use ⇒ Sorting → merge sort | quick sort —
Trees | BST | Heaps | Tries | Segment Trees
DP (Dynamic Programming)
Back tracking

Observations :

i) size keeps decreasing

ii) similar code only size changes

iii) end code (last code to stop).

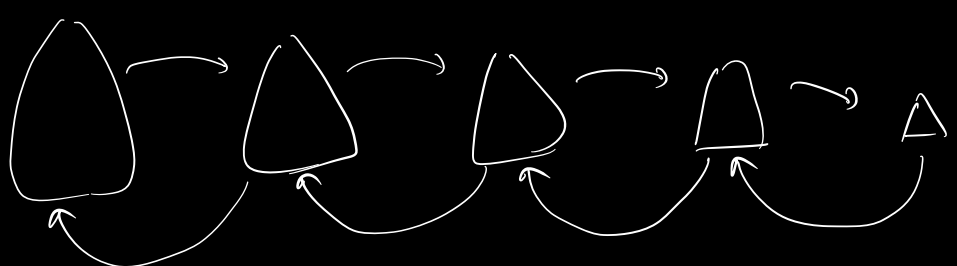
Recursion ⇒ a function calling itself.

technique of solving a problem using soln of

[smaller instances of same problem]

↓

subproblem



$SUM(N) = 1 + 2 + 3 + 4 + \cdots + (N-1) + N$

$= \underbrace{1 + 2 + 3 + 4 + \cdots + (N-1)}_{SUM(N-1)} + N$

$\Rightarrow SUM(N) = SUM(N-1) + N$

sub problem

$\Rightarrow SUM(N-1) = \underbrace{1 + 2 + 3 + 4 + \cdots + (N-2)}_{SUM(N-2)} + (N-1)$

$\Rightarrow SUM(N-1) = SUM(N-2) + (N-1)$

- ⇒ learn how to write recursive code
- ⇒ How it works | dry run
- ⇒ TC | SC

How do we write recursive code :-

Steps

1) Assumption

decide, what your func does?

2) main logic

solving the main problem using sub problem

3) base condn

when should recursion stop.

Find the sum of N natural nos. using recursion

```
int sum(N) { // assume :- sum of N natural nos.
    // base case
    if(N==1)
        return 1;
    // main logic
    return sum(N-1) + N;
}
```

Find the factorial of N.

$5! = 5 \times 4 \times 3 \times 2 \times 1$

$N! = N \times \underbrace{N-1 \times N-2 \times N-3 \times \cdots \times 1}_{(N-1)!}$

$\Rightarrow N! = N \times (N-1)!$

$1! = 1$

$0! = 1$

Code

```
int fact(N) { // assume :- that this func calculates fact(N)
  // base condn
  if( N == 0 )
      return 1;

  // main logic
  return fact(N-1) * N;
}
```

Given N, find N^{th} fibonacci number.

N	0	1	2	3	4	5	6	7	8	9	10
fib →	0	1	1	2	3	5	8	13	21	34	55

Fibonacci series ⇒ sum of last two nos.

[* fibonacci series & golden ratio]

fib(8) = 21 ⇒ 8 + 13 ⇒ fib(6) + fib(7)

fib(8) = fib(6) + fib(7)

⇒ fib(N) = fib(N-1) + fib(N-2) ⟸

```
int fib(N) {  // assume : Calculates fibonacci for Nth position
    if (N == 0)
        return 0
    else if (N == 1)
        return 1
    // main logic
    return fib(N-1) + fib(N-2)
}
```

base case

N = 0

fib(0) = ~~fib(-1) + fib(-2)~~

fib(1) = fib(0) + ~~fib(-1)~~

fib(2) = fib(0) + fib(1)

