

# Problem 2075: Decode the Slanted Ciphertext

## Problem Information

**Difficulty:** Medium

**Acceptance Rate:** 0.00%

**Paid Only:** No

## Problem Description

A string

`originalText`

is encoded using a

slanted transposition cipher

to a string

`encodedText`

with the help of a matrix having a

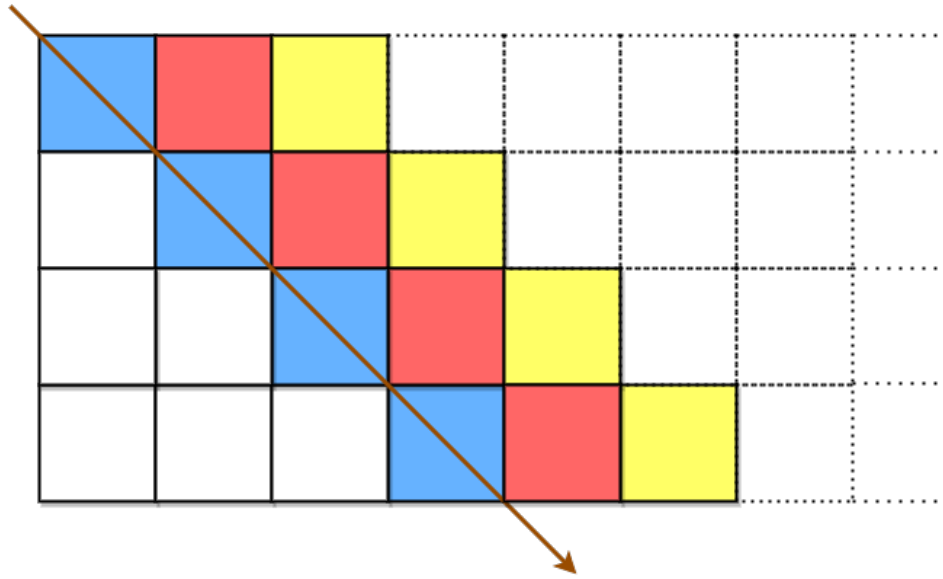
fixed number of rows

rows

.

`originalText`

is placed first in a top-left to bottom-right manner.



The blue cells are filled first, followed by the red cells, then the yellow cells, and so on, until we reach the end of

originalText

. The arrow indicates the order in which the cells are filled. All empty cells are filled with

''

. The number of columns is chosen such that the rightmost column will

not be empty

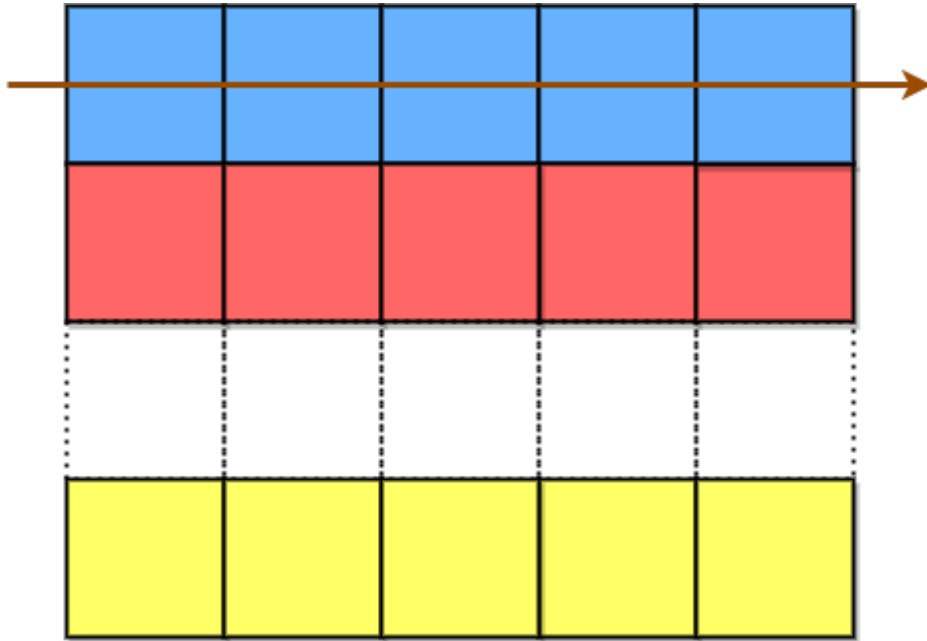
after filling in

originalText

.

encodedText

is then formed by appending all characters of the matrix in a row-wise fashion.



The characters in the blue cells are appended first to

`encodedText`

, then the red cells, and so on, and finally the yellow cells. The arrow indicates the order in which the cells are accessed.

