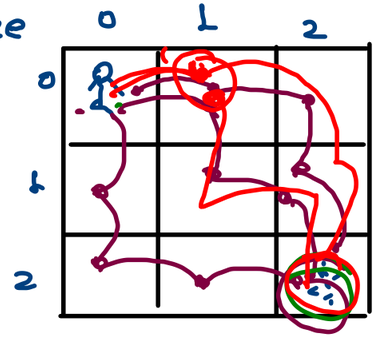
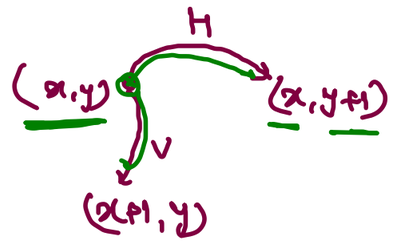
get maze path →



src - x = 0
src - y = 0

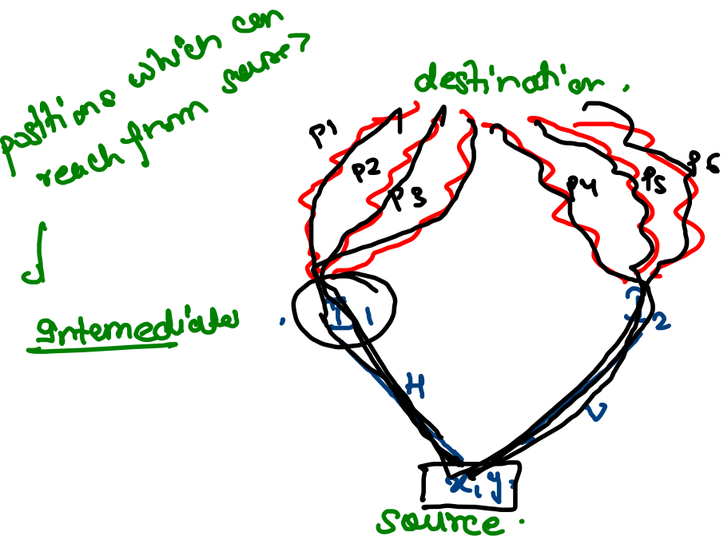
dst x = 2
dst y = 2

steps allowed



Arraylist → [HHVV, VVHH, HVHV ----- all paths] return.

positions which can reach from source



intermediate

(I1 + I2) → Destination → 6

source to destination = 3 from I1 + 3 from I2

Total = 6

[P4, P5, P1] Path from faith.

Path from faith [P1, P2, P3]. faith by Recurs

faith by Recurs

(x, y+1) (x+1, y)

H V

[HP1, HP2, HP3] [VP4, VP5, VP6]

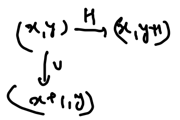
x, y

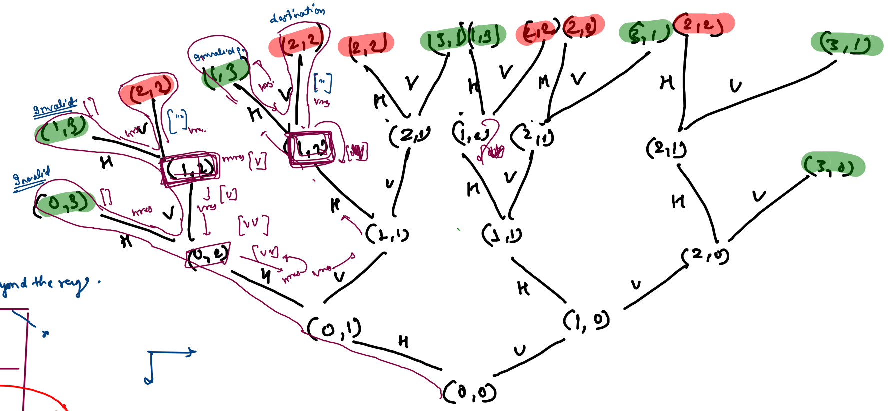
```java
public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc
    // answer
    if (sr == dr && sc == dc) {
        ArrayList<String> bres = new ArrayList<>();
        bres.add(""); // don't move path
        return bres;
    }
    // invalid
    if (sr > dr || sc > dc) {
        ArrayList<String> bres = new ArrayList<>();
        return bres;
    }

    // get path after horizontal jump
    ArrayList<String> hres = getMazePaths(sr, sc + 1, dr, dc);
    ArrayList<String> vres = getMazePaths(sr + 1, sc, dr, dc);

    ArrayList<String> mres = new ArrayList<>();
    // add path of horizontal jump
    for (String path : hres) {
        mres.add("h" + path);
    }
    // add path of vertical jump
    for (String path : vres) {
        mres.add("v" + path);
    }

    return mres;
}
```

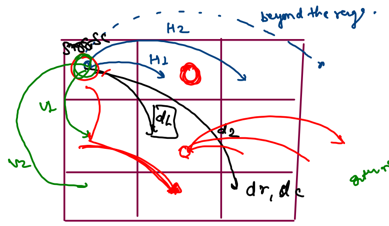
Base → Invalid

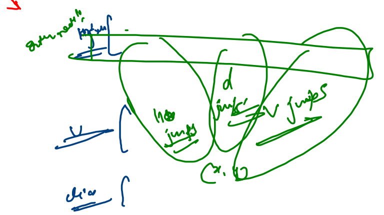
n=3
m=3

sr=0 sc=0
dr=2 dc=2









level → base case → smallest of the Input
options → calls → Include valid + Invalids
no. of calls

print Subsequence

```java
ArrayList<String> mres = new ArrayList<>();
// Horizontal Jumps
for (int jump = 1; jump + y <= dc; jump++) {
    ArrayList<String> hres = getMazePathsWithJump(x, y + jump, dr, dc);
    for (String path : hres) {
        mres.add("h" + jump + path);
    }
}
// Vertical
for (int jump = 1; jump + x <= dr; jump++) {
    ArrayList<String> vres = getMazePathsWithJump(x + jump, y, dr, dc);
    for (String path : vres) {
        mres.add("v" + jump + path);
    }
}

// Diagonal
for (int jump = 1; jump + x <= dr && jump + y <= dc; jump++) {
    ArrayList<String> dres = getMazePathsWithJump(x + jump, y + jump, dr, dc);
    for (String path : dres) {
        mres.add("d" + jump + path);
    }
}
```

dr=1
dc=1