0 1

Working of recursion :-

1) Sum of N natural nos :-

```
int sum(N) {
    if (N == 1)
        return 1
    return sum(N-1) + N;
}
```

sum(5)
return sum(4) + 5 → 10 → 15

sum(4)
return sum(3) + 4 → 6 → 10

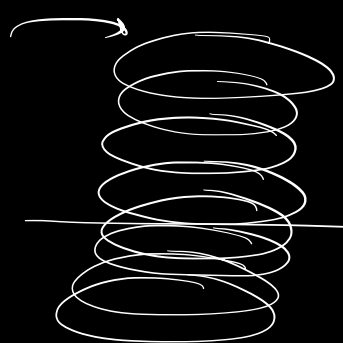
sum(3)
return sum(2) + 3 → 3 → 6

sum(2)
return sum(1) + 2 → 1 → 3

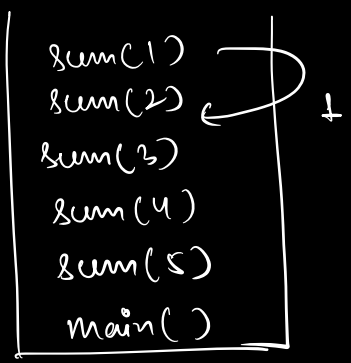
sum(1) → 1

Call Stack

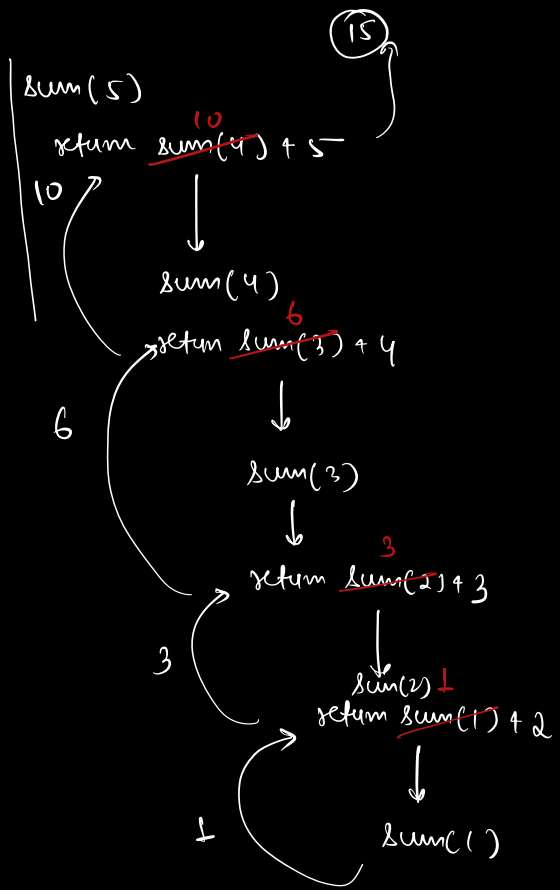
Stack :-



Stack → L I F O
last in first out

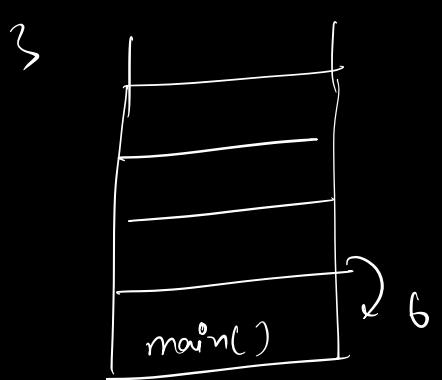


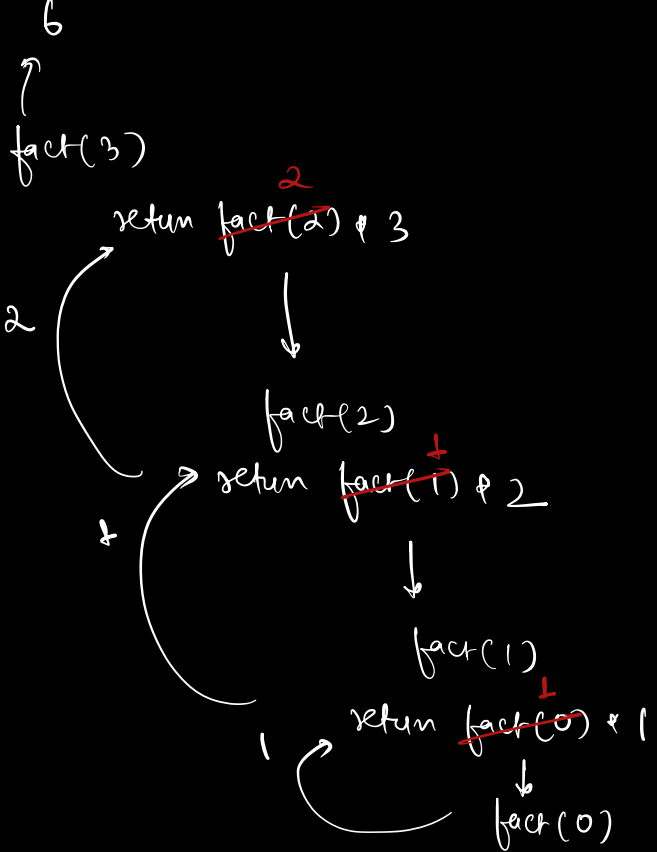
