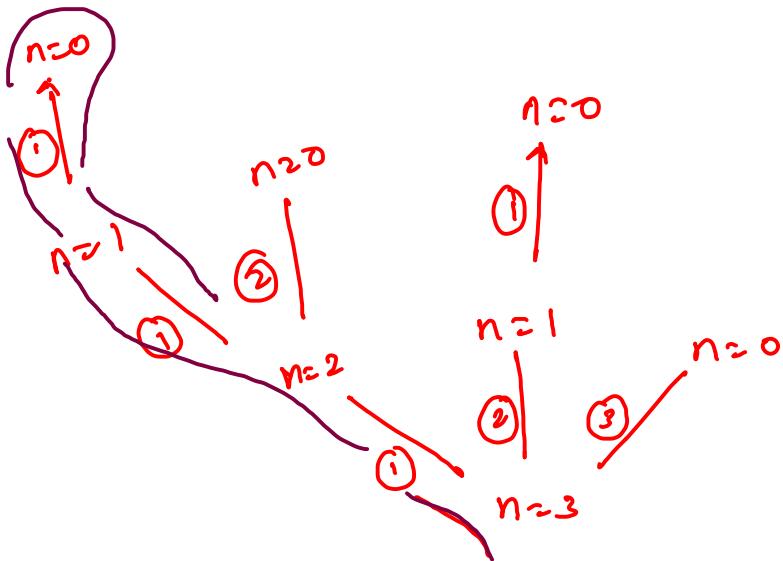
Calls

Base case.

✓ Smart → Stupid

Stupid → Smart

tree -



base or c - smart
if($n \leq 0$)
if($n \geq 0$)
br or .call(n)

↙
{ call($n-1$)
call($n-2$)
call($n-3$) } } call
stupid