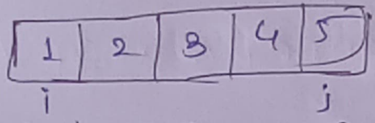
90-DAYS DSA PATTERN MASTERY CHALLENGE !!

Two Pointer Pattern

Dsa_journey_with_Vaishnavi

Day 1 Two Pointer Pattern 24/12/25

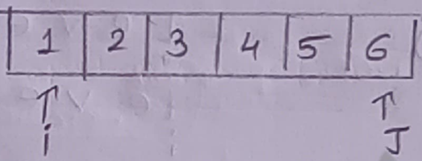


used in Linear Data Structure
- array
- strings
- Linked List

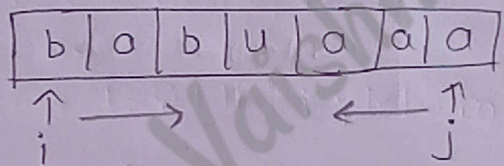
Defination :-

A Technique where two variables (pointers) traverse a Data structure (like an array, string & Linked List) simultaneously different positions.

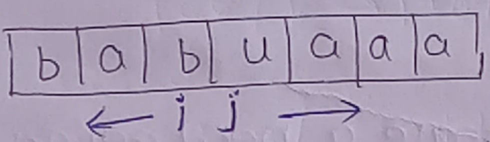
Linear Data Structure



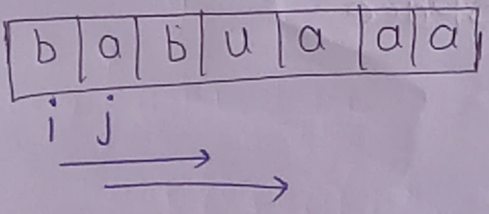
Two pointer either move towards each other, away from each other, or in the same direction



- Towards



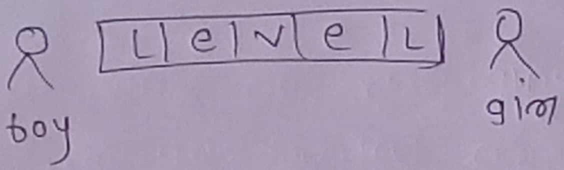
- Away



→ Same/ Fast

Efficiently Solving Problems involving comparisons, searching or partitioning → $\boxed{\text{PCS}}$ ☆

Ex - palindrome



boy's see → Level } same paildrome
girl see → Level

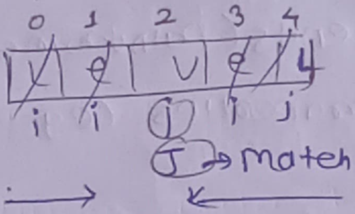
Brute Force way :-

```
S = "Level"                            matten
S_~~temp~~ rev = S.reverse ()          L e v e L
                                       L e v e L
i=0 :                                  i

while i<S.length
    IF S[i] != S_rev [i];
        print ("No");
        return
    i++
print ("yes");
```

TC = O(n)

optimization :-



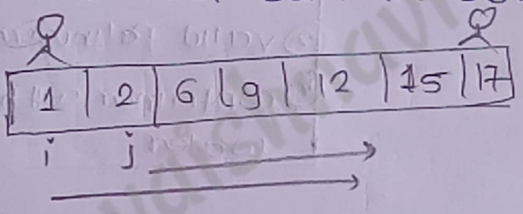
→ s = "level"
i = 0
j = s.length - 1
(5-1) → 4

```
while i<j:
  If s[i] != s[j]:
    Print ("No palindrome");
    return
    i = i+1;
    j = j-1;
print ("yes palindrome");
```

why use Two pointers :-

① No unncescessary iterations :-

Eg - Target sum in sorted array



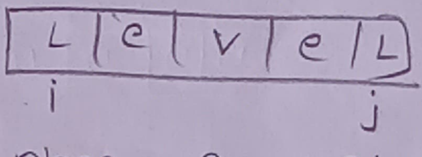
$x + y = target$

Brute force = $O(n^2)$

Two pointer = $O(n)$

② Early stopping as soon as mismatch / condition fails

Eg- Palindromes



③ works in place - no need to create new arrays or copies → No extra space create.

→ Eg palindroms

④ Improves Both speed & space efficiency

speed $O(n^2) \rightarrow O(n)$

space $(On) - (O(1))$

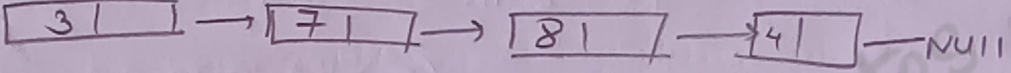
When to use it :-

1) Comparing elements from Both ends (Palindrome, Pair sum)

2) Finding Pairs with specific properties (eg
sum = target)

3) Removing or merging (eg remove Duplicates,
merge sorted arrays)

4) Detecting loops or middle elements
   - Linked list cycle
   - middle Node.



Template :-
+ Input, opposite side se Dono Pointer

```
a = [1, 4, 3, 2, 9]
     ↑           ↑
     i           j

left = 0
Right = a.length - 1

while left < right
    IF ~~left~~ condition ;
        left = left + 1 ;

else
    Right = Right - 1 ;

return ans
```

Examples
1. Two sum - Array sorted
2. Remove Duplicates from sorted array
3. valid palindrome
4. container with most water.

2 Inputs, Exhaust Both $\rightarrow$ exit

```
a = [1, 4, 3, 2, 9]
b = [9, 7, 6, 2, 0, 8]

left = 0
Right = 0

while left < a.length & Right < b.length
    IF Condition :
        left = left + 1 ;
    else
        Right = Right + 1
while left < a.length
    left = left + 1
while right < b.length;
    right = right + 1
```

Eg.

```
1   4   6   9   10
i   i   i   i   i
9   10  16  22  30
j   j   j
```

merge [1, 4, 6, 9, 9, 10, 16, 22, 30]

here i compare with j if i small then put in ans