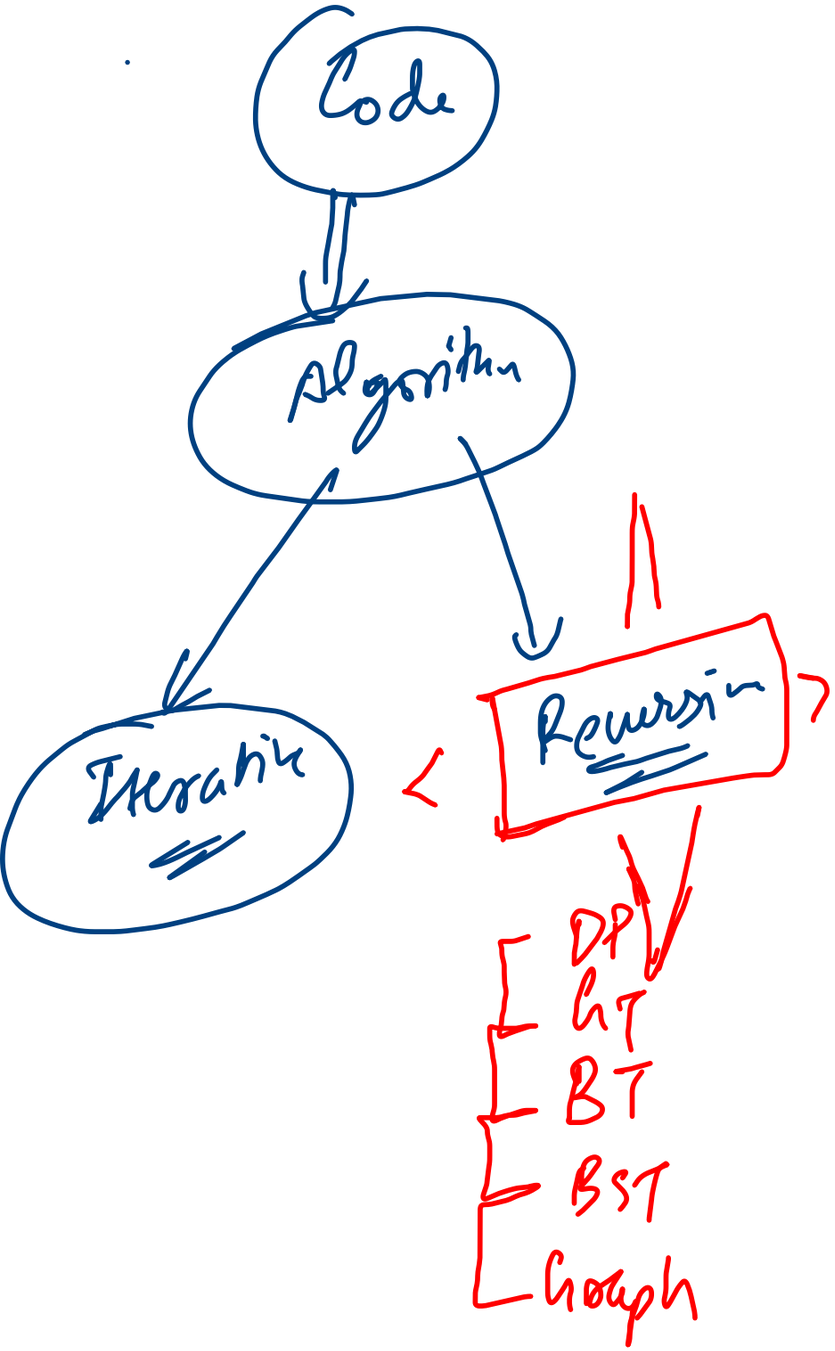
String, SB & ArrayList → Revision

Tues → 03:00 – 03:30 → string basics

Thursday → String, SB & ArrayList → Extra days (03:30 – 05:30)

Recursion in Mobile


Code
Algorithm
Iterative
Recursion
DP
GT
BT
BST
Graph

Recursion

```
public static returntype fun ( ) {

    ____________________________
    ____________________________
    ____________________________

    fun();

    ____________________________
    ____________________________
    ____________________________

}
```

Input
5

Output
5
4
3
2
1

Expectation (n): Given a number n, fun is expected to print decreasing sequence.

