→ We will be solving problems. (Puzzle + math + coding)

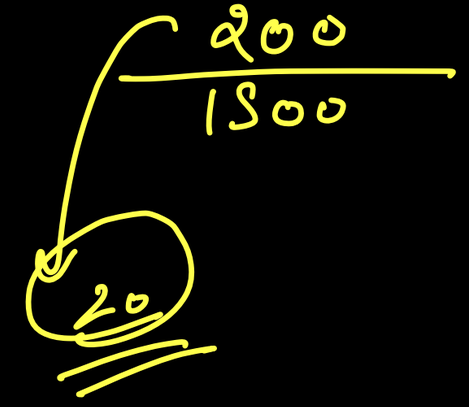
→ Relevant to interviews

→ Donot make notes in class.

→ Revise every class day wise.

→ Every class → HW.

→ Leetcode



20
↓
understood
while solving
noon

200
actually

1500
think

Most Recommended

[99% 1%] → fake fake

C++ and Java → they are fast

Show their code

Online → [20% → Do not ca
30% Ca... by]

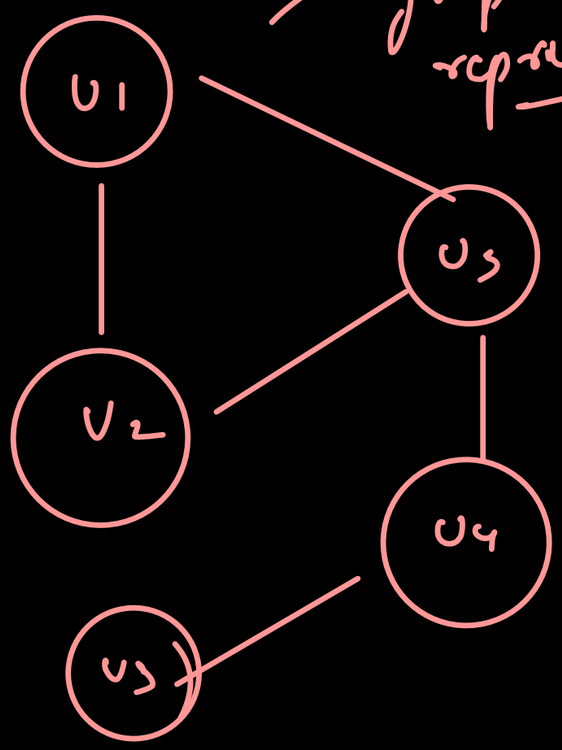
Python and DS → app driven

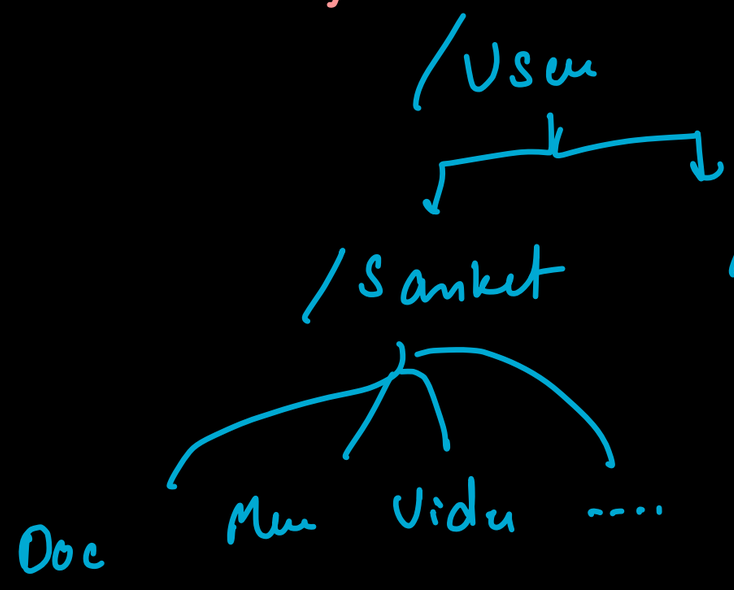
Do DSA with DS

→ 7-8 hr face tutorial or Java (By next week) Maketing
watch that.

