# Greedy → Optimisation Problem

DP

B&B

Largest No

2 9 5

↓

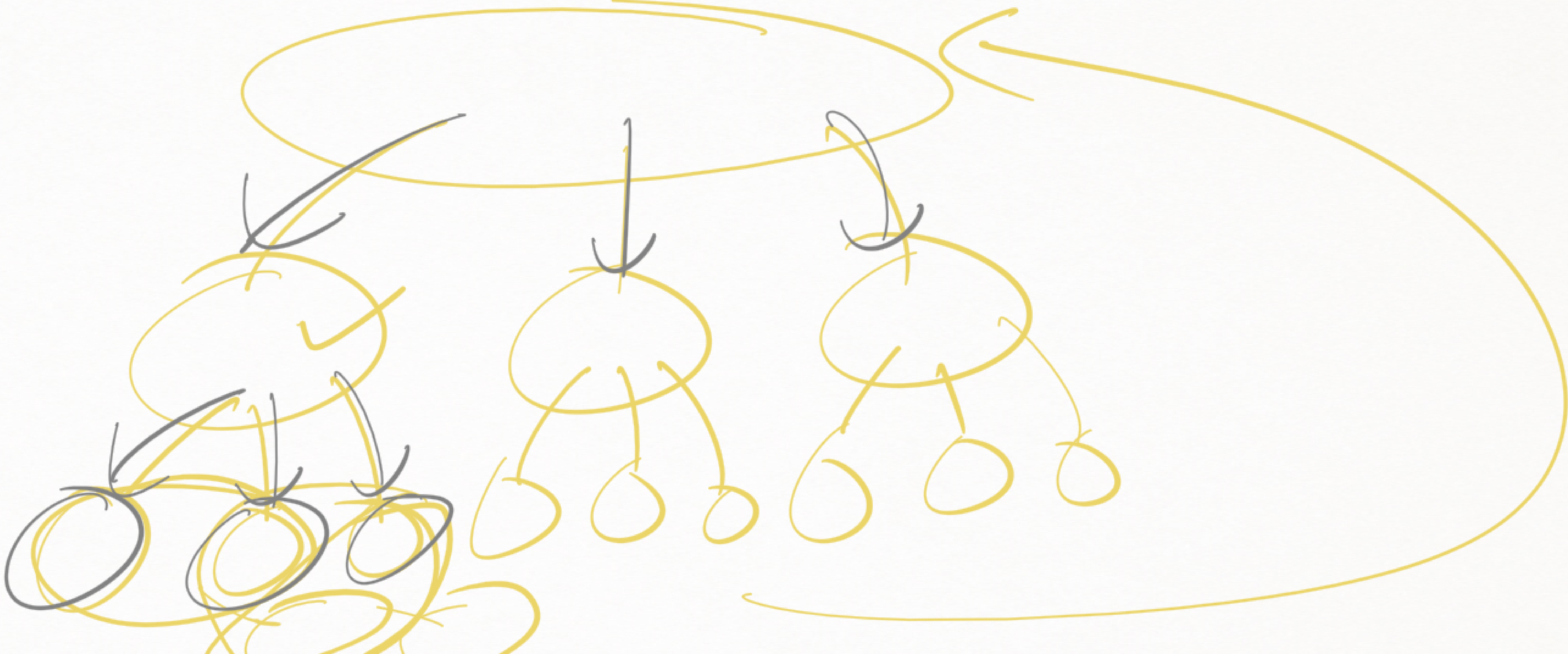
952

Greedy choice

Next subproblem

# Divide & Conquer

Divide a problem into similar but non overlapping subproblem to solve them recursively and combine them in the end.

