Recursion with arraylist

~~1~~	Get Subseq.	→	Print Subseq.
~~2~~	Get KPC	→	Print KPC
~~3~~	Get Maze path		
~~4~~	Get mazepath with jump.		
~~5~~	Stair paths.	→	Print stair paths.

Print

Signature freeze

ArrayList

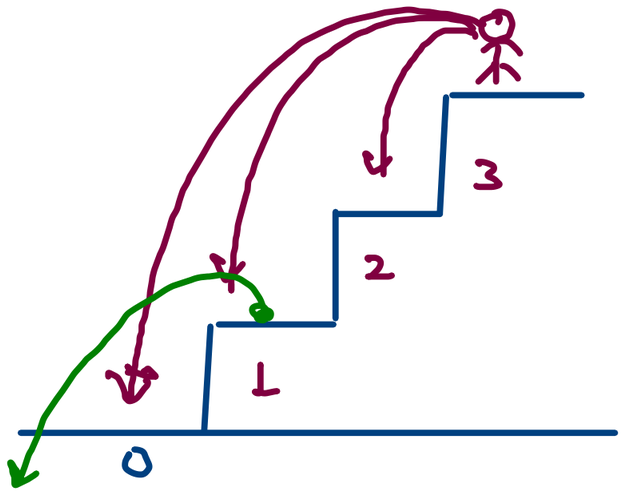
Stair path: (print)

→ answer so far

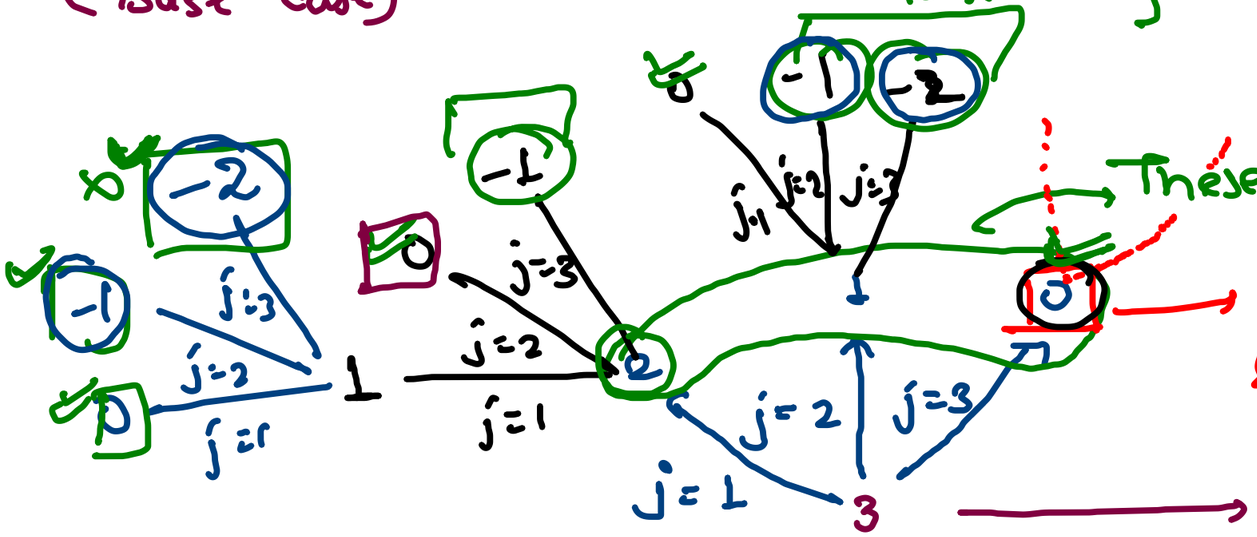
→ level & options.

✓ Option → 'jumps' → calls (calls)

✓ level → Stairs → Basecalls (Base case)



Base case → Not valid answer



Recursion

① calls → Smart
Base case → fool

② calls → fool
Base case → smart

③ → calls → fool } Error
→ Base → fool }

Error

fool - calls , smart Base case

```
public static void printSP1(int n, String asf) {
    // Smart Base case, why? because it can manag
    System.out.println(n);
    if (n <= 0) {
        // if (n == 0)
        //     System.out.println(asf);
        return;
    }
    printSP1(n - 1, asf + "1");
    printSP1(n - 2, asf + "2");
    printSP1(n - 3, asf + "3");
}
```

ANSWER.

1 1 1
1 2
2 1
3

n.