extra space required for call stack



factorial dry run :-

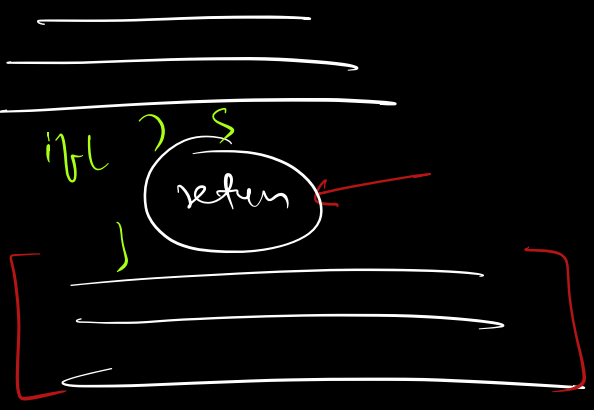
```
int fact(N) {
    if (N==0)
        return 1
    return fact(N-1) * N
```





- return ⇒ keyword stops the execution of func

```
fun() {
```




```
}
```

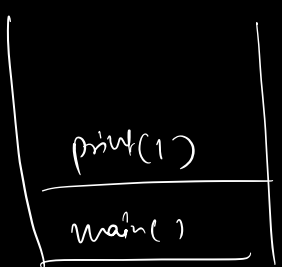
Q 1. Given a no. N, print all the nos. from 1 to N, increasing order using recursion.

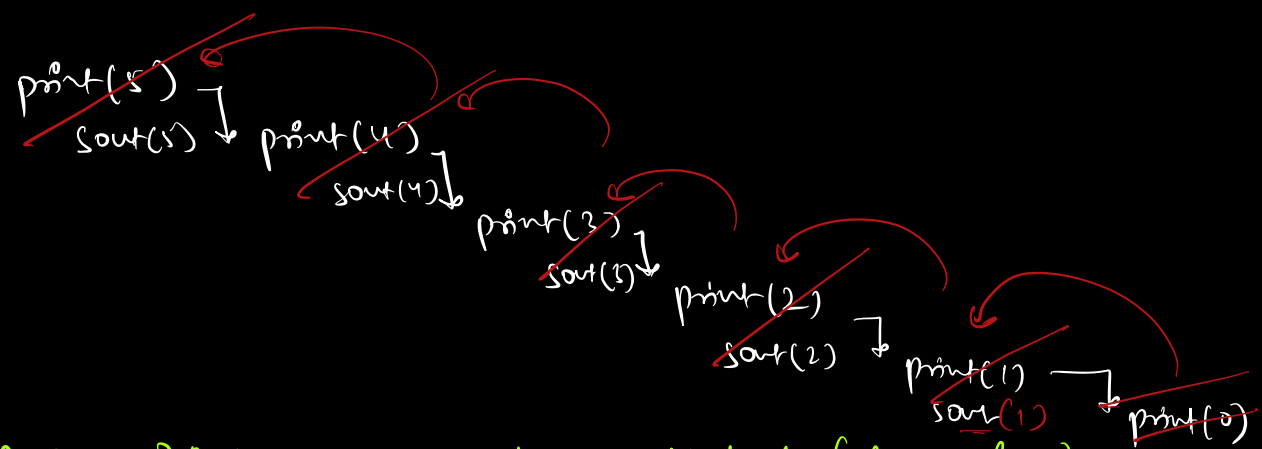
If N = 5, O/P ⇒ 1 2 3 4 5

```
void print(N) {
    if (N==0)
        return;
    print(N-1)
    sout(N)
}
```

print(5) ⇒ 1 2 3 4 5

print(1)