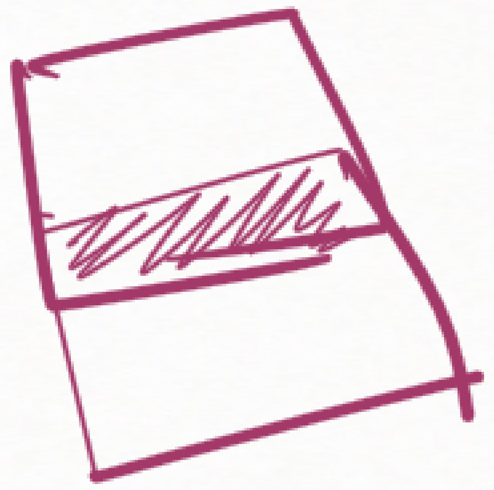


British
↓
India

BIT

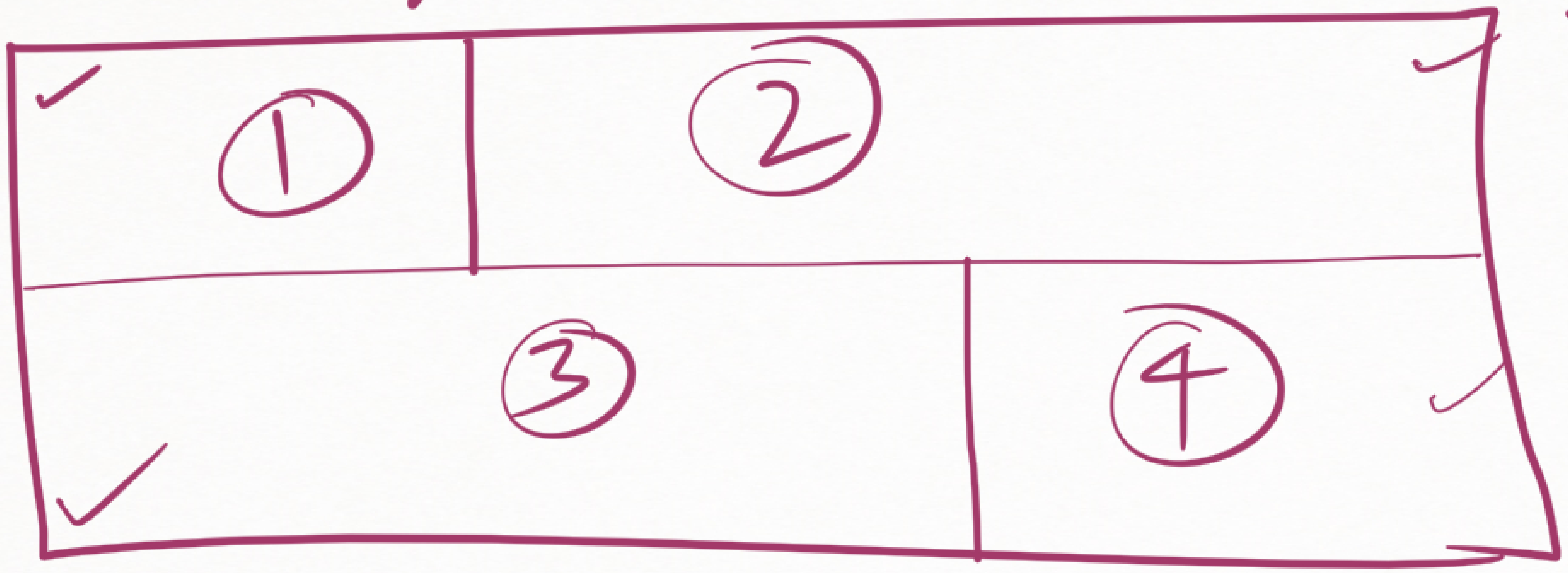
# Divide

A problem to be solved



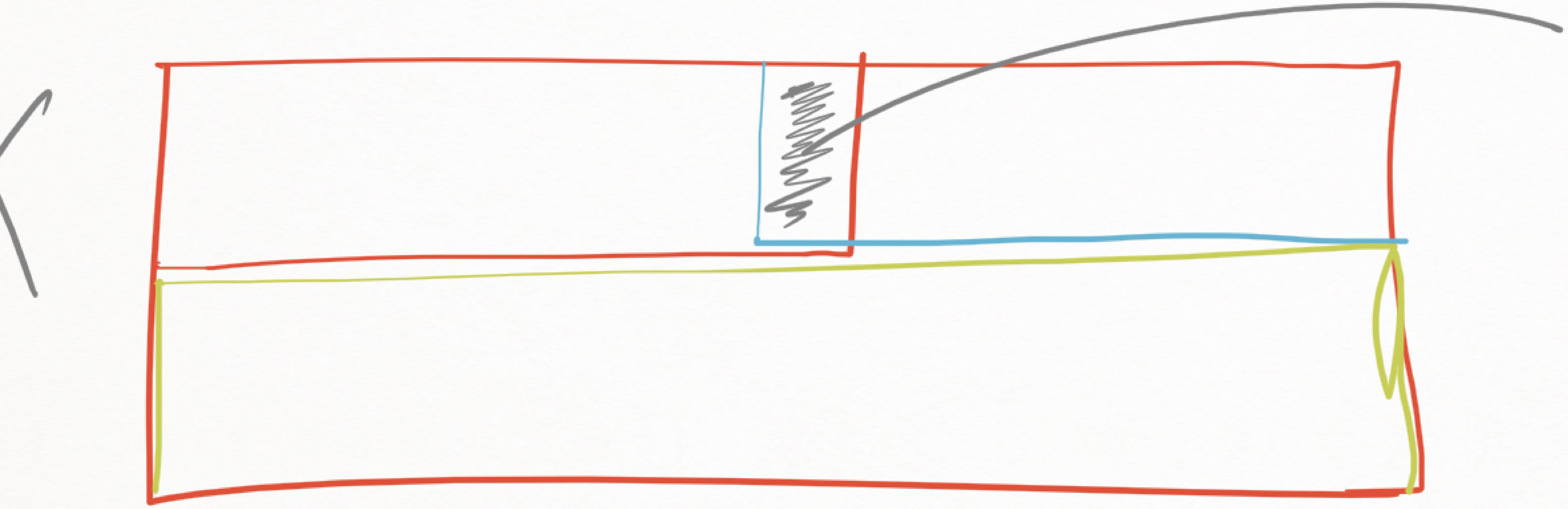
Break Problem into non-overlapping subproblems the same type

