

# Problem 1961: Check If String Is a Prefix of Array

## Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given a string

s

and an array of strings

words

, determine whether

s

is a

prefix string

of

words

.

A string

s

is a

prefix string

of

words

if

s

can be made by concatenating the first

k

strings in

words

for some

positive

k

no larger than

words.length

.

Return

true

if

s

is a

prefix string

of

words

, or

false

otherwise

.

Example 1:

Input:

```
s = "iloveleetcode", words = ["i", "love", "leetcode", "apples"]
```

Output:

true

Explanation:

s can be made by concatenating "i", "love", and "leetcode" together.

Example 2:

Input:

```
s = "iloveleetcode", words = ["apples", "i", "love", "leetcode"]
```

Output:

false

Explanation:

It is impossible to make s using a prefix of arr.

Constraints:

$1 \leq \text{words.length} \leq 100$

$1 \leq \text{words[i].length} \leq 20$

$1 \leq s.length \leq 1000$

words[i]

and

s

consist of only lowercase English letters.

## Code Snippets

C++:

```
class Solution {
public:
    bool isPrefixString(string s, vector<string>& words) {
        }
};
```

Java:

```
class Solution {
public boolean isPrefixString(String s, String[] words) {
    }
```

```
}
```

### Python3:

```
class Solution:  
    def isPrefixString(self, s: str, words: List[str]) -> bool:
```

### Python:

```
class Solution(object):  
    def isPrefixString(self, s, words):  
        """  
        :type s: str  
        :type words: List[str]  
        :rtype: bool  
        """
```

### JavaScript:

```
/**  
 * @param {string} s  
 * @param {string[]} words  
 * @return {boolean}  
 */  
var isPrefixString = function(s, words) {  
  
};
```

### TypeScript:

```
function isPrefixString(s: string, words: string[]): boolean {  
  
};
```

### C#:

```
public class Solution {  
    public bool IsPrefixString(string s, string[] words) {  
  
    }  
}
```

**C:**

```
bool isPrefixString(char* s, char** words, int wordsSize) {  
  
}
```

**Go:**

```
func isPrefixString(s string, words []string) bool {  
  
}
```

**Kotlin:**

```
class Solution {  
    fun isPrefixString(s: String, words: Array<String>): Boolean {  
  
    }  
}
```

**Swift:**

```
class Solution {  
    func isPrefixString(_ s: String, _ words: [String]) -> Bool {  
  
    }  
}
```

**Rust:**

```
impl Solution {  
    pub fn is_prefix_string(s: String, words: Vec<String>) -> bool {  
  
    }  
}
```

**Ruby:**

```
# @param {String} s  
# @param {String[]} words  
# @return {Boolean}  
def is_prefix_string(s, words)
```

```
end
```

### PHP:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @param String[] $words  
     * @return Boolean  
     */  
    function isPrefixString($s, $words) {  
  
    }  
}
```

### Dart:

```
class Solution {  
  bool isPrefixString(String s, List<String> words) {  
  
  }  
}
```

### Scala:

```
object Solution {  
  def isPrefixString(s: String, words: Array[String]): Boolean = {  
  
  }  
}
```

### Elixir:

```
defmodule Solution do  
  @spec is_prefix_string(s :: String.t, words :: [String.t]) :: boolean  
  def is_prefix_string(s, words) do  
  
  end  
end
```

### Erlang:

```
-spec is_prefix_string(S :: unicode:unicode_binary(), Words :: [unicode:unicode_binary()]) -> boolean().  
is_prefix_string(S, Words) ->  
.
```

## Racket:

```
(define/contract (is-prefix-string s words)  
  (-> string? (listof string?) boolean?)  
)
```

# Solutions

## C++ Solution:

```
/*  
 * Problem: Check If String Is a Prefix of Array  
 * Difficulty: Easy  
 * Tags: array, string  
 *  
 * 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:  
    bool isPrefixString(string s, vector<string>& words) {  
  
    }  
};
```

## Java Solution:

```
/**  
 * Problem: Check If String Is a Prefix of Array  
 * Difficulty: Easy  
 * Tags: array, string  
 *  
 * 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 boolean isPrefixString(String s, String[] words) {
        }

    }
}

```

### Python3 Solution:

```

"""
Problem: Check If String Is a Prefix of Array
Difficulty: Easy
Tags: array, string

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 isPrefixString(self, s: str, words: List[str]) -> bool:
        # TODO: Implement optimized solution
        pass

```

### Python Solution:

```

class Solution(object):
    def isPrefixString(self, s, words):
        """
        :type s: str
        :type words: List[str]
        :rtype: bool
        """

```

### JavaScript Solution:

```

/**
 * Problem: Check If String Is a Prefix of Array
 * Difficulty: Easy

```

```

* Tags: array, string
*
* 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 {string} s
* @param {string[]} words
* @return {boolean}
*/
var isPrefixString = function(s, words) {

};

```

### TypeScript Solution:

```

/**

* Problem: Check If String Is a Prefix of Array
* Difficulty: Easy
* Tags: array, string
*
* 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 isPrefixString(s: string, words: string[]): boolean {

};

```

### C# Solution:

```

/*
* Problem: Check If String Is a Prefix of Array
* Difficulty: Easy
* Tags: array, string
*
* 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 bool IsPrefixString(string s, string[] words) {
        return true;
    }
}

```

### C Solution:

```

/*
 * Problem: Check If String Is a Prefix of Array
 * Difficulty: Easy
 * Tags: array, string
 *
 * 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
*/

bool isPrefixString(char* s, char** words, int wordsSize) {
    return true;
}

```

### Go Solution:

```

// Problem: Check If String Is a Prefix of Array
// Difficulty: Easy
// Tags: array, string
//
// 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 isPrefixString(s string, words []string) bool {
    return true
}

```

### Kotlin Solution:

```
class Solution {  
    fun isPrefixString(s: String, words: Array<String>): Boolean {  
        }  
        }  
}
```

### Swift Solution:

```
class Solution {  
    func isPrefixString(_ s: String, _ words: [String]) -> Bool {  
        }  
        }
```

### Rust Solution:

```
// Problem: Check If String Is a Prefix of Array  
// Difficulty: Easy  
// Tags: array, string  
//  
// 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 is_prefix_string(s: String, words: Vec<String>) -> bool {  
        }  
        }
```

### Ruby Solution:

```
# @param {String} s  
# @param {String[]} words  
# @return {Boolean}  
def is_prefix_string(s, words)  
  
end
```

### PHP Solution:

```

class Solution {

    /**
     * @param String $s
     * @param String[] $words
     * @return Boolean
     */
    function isPrefixString($s, $words) {

    }
}

```

### Dart Solution:

```

class Solution {
bool isPrefixString(String s, List<String> words) {

}
}

```

### Scala Solution:

```

object Solution {
def isPrefixString(s: String, words: Array[String]): Boolean = {

}
}

```

### Elixir Solution:

```

defmodule Solution do
@spec is_prefix_string(s :: String.t, words :: [String.t]) :: boolean
def is_prefix_string(s, words) do

end
end

```

### Erlang Solution:

```

-spec is_prefix_string(S :: unicode:unicode_binary(), Words :: [unicode:unicode_binary()]) -> boolean().
is_prefix_string(S, Words) ->

```

**Racket Solution:**

```
(define/contract (is-prefix-string s words)
  (-> string? (listof string?) boolean?)
  )
```