

Input 4

Input 4  
 $A = [3 \quad 4 \quad 5 \quad 3] \quad n$

Output: 5 5 3 -1

for  
s

A hand-drawn diagram on a white background. It features a large circle with a horizontal chord drawn through its center. The left endpoint of the chord is marked with a small 'X'. A vertical line segment connects the midpoint of the chord to the center of the circle. On the right side of the circle, there is a vertical bracket with a small '1' at its right end, likely indicating the radius or diameter of the circle.

index  $i (0 - n)$

for index (1 - n)

find\_max

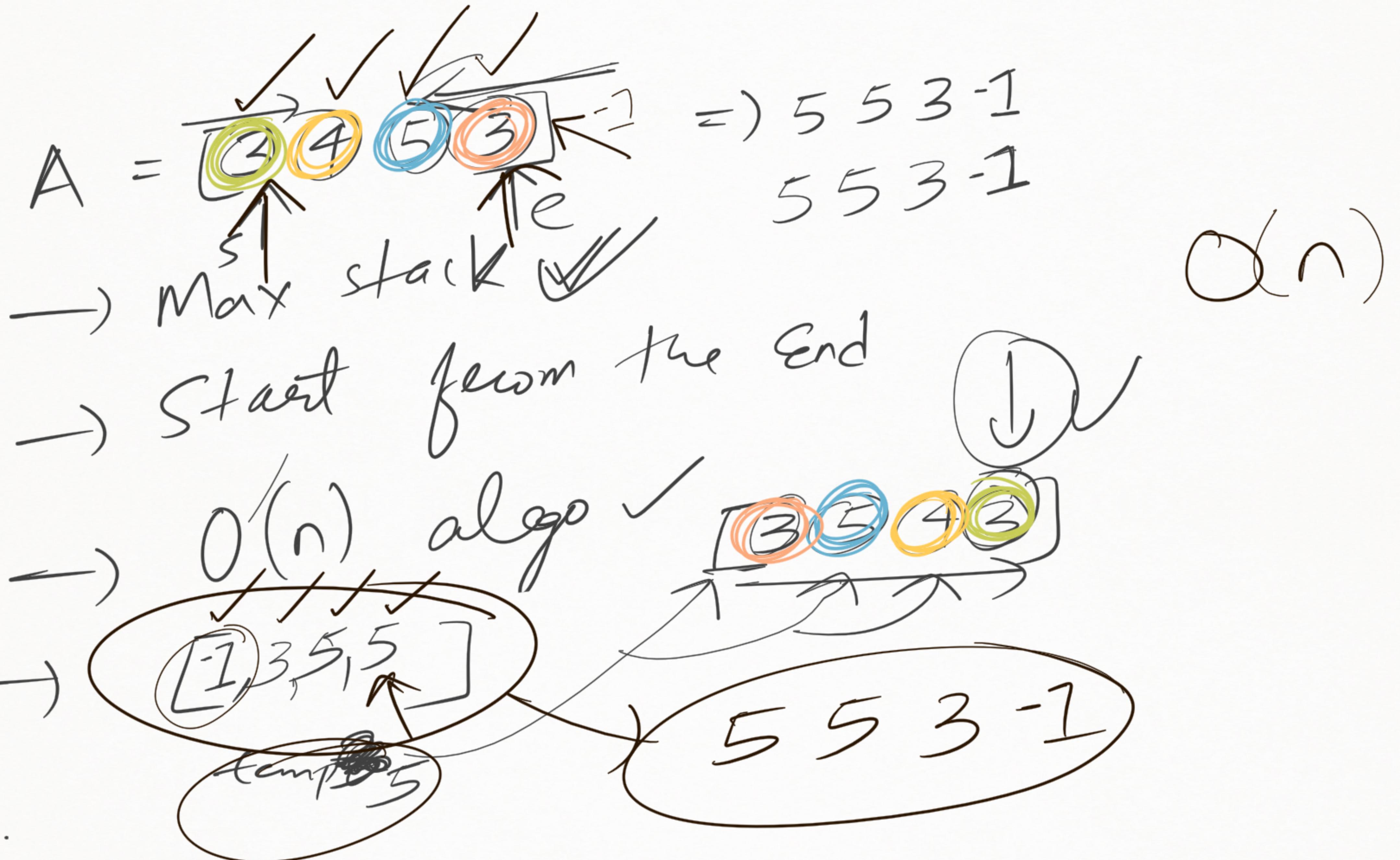
Takes a lot

of time

$O(n^2)$

A hand-drawn diagram consisting of several black ink lines. On the left, there is a large, open upward-pointing arrow. To its right, at the top, are two small, closed circular arcs. Below these, further to the right, are two parallel horizontal lines.