alid
Divide

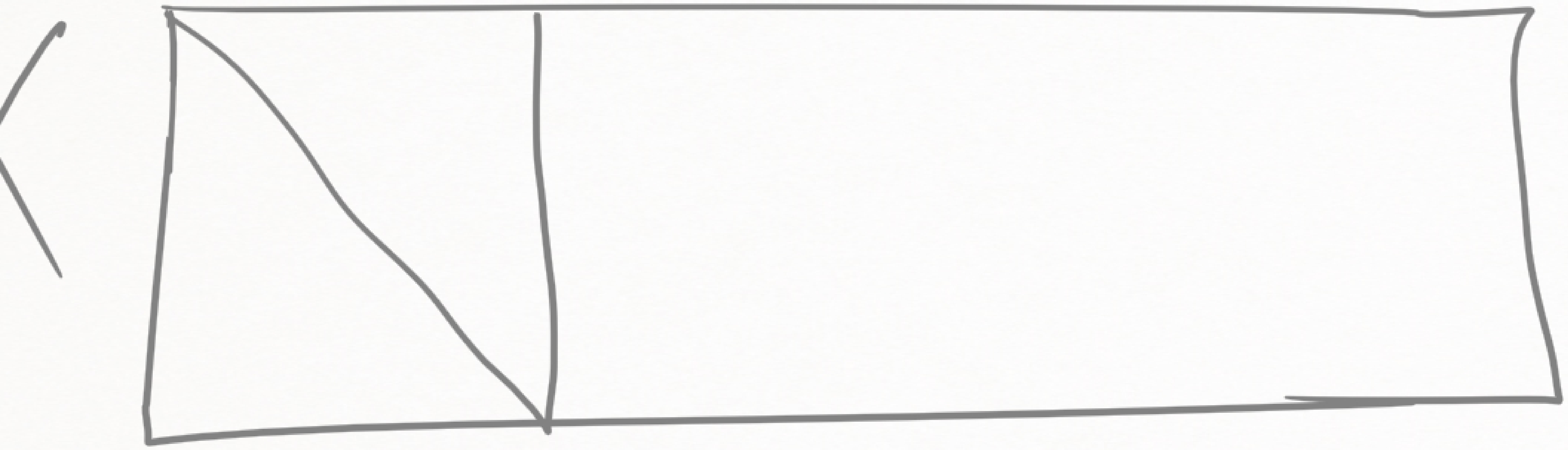



4
Diff
subprob.

Invalid Divide $\boxed{\text{Problem}}$

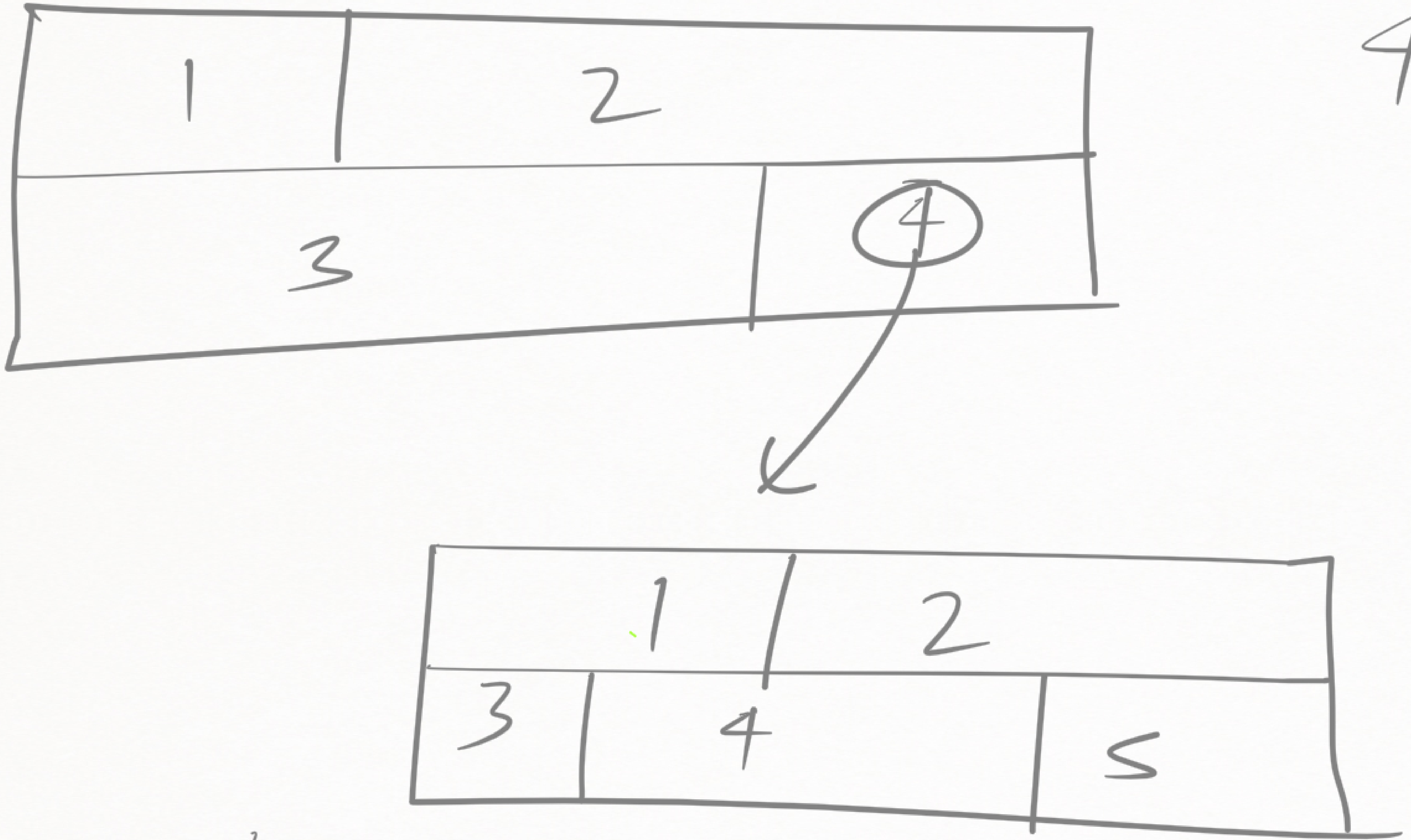


Overlap



No overlap
But diff.
Subproblem.

How each subproblem is solved!



