<https://leetcode.com/problems/zigzag-conversion/>

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

**Attempt 1: 2023-10-11**

**Solution 1: Find String pattern (30 min)**

class Solution {

public String convert(String s, int numRows) {

// Must check if the definition of row is 1 or not

// if numRows == 1 will go into infinite loop

// Test out:

// s = "A", numRows = 1 -> Memory Limit Exceeded

if(numRows == 1) {

return s;

}

int len = s.length();

int cycleLen = 2 \* numRows - 2;

StringBuilder sb = new StringBuilder();

for(int i = 0; i < numRows; i++) {

for(int j = 0; i + j < len; j += cycleLen) {

sb.append(s.charAt(i + j));

if(i != 0 && i != numRows - 1 && j + cycleLen - i < len) {

sb.append(s.charAt(j + cycleLen - i));

}

}

}

return sb.toString();

}

}

Time Complexity: O(N)

Space Complexity: O(N)

时间复杂度：O（n），虽然是两层循环，但第二次循环每次加的是 cycleLen ，无非是把每个字符遍历了 1 次，所以两层循环内执行的次数肯定是字符串的长度。

空间复杂度：O（n），保存字符串。

**Step by Step process**

0 1 2 3 4 5 6 7 8 9 10 11 12 13

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

0 6 12

P I N row 0(i)

1 (5) 7 (11) 13

A L S I G row 1

2 (4) 8 (10)

Y A H R row 2

3 9

P I row 3

Above is the actual index -> char mapping on each row,

we find the gap between normal indexed chars(no brackets) is 6,

and the gap between special indexed chars(brackets) also 6,

which exactly equal to one zigzag cycle length, one zigzag

cycle in s = "PAYPALISHIRING" is "PAYPAL", the length is 6.

But only first and last row only have normal indexed chars,

all other rows besides normal indexed chars will also contain

special indexed chars, we have to handle these special indexed

chars separately: e.g 'L' at index = 5 and 'I' at index = 11

need to add in row 2, same for 'A' at index = 4 and 'R' at

index = 10 need to add in row 3. We can find regular pattern:

5 = 0(j) + 6(cycleLen) - 1(i)

11 = 6(j) + 6(cycleLen) - 1(i)

4 = 0(j) + 6(cycleLen) - 2(i)

10 = 6(j) + 6(cycleLen) - 2(i)

**Solution 2: Flip the traverse direction when hitting first and last row (30 min)**

class Solution {

public String convert(String s, int numRows) {

// Must check if the definition of row is 1 or not

// if numRows == 1 will go into infinite loop

// Test out:

// s = "A", numRows = 1 -> Memory Limit Exceeded

if(numRows == 1) {

return s;

}

int len = s.length();

List<StringBuilder> list = new ArrayList<>();

for(int i = 0; i < numRows; i++) {

list.add(new StringBuilder());

}

int row = 0;

boolean goingDown = true;

for(char c : s.toCharArray()) {

list.get(row).append(c);

row += goingDown ? 1 : -1;

// Flip the direction when hit first or last row

if(row == 0 || row == numRows - 1) {

goingDown = !goingDown;

}

}

StringBuilder result = new StringBuilder();

for(StringBuilder sb : list) {

result.append(sb);

}

return result.toString();

}

}

Time Complexity: O(N)

Space Complexity: O(N)

**Refer to**

<https://leetcode.wang/leetCode-6-ZigZag-Conversion.html>

## **2. 解法一**

按照写 Z 的过程，遍历每个字符，然后将字符存到对应的行中。用 goingDown 保存当前的遍历方向，如果遍历到两端，就改变方向。

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; //遍历到两端，改变方向

curRow += goingDown ? 1 : -1;

}

StringBuilder ret = new StringBuilder();

for (StringBuilder row : rows) ret.append(row);

return ret.toString();

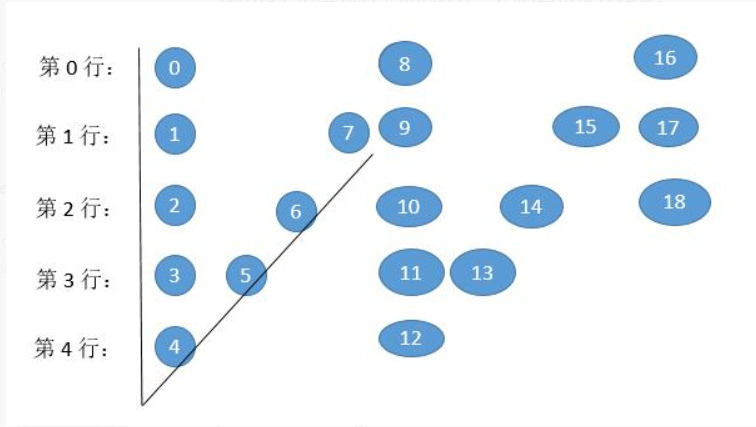
}

时间复杂度：O（n），n 是字符串的长度。

空间复杂度：O（n），保存每个字符需要的空间。

## **3. 解法二**

找出按 Z 形排列后字符的规律，然后直接保存起来。



我们可以看到，图形其实是有周期的，0，1，2 ... 7 总过 8 个，然后就又开始重复相同的路径。周期的计算就是 cycleLen = 2 × numRows - 2 = 2 × 5 - 2 = 8 个。

我们发现第 0 行和最后一行一个周期内有一个字符，所以第一个字符下标是 0 ，第二个字符下标是 0 + cycleLen = 8，第三个字符下标是 8 + cycleLen = 16 。

其他行都是两个字符。

第 1 个字符和第 0 行的规律是一样的。

第 2 个字符其实就是下一个周期的第 0 行的下标减去当前行。什么意思呢？

我们求一下第 1 行第 1 个周期内的第 2 个字符，下一个周期的第 0 行的下标是 8 ，减去当前行 1 ，就是 7 了。

我们求一下第 1 行第 2 个而周期内的第 2 个字符，下一个周期的第 0 行的下标是 16 ，减去当前行 1 ，就是 15 了。

我们求一下第 2 行第 1 个周期内的第 2 个字符，下一个周期的第 0 行的下标是 8 ，减去当前行 2 ，就是 6 了。

当然期间一定要保证下标小于 n ，防止越界。

可以写代码了。

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) { //每次加一个周期

ret.append(s.charAt(j + i));

if (i != 0 && i != numRows - 1 && j + cycleLen - i < n) //除去第 0 行和最后一行

ret.append(s.charAt(j + cycleLen - i));

}

}

return ret.toString();

}

时间复杂度：O（n），虽然是两层循环，但第二次循环每次加的是 cycleLen ，无非是把每个字符遍历了 1 次，所以两层循环内执行的次数肯定是字符串的长度。

空间复杂度：O（n），保存字符串。

## **4. 总结**

这次算是总结起来最轻松的了，这道题有些找规律的意思。解法一顺着排列的方式遍历，解法二直接从答案入口找出下标的规律。