Problem Statement

There are N stones, numbered $1, 2, \ldots, N$. For each i ($1 \leq i \leq N$), the height of Stone i is h_i.

There is a frog who is initially on Stone 1. He will repeat the following action some number of times to reach Stone N:

- If the frog is currently on Stone i, jump to one of the following: Stone $i+1, i+2, \ldots, i+K$. Here, a cost of $|h_i - h_j|$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches Stone N.

Input

Input is given from Standard Input in the following format:

```
N K
h_1 h_2 ... h_N
```

Output

Print the minimum possible total cost incurred.

Sample Input 1 Copy

```
5 3
10 30 40 50 20
```

Sample Output 1 Copy

```
30
```

If we follow the path $1 \rightarrow 2 \rightarrow 5$, the total cost incurred would be $|10 - 30| + |30 - 20| = 30$.

hint → frog jump fibonacci

Recursively

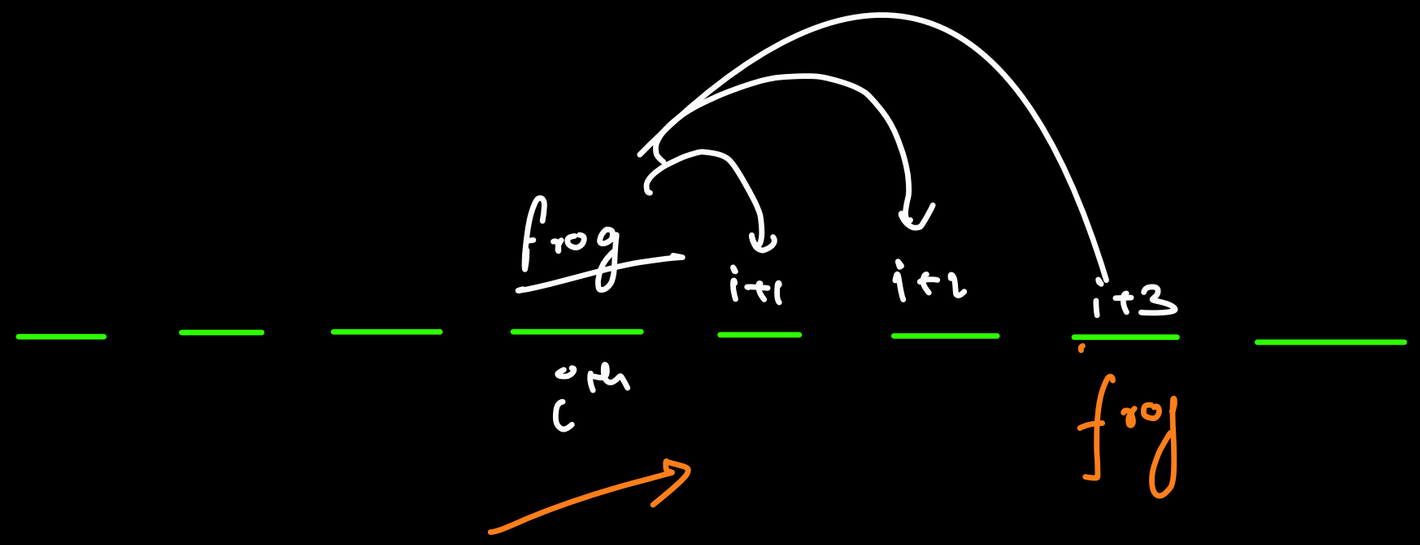