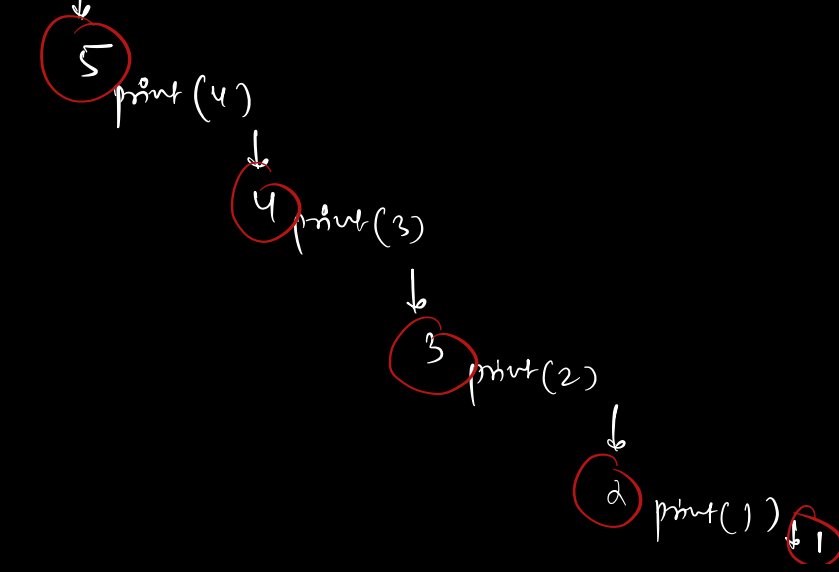
Q 2. Print all nos. from N to 1 (dec. order)

```
void print(N) {
    if (N==0)
        return;
    sout(N)
    print(N-1)
}
```

print(5) ⇒ 5 4 3 2 1

print(4) → 4 3 2 1

print(5) ⇒ sout(5) + print(4)

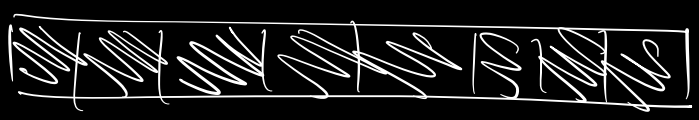


Q 3. Given a string, check if it is a palindrome using recursion.

Palindrome ⇒ MADAM

LEVEL

MALAYALAM



e s

Stopping condn ⇒ $s \geq e$

base case

```
boolean isPalindrome(str, s, e) {
    if (s > e)
        return true
    if (str[s] == str[e]) {
        return isPalindrome(str, s+1, e-1)
    } else {
        return false
    }
}
```

0 1 2 3 4

M A D A M

isPalin(str, 0, 4)

isPalind(str, 1, 3)

isPalin(str, 2, 2)

isPalin(str, 3, 1)