(DSA) Data Structures & Algorithms

↳ Data Structures → folder



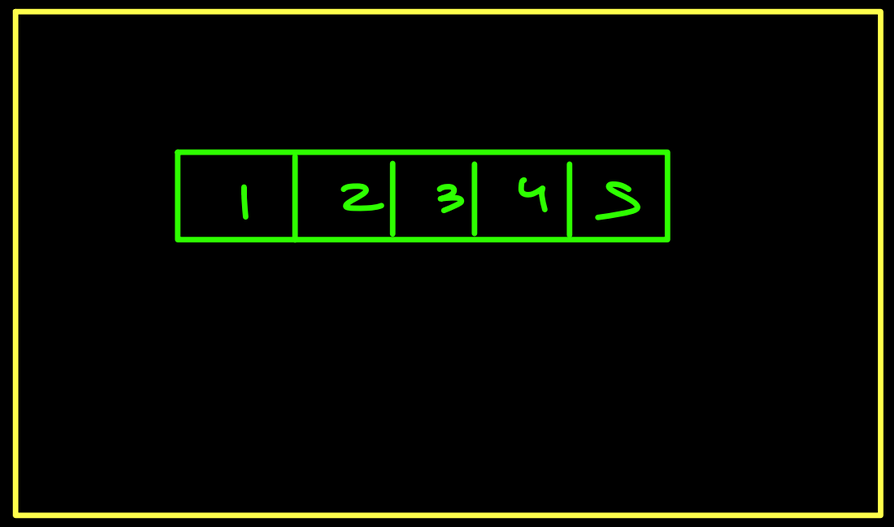


hierarchy

Data structures are storage models which helps us to store data for diff use cases.

↳ arrays

[1, 2, 3, 4, 5] → linear fashion



arrays
linked list
Stacks
queue
trees
etc

Algorithms → Defined steps that helps us to achieve a task is called algorithm.

Problem-1 → 2 sum

return array of indices of two pairs.

nums → [2, 7, 11, 15] → idx (0, 1, 2, 3)

target → 9

ans → [0, 1]

from this array find a pair of elements that sums up to target.

(Do not use same element twice)

$2 \leq$ length of array $\leq 10^4$

$-10^9 \leq$ element of array $\leq 10^9$

$-10^9 \leq$ target $\leq 10^9$

TLE → time limit exceeded

1 sec to run

In one sec approx 10^8 instructions we can execute.

Pair → target

$[a_0\ a_1\ a_2\ a_3\ \ldots\ a_{n-1}]$

worst solution / brute force

why not prepare all possible pairs & check the Sum.

i j

0 1 2 3 → index

[2, 7, 11, 15]

(2,7) (7,11) (11,15)

(2,11) (7,15)

(2,15)

j → 2nd elt of pair

i → first value of pair

(2

we can fix the first element of the pair & then combine it with the remaining elements.

n ← length of arr

```
for (i=0; i<=n-2 ; i++) {
    for (j=i+1 ; j<n ; j++) {
        print (arr[i], arr[j])
    }
}
```

print all pairs from arr

outer

n-2 times

```
for (i=0; i<=n-2 ; i++) {
    for (j=i+1 ; j<n ; j++) {
        if (num[i] + num[j] == target) {
            return [i,j];
        }
    }
}
```

last

in every internal loop iteration we execute 3-4 instructions

inner & space

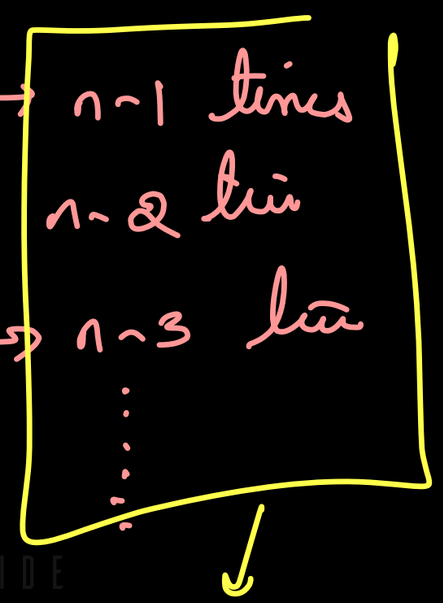
outer

i = 0 → internal loop → n-1 times

i = 1 → internal loop → n-2 times

i = 2 → " " → n-3 times

⋮



Total iterations

$$(n-1) + (n-2) + (n-3) + \cdots\cdots 2 + 1$$

Sum of first $(n-1)$ natural no $\Rightarrow \dfrac{(n)(n-1)}{2}$

4 Proof

Total instructs is approx $\dfrac{2 \times (n)(n-1)}{2}$

$\rightarrow 2(n)(n-1)$

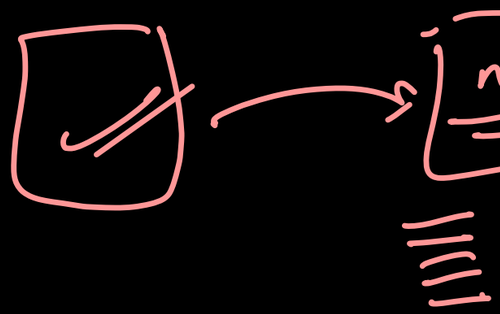
$\rightarrow 2 \times 10^4 (10^4 - 1) \approx 10^8$

