Flood Fill →

main

mark - block
unblock → unmark

0 → visitable
1 - obstacle/hurdle

Base case
→ print

mark

top valid(rr, cc)
board[rr][cc] != 1

lef

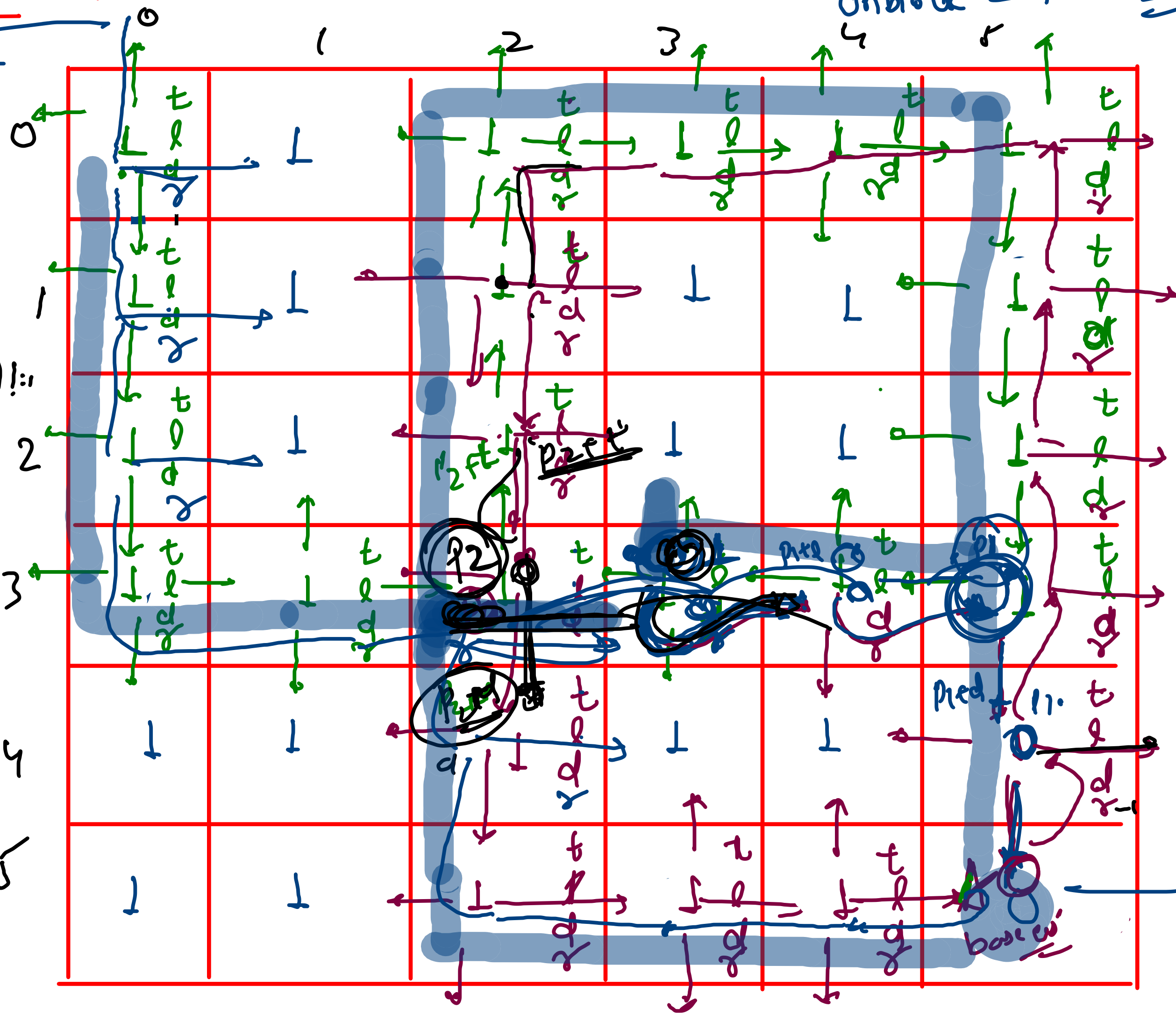
down

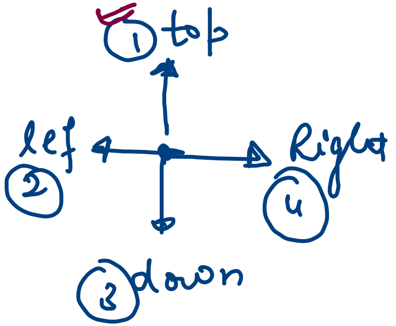
rig

dddrrtttrr

ddddd

dddrrdddrrr





print all path from which we can reach from top left to bottom right

print right

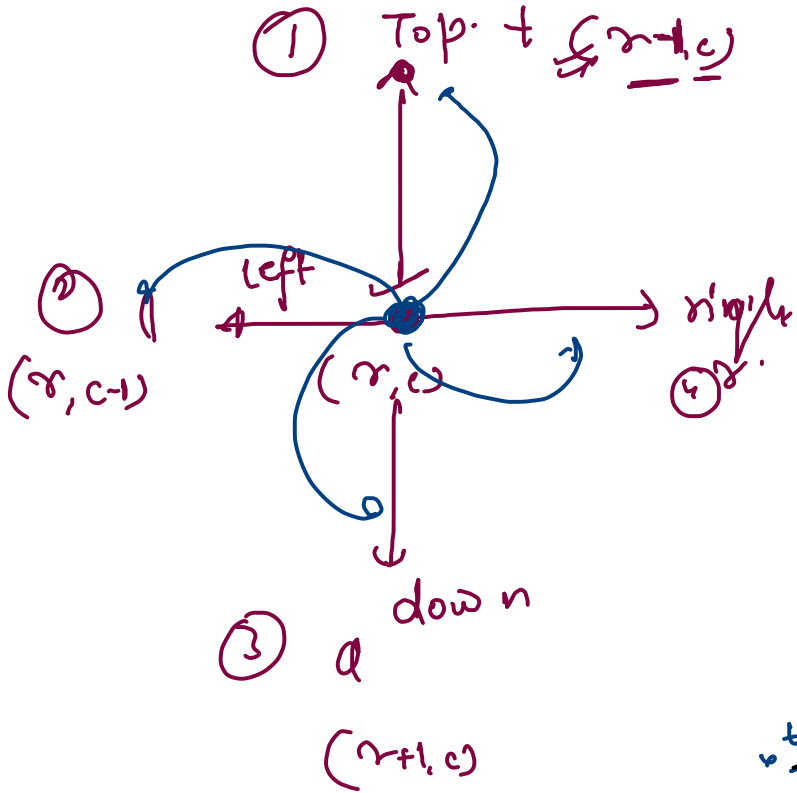