2. Beste farce  
Naïve



## # Recurit Recursion

Problem: Sum all values from 1 to n.

```
sum_natural(n)
{ }
```

Solution Step 1) Keep in mind what fun should do?

$$1 + 2 + \dots + n$$

~~Step~~ Pick a sub-prob & assume that your fun  
already solves it

$$\text{sum\_natural}(n-1) \dots \text{sum\_natural}(n-2)$$
$$\dots \text{sum\_natural}(1)$$

Choose the closest subproblem to the original prob  
 $\Rightarrow n-1$

$\text{sum\_natural}(n)$

{  
   $\text{sub\_sol} = \text{sum\_natural}(n-1)$   
}

Step 3) Take sub-sol & use it to solve  
using prob.

Defn = I to  $n$  ✓

Sub-sol = I to  $n-1$  ✓ + n  
sum\_natural(n)

{ sub-sol = sum\_natural( $n-1$ )  
return  $n + \text{sub-sol}$   
}

Step 4.) Base Case

→ (termination condition)

→ Ending Condition

→ Usually it is if else in  
the beginning.

→ Edict Case, that can be  
written like Hard code

$n \rightarrow n-1 \rightarrow n-2 \dots$  ← ① ~~2~~

If  $n == 1$   
return 1

```
sum_natural(n)
{   if n == 1
    return 1
    sub_sol = sum_natural(n-1)
    return n + sub_sol
}
```

## Example ② Reverse a String Using Recursion

①

Input = qweat~~y~~

Output = ~~yt~~reuwq

reverse-string (input-str)

{

}

Step 2) Sub-prob?

In → qweerty  
Op = gfreng

String without last char

reverse\_string(inp\_star)

{

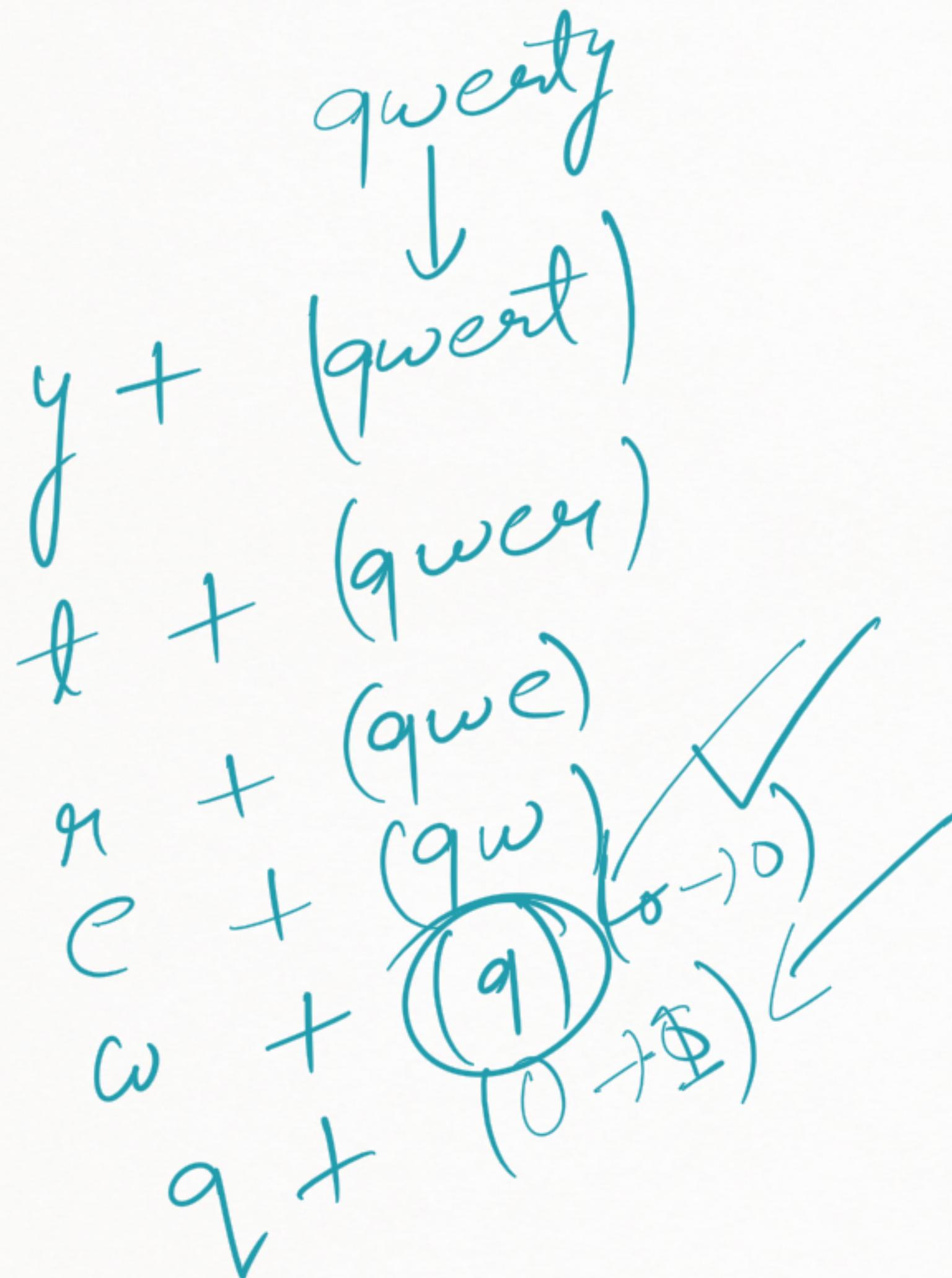
sub-sol = reverse\_string(inp\_star  
[0 → end - 1])