3 2 1 0 -1 -2 0 -1

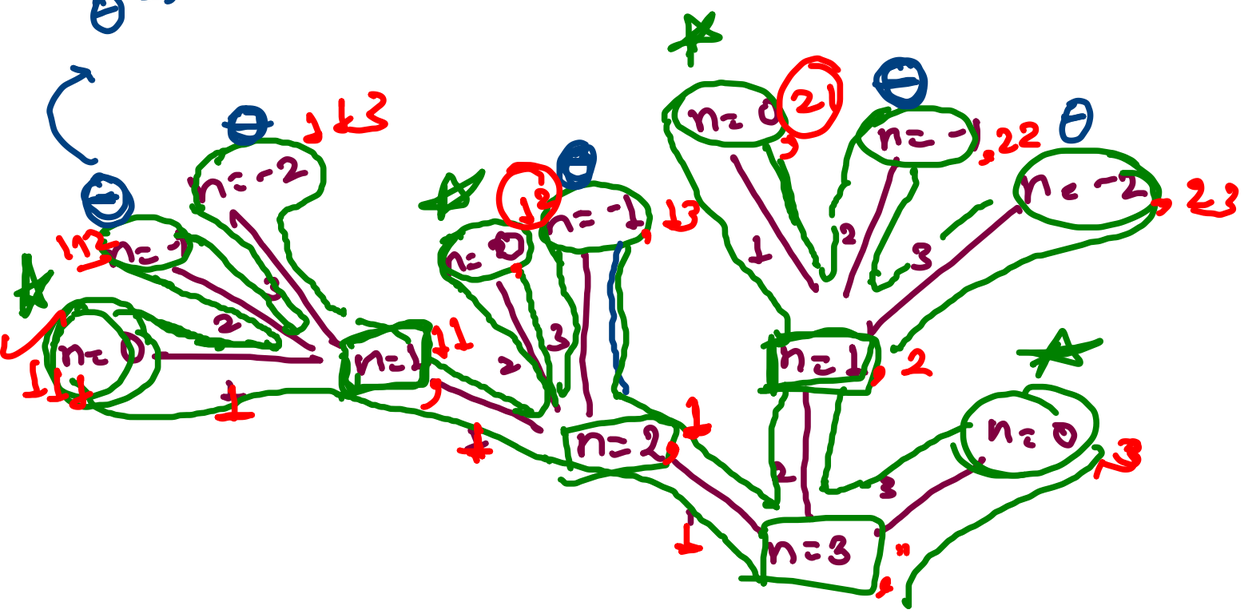
1 0 -1 -2 0

3 2 1 0 -1 -2 0 -1 1 2 0

=====

3 2 1 0 0 -1 0 0

Base most



DENSE RECURSIVE TREE

Smart -calls fool- Base case.

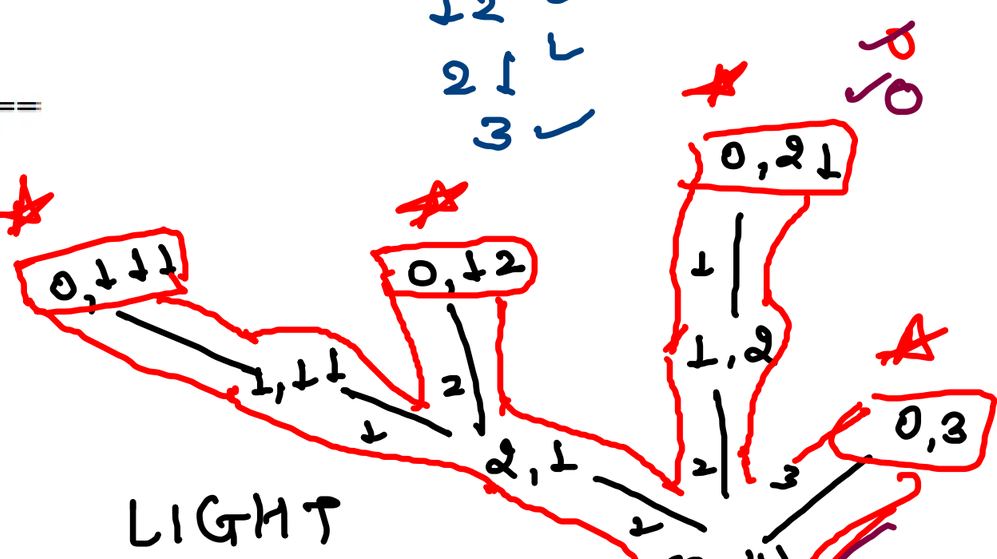
```
public static void printSP2(int n, String asf) {
    System.out.println(n);
    // fool base case
    if (n == 0) {
        // System.out.println(asf);
        return;
    }

    // smart calls
    if (n - 1 >= 0)
        printSP2(n - 1, asf + "1");
    if (n - 2 >= 0)
        printSP2(n - 2, asf + "2");
    if (n - 3 >= 0)
        printSP2(n - 3, asf + "3");
}
```