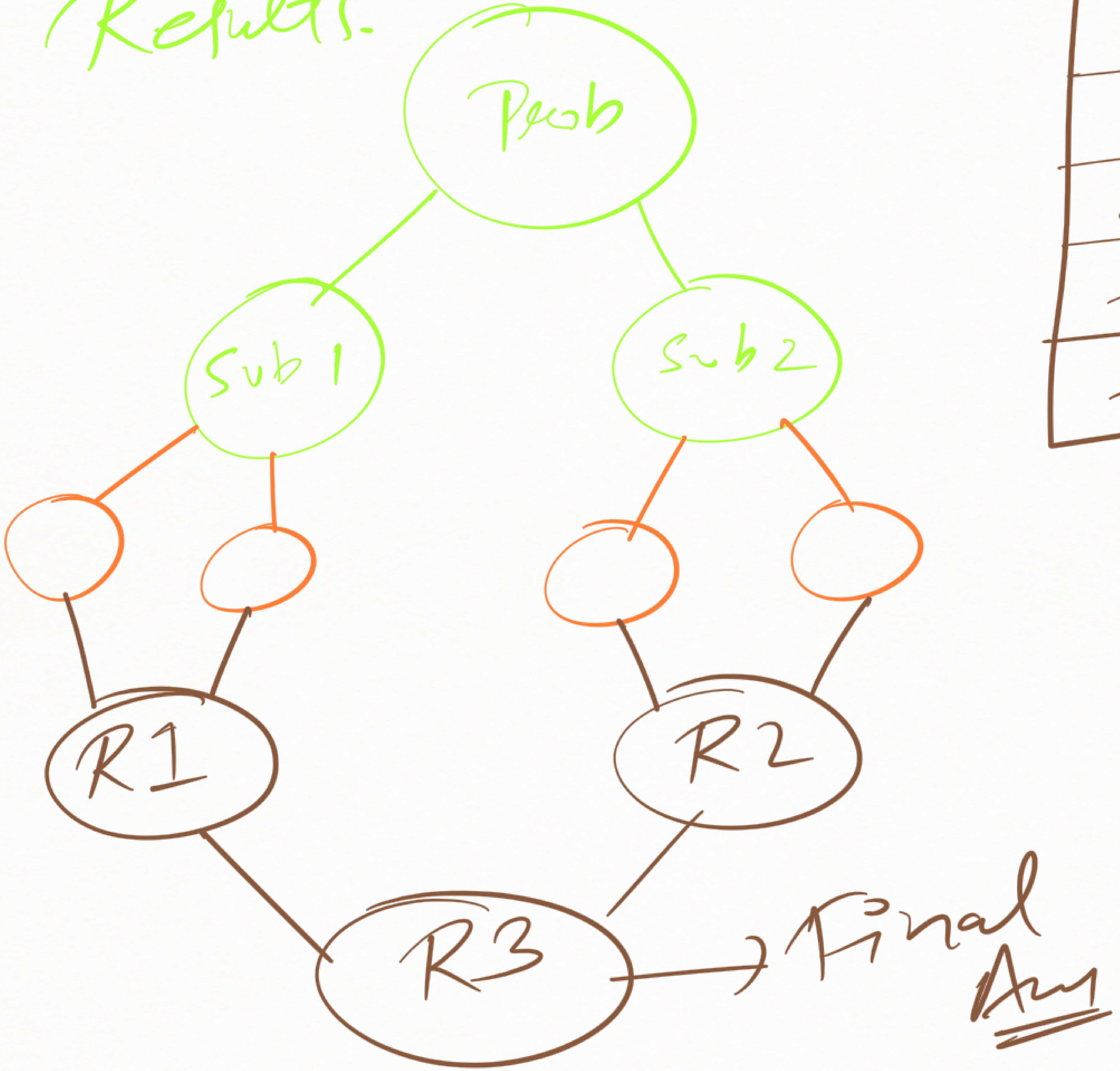

4 Subprob.

5 Sub prob

→ Do it till Base case is arrived

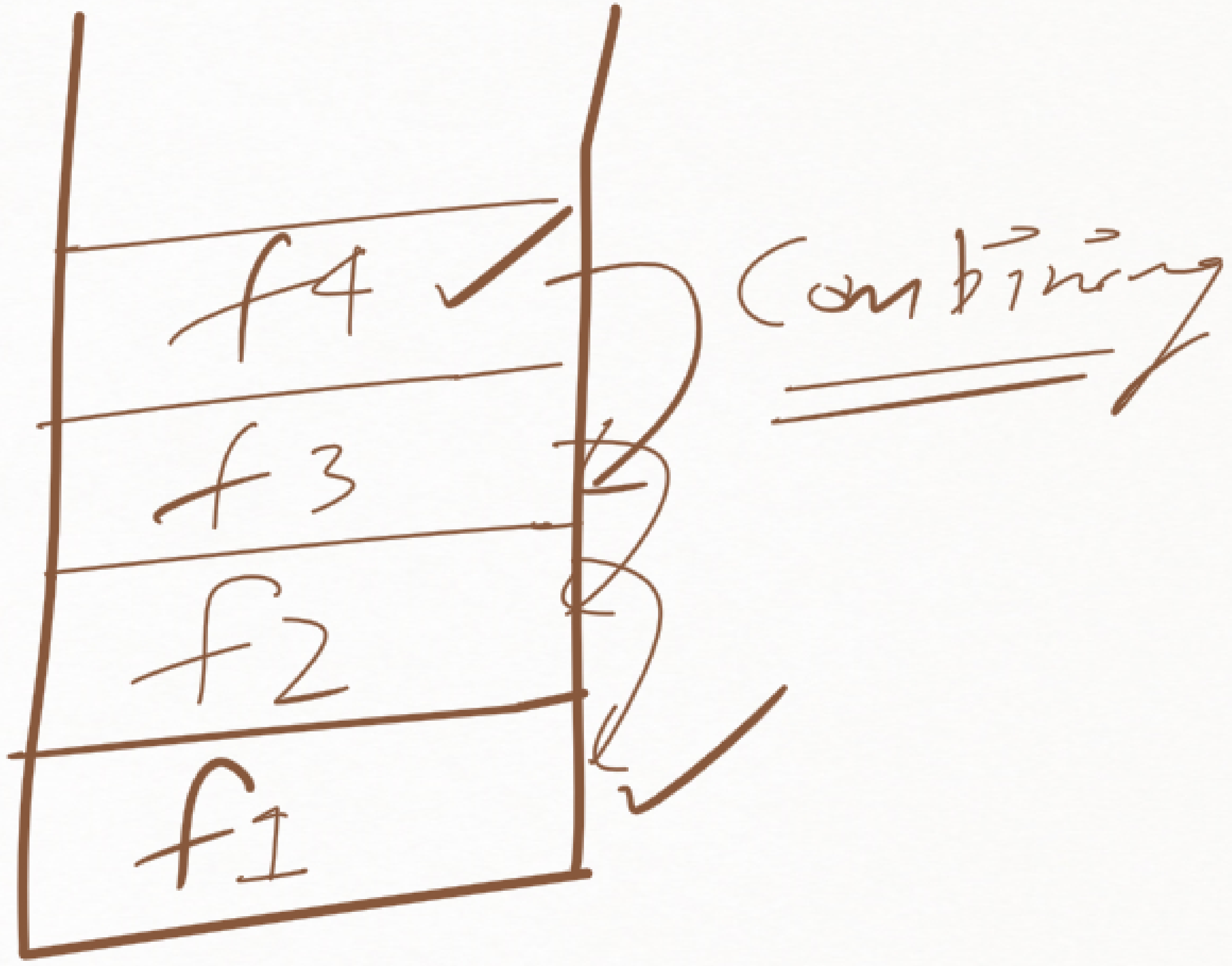
Terminating condition for recursion.