cost → $|h_3 - h_6|$

cost = $|h_1 - h_2| + |h_2 - h_5|$

⇒ $|10 - 30| + |30 - 20|$

⇒ $20 + 10$

⇒ 30



#approach1

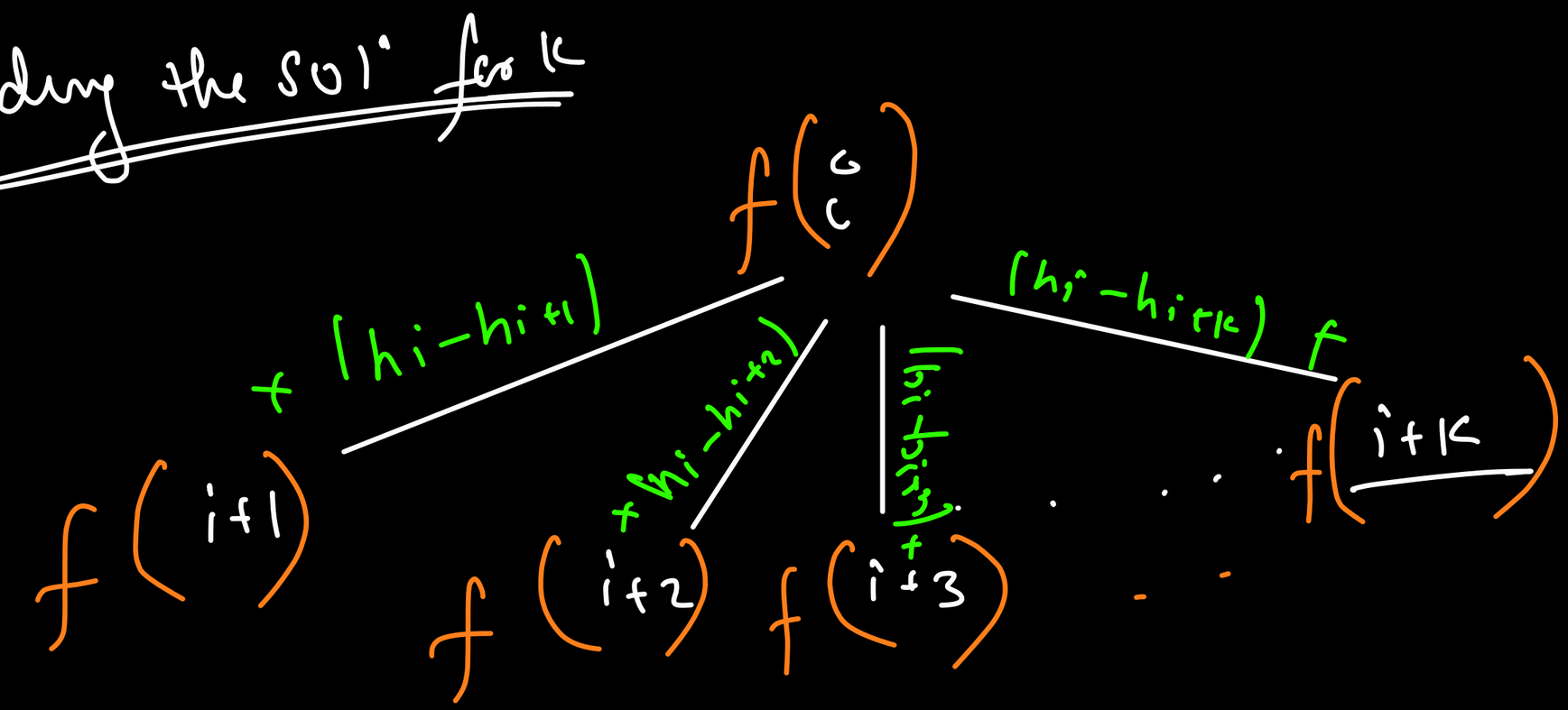
$$f(i) = \min\left(f(i+1) + |h_i - h_{i+1}|,\ f(i+2) + |h_i - h_{i+2}| \right)$$

#approach2

$$f(n) = \min\left(f(n-1) + |h_{n-1} + h_n|,\ f(n-2) + |h_{n-2} - h_n| \right)$$

$$f(i) = \min\left(f(i+1) + |h_i - h_{i+1}|,\ f(i+2) + |h_i - h_{i+2}|,\ f(i+3) + |h_i - h_{i+3}| \right)$$

Extending the soln for K



$$f(i) = \min\left(f(i+j) + |h_i - h_{i+j}| \right) \quad \forall\ j \in [1, K]$$

$$\min\left(f(i+1) + |h_i - h_{i+1}|,\ f(i+2) + |h_i - h_{i+2}| \cdots\cdots f(i+K) + |h_i - h_{i+K}| \right)$$

```
ans = +∞;
for (j=1; j<=K; j++) {
    if (i+j >= n) break;
    ans = min(ans, f(i+j) + abs(h[i] - h[i+j]))
}
```

Q⇒ Given an array of n integers, calculate the sum of array elements, recursively.

