Problem 6: Zigzag Conversion

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

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

Example 3:

Input:

s = "A", numRows = 1

Output:

"A"

Constraints:

1 <= s.length <= 1000

s

consists of English letters (lower-case and upper-case),

','

and

'.'

.

1 <= numRows <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

```
class Solution:
def convert(self, s: str, numRows: int) -> str:
```

Python:

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

JavaScript:

```
/**
* @param {string} s
* @param {number} numRows
* @return {string}
*/
var convert = function(s, numRows) {

};
```

TypeScript:

```
function convert(s: string, numRows: number): string {

};
```

C#:

```
public class Solution {
public string Convert(string s, int numRows) {

}
}
```

C:

```
char* convert(char* s, int numRows) {

}
```

Go:

```
func convert(s string, numRows int) string {

}
```

Kotlin:

```
class Solution {
fun convert(s: String, numRows: Int): String {

}
}
```

Swift:

```
class Solution {
func convert(_ s: String, _ numRows: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn convert(s: String, num_rows: i32) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} num_rows
# @return {String}
def convert(s, num_rows)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $numRows
* @return String
*/
function convert($s, $numRows) {

}
}
```

Dart:

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

```
}
}
```

Scala:

```
object Solution {
def convert(s: String, numRows: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec convert(s :: String.t, num_rows :: integer) :: String.t
def convert(s, num_rows) do

end
end
```

Erlang:

```
-spec convert(S :: unicode:unicode_binary(), NumRows :: integer()) ->
unicode:unicode_binary().
convert(S, NumRows) ->
.
```

Racket:

```
(define/contract (convert s numRows)
(-> string? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Zigzag Conversion
```

```
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Zigzag Conversion
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Zigzag Conversion
Difficulty: Medium
Tags: string

Approach: String manipulation with hash map or two pointers
```

```
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def convert(self, s: str, numRows: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Zigzag Conversion
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @param {number} numRows
* @return {string}
*/
var convert = function(s, numRows) {

};
```

TypeScript Solution:

```
/**
* Problem: Zigzag Conversion
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function convert(s: string, numRows: number): string {

};
```

C# Solution:

```
/*
* Problem: Zigzag Conversion
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string Convert(string s, int numRows) {

}
}
```

C Solution:

```
/*
* Problem: Zigzag Conversion
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

char* convert(char* s, int numRows) {

}
```

Go Solution:

```
// Problem: Zigzag Conversion
// Difficulty: Medium
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func convert(s string, numRows int) string {

}
```

Kotlin Solution:

```
class Solution {
fun convert(s: String, numRows: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func convert(_ s: String, _ numRows: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Zigzag Conversion
// Difficulty: Medium
// Tags: string
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn convert(s: String, num_rows: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} num_rows
# @return {String}
def convert(s, num_rows)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $numRows
* @return String
*/
function convert($s, $numRows) {

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def convert(s: String, numRows: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec convert(s :: String.t, num_rows :: integer) :: String.t
def convert(s, num_rows) do

end
end
```

Erlang Solution:

```
-spec convert(S :: unicode:unicode_binary(), NumRows :: integer()) ->
unicode:unicode_binary().
convert(S, NumRows) ->
.
```

Racket Solution:

```
(define/contract (convert s numRows)
(-> string? exact-integer? string?)
)
```