faith (n-1): for num "n-1" fun will print decreasing sequence

Expectation

fun(5) → 5
4
3
2
1

faith

fun(4) → 4
3
2
1

fun(5) → print(5)
fun(4)

fun(n) → print(n)
fun(n-1)

80%

Generalized code

HLC

A → B → C

0 (5) (5)

A → B → (C)

1 (5) (4)

Back

limiting point / Base Case

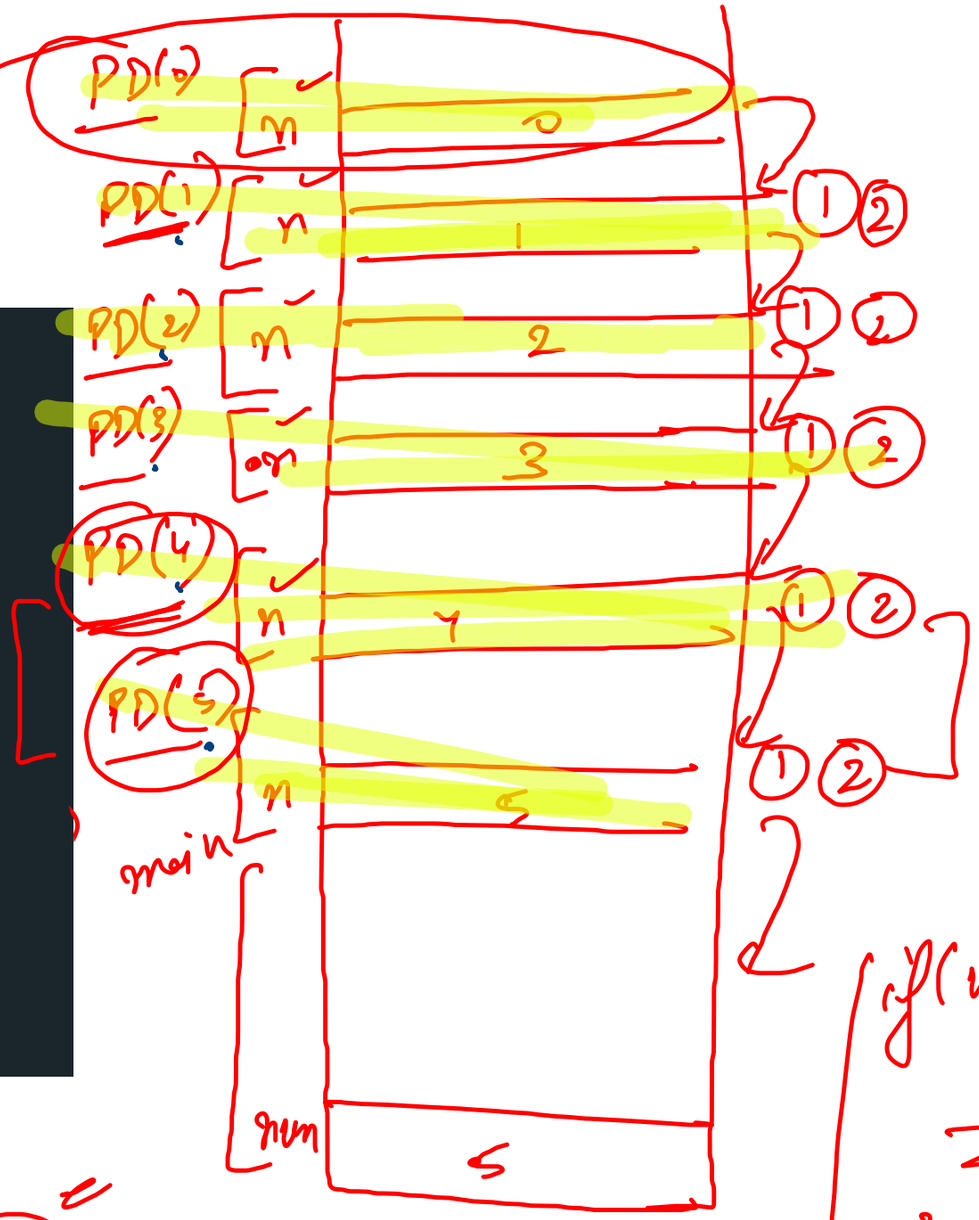
Dry run

1. return [forcefull]
2. nothing to execute

```java
public static void main(String[] args){
    Scanner scn = new Scanner(System.in);

    int num = scn.nextInt();

    printDecreasing(num); // expectation
}

public static void printDecreasing(int n){
    System.out.println(n);      1
    printDecreasing(n-1);       2
}
```

```java
if(n == 0){
    return;      1
}
```