# Combine Results.




ase
es

tarting the
combine
stage.



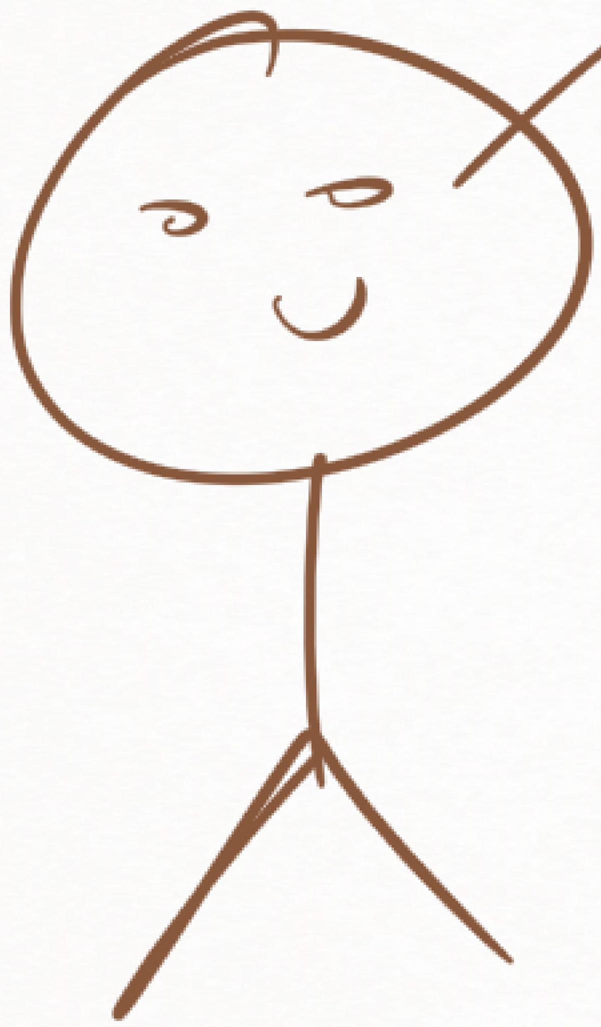
# Divide & Conquer Strategy

1) Break into non-overlapping subproblems of the same type.

2) Solve subproblems.

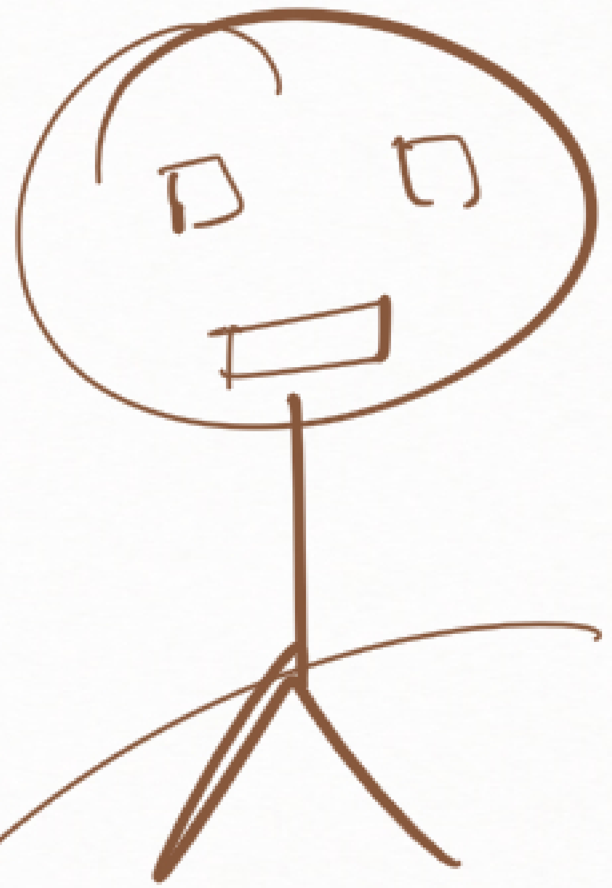
3) Combine Results.

