Problem 1138: Alphabet Board Path

Problem Information

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

Problem Description

On an alphabet board, we start at position

(0, 0)

, corresponding to character

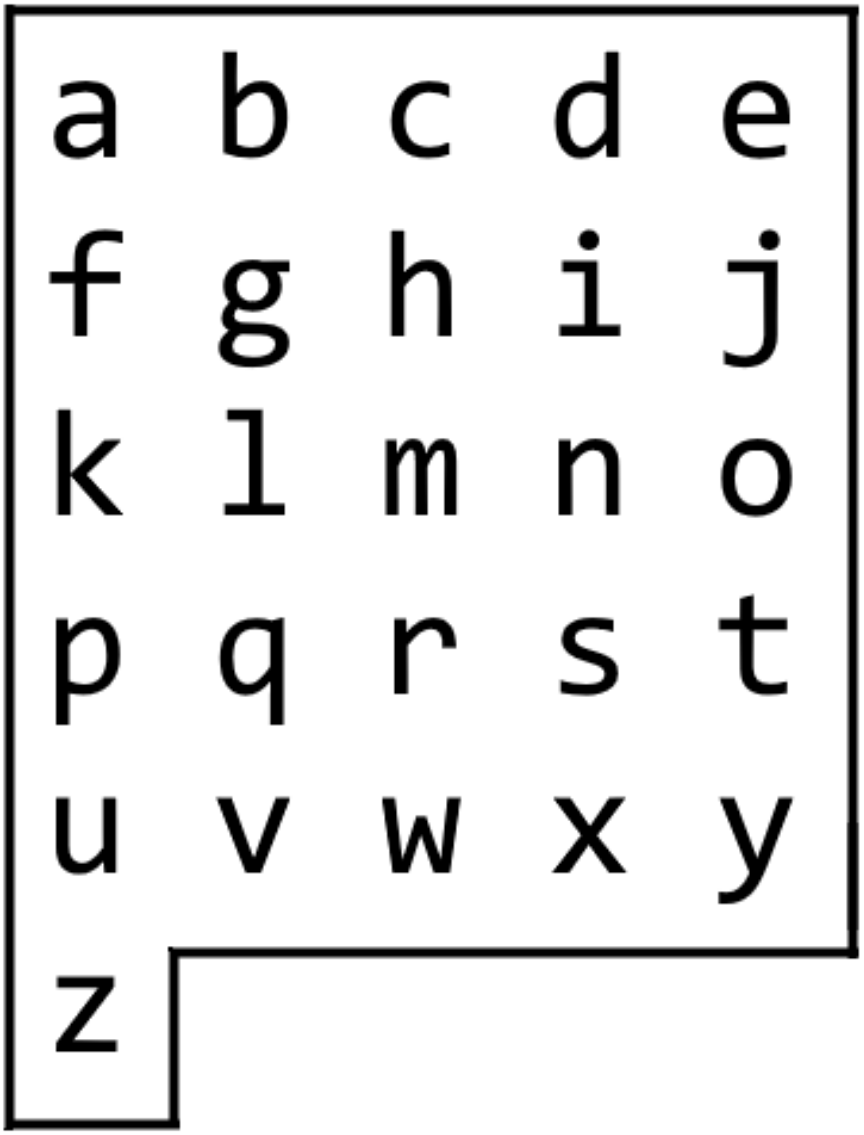
board[0][0]

.

Here,

board = ["abcde", "fghij", "klmno", "pqrst", "uvwxy", "z"]

, as shown in the diagram below.



We may make the following moves:

'U'

moves our position up one row, if the position exists on the board;

'D'

moves our position down one row, if the position exists on the board;

'L'

moves our position left one column, if the position exists on the board;

'R'

moves our position right one column, if the position exists on the board;

'!'

adds the character

board[r][c]

at our current position

(r, c)

to the answer.

(Here, the only positions that exist on the board are positions with letters on them.)

Return a sequence of moves that makes our answer equal to

target

in the minimum number of moves. You may return any path that does so.

Example 1:

Input:

target = "leet"

Output:

"DDR!UURRR!!DDD!"

Example 2:

Input:

target = "code"

Output:

"RR!DDRR!UUL!R!"

Constraints:

1 <= target.length <= 100

target

consists only of English lowercase letters.

Code Snippets

C++:

```
class Solution {
public:
string alphabetBoardPath(string target) {

}
};
```

Java:

```
class Solution {
public String alphabetBoardPath(String target) {

}
}
```

Python3:

```
class Solution:
def alphabetBoardPath(self, target: str) -> str:
```

Python:

```
class Solution(object):
def alphabetBoardPath(self, target):
"""
:type target: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} target
* @return {string}
*/
var alphabetBoardPath = function(target) {

};
```

TypeScript:

```
function alphabetBoardPath(target: string): string {

};
```

C#:

```
public class Solution {
public string AlphabetBoardPath(string target) {

}
}
```

C:

```



char * alphabetBoardPath(char * target){

}
```

Go:

```
func alphabetBoardPath(target string) string {
```

```
}
```

Kotlin:

```
class Solution {
fun alphabetBoardPath(target: String): String {

}
}
```

Swift:

```
class Solution {
func alphabetBoardPath(_ target: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn alphabet_board_path(target: String) -> String {

}
}
```

Ruby:

```
# @param {String} target
# @return {String}
def alphabet_board_path(target)

end
```

PHP:

```
class Solution {

/**
* @param String $target
* @return String
*/
```

```
function alphabetBoardPath($target) {

}
}
```

Scala:

```
object Solution {
def alphabetBoardPath(target: String): String = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Alphabet Board Path
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string alphabetBoardPath(string target) {

}
};
```

Java Solution:

```
/**
* Problem: Alphabet Board Path
* Difficulty: Medium
* Tags: string, hash
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public String alphabetBoardPath(String target) {

}
}
```

Python3 Solution:

```
"""
Problem: Alphabet Board Path
Difficulty: Medium
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def alphabetBoardPath(self, target: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def alphabetBoardPath(self, target):
"""
:type target: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Alphabet Board Path
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} target
* @return {string}
*/
var alphabetBoardPath = function(target) {

};
```

TypeScript Solution:

```
/**
* Problem: Alphabet Board Path
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function alphabetBoardPath(target: string): string {

};
```

C# Solution:

```
/*
* Problem: Alphabet Board Path
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public string AlphabetBoardPath(string target) {

}
}
```

C Solution:

```
/*
* Problem: Alphabet Board Path
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/



char * alphabetBoardPath(char * target){

}
```

Go Solution:

```
// Problem: Alphabet Board Path
// Difficulty: Medium
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func alphabetBoardPath(target string) string {

}
```

Kotlin Solution:

```
class Solution {
fun alphabetBoardPath(target: String): String {

}
}
```

Swift Solution:

```
class Solution {
func alphabetBoardPath(_ target: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Alphabet Board Path
// Difficulty: Medium
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn alphabet_board_path(target: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} target
# @return {String}
def alphabet_board_path(target)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $target
* @return String
*/
function alphabetBoardPath($target) {

}
}
```

Scala Solution:

```
object Solution {
def alphabetBoardPath(target: String): String = {

}
}
```