A → B → C → D



5
4
3
2
1

if(n > 0) {
}
else { return; }

5 →

1
2
3
4
5

80%

Expectation

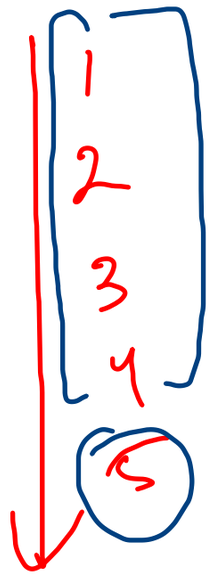
fun(5) → 1 2 3 4 5

fun(5) ⇒ fun(4)
print(5)

fun(n) ⇒ fun(n-1)
print(n)

fun(4) = 1 2 3 4

A ⇄ B



if(n==0){
return;
}

```
public static void printIncreasing(int n){
    printIncreasing(n - 1);
    System.out.println(n);
}
```

3

Sample Output

```
3
2
1
1
2
3
```

fun(3) = 3

2
1
1
2

3

fun(2) ⇒ 2
1
1
2

n-1
↑
n

Expectation = faith + my work

fun(3) = print(3)
fun(2)
print(3)

fun(n) = print(n)
fun(n-1)
print(n)

recursion
↳ Assumption of being

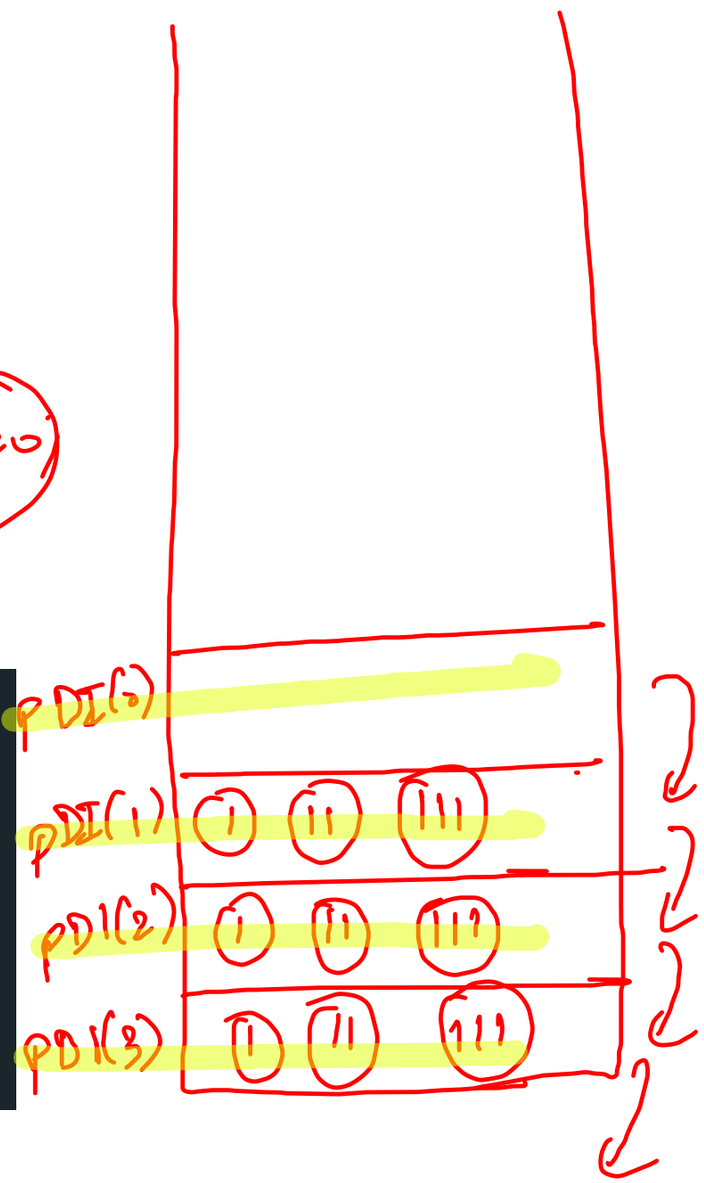
3
2
1
1
2
3

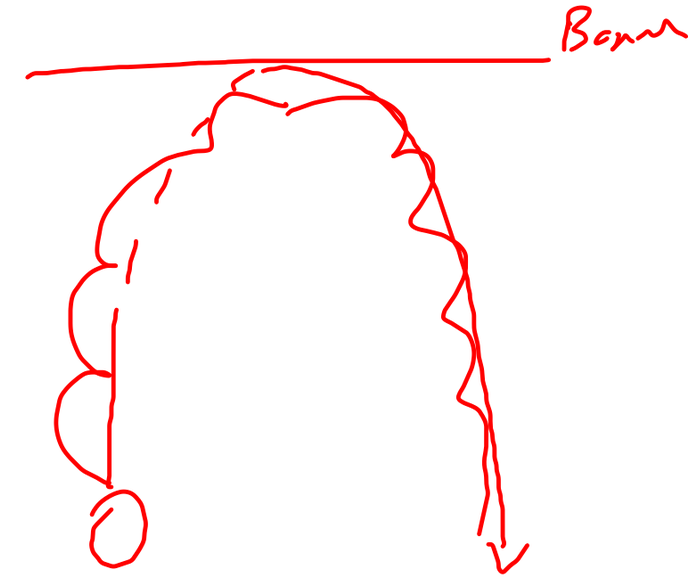
if(n==0){
return;
}

return;

return 20;

```
public static void pdi(int n){
    System.out.println(n);
    pdi(n-1);
    System.out.println(n);
}
```





n = 5

$$\boxed{5! = 5 \times 4!}$$

Expectation: $fact(5) = 5 \times 4 \times 3 \times 2 \times 1$

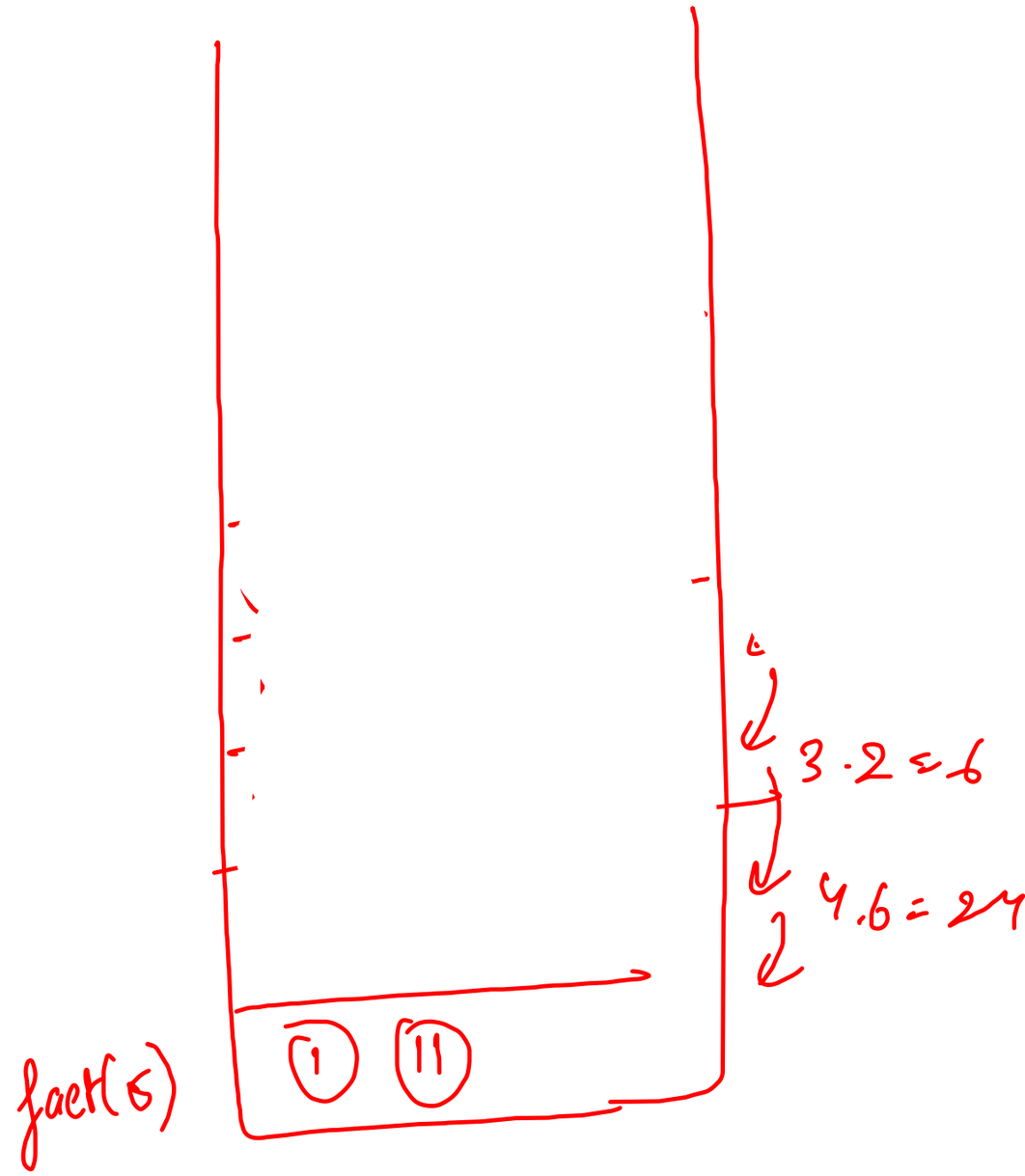
Faith: $fact(4) = 4 \times 3 \times 2 \times 1$