## 21 Q'sn Game



You

Guess that Number



Computer

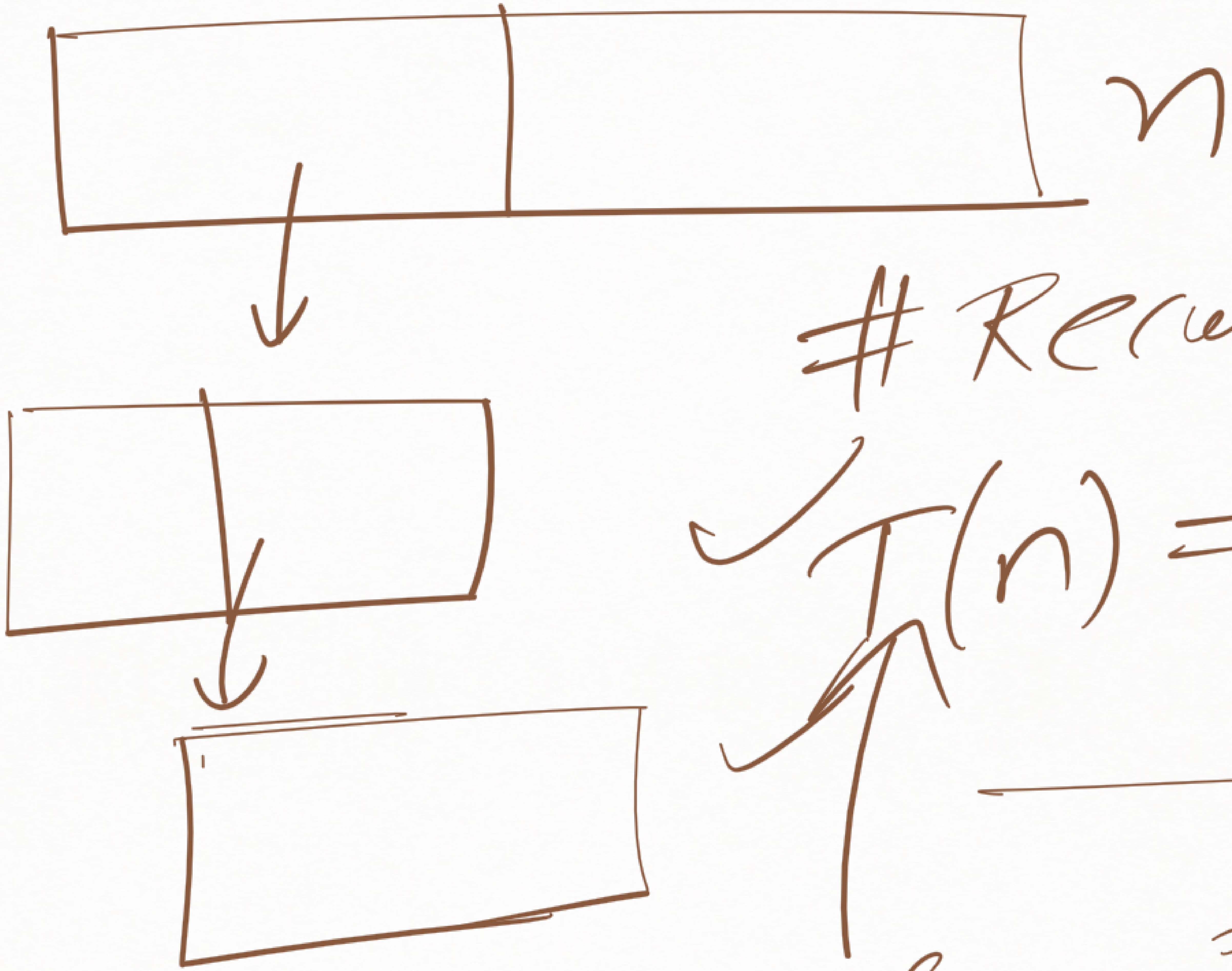
Has a hidden number $1 \le x \le N$

Computer will ans if your guessed no is equal, less, or greater than the hidden no.