Ex ⇒ [1, 2, 3, -1, 5]

ans ⇒ f(arr, 0)

ans ⇒ 10

Help us in iterating the array

[a_0, a_1, a_2, a_3 ..... a_{n-1}]

$f(arr, i) = f(arr, i+1) + arr[i]$

returns sum of elements of array from index i to n-1

if (i == n) → array is having no elements

return 0;

assume the function f works correctly for i+1.

Self work → add i^{th} index element to the sum of (i+1) to (-1)

```
  0  1  2  3  4
[ 1, 2, 3, -1, 5]
```

Time → O(n)

Space → O(n)

Q. Given an array of integers of length n, and a target value x, check whether x exists in the array or not?

Ex → [1, 9, -1, 3, 8, 6]

x = 8

ans → true.

2 recursively

2.0

↳ f(arr, i, x) = (arr[i] == x) or f(arr, i+1, x)

returns whether x is present in the array from index i to n-1

Base → if (i == n) return false

selfwork ⇒ to check if the current element is equal to x.

assume ⇒ function f works correctly for i+1, i.e. it correctly check if x exists in the range [i+1, n-1] or not.

Q:- Given an array of n integers, Check if the array elements are arranged in asc order or not ??

Ex→ [2, 3, 9, 10, 16]

Ans→ true

2

Recursively

Ex→ [2, -3, 9, 10, 16]

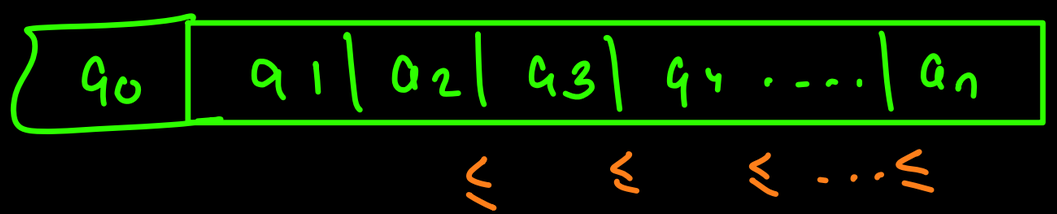
Ans→ false

$f(arr, i) = (arr[i] <= arr[i+1])$ and $f(arr, i+1)$

returns whether elements from index $[i, n-1]$ are arranged in inc order or not ??

check if i^{th} element has been correctly placed or not ??