1 1 1
1 2
2 1
3

3 2 1 0 -1 -2 0 -1 1 2 0

=====

3 2 1 0 0 -1 0 0



LIGHT RECURSIVE TREE

Better - Slightly

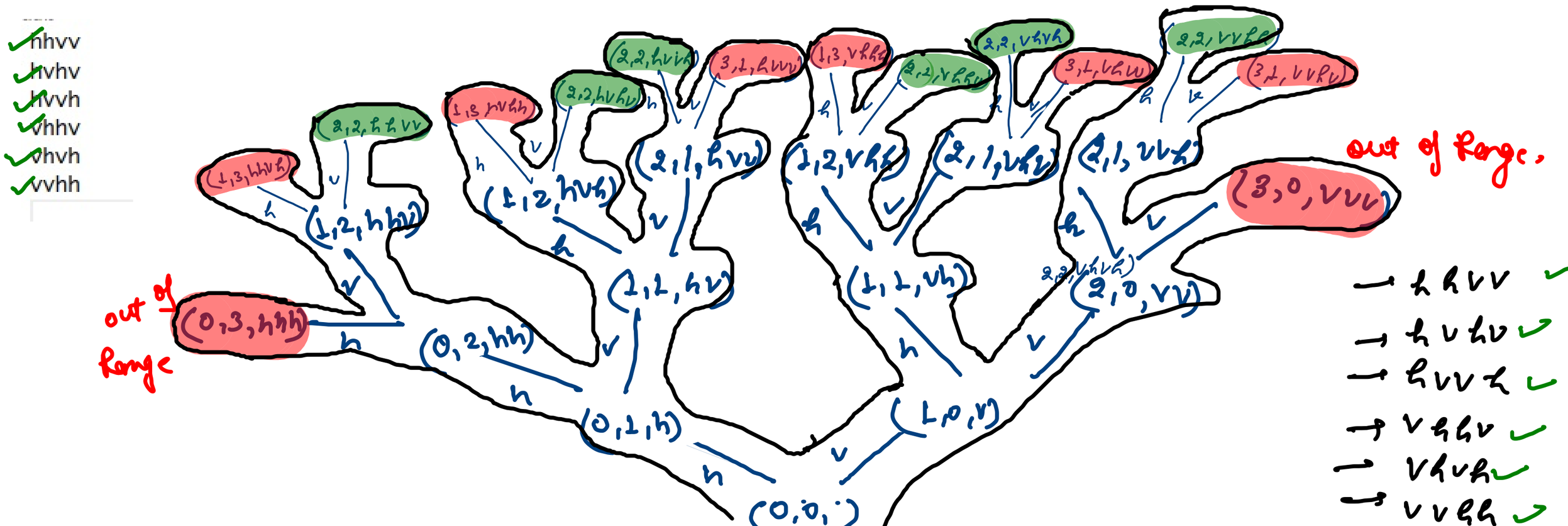
print Maze path →

x,y —h→ initial → 0,0 → source.
x,y —v→ dst → n-1, m-1
path so far.

options → horizontal, vertical } Responsible for all
level → coordinates. } Responsible for base case.

x,y —horizontal→ (x, y+1)
x,y —vertical→ (x+1, y)

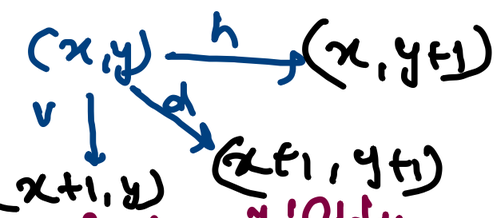
DRY RUN → Base case - smart calls → stupid/ fool



