

# ① Addition using Recursion.

## Problem Statement

Add two nonnegative integers  $a$  and  $b$ . We are only allowed to add or subtract single units from the numbers.

Size of the problem.:

$$a + b$$

Base Cases:

~~1:~~  $a = b = 0$

✓2:  $a = 0, \text{result} = b$

✓3:  $b = 0, \text{result} = a$

These two base cases makes the first base case redundant.

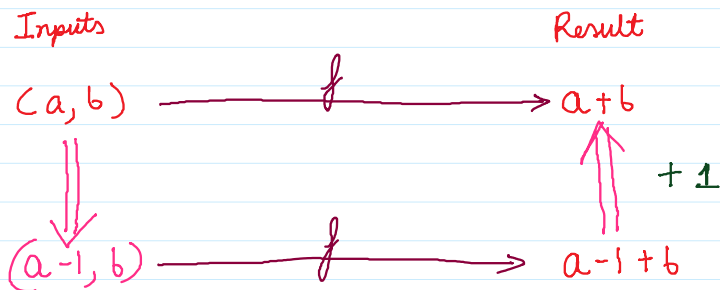
The second & third base cases guarantees that  $a$  &  $b$  will be positive for all the recursive cases.

## Problem Decomposition

Reduce the size of the problem by 1 unit.

We can subtract one unit from either  $a$  or  $b$ .

## Recursion Diagram



(We need to extend the solution to the sub problem in order to find solution to the original problem)

↓  
This can be done by adding 1 to the solution to the sub problem.

Recursive Case:-

$$f(a, b) = f(a-1, b) + 1$$

↖ function to implement

function to implement

### Mathematical Representation :-

$$f(a, b) = \begin{cases} a & \text{if } b = 0 \\ b & \text{if } a = 0 \\ f(a-1, b) + 1 & \text{if } a > 0 \text{ \& } b > 0 \end{cases}$$

### Code & mathematical Representation :-

```
def add(a, b)
  if a = 0
    return b
  elif b = 0
    return a
  else
    return add(a-1, b) + 1
end
```

Diagram showing the mapping between the code and the mathematical representation:

- The `if a = 0` branch maps to the first case of the mathematical representation:  $f(a, b) = a$  if  $b = 0$ .
- The `elif b = 0` branch maps to the second case:  $f(a, b) = b$  if  $a = 0$ .
- The `else` branch maps to the third case:  $f(a, b) = f(a-1, b) + 1$  if  $a > 0$  &  $b > 0$ .

### Option 2

Reduce the size of the problem by subtracting a unit from  $b$  instead of  $a$

$$f(a, b) = f(a, b-1) + 1$$

Speed :- Slow if  $b$  is large ( $\because$  more calls needed to reach base case)

### Efficient solution :-

Choose smallest input parameter to reduce

| $a$    | $b$    | Number of calls |
|--------|--------|-----------------|
| 10,000 | 2      | 2               |
| 2      | 10,000 | 10,000          |

### Decomposition Approach 1 :-

Decrement the smallest input parameter.

Size of the subproblem

$$\min(a, b) - 1$$

Recursive Rule

if  $a < b$

$$f(a, b) = f(a-1, b) + 1$$

Else

$$f(a, b) = f(a, b-1) + 1$$

Mathematical function

$$f(a, b) = \begin{cases} a & \text{if } b = 0 \\ b & \text{if } a = 0 \\ f(a-1, b) & \text{if } a < b \text{ ( } a \neq 0 \text{ \& } b \neq 0 \text{)} \\ f(a, b-1) & \text{if } b \leq a \text{ ( } a \neq 0 \text{ \& } b \neq 0 \text{)} \end{cases}$$

pseudo code :-

```
def add(a, b)
    if a = 0
        return b
    else if b = 0
        return a
    else if a < b
        return add(a-1, b) + 1
    else
        return add(a, b-1) + 1
    end
end
```

Base cases

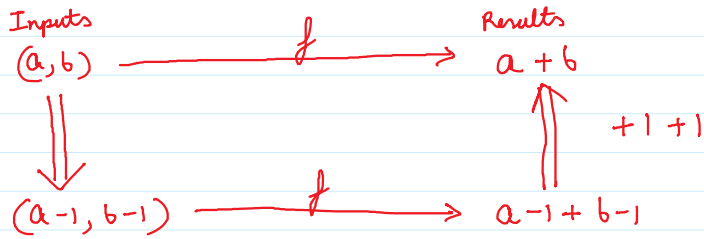
Recursive cases

Decomposition Approach #2

Decrement both parameters (reduces the size of the problem)

Recursive diagram

## Recursive diagram



## Recursive Rule :-

$$f(a, b) = f(a-1, b-1) + 2 \quad \text{for } a > 0 \text{ \& } b > 0$$

## pseudo code

```
def add(a, b)
    if a = 0
        return b
    else if b = 0
        return a
    else
        return add(a-1, b-1) + 2
    end
end
```

} Base cases

} Recursive Case

Aaaaa