Binary Search

Exact Same Subproblem

# Divide

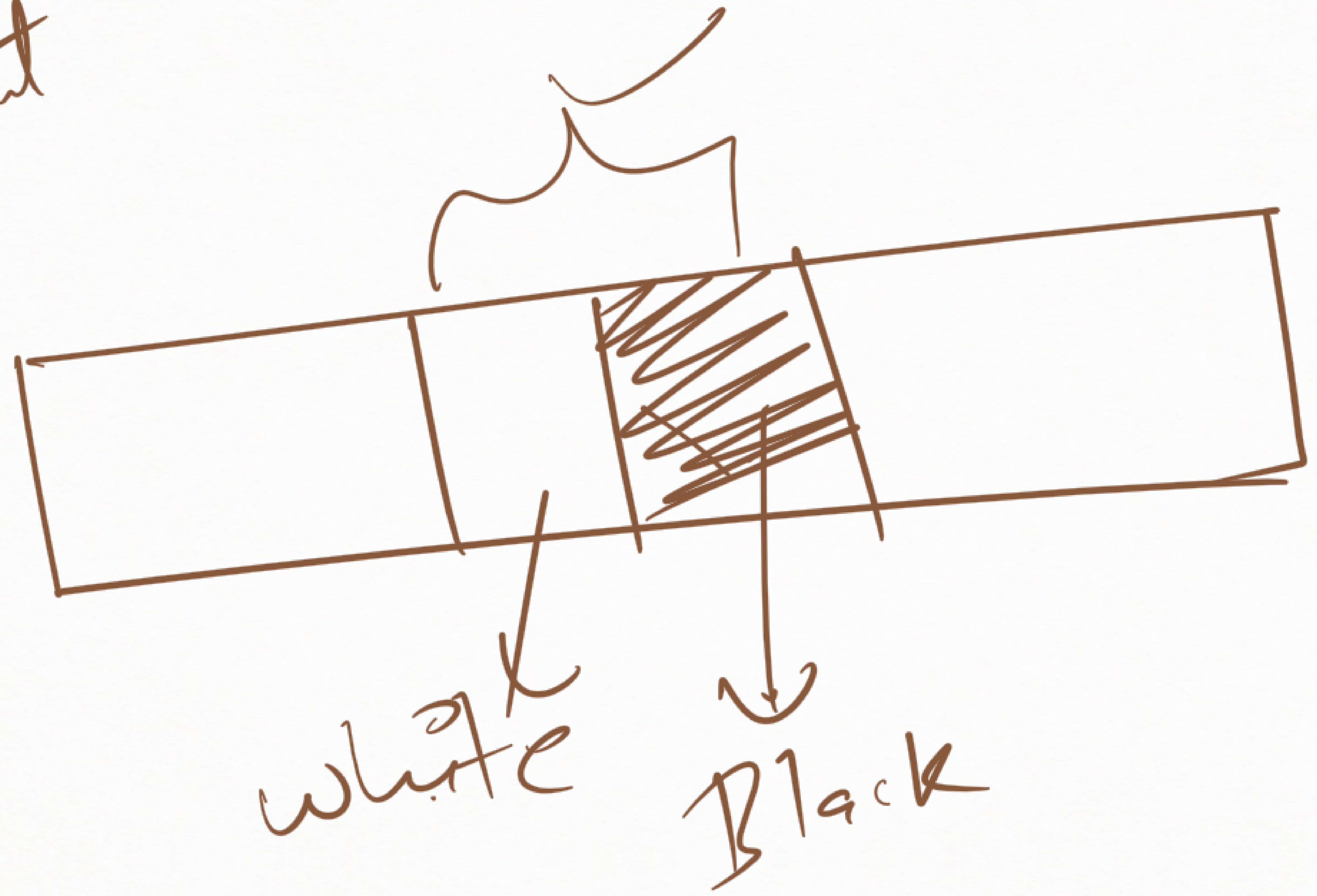



# Recurrence Relation

$$T(n) = T\left(\frac{n}{2}\right) + C$$

Time complexity

Two adjacent cell game




Binary Search

- Polynomial Multiplication (Divide & Conquer)

$$A(x) = 3x^2 + 2x + 5 \qquad 3x^3$$

$$B(x) = 5x^2 + x + 2$$

$$A(x) \cdot B(x) = 15x^4 + 3x^3 + 6x^2 + 10x^3 + 2x^2 + 4x + 25x^2 + 5x + 10$$

$$= 15x^4 + 13x^3 + 33x^2 + 9x + 10$$

$10 + 3 + 10$

$\Downarrow$

$13$

Page rank algo

→ IB Search engine

→ Google

## ultiplying polynomials

Variable

put: Two $n-1$ degree polynomials:

$$a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$$

$$b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1x + b_0$$

tput: The product polynomial:

$$c_{2n-2}x^{2n-2} + c_{2n-3}x^{2n-3} + \cdots + c_1x + c_0$$

where:

$$c_{2n-2} = a_{n-1}b_{n-1}$$

$$c_{2n-3} = a_{n-1}b_{n-2} + a_{n-2}b_{n-1}$$

$$\ldots$$

$$c_2 = a_2b_0 + a_1b_1 + a_0b_2 \quad = 1+1, 0+2, 2+0$$

$$c_1 = a_1b_0 + a_0b_1 = 0+1 = 1$$

$$c_0 = a_0b_0$$

Coefficients

Diff.

$x^1$ $\quad 3 = 1+1+1, \; 0+2+1,$

$x^2 \to$ Power

$A = 3x^2 + 8x + 5$

$B = 5x^2 + x + 2$

$2 = 6x^2 + 2x^2$

$2 = 1+1, \; 0+2, \; 2+0$

# Multiplying Polynomials

mple

: $n = 3$, $A = (3, 2, 5)$, $B = (5, 1, 2)$

$A(x) = 3x^2 + 2x + 5$ 3

$B(x) = 5x^2 + x + 2$

$B(x) = 15x^4 + 13x^3 + 33x^2 + 9x + 10$

put: $C = (15, 13, 33, 9, 10)$

| | | |
|---|---|---|
| 3 | 2 | 5 |

$3x^2 + 2x + 5$

| $x^2$ | $x^1$ | $x^0$ |
|---|---|---|
| 5 | 1 | 2 |

$5x^2 + 1x + 2$

| 4 | 3 | 2 | 1 | |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |

| $x^1$ | $x^0$ |
|---|---|
| 1 | 1 |

$1x^4 + 1x$

$x + 1$

| $x^4$ | $x^3$ | $x^2$ | $x^1$ | $x^0$ |
|---|---|---|---|---|
| 1 | 3 | 7 | 2 | 8 |

$x^4 + 3x^3 + 7x^2 + 2x + 8$

$2 * 3 - 1$

Every polynomial can b written in an Array.

$$3x^2 + 7x^5 + 2$$

| 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| 7 | 0 | 0 | 3 | 0 | 2 |

| 3 | 2 | 1 | 0 |
|---|---|---|---|
| 4 | 0 | 2 | 0 |

$\rightarrow 4x^3 + 2x$

Naive → Algo, without any efficient approach, just as the way we can solve a problem.

→ w/o applying any algo technique like DP, Greedy, Divide & Conq.