Print Mazepath with Jumps

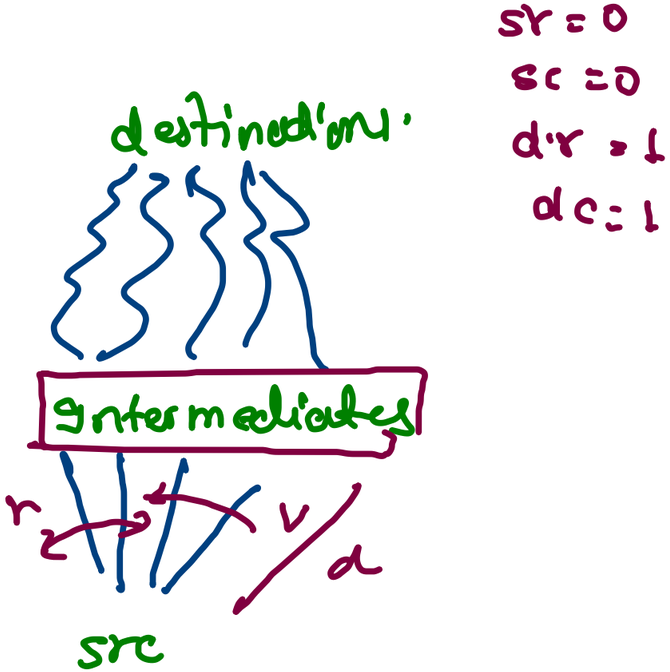
Devel → Coordinates is [levels] helpful for base case

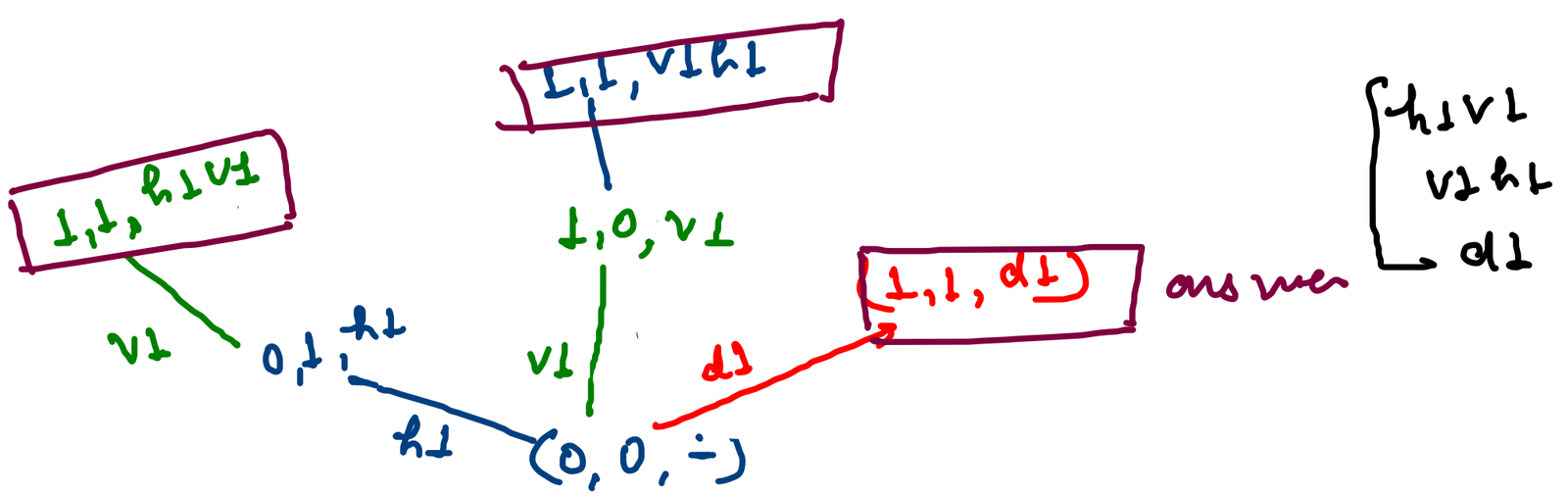
opts → horizontal valid calls + vertical valid calls + Diagonal valid calls.] calls Responsibility.



