Problem 1366: Rank Teams by Votes

Problem Information

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

Problem Description

In a special ranking system, each voter gives a rank from highest to lowest to all teams participating in the competition.

The ordering of teams is decided by who received the most position-one votes. If two or more teams tie in the first position, we consider the second position to resolve the conflict, if they tie again, we continue this process until the ties are resolved. If two or more teams are still tied after considering all positions, we rank them alphabetically based on their team letter.

You are given an array of strings

votes

which is the votes of all voters in the ranking systems. Sort all teams according to the ranking system described above.

Return

a string of all teams

sorted

by the ranking system

.

Example 1:

Input:

votes = ["ABC","ACB","ABC","ACB","ACB"]

Output:

"ACB"

Explanation:

Team A was ranked first place by 5 voters. No other team was voted as first place, so team A is the first team. Team B was ranked second by 2 voters and ranked third by 3 voters. Team C was ranked second by 3 voters and ranked third by 2 voters. As most of the voters ranked C second, team C is the second team, and team B is the third.

Example 2:

Input:

votes = ["WXYZ","XYZW"]

Output:

"XWYZ"

Explanation:

X is the winner due to the tie-breaking rule. X has the same votes as W for the first position, but X has one vote in the second position, while W does not have any votes in the second position.

Example 3:

Input:

votes = ["ZMNAGUEDSJYLBOPHRQICWFXTVK"]

Output:

"ZMNAGUEDSJYLBOPHRQICWFXTVK"

Explanation:

Only one voter, so their votes are used for the ranking.

Constraints:

1 <= votes.length <= 1000

1 <= votes[i].length <= 26

votes[i].length == votes[j].length

for

0 <= i, j < votes.length

.

votes[i][j]

is an English

uppercase

letter.

All characters of

votes[i]

are unique.

All the characters that occur in

votes[0]

also occur

in

votes[j]

where

1 <= j < votes.length

.

Code Snippets

C++:

```
class Solution {
public:
string rankTeams(vector<string>& votes) {

}
};
```

Java:

```
class Solution {
public String rankTeams(String[] votes) {

}
}
```

Python3:

```
class Solution:
def rankTeams(self, votes: List[str]) -> str:
```

Python:

```
class Solution(object):
def rankTeams(self, votes):
"""
:type votes: List[str]
```

```
:rtype: str
"""
```

JavaScript:

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

};
```

TypeScript:

```
function rankTeams(votes: string[]): string {

};
```

C#:

```
public class Solution {
public string RankTeams(string[] votes) {

}
}
```

C:

```
char* rankTeams(char** votes, int votesSize) {

}
```

Go:

```
func rankTeams(votes []string) string {

}
```

Kotlin:

```
class Solution {
fun rankTeams(votes: Array<String>): String {

}
}
```

Swift:

```
class Solution {
func rankTeams(_ votes: [String]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn rank_teams(votes: Vec<String>) -> String {

}
}
```

Ruby:

```
# @param {String[]} votes
# @return {String}
def rank_teams(votes)

end
```

PHP:

```
class Solution {

/**
* @param String[] $votes
* @return String
*/
function rankTeams($votes) {

}
}
```

Dart:

```
class Solution {
String rankTeams(List<String> votes) {

}
}
```

Scala:

```
object Solution {
def rankTeams(votes: Array[String]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec rank_teams(votes :: [String.t]) :: String.t
def rank_teams(votes) do

end
end
```

Erlang:

```
-spec rank_teams(Votes :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
rank_teams(Votes) ->
.
```

Racket:

```
(define/contract (rank-teams votes)
(-> (listof string?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Rank Teams by Votes
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string rankTeams(vector<string>& votes) {

}
};
```

Java Solution:

```
/**
* Problem: Rank Teams by Votes
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public String rankTeams(String[] votes) {

}
}
```

Python3 Solution:

```
"""
Problem: Rank Teams by Votes
Difficulty: Medium
Tags: array, string, hash, sort
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Rank Teams by Votes
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string[]} votes
* @return {string}
*/
var rankTeams = function(votes) {

};
```

TypeScript Solution:

```
/**
* Problem: Rank Teams by Votes
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function rankTeams(votes: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Rank Teams by Votes
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public string RankTeams(string[] votes) {

}
}
```

C Solution:

```
/*
* Problem: Rank Teams by Votes
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

char* rankTeams(char** votes, int votesSize) {

}
```

Go Solution:

```
// Problem: Rank Teams by Votes
// Difficulty: Medium
// Tags: array, string, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func rankTeams(votes []string) string {

}
```

Kotlin Solution:

```
class Solution {
fun rankTeams(votes: Array<String>): String {

}
}
```

Swift Solution:

```
class Solution {
func rankTeams(_ votes: [String]) -> String {

}
}
```

Rust Solution:

```
// Problem: Rank Teams by Votes
// Difficulty: Medium
// Tags: array, string, hash, sort
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn rank_teams(votes: Vec<String>) -> String {

}
}
```

Ruby Solution:

```
# @param {String[]} votes
# @return {String}
def rank_teams(votes)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $votes
* @return String
*/
function rankTeams($votes) {

}
}
```

Dart Solution:

```
class Solution {
String rankTeams(List<String> votes) {

}
}
```

Scala Solution:

```
object Solution {
def rankTeams(votes: Array[String]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec rank_teams(votes :: [String.t]) :: String.t
def rank_teams(votes) do

end
end
```

Erlang Solution:

```
-spec rank_teams(Votes :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
rank_teams(Votes) ->
.
```

Racket Solution:

```
(define/contract (rank-teams votes)
(-> (listof string?) string?)
)
```