6. ZigZag Conversion

June 11, 2018 | 189.9K views

Previous Next

The string `"PAYPALISHIRING"` is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

```
P   A   H   N
A P L S I I G
Y   I   R
```

And then read line by line: `"PAHNAPLSIIGYIR"`

Write the code that will take a string and make this conversion given a number of rows:

```
string convert(string s, int numRows);
```

Example 1:

```
Input: s = "PAYPALISHIRING", numRows = 3
Output: "PAHNAPLSIIGYIR"
```

Example 2:

```
Input: s = "PAYPALISHIRING", numRows = 4
Output: "PINALSIGYAHRPI"
Explanation:

P     I    N
A   L S  I G
Y A   H R
P     I
```

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 $\min(\text{numRows}, \text{len}(s))$ 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.

C++ | Java | Copy

```cpp
class Solution {
public:
    string convert(string s, int numRows) {

        if (numRows == 1) return s;

        vector<string> rows(min(numRows, int(s.size())));
        int curRow = 0;
        bool goingDown = false;

        for (char c : s) {
            rows[curRow] += c;
            if (curRow == 0 || curRow == numRows - 1) goingDown = !goingDown;
            curRow += goingDown ? 1 : -1;
        }

        string ret;
        for (string row : rows) ret += row;
        return ret;
    }
};
```

Complexity Analysis

- Time Complexity: $O(n)$, where $n == \text{len}(s)$
- 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 $k\ (2 \cdot \text{numRows} - 2)$
- Characters in row $\text{numRows} - 1$ are located at indexes $k\ (2 \cdot \text{numRows} - 2) + \text{numRows} - 1$
- Characters in inner row i are located at indexes $k\ (2 \cdot \text{numRows} - 2) + i$ and $(k+1)(2 \cdot \text{numRows} - 2) - i$.

C++ | Java | Copy

```cpp
class Solution {
public:
    string convert(string s, int numRows) {

        if (numRows == 1) return s;

        string ret;
        int n = s.size();
        int cycleLen = 2 * numRows - 2;

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

Complexity Analysis

- Time Complexity: $O(n)$, where $n == \text{len}(s)$. Each index is visited once.
- Space Complexity: $O(n)$. For the cpp implementation, $O(1)$ if return string is not considered extra space.

‹ 1 2 3 4 5 6 ... 15 16 ›