rdir = {-1, 0, 1, 0}

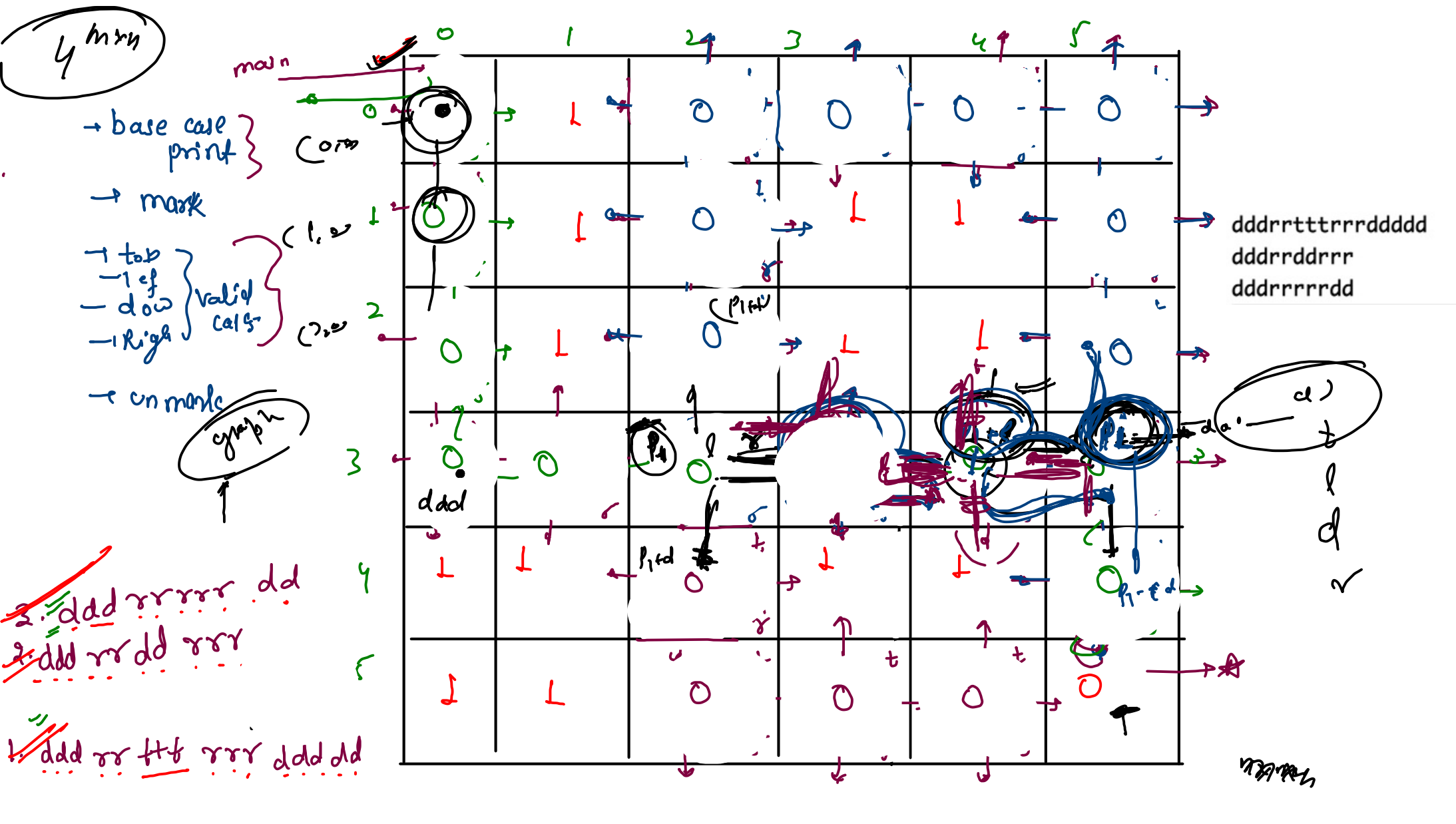
cdir = {0, -1, 0, 1}

```java
public static void floodfill(int[][] board, int sr, int sc, String asf) {
    if(sr == board.length - 1 && sc == board[0].length - 1) {
        System.out.println(asf);
        return;
    }

    for(int d = 0; d < rdir.length; d++) {
        int rr = sr + rdir[d];
        int cc = sc + cdir[d];
        char dir = chArr[d];

        if(rr >= 0 && rr < board.length && cc >= 0 && cc < board[0].length) {
            floodfill(board, rr, cc, asf + dir);
        }
    }
}
```




4 m x n
main
→ base case print
→ mark
→ top
→ left
→ down
→ Right
valid calls
→ unmark
graph
dddrrtttrrrddddd
dddrrddrrr
dddrrrrrdd

Target Sum Subset

arr = { 1, 4, 2, 3, 0 }

target = 5.