$$\boxed{fact(5) = 5 \times fact(4)}$$

$$\boxed{fact(n) = 5 \times fact(n-1)}$$

```java
if(n == 0){
    return 1;
}
```

```java
public static int factorial(int n){
    int factNm1 = factorial(n-1);
    int factN = n * factNm1;
    return factN;
}
```



2 → x

5 → n

Sample Ou

32

$pow(2, 5) \Rightarrow 2^5$

$\boxed{2^5} = 2 \cdot \boxed{2^4}$

$\boxed{pow(2,5) = 2 \cdot pow(2, 4)}$

$\boxed{pow(x, n) = x \cdot pow(x, n-1)}$

if(n==0){

$2^0 =$

2 5

```
public static int power(int x, int n){
    int xPowNm1 = power(x,n-1);
    int xPowN = x * xPowNm1;
    return xPowN;
}
```

i

ii

iii

pow(2,0)	x=2, n=0	
pow(2,1)	x=2, n=1	1
pow(2,2)	x=2, n=2	1
pow(2,3)	x=2, n=3	1
pow(2,4)	x=2, n=4	1
pow(2,5)	x=2, n=5	1

$$x^n = x \cdot x^{n-1}$$

if (n % 2 == 0) { even }

if (n % 2 == 1) { odd }

Implementation

try sol. → Doubt support

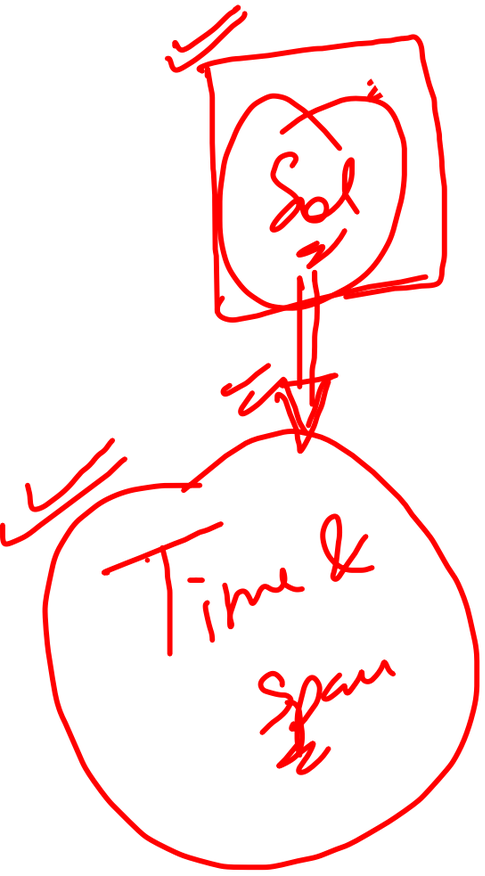
$$x^n = x^{n/2} \cdot x^{n/2}$$

$$2^4 = 2^2 \cdot 2^2$$

$$2^5 = 2^2 \cdot 2^2 \cdot 2$$

$$5/2 = 2$$

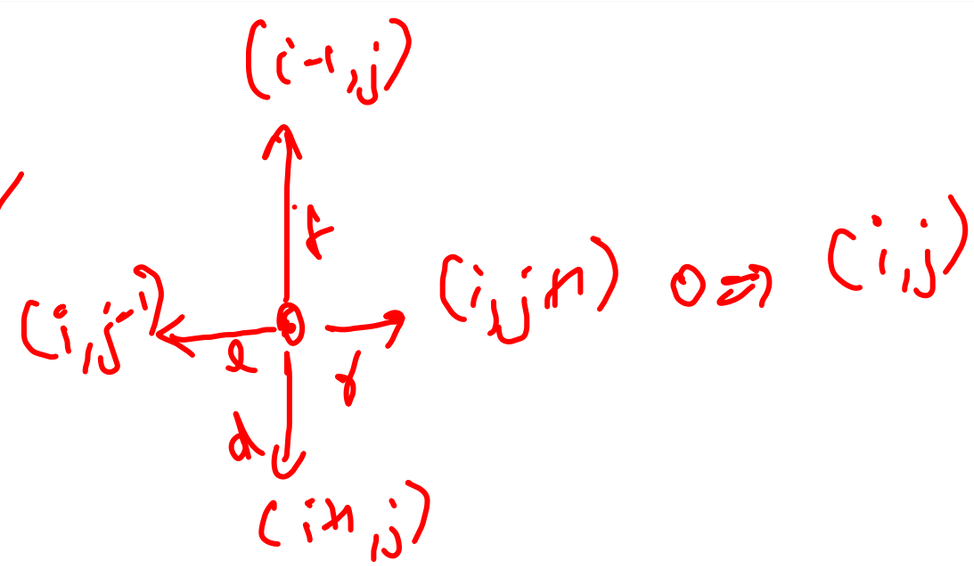