0 1 2 3

A B A A

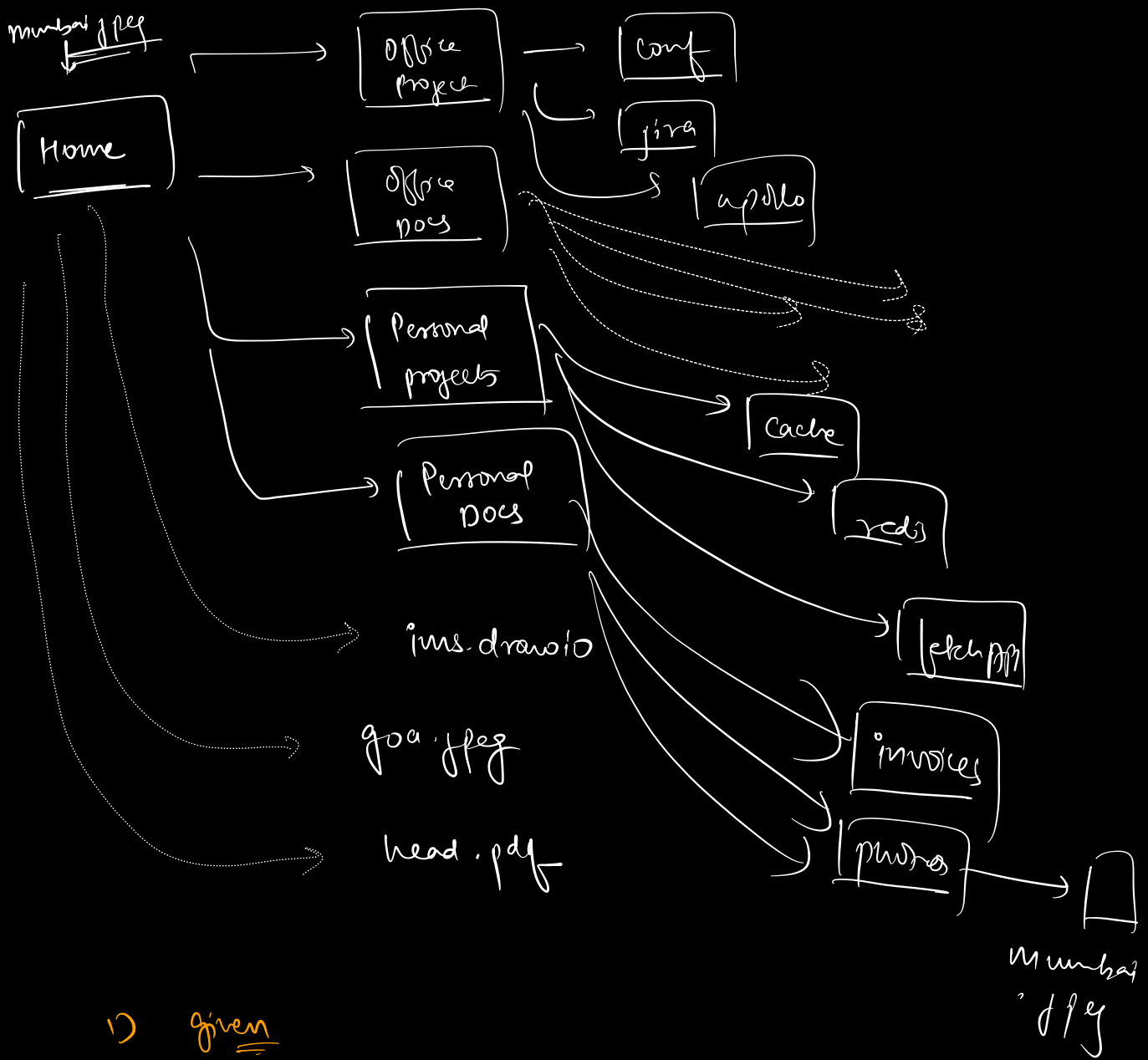
isPalin(str, 0, 3)

isPalin(str, 1, 2)

false

Facebook, Google

Q4. Given a directory structure and you have to search a file.