generate → all Subset

↳ take contribution from all subset

Check in the base cas. Sum == ter

arr → {2, 1, 3, 0}

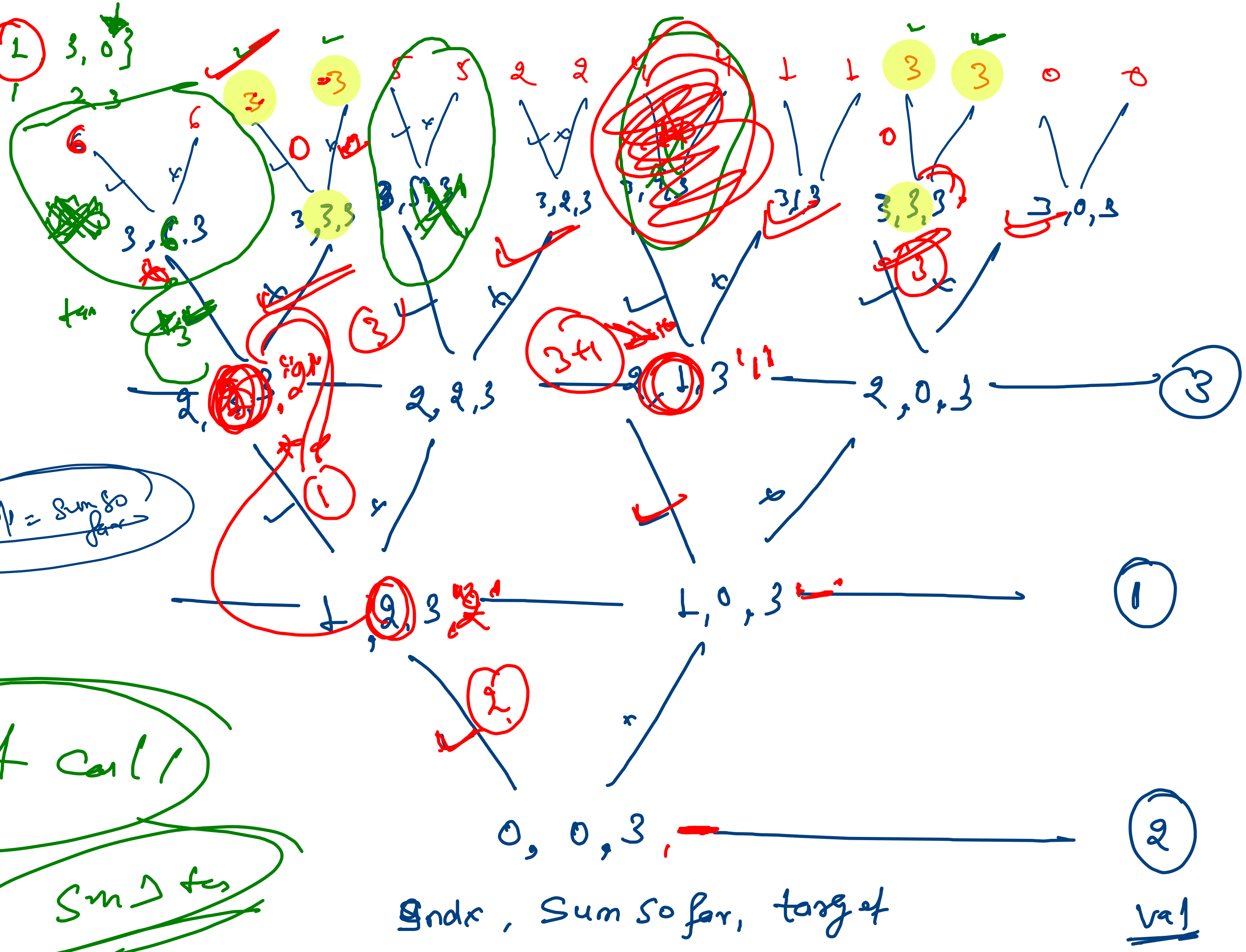
target = 3

level → index

option → Yes / No

target = 3

tot = sum so far



Smart Call

Sum fer