$$\text{Power}(x, n) = \begin{cases} \text{power}(x, n/2) \cdot \text{power}(x, n/2), & n == \text{even} \\ \text{power}(x, n/2) \cdot \text{power}(x, n/2) \cdot x, & n = \text{odd} \end{cases}$$


Sol
Time &
Span

1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers, representing elements of 2d array a. The numbers can be 1 or 0 only.
4. You are standing in the top-left corner and have to reach the bottom-right corner.
Only four moves are allowed 't' (1-step up), 'l' (1-step left), 'd' (1-step down) 'r' (1-step right). You can only move to cells which have 0 value in them. You can't move out of the boundaries or in the cells which have value 1 in them (1 means obstacle)
5. Complete the body of floodfill function - without changing signature - to print all paths that can be used to move from top-left to bottom-right.

Note1 -> Please check the sample input and output for details
Note2 -> If all four moves are available make moves in the order 't', 'l', 'd' and 'r'

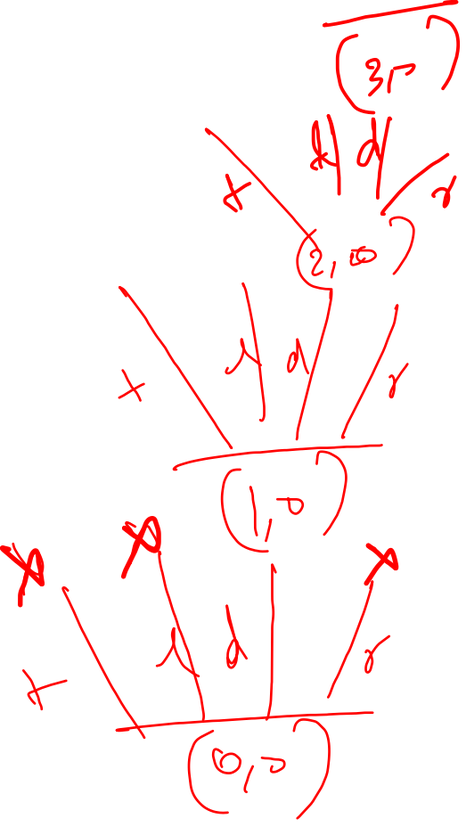
t l d r

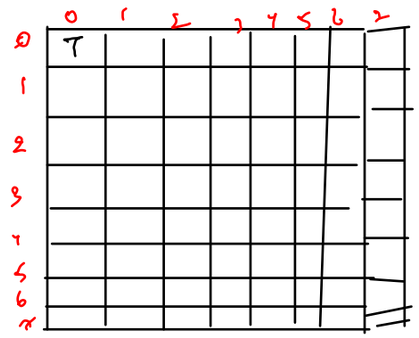


0 ⇒ (i,j)