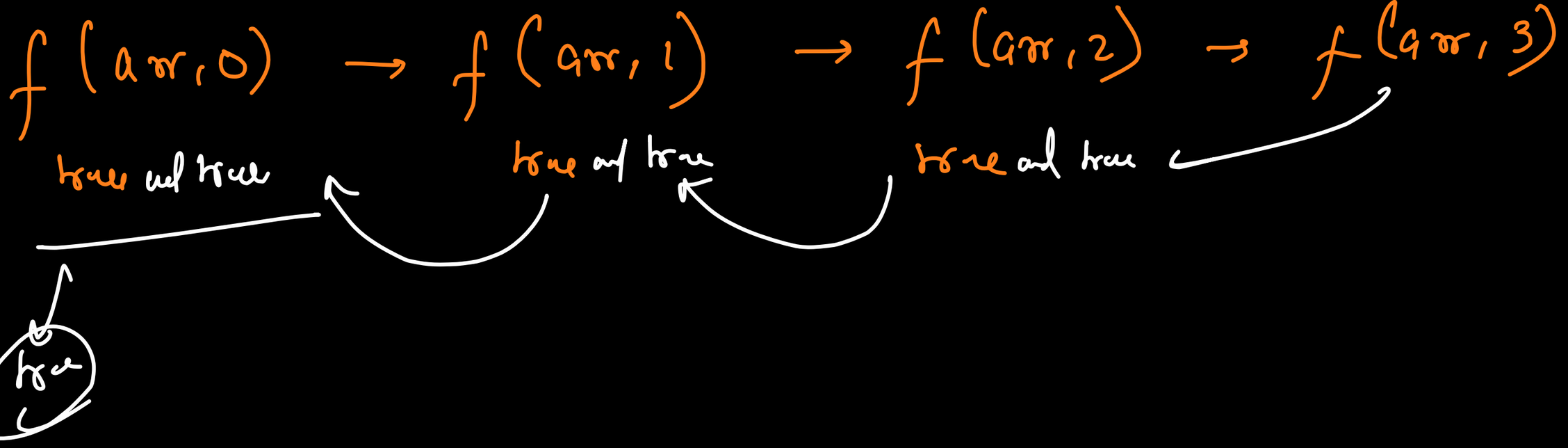
assume that f works fo i+1 & returns while $[i+1, n-1]$ is in asc order or not



ans → f(arr, 0)

```
if (i == n-1) {
    return true;
}
```

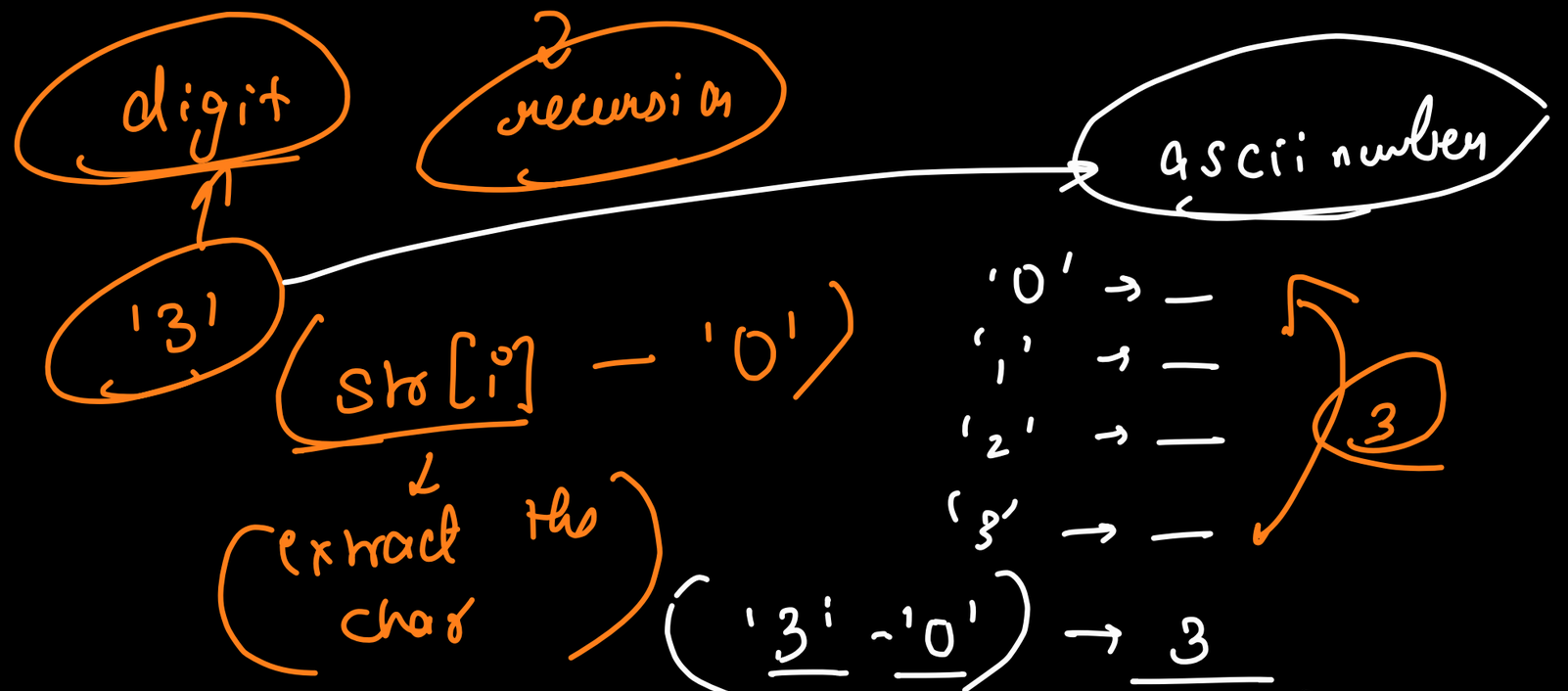
[1, [2, 3, 4]]