}

Step 3

$\text{sub\_sol} = \text{?inp-star}(0 \rightarrow \text{end}-1)$       *Ouiq*  
 $I|P = \boxed{\text{qweat}}$        $I|P = \text{qweaty}$   
 $O|P = \boxed{\text{treewq}}$        $O|P = \boxed{\text{ytrewq}}$   
                ↓  
 $\text{reverse-star}(\text{?inp-star})$   
{       $\text{sub\_sol} = \text{reverse-star}(\text{?inp-star } (0 \rightarrow \text{end}-1))$   
        return  $\text{?inp-star}(\text{end}) + \text{sub\_sol}$   
        }  
    }

Step 4) Bare Calc



0 → end-1

?if Len(^inp-star) = -1  
return ^inp-star



I|P = qweerty  
O|P = yfewq

reverse\_steing( $\hat{\text{inp\_str}}$ )

{ if  $\text{len}(\hat{\text{inp\_str}}) == 1$   
return  $\hat{\text{inp\_str}}$ .

$\text{sub\_sol} = \text{reversesteing}(\hat{\text{inp\_str}}[0 \rightarrow \text{end}-1])$

return  $\hat{\text{inp\_str}}(\text{end}) + \text{sub\_sol}$

}

Example \*

## Common Subsequence

String 1

= abcde~~fghij~~

X <sup>char</sup>  
q com sub  
have

String 2

= c~~e~~hgi

Both  
to be in  
same

✓  
C, e

ce ~~gj~~

order

egj

If I say egj is  
a common subseq.  
than egj should appear  
in the same order egj  
in both strings.

String 1 = apopxime  
String 2 = apple

# Largest common subseq.

out of all

String 1 = apopxime  
String 2 = apple

LCS

able

LCS  $\Rightarrow$  apple

(4) apple?

O/P: length  
of LCS

A = apopxime  
B = apple

String 1 = abcde~~fghi~~<sup>j</sup> LCS

String 2 = e~~c~~d~~q~~i

Same  
order  
important

is important

abcde~~fghi~~<sup>j</sup>

e~~c~~d~~g~~i

$$A = \begin{array}{|c|c|} \hline \circ & 1 \\ \hline b & d \\ \hline \end{array}$$

$$B = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline \end{array} .$$

0 1 2 3

~~Step A, B~~ LCS( $i, j$ )

$\{$

Brute force/Naive?

$$A = \cancel{\begin{array}{c} \circ \\ b \\ d \\ e \end{array}} \overset{i}{\cancel{e}}$$

$$B = \cancel{\begin{array}{cccc} a & x & \cancel{d} & e \end{array}}$$

$\uparrow$

Input  $\Rightarrow$  Pointers to both strings  
 O/P  $\Rightarrow$  length of lcs