0 → movement allowed

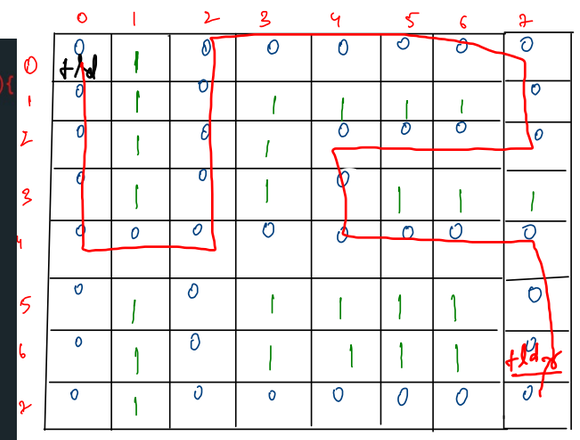
1 → obstacle

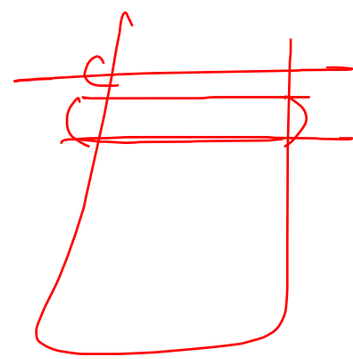


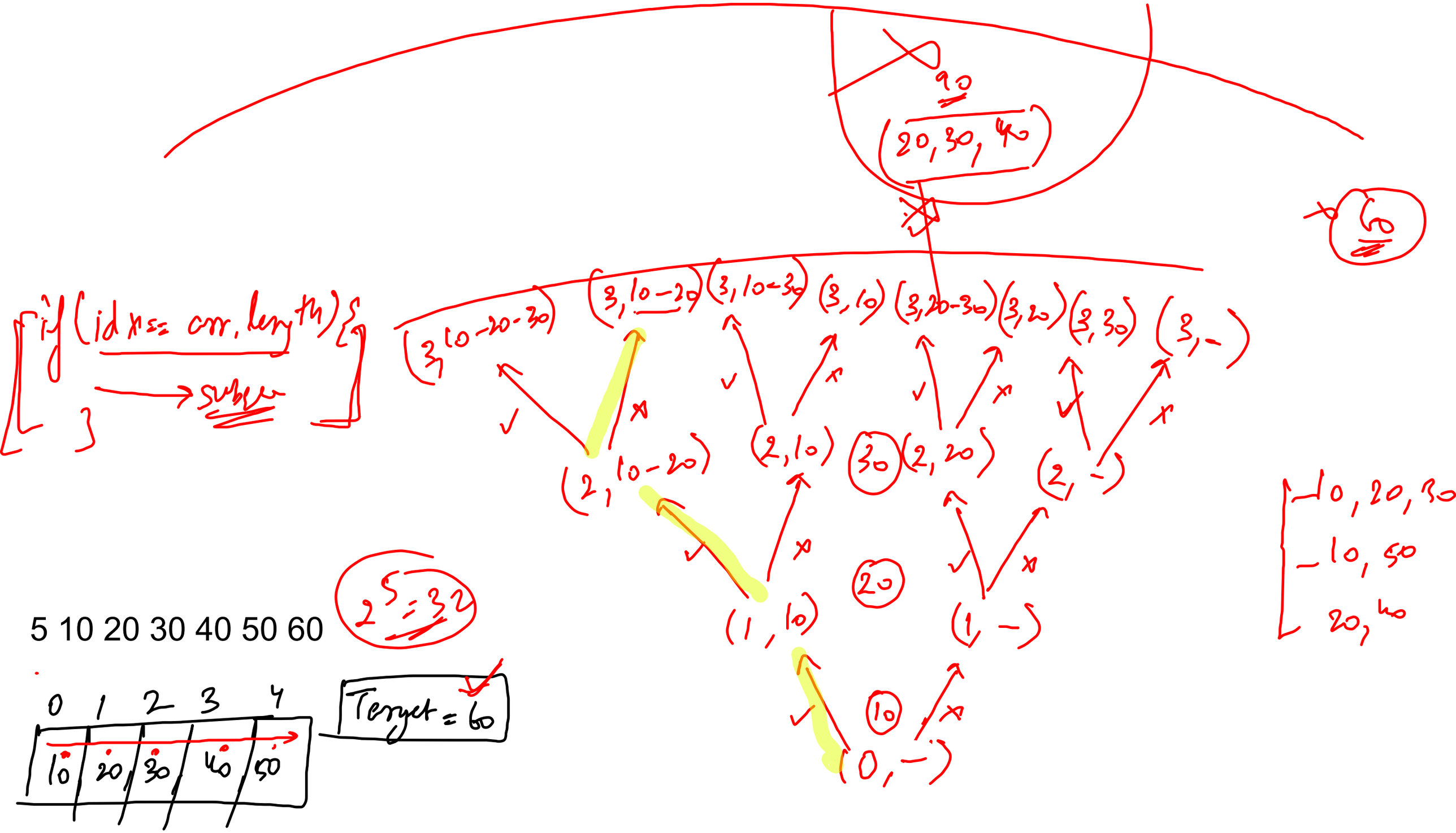


```
lic static void floodfill(int[][] maze , int r , int c , String psf , boolean [][]visited) {
 if(r < 0  || r >= maze.length || c < 0 || c >= maze[0].length || maze[r][c] == 1 || visited[r][c]){
     // invalid
     return;
 }
 if(r == maze.length-1 && c == maze[0].length-1){
     System.out.println(psf);
     return;
 }
 visited[r][c] = true;

 floodfill(maze,r-1,c,psf+'t',visited);// top
 floodfill(maze,r,c-1,psf+'l',visited);// left
 floodfill(maze,r+1,c,psf+'d',visited);// down
 floodfill(maze,r,c+1,psf+'r',visited);// right

 visited[r][c] = false;
```