Q» Given a string of digits, calculate the sum of digits of the string.

Ex → "1234"

ans → 10



std::String str = "12345";

$f(str, i) = f(str, i+1) + (str[i] - '0');$

2

Sum of digits of string from index i - n-1

```
if (i == str.size())
    return 0;
```

Printing Technique for Recursion

f(----, output)

↓

string

Str = "abc"

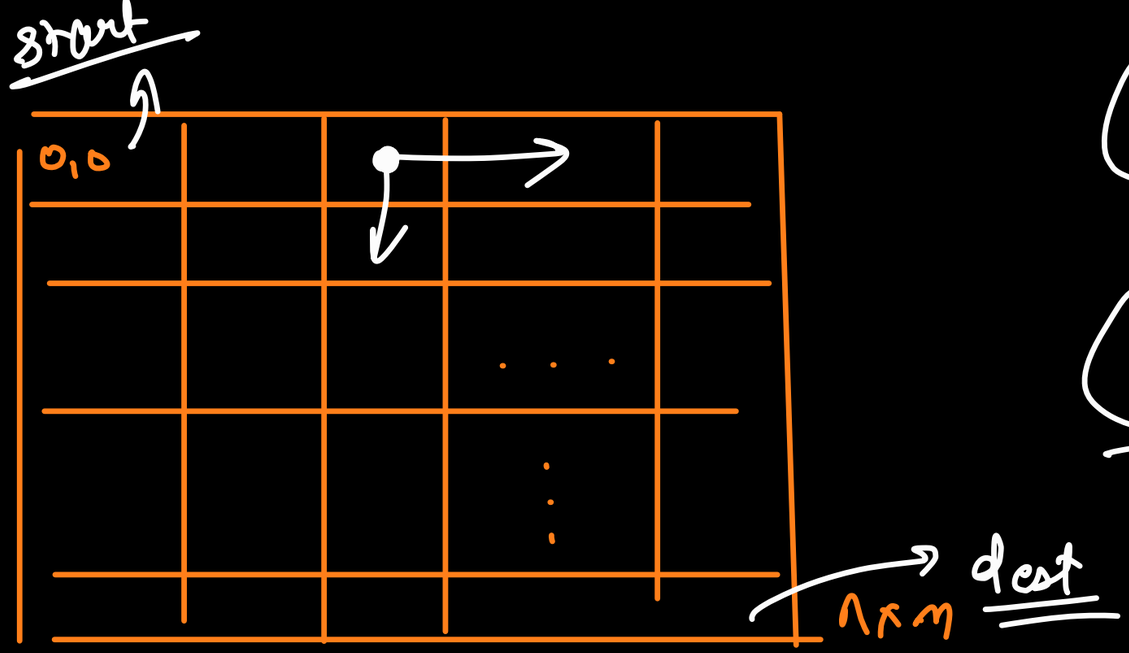
Str2 = Str + "xyz"

→ "abcxyz"

O(n)