return  $\Rightarrow$  length of lcs



Recursion with ~~optimisation~~  
Decision → Largest

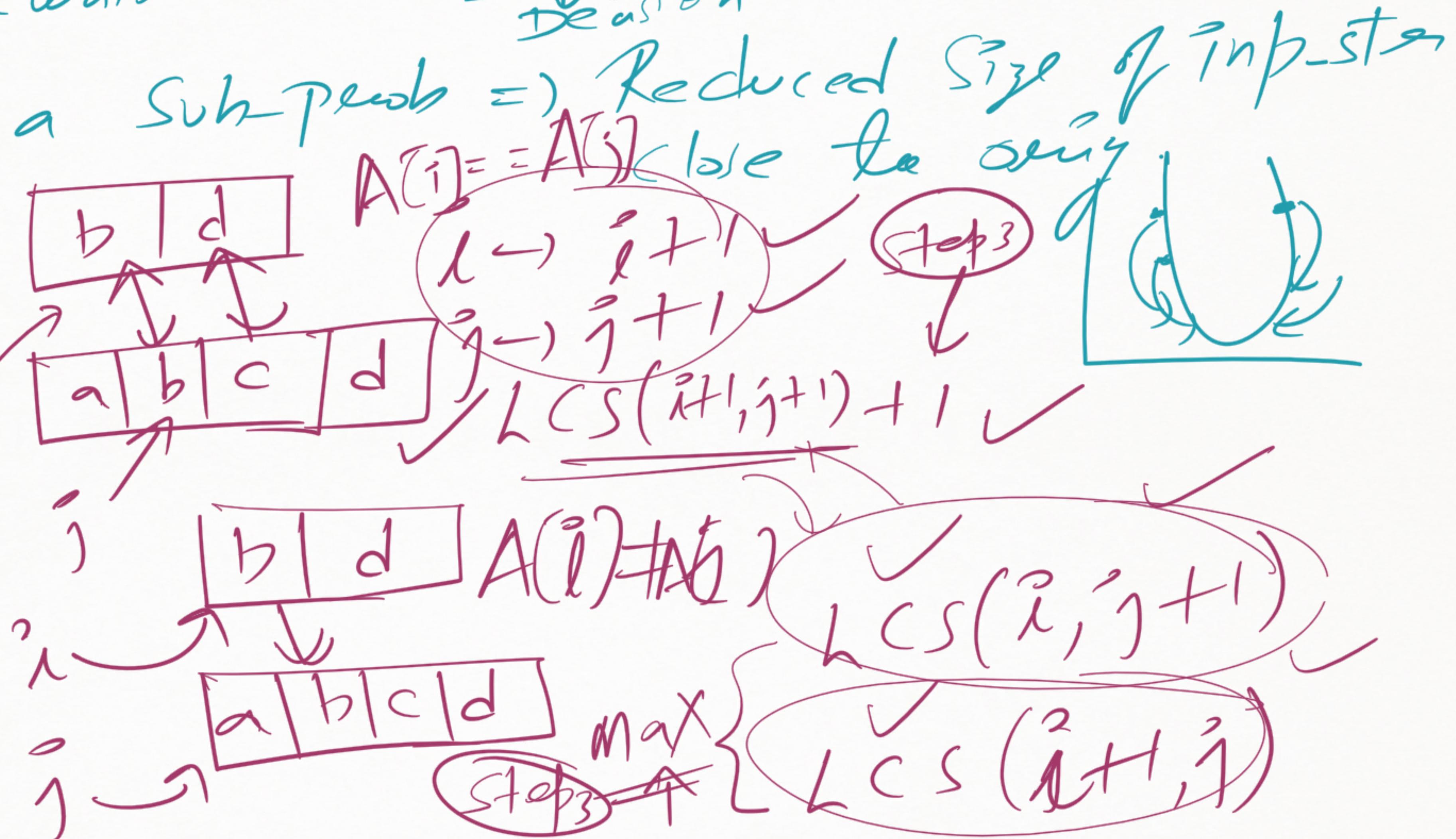
Step 2

Pick a subprob

$A =$	b	d		
$B =$	a	b	c	b
$i \rightarrow$	↑	↑	↑	↑

Match?

Non Match?



~~Step 2  
Continue~~

$LCS(i, j)$

{

if ( $A[i] = A[j]$ )