possibilities





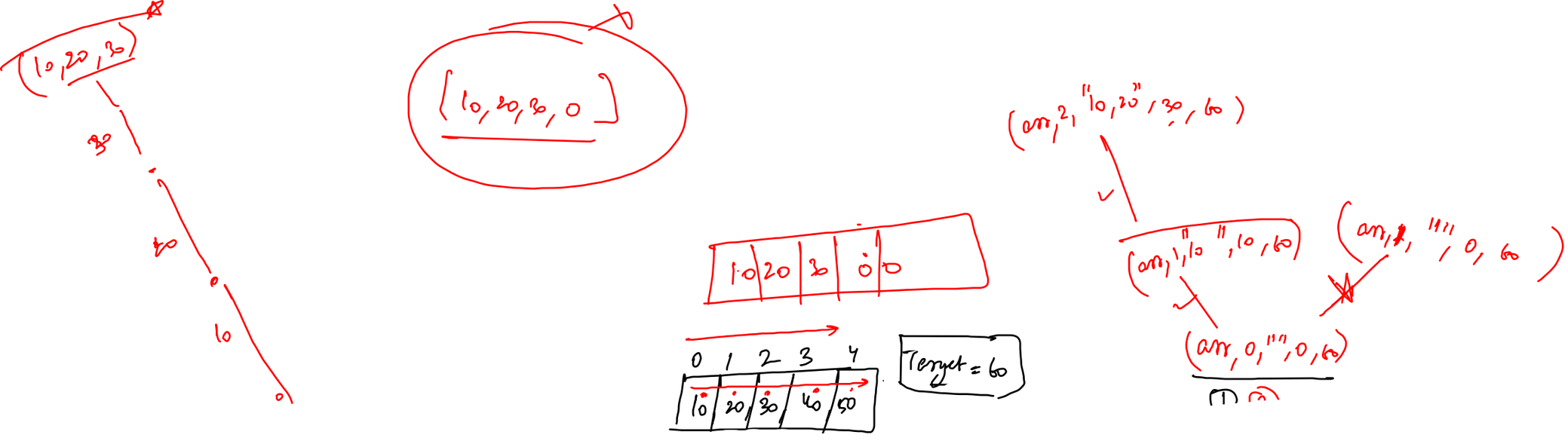


5 10 20 30 40 50 60

$2^5 = 32$

0	1	2	3	4
10	20	30	40	50

Target = 60




```java
public static void printTargetSumSubsets(int[] arr, int idx, String set, int sos, int tar) {
    if(idx ==  arr.length){
        if(sos == tar){
            System.out.println(set+".");
        }
        return;
    }
    printTargetSumSubsets(arr,idx+1,set+arr[idx]+", ",sos+arr[idx],tar);// inc
    printTargetSumSubsets(arr,idx+1,set,sos,tar);// exc
}
```