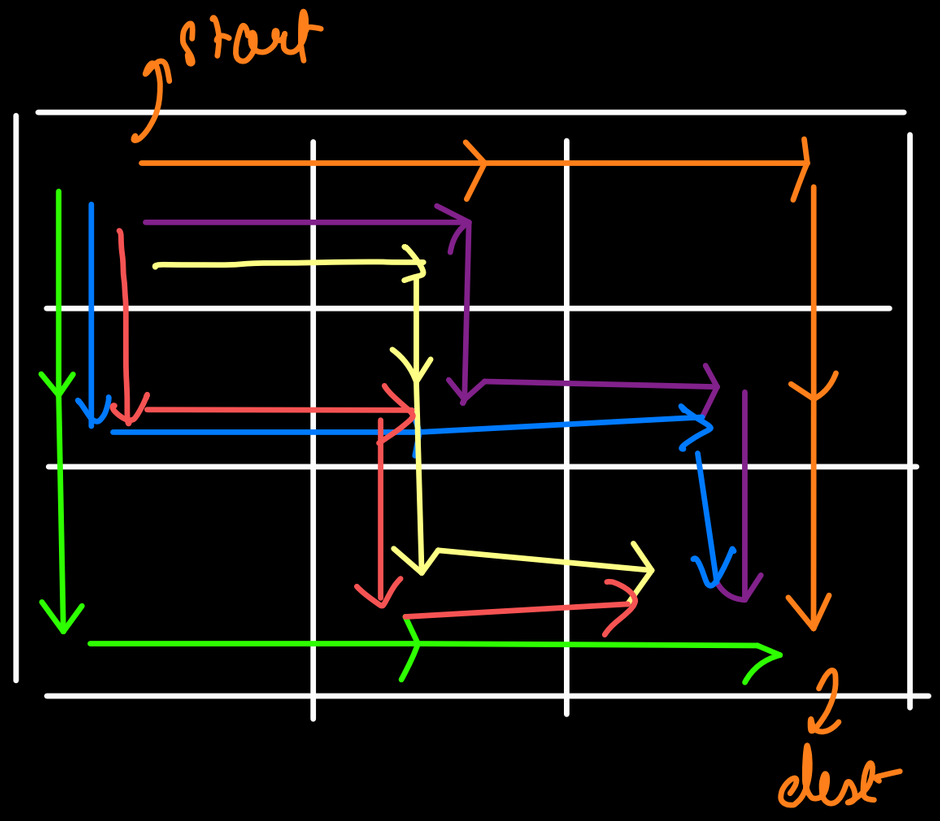
concatenate

Q: Given dimensions of a 2d grid (n, m), print total different ways in which you can start from $(0,0)$ and reach $(n-1, m-1)$ such that from any index you can move one step down or right.