DRY RUN for Get →

```
for(int jump = 1; jump+sr <= dr; jump++)
for(int jump=1; jump+sc <= dc; jump++)
for(int jump=1; jump+sr <= dr && jump+sc<=dc; jump++)
```



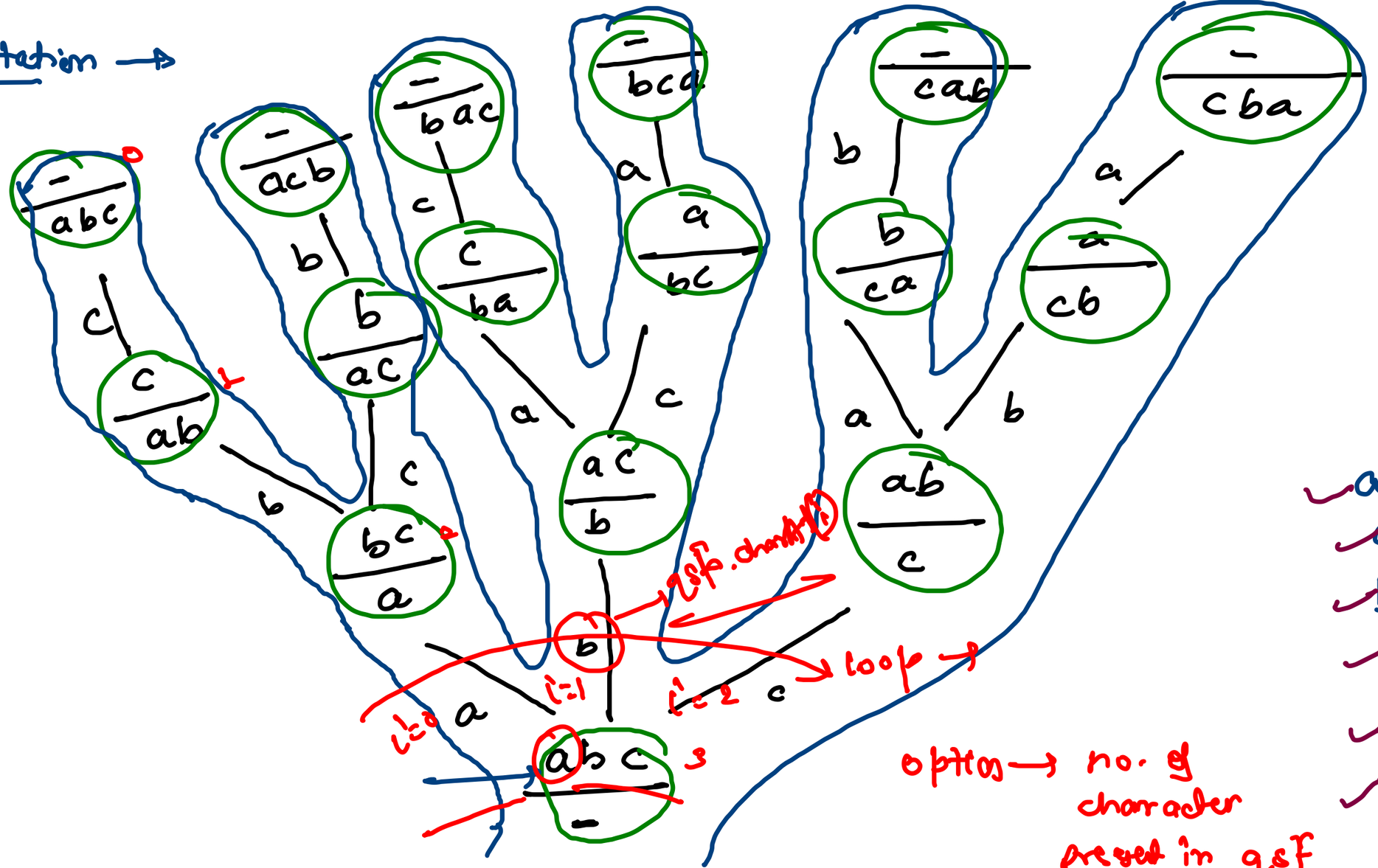


print per mutation →

abc →

all possible arrangemet

✓abc
✓acb
✓bac
✓bca
✓cab
✓cba



✓abc
✓acb
✓bac
✓bca
✓cab
✓cba

option → no. of character present in qsf

level => depend on length of qsf

print Encodings →

6 5 5 1 9 6

a → 1
b → 2
c → 3
d → 4
e → 5
f → 6
g → 7
h → 8
i → 9
j → 10
k → 11
l → 12
m → 13
n → 14
o → 15
p → 16
q → 17
r → 18
s → 19
t → 20
u → 21
v → 22
w → 23
x → 24
y → 25
z → 26