Efficient → Opposite of the Above ✓ DP, Greedy DS

Poly($A, B, n$) #Naive

Input: A, B → Polynomial

n → Size of Array.

```
Array[n][n]
rom 0 to n − 1:
 from 0 to n − 1:
r[i][j] ← A[i] * B[j]
← Array[2n − 1]
rom 0 to 2n − 1:
ct[i] ← 0
rom 0 to n − 1:
 from 0 to n − 1:
duct[i + j] ← product[i + j] + pair[i][j]
product
```

9 times for the earlier problem

pair[i][j] → 2D Array

[[ ], [ ]]

Product Array (2n-1)

$1 + n^2 + 1 + n^2$

2 3]

5 6]

$3n^2$

Time Complexity of Naive Approach.

$O(n^2) \to$ multiplications

$O(n^2) \to$ Addition

$O(n^2) \to$ ~~0~~

$n, \log n$

Divide & Conquer App.

$O(n^{1.58})$

→ Why $O(n^{1.58})$ makes it better sense

→ We talk about $n$ values which are very very huge.

$2^{1.58}$ $2^2$

→ $n = 10$ billion

$(10 \text{ billion})^{1.58} << (10b)^2$

# Master Theorem. (Time Complexity topic)

Recurrence Relation (Recursion Algo)

**Theorem**

If $T(n) = aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + O(n^d)$ (for constants $a > 0, b > 1, d \geq 0$), then:

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

① ② ③

$\log_b a$

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n) = O(n^2)$$

## er Theorem Example 1

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n)$$

a

b

d

- $a = 4$
- $b = 2$
- $d = 1$

$d < \log_b a$, $T(n) = O(n^{\log_b a}) = O(n^2)$

Recurrence relation

fun(n)

$O(n^2)$

n task

fun(n/2)

fun(n/2)

fun(n/2)

fun(n/2)

$\log_2 4 = \log_2 2^2 = 2 > 1$

# Recurrence Relation [Recursive Algo]

function(n)
{
print() $\rightarrow$ 1 } $T(n)$
function($\frac{n}{2}$) $\rightarrow T(\frac{n}{2})$
}

Suppose $T(n)$ is the time taken

$T(n) = 1 + T(\frac{n}{2})$

# Linear search

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 7 | 3 | 6 | 5 | 9 |

5 Units of time

| 1 | 2 |
|---|---|
| 7 | 3 |

2 Units of time

$O(n)$ 5=

2

Algo

```
function(n)
{
    for (i = 0 to n)
    {
        print
    }
    function(n/2)
    function(n/2)
    function(n/4)
}
```

$$\text{for}(i=0 \text{ to } n)\{\text{print}\} \rightarrow O(n)$$

$$\text{function}\left(\tfrac{n}{2}\right) \rightarrow T\left(\tfrac{n}{2}\right)$$

$$\text{function}\left(\tfrac{n}{2}\right) \rightarrow T\left(\tfrac{n}{2}\right)$$

$$\text{function}\left(\tfrac{n}{4}\right) \rightarrow T\left(\tfrac{n}{4}\right)$$

Suppose it ta[kes]

$T(n)$ time

$$T(n) = 2T\left(\tfrac{n}{2}\right) + T\left(\tfrac{n}{4}\right) + O(n)$$

fun(n){

print( )

fun(1)

fun(1)

fun($\frac{n}{2}$)

}

$T(n)$

$O(n)$ ✓ n units of time

But $T(n)$ can take an amount of time, tho d... upon n.

$2^n$, log n

just a math f...

$T\left(\frac{n}{2}\right)$

$$\boxed{T(n) = 3O(1) + T\left(\frac{n}{2}\right)}$$

$$T(n) = O(n^2) + T\left(\frac{n}{4}\right)$$

```
function(n){
    for i = 0 to n
    { for j = 0 to n
          print(i, j)
    }
    function(n/4)
}
```

$O(n^2)$

$T(n)$

$T\left(\frac{n}{4}\right)$

## er Theorem Example 2

currence eqn

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$$a = 3$$

$$b = 2$$

$$d = 1$$

$d < \log_b a$,

$= O(n^{\log_b a}) = O(n^{\log_2 3})$

$d = 1$

$a = 3$

$b = 2$

log

$> 1$

$< 2$

$=$

$O$

## er Theorem Example 3

$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

$$a = 2$$

$$b = 2$$

$$d = 1$$

$d = \log_b a,$

$= O(n^d \log n) = O(n \log n)$

$d = 1$

$\log_b a = 2 = 1$

$b = 2$

$O(n^1 \log n)$

## er Theorem Example 4

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

$$a = 1$$

$$b = 2$$

$$d = 0$$

$d = \log_b a$, $T(n) = O(n^d \log n) =$
og $n) = O(\log n)$

$O(n^0) = O(1)$

$O(n^0 \log n) = O(\log$

$a = 1$

$b = 2$

$d = 0$

$n^0 = 1$

## ter Theorem Example 5

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n^2)$$

$a = 2$

$b = 2$

$d = 2$

$d > \log_b a$, $T(n) = O(n^d) = O(n^2)$

$a = 2$

$\log$

$b = 2$

$= 1$

$O(n^d)$

$d = 2$

$2 > 1$

$O(n^2)$

orem

$$n) = aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + O(n^d) \text{ (for constants}$$

$$b > 1, d \geq 0), \text{ then:}$$

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

The bigger one is kept

$O(n^{\text{bigger one}})$

Deve

Webhooks

Payload

API

Output/Response

Reverse API

Event

Web hook

...om
...on
...s

Payload

API

Person

Semantic Analysis

Ganga is taking bath

in Ganga

River

NLP