

Problem 937: Reorder Data in Log Files

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given an array of

logs

. Each log is a space-delimited string of words, where the first word is the identifier

There are two types of logs:

Letter-logs

: All words (except the identifier) consist of lowercase English letters.

Digit-logs

: All words (except the identifier) consist of digits.

Reorder these logs so that:

The

letter-logs

come before all

digit-logs

The

letter-logs

are sorted lexicographically by their contents. If their contents are the same, then sort them lexicographically by their identifiers.

The

digit-logs

maintain their relative ordering.

Return

the final order of the logs

Example 1:

Input:

```
logs = ["dig1 8 1 5 1", "let1 art can", "dig2 3 6", "let2 own kit dig", "let3 art zero"]
```

Output:

```
["let1 art can", "let3 art zero", "let2 own kit dig", "dig1 8 1 5 1", "dig2 3 6"]
```

Explanation:

The letter-log contents are all different, so their ordering is "art can", "art zero", "own kit dig". The digit-logs have a relative order of "dig1 8 1 5 1", "dig2 3 6".

Example 2:

Input:

```
logs = ["a1 9 2 3 1", "g1 act car", "zo4 4 7", "ab1 off key dog", "a8 act zoo"]
```

Output:

```
["g1 act car", "a8 act zoo", "ab1 off key dog", "a1 9 2 3 1", "zo4 4 7"]
```

Constraints:

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

$3 \leq \text{logs[i].length} \leq 100$

All the tokens of

`logs[i]`

are separated by a

single

space.

`logs[i]`

is guaranteed to have an identifier and at least one word after the identifier.

Code Snippets

C++:

```
class Solution {
public:
    vector<string> reorderLogFiles(vector<string>& logs) {
```

```
    }
};
```

Java:

```
class Solution {
public String[] reorderLogFiles(String[] logs) {
    }
}
```

Python3:

```
class Solution:
    def reorderLogFiles(self, logs: List[str]) -> List[str]:
```

Python:

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

```

JavaScript:

```
/**
 * @param {string[]} logs
 * @return {string[]}
 */
var reorderLogFiles = function(logs) {
};

}
```

TypeScript:

```
function reorderLogFiles(logs: string[]): string[] {
    };
}
```

C#:

```
public class Solution {  
    public string[] ReorderLogFiles(string[] logs) {  
  
    }  
}
```

C:

```
/**  
 * Note: The returned array must be malloced, assume caller calls free().  
 */  
char** reorderLogFiles(char** logs, int logsSize, int* returnSize) {  
  
}
```

Go:

```
func reorderLogFiles(logs []string) []string {  
  
}
```

Kotlin:

```
class Solution {  
    fun reorderLogFiles(logs: Array<String>): Array<String> {  
  
    }  
}
```

Swift:

```
class Solution {  
    func reorderLogFiles(_ logs: [String]) -> [String] {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn reorder_log_files(logs: Vec<String>) -> Vec<String> {  
  
    }
```

```
}
```

Ruby:

```
# @param {String[]} logs
# @return {String[]}
def reorder_log_files(logs)

end
```

PHP:

```
class Solution {

    /**
     * @param String[] $logs
     * @return String[]
     */
    function reorderLogFiles($logs) {

    }
}
```

Dart:

```
class Solution {
List<String> reorderLogFiles(List<String> logs) {

}
```

Scala:

```
object Solution {
def reorderLogFiles(logs: Array[String]): Array[String] = {

}
```

Elixir:

```

defmodule Solution do
@spec reorder_log_files(logs :: [String.t]) :: [String.t]
def reorder_log_files(logs) do

end
end

```

Erlang:

```

-spec reorder_log_files(Logs :: [unicode:unicode_binary()]) ->
[unicode:unicode_binary()].
reorder_log_files(Logs) ->
.

```

Racket:

```

(define/contract (reorder-log-files logs)
(-> (listof string?) (listof string?)))
)
```

Solutions

C++ Solution:

```

/*
 * Problem: Reorder Data in Log Files
 * Difficulty: Medium
 * Tags: array, string, graph, sort
 *
 * 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:
vector<string> reorderLogFiles(vector<string>& logs) {

}
};
```

Java Solution:

```
/**  
 * Problem: Reorder Data in Log Files  
 * Difficulty: Medium  
 * Tags: array, string, graph, sort  
 *  
 * 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[] reorderLogFiles(String[] logs) {  
        }  
    }  
}
```

Python3 Solution:

```
"""  
Problem: Reorder Data in Log Files  
Difficulty: Medium  
Tags: array, string, graph, sort  
  
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 reorderLogFiles(self, logs: List[str]) -> List[str]:  
        # TODO: Implement optimized solution  
        pass
```

Python Solution:

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

JavaScript Solution:

```
/**  
 * Problem: Reorder Data in Log Files  
 * Difficulty: Medium  
 * Tags: array, string, graph, sort  
  
 * 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[]} logs  
 * @return {string[]}  
 */  
var reorderLogFiles = function(logs) {  
  
};
```

TypeScript Solution:

```
/**  
 * Problem: Reorder Data in Log Files  
 * Difficulty: Medium  
 * Tags: array, string, graph, sort  
  
 * 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 reorderLogFiles(logs: string[]): string[] {  
  
};
```

C# Solution:

```
/*  
 * Problem: Reorder Data in Log Files  
 * Difficulty: Medium  
 * Tags: array, string, graph, sort
```

```

/*
 * 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[] ReorderLogFiles(string[] logs) {
        }

    }
}

```

C Solution:

```

/*
 * Problem: Reorder Data in Log Files
 * Difficulty: Medium
 * Tags: array, string, graph, sort
 *
 * 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
 */

/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
char** reorderLogFiles(char** logs, int logsSize, int* returnSize) {

}

```

Go Solution:

```

// Problem: Reorder Data in Log Files
// Difficulty: Medium
// Tags: array, string, graph, sort
//
// 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 reorderLogFiles(logs []string) []string {  
    }  
}
```

Kotlin Solution:

```
class Solution {  
    fun reorderLogFiles(logs: Array<String>): Array<String> {  
        }  
    }
```

Swift Solution:

```
class Solution {  
    func reorderLogFiles(_ logs: [String]) -> [String] {  
        }  
    }
```

Rust Solution:

```
// Problem: Reorder Data in Log Files  
// Difficulty: Medium  
// Tags: array, string, graph, sort  
//  
// 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 reorder_log_files(logs: Vec<String>) -> Vec<String> {  
        }  
    }
```

Ruby Solution:

```
# @param {String[]} logs  
# @return {String[]}  
def reorder_log_files(logs)
```

```
end
```

PHP Solution:

```
class Solution {  
  
    /**  
     * @param String[] $logs  
     * @return String[]  
     */  
    function reorderLogFiles($logs) {  
  
    }  
}
```

Dart Solution:

```
class Solution {  
List<String> reorderLogFiles(List<String> logs) {  
  
}  
}
```

Scala Solution:

```
object Solution {  
def reorderLogFiles(logs: Array[String]): Array[String] = {  
  
}  
}
```

Elixir Solution:

```
defmodule Solution do  
@spec reorder_log_files(logs :: [String.t]) :: [String.t]  
def reorder_log_files(logs) do  
  
end  
end
```

Erlang Solution:

```
-spec reorder_log_files(Logs :: [unicode:unicode_binary()]) ->
[unicode:unicode_binary()].
reorder_log_files(Logs) ->
.
```

Racket Solution:

```
(define/contract (reorder-log-files logs)
(-> (listof string?) (listof string?))
)
```