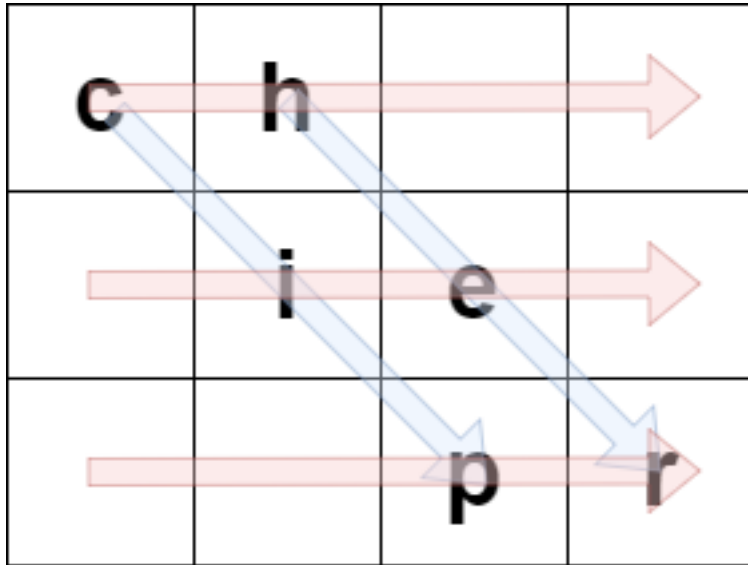
For example, if

`originalText = "cipher"`

and

`rows = 3`

, then we encode it in the following manner:



The blue arrows depict how

originalText

is placed in the matrix, and the red arrows denote the order in which

encodedText

is formed. In the above example,

encodedText = "ch ie pr"

.

Given the encoded string

encodedText

and number of rows

rows

, return

the original string

originalText

.

Note:

originalText

does not

have any trailing spaces

''

. The test cases are generated such that there is only one possible

originalText

.

Example 1:

Input:

encodedText = "ch ie pr", rows = 3

Output:

"cipher"

Explanation:

This is the same example described in the problem description.

Example 2:

<b>i</b>	<b>v</b>	<b>e</b>	<b>o</b>		
		<b>e</b>	<b>e</b>	<b>d</b>	
		<b>l</b>		<b>t</b>	<b>e</b>
			<b>o</b>	<b>l</b>	<b>c</b>

Input:

encodedText = "iveo eed l te olc", rows = 4

Output:

"i love leetcode"

Explanation:

The figure above denotes the matrix that was used to encode originalText. The blue arrows show how we can find originalText from encodedText.

Example 3:

<b>c</b>	<b>o</b>	<b>d</b>	<b>i</b>	<b>n</b>	<b>g</b>
----------	----------	----------	----------	----------	----------

Input:

encodedText = "coding", rows = 1

Output:

"coding"

Explanation:

Since there is only 1 row, both originalText and encodedText are the same.

Constraints:

$0 \leq \text{encodedText.length} \leq 10$

6

encodedText

consists of lowercase English letters and

, ' ,

only.

encodedText

is a valid encoding of some

originalText

that

does not

have trailing spaces.

$1 \leq \text{rows} \leq 1000$

The testcases are generated such that there is

only one

possible

originalText

.

## Code Snippets

### C++:

```
class Solution {  
public:  
    string decodeCiphertext(string encodedText, int rows) {  
  
    }  
};
```

### Java:

```
class Solution {  
    public String decodeCiphertext(String encodedText, int rows) {  
  
    }  
}
```

### Python3:

```
class Solution:  
    def decodeCiphertext(self, encodedText: str, rows: int) -> str:
```

### Python:

```
class Solution(object):  
    def decodeCiphertext(self, encodedText, rows):  
        """  
        :type encodedText: str  
        :type rows: int  
        :rtype: str  
        """
```

### JavaScript:

```

/**
 * @param {string} encodedText
 * @param {number} rows
 * @return {string}
 */
var decodeCiphertext = function(encodedText, rows) {

};

```

### TypeScript:

```

function decodeCiphertext(encodedText: string, rows: number): string {

};

```

### C#:

```

public class Solution {
    public string DecodeCiphertext(string encodedText, int rows) {

    }
}

```

### C:

```

char* decodeCiphertext(char* encodedText, int rows) {

}

```

### Go:

```

func decodeCiphertext(encodedText string, rows int) string {

}

```

### Kotlin:

```

class Solution {
    fun decodeCiphertext(encodedText: String, rows: Int): String {

    }
}

```

### Swift:

```
class Solution {  
    func decodeCiphertext(_ encodedText: String, _ rows: Int) -> String {  
  
    }  
}
```

### Rust:

```
impl Solution {  
    pub fn decode_ciphertext(encoded_text: String, rows: i32) -> String {  
  
    }  
}
```

