ZigZag Conversion

Problem

The string “PAYPALISHIRING” is written in a zigzag pattern on a given number of rows like this:

1	2	3	4	5	6	7
P		A		H		N
A	P	L	S	I	I	G
Y		I		R		

And then read line by line: “PAHNAPLSIIGYIR”

example 1:

Input: s = “PAYPALISHIRING”, numRows = 3
Output: “PAHNAPLSIIGYIR”

example 2:

Input: s = “PAYPALISHIRING”, numRows = 4
Output: “PINALSIGYAHRPI”

Solution

Approach 1: Sort by Row

Intuition

By iterating through the string from left to right, we can easily determine which row in the Zig-Zag pattern that a character belongs to.

Algorithm

We can use $min(numRows, len(s))$ lists to represent the non-empty rows of the Zig-Zag Pattern.

Iterate through s from left to right, appending each character to the appropriate row. The appropriate row can be tracked using two variables: the current row and the current direction.

The current direction changes only when we moved up to the topmost row or moved down to the bottommost row.

Java

```
class Solution {
    public String convert(String s, int numRows) {

        if (numRows == 1) return s;
```

```
        List<StringBuilder> rows = new ArrayList<>();
        for (int i = 0; i < Math.min(numRows, s.length()); i++)
            rows.add(new StringBuilder());

        int curRow = 0;
        boolean goingDown = false;

        for (char c : s.toCharArray()) {
            rows.get(curRow).append(c);
            if (curRow == 0 || curRow == numRows - 1) goingDown = !goingDown;
            curRow += goingDown ? 1 : -1;
        }

        StringBuilder ret = new StringBuilder();
        for (StringBuilder row : rows) ret.append(row);
        return ret.toString();
    }
}
```

Python

```
class Solution(object):
    def convert(self, s, numRows):
        """
        :type s: str
        :type numRows: int
        :rtype: str
        """
        if numRows == 1:
            return s
        rows = ["" for i in range(numRows)]
        curRow = 0
        goingDown = False
        for i in range(len(s)):
            rows[curRow] += s[i]
            if (curRow == 0 or curRow == numRows - 1):
                goingDown = ~goingDown
            curRow += 1 if goingDown else -1

        ret = ""
        for row in rows:
            ret += row

        return ret
```

Complexity Analysis

- Time Complexity: $O(n)$, where $n == len(s)$
- Space Complexity: $O(n)$

Approach 2: Visit by Row

Intuition

Visit the characters in the same order as reading the Zig-Zag pattern line by line.

Algorithm

Visit all characters in row 0 first, then row 1, then row 2, and so on...

For all whole numbers k,

- Characters in row 0 are located at indexes $k(2 * numRows - 2)$
- Characters in row $numRows - 1$ are located at indexes $k(2 * numRows - 2) + numRows - 1$
- Characters in inner row i are located at indexes $k(2 * numRows - 2) + i$ and $(k + 1)(2 * numRows - 2) - i$

Java(Official)

```
class Solution {
    public String convert(String s, int numRows) {

        if (numRows == 1) return s;

        StringBuilder ret = new StringBuilder();
        int n = s.length();
        int cycleLen = 2 * numRows - 2;

        for (int i = 0; i < numRows; i++) {
            for (int j = 0; j + i < n; j += cycleLen) {
                ret.append(s.charAt(j + i));
                if (i != 0 && i != numRows - 1 && j + cycleLen - i < n)
                    ret.append(s.charAt(j + cycleLen - i));
            }
        }
        return ret.toString();
    }
}
```

Java(My Own)

```
class Solution {
    public String convert(String s, int numRows) {
        String news = "";
        if(s.length() <= 0 || numRows == 1)
            return s;
        int index = 0;
        for(int i = 0;i<numRows;i++){
            index = i;
            Boolean flag = true;
            int fold = 1;
            if(i==numRows-1)
                fold = 2;
            while(index<s.length()){
                news += s.charAt(index);
                index = (fold * (numRows - 1) - index) * 2 + index;
                if(i==0 || i==numRows-1)
                    fold += 2;
                else
                    fold ++;
            }
        }
        return news;
    }
}
```

Python

```
class Solution:
    def convert(self, s, numRows):
        """
        :type s: str
        :type numRows: int
        :rtype: str
```

```
        """
        if numRows == 1:
            return s
        strlen = len(s)
        cycleLen = 2 * numRows - 2
        ret = ""
        for i in range(numRows):
            for j in range(0, strlen - i, cycleLen):
                ret += s[j+i]
                if (i != 0  and i != numRows - 1 and j + cycleLen - i < strlen):
                    ret += s[j + cycleLen - i]

        return ret
```

Complexity Analysis

- Time Complexity: $O(n)$, where $n == len(s)$
- Space Complexity: $O(n)$