Problem 744: Find Smallest Letter Greater Than Target

Problem Information

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

Problem Description

You are given an array of characters

letters

that is sorted in

non-decreasing order

, and a character

target

. There are

at least two different

characters in

letters

.

Return

the smallest character in

letters

that is lexicographically greater than

target

. If such a character does not exist, return the first character in

letters

.

Example 1:

Input:

letters = ["c","f","j"], target = "a"

Output:

"c"

Explanation:

The smallest character that is lexicographically greater than 'a' in letters is 'c'.

Example 2:

Input:

letters = ["c","f","j"], target = "c"

Output:

"f"

Explanation:

The smallest character that is lexicographically greater than 'c' in letters is 'f'.

Example 3:

Input:

letters = ["x","x","y","y"], target = "z"

Output:

"x"

Explanation:

There are no characters in letters that is lexicographically greater than 'z' so we return letters[0].

Constraints:

2 <= letters.length <= 10

4

letters[i]

is a lowercase English letter.

letters

is sorted in

non-decreasing

order.

letters

contains at least two different characters.

target

is a lowercase English letter.

Code Snippets

C++:

```
class Solution {
public:
char nextGreatestLetter(vector<char>& letters, char target) {

}
};
```

Java:

```
class Solution {
public char nextGreatestLetter(char[] letters, char target) {

}
}
```

Python3:

```
class Solution:
def nextGreatestLetter(self, letters: List[str], target: str) -> str:
```

Python:

```
class Solution(object):
def nextGreatestLetter(self, letters, target):
"""
:type letters: List[str]
:type target: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {character[]} letters
* @param {character} target
* @return {character}
*/
var nextGreatestLetter = function(letters, target) {

};
```

TypeScript:

```
function nextGreatestLetter(letters: string[], target: string): string {

};
```

C#:

```
public class Solution {
public char NextGreatestLetter(char[] letters, char target) {

}
}
```

C:

```
char nextGreatestLetter(char* letters, int lettersSize, char target) {

}
```

Go:

```
func nextGreatestLetter(letters []byte, target byte) byte {

}
```

Kotlin:

```
class Solution {
fun nextGreatestLetter(letters: CharArray, target: Char): Char {

}
}
```

Swift:

```
class Solution {
func nextGreatestLetter(_ letters: [Character], _ target: Character) ->
Character {

}
}
```

Rust:

```
impl Solution {
pub fn next_greatest_letter(letters: Vec<char>, target: char) -> char {

}
}
```

Ruby:

```
# @param {Character[]} letters
# @param {Character} target
# @return {Character}
def next_greatest_letter(letters, target)

end
```

PHP:

```
class Solution {

/**
* @param String[] $letters
* @param String $target
* @return String
*/
function nextGreatestLetter($letters, $target) {

}
}
```

Dart:

```
class Solution {
String nextGreatestLetter(List<String> letters, String target) {

}
}
```

Scala:

```
object Solution {
def nextGreatestLetter(letters: Array[Char], target: Char): Char = {

}
}
```

Elixir:

```
defmodule Solution do
@spec next_greatest_letter(letters :: [char], target :: char) :: char
def next_greatest_letter(letters, target) do

end
end
```

Erlang:

```
-spec next_greatest_letter(Letters :: [char()], Target :: char()) -> char().
next_greatest_letter(Letters, Target) ->
.
```

Racket:

```
(define/contract (next-greatest-letter letters target)
(-> (listof char?) char? char?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Smallest Letter Greater Than Target
```

```
* Difficulty: Easy
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
char nextGreatestLetter(vector<char>& letters, char target) {

}
};
```

Java Solution:

```
/**
* Problem: Find Smallest Letter Greater Than Target
* Difficulty: Easy
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public char nextGreatestLetter(char[] letters, char target) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Smallest Letter Greater Than Target
Difficulty: Easy
Tags: array, graph, sort, search

Approach: Use two pointers or sliding window technique
```

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

class Solution:
def nextGreatestLetter(self, letters: List[str], target: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def nextGreatestLetter(self, letters, target):
"""
:type letters: List[str]
:type target: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Find Smallest Letter Greater Than Target
* Difficulty: Easy
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {character[]} letters
* @param {character} target
* @return {character}
*/
var nextGreatestLetter = function(letters, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Smallest Letter Greater Than Target
* Difficulty: Easy
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function nextGreatestLetter(letters: string[], target: string): string {

};
```

C# Solution:

```
/*
* Problem: Find Smallest Letter Greater Than Target
* Difficulty: Easy
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public char NextGreatestLetter(char[] letters, char target) {

}
}
```

C Solution:

```
/*
* Problem: Find Smallest Letter Greater Than Target
* Difficulty: Easy
* Tags: array, graph, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

char nextGreatestLetter(char* letters, int lettersSize, char target) {

}
```

Go Solution:

```
// Problem: Find Smallest Letter Greater Than Target
// Difficulty: Easy
// Tags: array, graph, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func nextGreatestLetter(letters []byte, target byte) byte {

}
```

Kotlin Solution:

```
class Solution {
fun nextGreatestLetter(letters: CharArray, target: Char): Char {

}
}
```

Swift Solution:

```
class Solution {
func nextGreatestLetter(_ letters: [Character], _ target: Character) ->
Character {

}
}
```

Rust Solution:

```
// Problem: Find Smallest Letter Greater Than Target
// Difficulty: Easy
```

```
// Tags: array, graph, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn next_greatest_letter(letters: Vec<char>, target: char) -> char {

}
}
```

Ruby Solution:

```
# @param {Character[]} letters
# @param {Character} target
# @return {Character}
def next_greatest_letter(letters, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $letters
* @param String $target
* @return String
*/
function nextGreatestLetter($letters, $target) {

}
}
```

Dart Solution:

```
class Solution {
String nextGreatestLetter(List<String> letters, String target) {

}
```

```
}
```

Scala Solution:

```
object Solution {
def nextGreatestLetter(letters: Array[Char], target: Char): Char = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec next_greatest_letter(letters :: [char], target :: char) :: char
def next_greatest_letter(letters, target) do

end
end
```

Erlang Solution:

```
-spec next_greatest_letter(Letters :: [char()], Target :: char()) -> char().
next_greatest_letter(Letters, Target) ->
.
```

Racket Solution:

```
(define/contract (next-greatest-letter letters target)
(-> (listof char?) char? char?)
)
```