sub-sol =  $LCS(i+1, j+1)$

else

sub-sol-1 =  $LCS(i+1, j)$

sub-sol-2 =  $LCS(i, j+1)$

}



$LCS(i, j)$

}

if  $A[i] == B[i]$

sub-sol =  $LCS(i+1, j+1)$

else return  $1 + \text{sub-sol}$

sub-sol-1 =  $LCS(i+1, j)$

sub-sol-2 =  $LCS(i, j+1)$

return  $\max(\text{sub-sol-1}, \text{sub-sol-2})$

}

Step 4 Base Case

~~A = D(1)~~

Tab [cd] ↑

B =

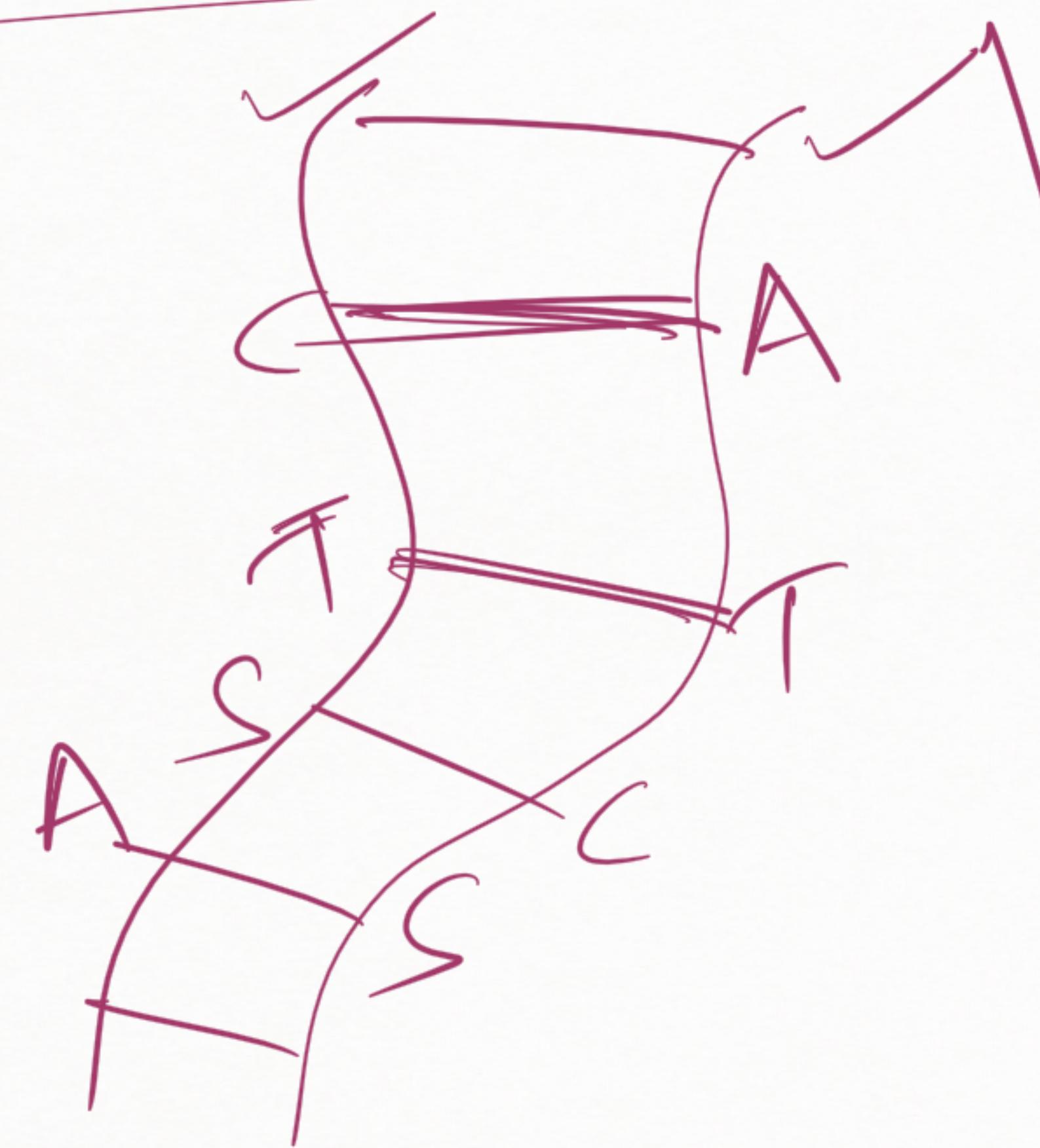
if anyone of them  
becomes empty | end  
if  $i == \text{len}(A)$  or  
 $j == \text{len}(B)$   
return 0

✓

LCS( $i, j$ )

```
{   if  $i = \text{len}(A)$  or  
     $j = \text{len}(B)$  return 0  
  else if  $A[i] == B[j]$ ,  
    return 1 + LCS( $i+1, j+1$ )  
  else  
    return max(LCS( $i+1, j$ ), LCS( $i, j+1$ ))  
}
```

DNA Strands



App of LCS

Edit Distance

Integrity  
of Matrix

Data Management  
Molit Systems