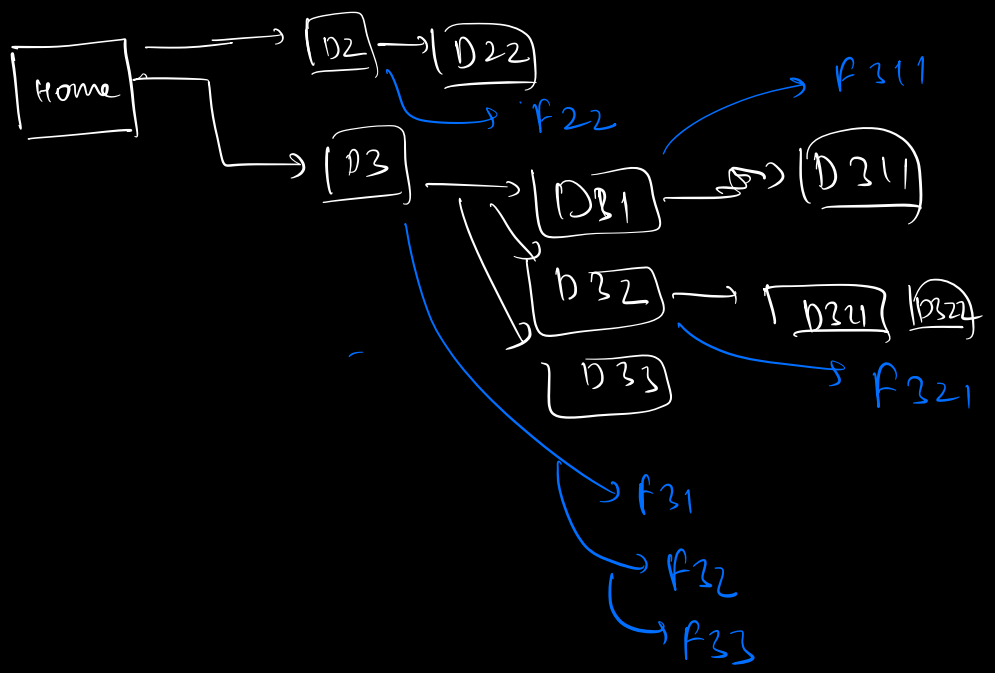
1) given

List< > getAllDirectory (directoryName)

List< > getAllFiles (directoryName)

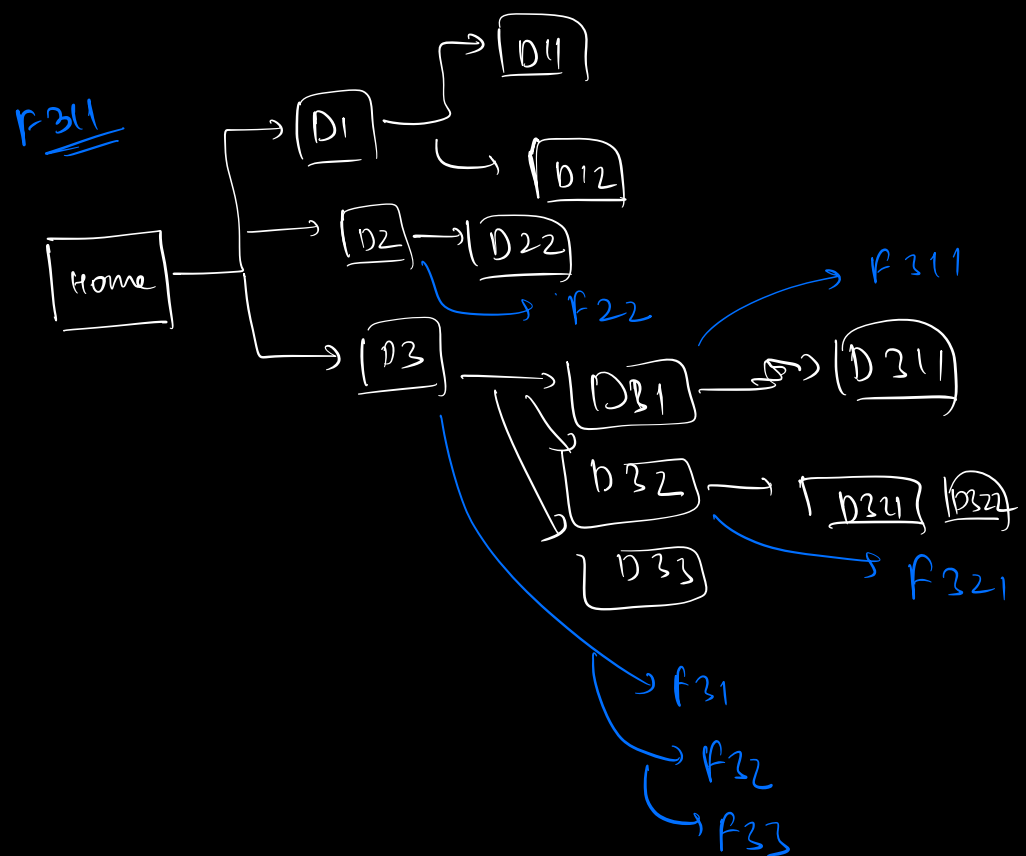
O/P = true / false = fileName

f311




```
boolean search(directoryName, fileName) {

    List<String> files = getAllFiles(directName);

    for(i=0; i< files.size(); i++) {

        if( files[i] == fileName )

            return true

    }

List<String> directories = getAllDirs(directoryName);
```

```
for( i=0; i< directories.size(); i++ ){
    if( search( directories[i], fileName ))
        return true
}
return false
}
```



search(home, f311)

Search (D1, F311)

Rec - 2

→ LinkedList ←

Queue

Tries 1

Tries 2

Someone else

Subsequence & Subsets ←

Advanced