Problem 14: Longest Common Prefix

Problem Information

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

Problem Description

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string

""

.

Example 1:

Input:

strs = ["flower","flow","flight"]

Output:

"fl"

Example 2:

Input:

strs = ["dog","racecar","car"]

Output:

""

Explanation:

There is no common prefix among the input strings.

Constraints:

1 <= strs.length <= 200

0 <= strs[i].length <= 200

strs[i]

consists of only lowercase English letters if it is non-empty.

Code Snippets

C++:

```
class Solution {
public:
string longestCommonPrefix(vector<string>& strs) {

}
};
```

Java:

```
class Solution {
public String longestCommonPrefix(String[] strs) {

}
}
```

Python3:

```
class Solution:
def longestCommonPrefix(self, strs: List[str]) -> str:
```

Python:

```
class Solution(object):
def longestCommonPrefix(self, strs):
"""
:type strs: List[str]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string[]} strs
* @return {string}
*/
var longestCommonPrefix = function(strs) {

};
```

TypeScript:

```
function longestCommonPrefix(strs: string[]): string {

};
```

C#:

```
public class Solution {
public string LongestCommonPrefix(string[] strs) {

}
}
```

C:

```
char* longestCommonPrefix(char** strs, int strsSize) {

}
```

Go:

```
func longestCommonPrefix(strs []string) string {
}
```

Kotlin:

```
class Solution {
fun longestCommonPrefix(strs: Array<String>): String {

}
}
```

Swift:

```
class Solution {
func longestCommonPrefix(_ strs: [String]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn longest_common_prefix(strs: Vec<String>) -> String {

}
}
```

Ruby:

```
# @param {String[]} strs
# @return {String}
def longest_common_prefix(strs)

end
```

PHP:

```
class Solution {

/**
* @param String[] $strs
* @return String
*/
function longestCommonPrefix($strs) {

}
}
```

Dart:

```
class Solution {
String longestCommonPrefix(List<String> strs) {

}
}
```

Scala:

```
object Solution {
def longestCommonPrefix(strs: Array[String]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_common_prefix(strs :: [String.t]) :: String.t
def longest_common_prefix(strs) do

end
end
```

Erlang:

```
-spec longest_common_prefix(Strs :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
longest_common_prefix(Strs) ->
.
```

Racket:

```
(define/contract (longest-common-prefix strs)
(-> (listof string?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Common Prefix
* 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:
string longestCommonPrefix(vector<string>& strs) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Common Prefix
* 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 String longestCommonPrefix(String[] strs) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Common Prefix
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 longestCommonPrefix(self, strs: List[str]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def longestCommonPrefix(self, strs):
"""
:type strs: List[str]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Longest Common Prefix
* 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[]} strs
* @return {string}
*/
var longestCommonPrefix = function(strs) {
};
```

TypeScript Solution:

```
/**
* Problem: Longest Common Prefix
* 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 longestCommonPrefix(strs: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Longest Common Prefix
* 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 string LongestCommonPrefix(string[] strs) {

}
}
```

C Solution:

```
/*
* Problem: Longest Common Prefix
* 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
*/

char* longestCommonPrefix(char** strs, int strsSize) {

}
```

Go Solution:

```
// Problem: Longest Common Prefix
// 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 longestCommonPrefix(strs []string) string {

}
```

Kotlin Solution:

```
class Solution {
fun longestCommonPrefix(strs: Array<String>): String {

}
}
```

Swift Solution:

```
class Solution {
func longestCommonPrefix(_ strs: [String]) -> String {

}
}
```

Rust Solution:

```
// Problem: Longest Common Prefix
// 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 longest_common_prefix(strs: Vec<String>) -> String {

}
}
```

Ruby Solution:

```
# @param {String[]} strs
# @return {String}
def longest_common_prefix(strs)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $strs
* @return String
*/
function longestCommonPrefix($strs) {

}
}
```

Dart Solution:

```
class Solution {
String longestCommonPrefix(List<String> strs) {
}
}
```

Scala Solution:

```
object Solution {
def longestCommonPrefix(strs: Array[String]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_common_prefix(strs :: [String.t]) :: String.t
def longest_common_prefix(strs) do

end
end
```

Erlang Solution:

```
-spec longest_common_prefix(Strs :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
longest_common_prefix(Strs) ->
.
```

Racket Solution:

```
(define/contract (longest-common-prefix strs)
(-> (listof string?) string?)
)
```