$(i, j) \rightarrow (i, j+1)$

$(i, j) \rightarrow (i+1, j)$



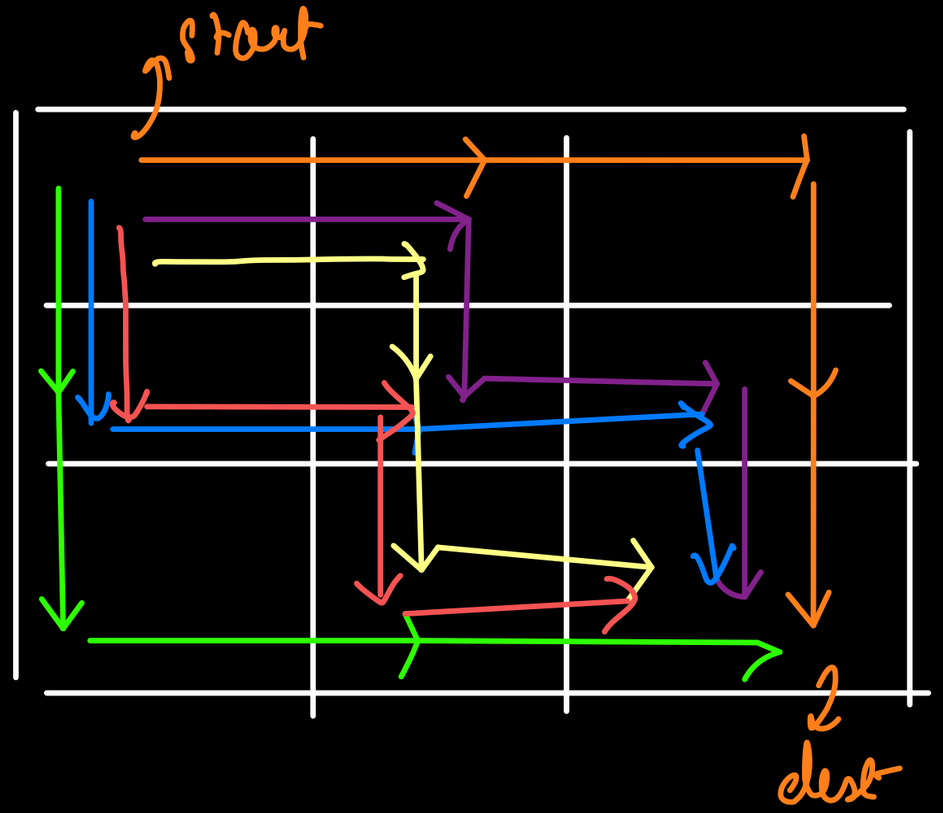
3x3

R R D D
D D R R
R D R D
D R R D
R D D R
D R D R

print all Groups

n=3 m=3

R R D D
D D R R
R D R D
D R R D
R D D R
D R D R



occur

2

$\frac{n!}{a!\,b!\,c!\ldots}$ $\frac{5!}{2!}$

$\frac{4!}{2!\,2!} \rightarrow \frac{24 \times 3}{4 \times 1} \rightarrow 6$

$n=3$ $m=3$

R R D D

D D R R

R D R D

D R R D

R D D R

D R D R

} 6

↳ all of these are permutation of RRDD

$\frac{(n-1+m-1)!}{(n-1)!\,(m-1)!}$ → Math

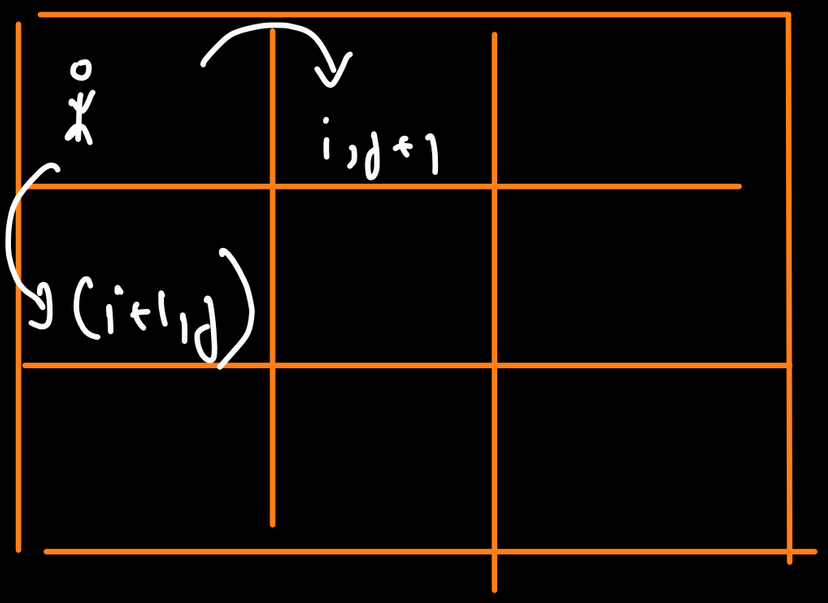
// count the total possible ways

$f(i, j, n, m, output) = f(i, j+1, n, m, output + "R")$

$f(i+1, j, n, m, output + "D")$

prints the total ways to reach from i, j to (n-1, m-1) using output string

```
if (i >= n or j >= m)
    return;
if (i == n-1 and j == m-1)
    print(output)
    return;
```



RRDD
RDRD
RDDR
$\in$
RDD
DRD
DDR

'D'
DRR
RRD
RDR