### Ruby:

```
# @param {String} encoded_text  
# @param {Integer} rows  
# @return {String}  
def decode_ciphertext(encoded_text, rows)  
  
end
```

### PHP:

```
class Solution {  
  
    /**  
     * @param String $encodedText  
     * @param Integer $rows  
     * @return String  
     */  
    function decodeCiphertext($encodedText, $rows) {  
  
    }  
}
```

### Dart:

```
class Solution {  
    String decodeCiphertext(String encodedText, int rows) {  
  
    }  
}
```

```
}  
}
```

### Scala:

```
object Solution {  
  def decodeCiphertext(encodedText: String, rows: Int): String = {  
  
  }  
}
```

### Elixir:

```
defmodule Solution do  
  @spec decode_ciphertext(encoded_text :: String.t, rows :: integer) ::  
    String.t  
  def decode_ciphertext(encoded_text, rows) do  
  
  end  
end
```

### Erlang:

```
-spec decode_ciphertext(EncodedText :: unicode:unicode_binary(), Rows ::  
integer()) -> unicode:unicode_binary().  
decode_ciphertext(EncodedText, Rows) ->  
.
```

### Racket:

```
(define/contract (decode-ciphertext encodedText rows)  
  (-> string? exact-integer? string?)  
)
```

## Solutions

### C++ Solution:

```

/*
 * Problem: Decode the Slanted Ciphertext
 * 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 decodeCiphertext(string encodedText, int rows) {

    }
};

```

### Java Solution:

```

/**
 * Problem: Decode the Slanted Ciphertext
 * 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 decodeCiphertext(String encodedText, int rows) {

    }
}

```

### Python3 Solution:

```

"""
Problem: Decode the Slanted Ciphertext
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 decodeCiphertext(self, encodedText: str, rows: int) -> str:
        # TODO: Implement optimized solution
        pass

```

### Python Solution:

```

class Solution(object):
    def decodeCiphertext(self, encodedText, rows):
        """
        :type encodedText: str
        :type rows: int
        :rtype: str
        """

```

### JavaScript Solution:

```

/**
 * Problem: Decode the Slanted Ciphertext
 * 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} encodedText
 * @param {number} rows
 * @return {string}
 */
var decodeCiphertext = function(encodedText, rows) {

};

```

## TypeScript Solution:

```
/**
 * Problem: Decode the Slanted Ciphertext
 * 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 decodeCiphertext(encodedText: string, rows: number): string {

};
```

## C# Solution:

```
/*
 * Problem: Decode the Slanted Ciphertext
 * 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 DecodeCiphertext(string encodedText, int rows) {

    }
}
```

## C Solution:

```
/*
 * Problem: Decode the Slanted Ciphertext
 * 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* decodeCiphertext(char* encodedText, int rows) {

}

```

### Go Solution:

```

// Problem: Decode the Slanted Ciphertext
// 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 decodeCiphertext(encodedText string, rows int) string {

}

```

### Kotlin Solution:

```

class Solution {
    fun decodeCiphertext(encodedText: String, rows: Int): String {

    }
}

```

### Swift Solution:

```

class Solution {
    func decodeCiphertext(_ encodedText: String, _ rows: Int) -> String {

    }
}

```

### Rust Solution:

```

// Problem: Decode the Slanted Ciphertext
// 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 decode_ciphertext(encoded_text: String, rows: i32) -> String {

    }
}

```

### Ruby Solution:

```

# @param {String} encoded_text
# @param {Integer} rows
# @return {String}
def decode_ciphertext(encoded_text, rows)

end

```

### PHP Solution:

```

class Solution {

    /**
     * @param String $encodedText
     * @param Integer $rows
     * @return String
     */
    function decodeCiphertext($encodedText, $rows) {

    }

}

```

### Dart Solution:

```

class Solution {
    String decodeCiphertext(String encodedText, int rows) {

    }
}

```

```
}
```

### Scala Solution:

```
object Solution {  
  def decodeCiphertext(encodedText: String, rows: Int): String = {  
  
  }  
}
```

### Elixir Solution:

```
defmodule Solution do  
  @spec decode_ciphertext(encoded_text :: String.t, rows :: integer) ::  
    String.t  
  def decode_ciphertext(encoded_text, rows) do  
  
  end  
end
```

### Erlang Solution:

```
-spec decode_ciphertext(EncodedText :: unicode:unicode_binary(), Rows ::  
integer()) -> unicode:unicode_binary().  
decode_ciphertext(EncodedText, Rows) ->  
.
```

### Racket Solution:

```
(define/contract (decode-ciphertext encodedText rows)  
  (-> string? exact-integer? string?)  
)
```