Note → Never randomly solve array & strings problems:

DSA → maths → Trigonometry → 10th

techniques → Examples → Exercises → J-J

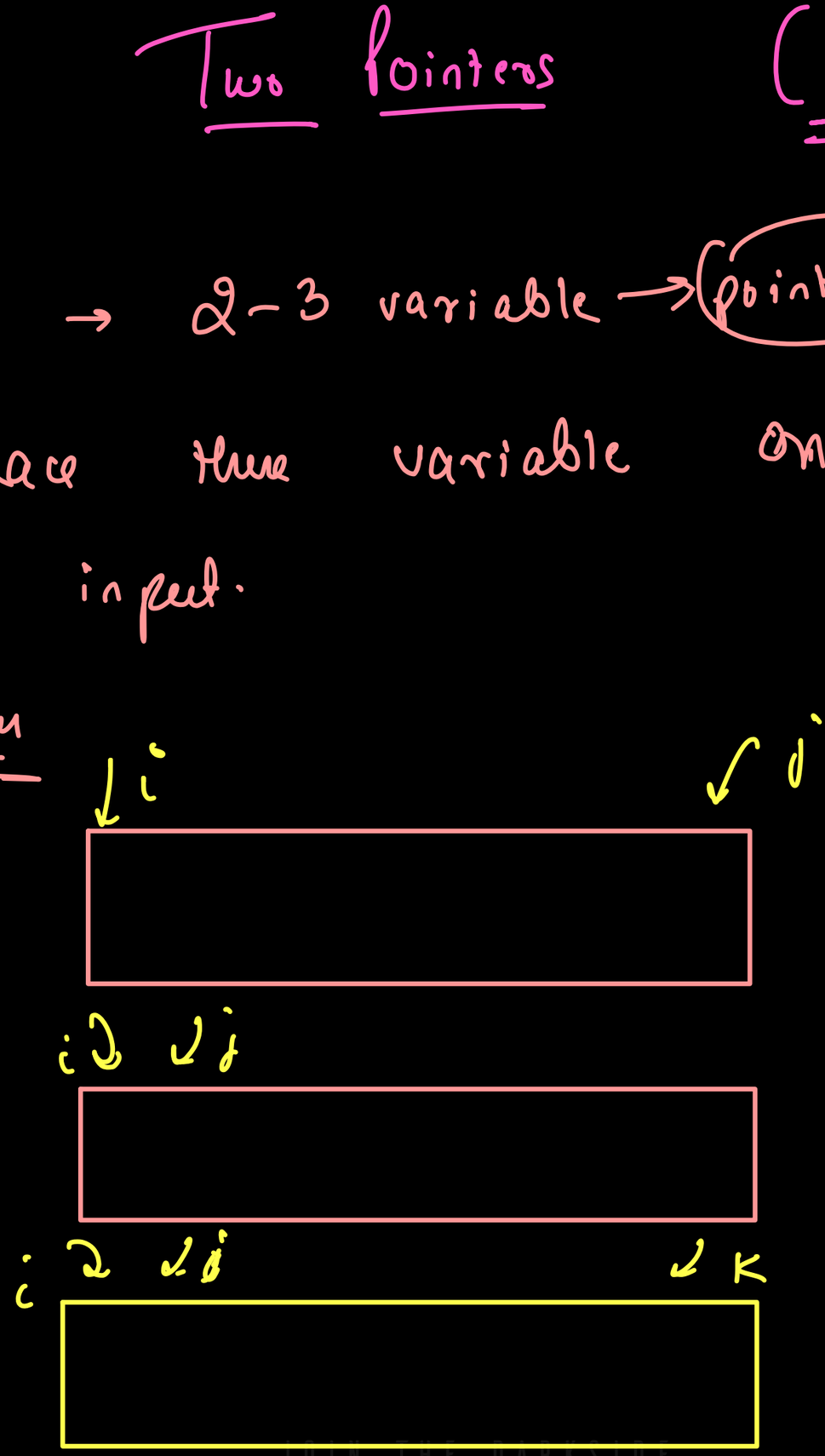
>100-200



Two Pointers (technique)

- prepare → 2-3 variable → (pointers)
- you place these variable on diffent parts of the input.
- now then

while moving you try to discard some part of data set



2 → Problem - 344 (Reverse a String)

→ S = ["h", "e", "l", "l", "o"] ["o", "l", "l", "e"]

ans

array

["o", "l", "l", "e", "h"]

#brute force → because we want data of array in reverse order, we can read the array in reverse order. And, one by one store the data in a new array.

```
n = s.length;
ans = [];

for ( i = n-1 ; i >= 0 ; i--) {
        ans.push (s[i]);
}

return ans;
```

Can we improve ??

