Minimum Characters to Add at front for Palindrome

Task :-

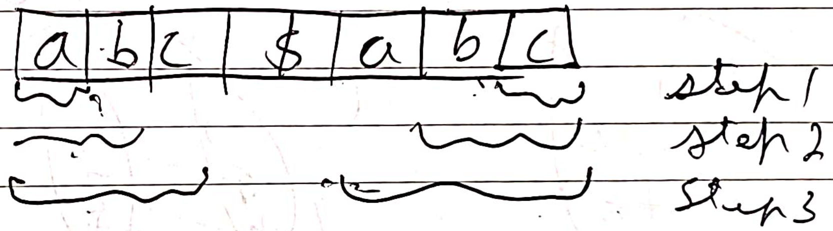
Given a string find the min number of charactors to be added to make palindrom

"abc" ⇒ $\boxed{cb}$abc

= $\underline{\underline{2}}$

"aacecaaaa" ⇒ $\boxed{aa}$ aacecaaaa

= $\underline{\underline{2}}$



index	Char	LPS value	Explanation
0	a	0	No prefix-sufix
1	b	0	No match with a
2	c	0	No match a or b
3	$	0	No match
4	a	0	No match
5	b	0	No Match
6	c	1	Matches with 'a'

LPS arr : [0, 0, 0, 0, 0, 0, 1]

To Find Min chars to Add

get the last value of LPS array

LPS[last] = 1

- Minimum chars to add = len(s) - LPS[last]

$\Rightarrow 3 - 1 = 2$

∴ To Make "abc" a palindrome we need to add "cb" at the front, resulting in "cbabc",

Pseudo Code:

```
Function minCharsToAdd(s):
        newstr = S + '$' + reverse(s)
        LPS = computedLPS(newStr)
        min_chars = length(S) - LPS[last]
        Return min_chars.

Function computeLPS(str):

        n = length(str)
        Initialize LPS array with all 0
        j = 0

        for i from 1 to n-1

            while j > 0 and str[i] .!= str[j]
                j = LPS[j-1]

            If str[i] == str[j]:
                j = j+1
                LPS[i] = j
    Return LPS[n-1]
```

Zig Zag Tree Traversal

Shapun S Poonja
1RV24MC098

Problem:

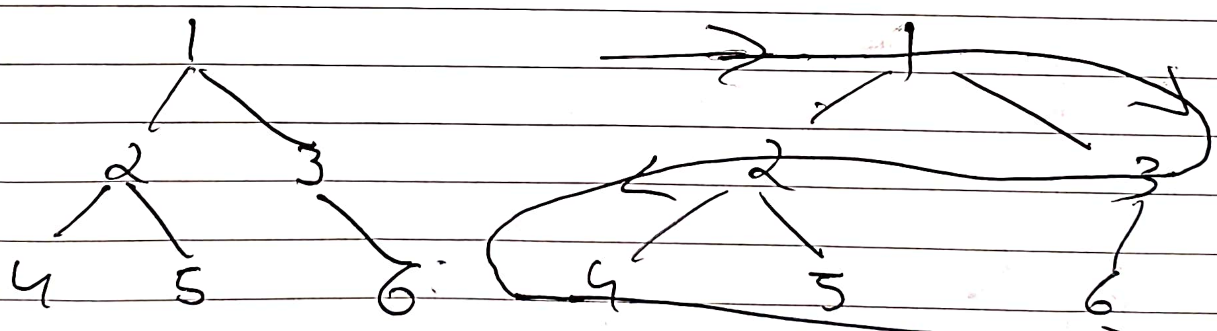
Gien a pinary tree print Zig-Zag level order Traversal this means

* The first level (root) is traversad from left to right

* The second level is traversad from right to left

* The third level is is traversed from ~~a~~ left to right and so on

Example



output:- 1 → 3 → 2 → 4 → 5 → 6

Using BFS (Breadth First Search)

steps to solve

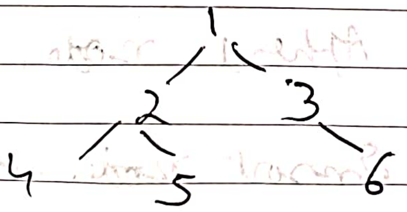
1. Use a Queue to store the nodes level by level

2. Maintain a boolean flag to switch between normal & reversed order

3. For each level;
   - if traversing left to right, add children normally (left → right).
   - If traversing right → left, add children in reverse order (right to left)

4. Print the final traversal order

Example Walkthrough



Level	Nodes Traversed	Order
1	1	Left to Right
2	3, 2	Right to Left
3	4, 5, 6	Left to Right

⇒ 1 3 2 4 5 6

Pseudo Code:

```
Function ZigZag(root);
    if root is Null:
        Return empty list

    Initialize result as empty array
    Initialize queue as empty queue
    Add root to queue
    set leftTo right = True

    while queue is not empty;
        Initialize level as empty list
        Get size of Queue

        Loop i from 1 to size;

            node = Remove front node

            If left to Right

                Append node.value to level list
            Else
                Insert node.value at begining

            if node.left is not NULL:
                Add node.left to Queue

            if node.right is not NULL:
                Add node.right to queue.
    Append all elements to level list
        Toggle leftTo Right
Return result.
```