↳ we are taking an extra space in memory because of the array. We can avoid it.

$i = 0$

$j = n-1$

i $\quad\quad\quad\quad$ j

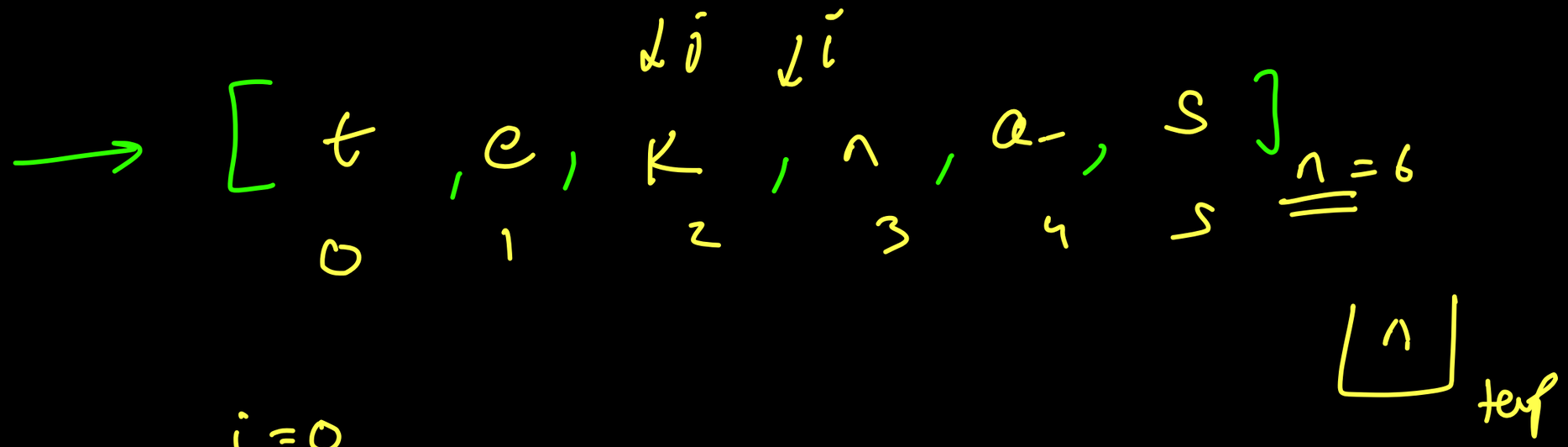
$[\ h, e, l, l, o\ ]$

$[0, i]$ $\longrightarrow$ left region

$[j, n-1]$ $\longrightarrow$ right region

already reversed

$[i+1, j-1]$ $\longrightarrow$ this region needs to be reversed




```
i = 0
j = n-1
while ( i <= j ) {
    temp = s[i]
    s[i] = s[j]
    s[j] = temp
    i++;
    j--;
}
```

Swapping

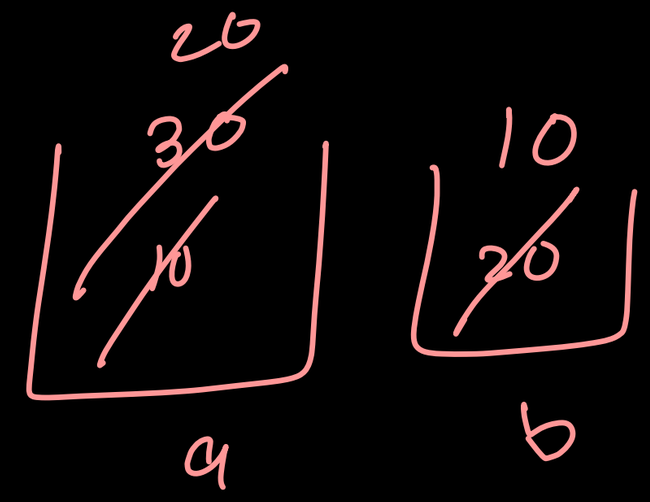
a = 10
b = 20

→ bitwise
→ arithmetic

a = a + b

b = a - b

a = a - b



Problem - 167

Can we optimise ??

target ⇒ 15

2^{i} j^{i}

[1, 3, 5, 7, 11, 16, 22]

i = 0

j = n-1

in each iteration you are eliminating one elem

```
while (i < j) {
    if (nums[i] + nums[j] == target)
        return [i+1, j+1]
    else if (nums[i] + num[j] > target)
        j--;
    else
        i++;
}
```

total iters → n

4 × n

4 × n

n ≤ 3 × 10^4

4 × 3 × 10^4

→ 12 × 10^4 ≃ 10^5

y