Problem 2512: Reward Top K Students

Problem Information

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

Problem Description

You are given two string arrays

positive_feedback

and

negative_feedback

, containing the words denoting positive and negative feedback, respectively. Note that

no

word is both positive and negative.

Initially every student has

0

points. Each positive word in a feedback report

increases

the points of a student by

3

, whereas each negative word

decreases

the points by

1

.

You are given

n

feedback reports, represented by a

0-indexed

string array

report

and a

0-indexed

integer array

student_id

, where

student_id[i]

represents the ID of the student who has received the feedback report

report[i]

. The ID of each student is

unique

.

Given an integer

k

, return

the top

k

students after ranking them in

non-increasing

order by their points

. In case more than one student has the same points, the one with the lower ID ranks higher.

Example 1:

Input:

positive_feedback = ["smart","brilliant","studious"], negative_feedback = ["not"], report = ["this student is studious","the student is smart"], student_id = [1,2], k = 2

Output:

[1,2]

Explanation:

Both the students have 1 positive feedback and 3 points but since student 1 has a lower ID he ranks higher.

Example 2:

Input:

positive_feedback = ["smart","brilliant","studious"], negative_feedback = ["not"], report = ["this student is not studious","the student is smart"], student_id = [1,2], k = 2

Output:

[2,1]

Explanation:

- The student with ID 1 has 1 positive feedback and 1 negative feedback, so he has 3-1=2 points. - The student with ID 2 has 1 positive feedback, so he has 3 points. Since student 2 has more points, [2,1] is returned.

Constraints:

1 <= positive_feedback.length, negative_feedback.length <= 10

4

1 <= positive_feedback[i].length, negative_feedback[j].length <= 100

Both

positive_feedback[i]

and

negative_feedback[j]

consists of lowercase English letters.

No word is present in both

positive_feedback

and

negative_feedback

.

n == report.length == student_id.length

1 <= n <= 10

4

report[i]

consists of lowercase English letters and spaces

' '

.

There is a single space between consecutive words of

report[i]

.

1 <= report[i].length <= 100

1 <= student_id[i] <= 10

9

All the values of

student_id[i]

are

unique

.

1 <= k <= n

Code Snippets

C++:

```
class Solution {
public:
vector<int> topStudents(vector<string>& positive_feedback, vector<string>&
negative_feedback, vector<string>& report, vector<int>& student_id, int k) {

}
};
```

Java:

```
class Solution {
public List<Integer> topStudents(String[] positive_feedback, String[]
negative_feedback, String[] report, int[] student_id, int k) {

}
}
```

Python3:

```
class Solution:
def topStudents(self, positive_feedback: List[str], negative_feedback:
List[str], report: List[str], student_id: List[int], k: int) -> List[int]:
```

Python:

```
class Solution(object):
def topStudents(self, positive_feedback, negative_feedback, report,
student_id, k):
"""
:type positive_feedback: List[str]
:type negative_feedback: List[str]
:type report: List[str]
:type student_id: List[int]
:type k: int
:rtype: List[int]
```

```
"""
```

JavaScript:

```
/**
* @param {string[]} positive_feedback
* @param {string[]} negative_feedback
* @param {string[]} report
* @param {number[]} student_id
* @param {number} k
* @return {number[]}
*/
var topStudents = function(positive_feedback, negative_feedback, report,
student_id, k) {

};
```

TypeScript:

```
function topStudents(positive_feedback: string[], negative_feedback:
string[], report: string[], student_id: number[], k: number): number[] {

};
```

C#:

```
public class Solution {
public IList<int> TopStudents(string[] positive_feedback, string[]
negative_feedback, string[] report, int[] student_id, int k) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* topStudents(char** positive_feedback, int positive_feedbackSize, char**
negative_feedback, int negative_feedbackSize, char** report, int reportSize,
int* student_id, int student_idSize, int k, int* returnSize) {
}
```

Go:

```
func topStudents(positive_feedback []string, negative_feedback []string,
report []string, student_id []int, k int) []int {

}
```

Kotlin:

```
class Solution {
fun topStudents(positive_feedback: Array<String>, negative_feedback:
Array<String>, report: Array<String>, student_id: IntArray, k: Int):
List<Int> {

}
}
```

Swift:

```
class Solution {
func topStudents(_ positive_feedback: [String], _ negative_feedback:
[String], _ report: [String], _ student_id: [Int], _ k: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn top_students(positive_feedback: Vec<String>, negative_feedback:
Vec<String>, report: Vec<String>, student_id: Vec<i32>, k: i32) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {String[]} positive_feedback
# @param {String[]} negative_feedback
# @param {String[]} report
# @param {Integer[]} student_id
# @param {Integer} k
# @return {Integer[]}
def top_students(positive_feedback, negative_feedback, report, student_id, k)

end
```

PHP:

```
class Solution {

/**
* @param String[] $positive_feedback
* @param String[] $negative_feedback
* @param String[] $report
* @param Integer[] $student_id
* @param Integer $k
* @return Integer[]
*/
function topStudents($positive_feedback, $negative_feedback, $report,
$student_id, $k) {

}
}
```

Dart:

```
class Solution {
List<int> topStudents(List<String> positive_feedback, List<String>
negative_feedback, List<String> report, List<int> student_id, int k) {

}
}
```

Scala:

```
object Solution {
def topStudents(positive_feedback: Array[String], negative_feedback:
Array[String], report: Array[String], student_id: Array[Int], k: Int):
List[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec top_students(positive_feedback :: [String.t], negative_feedback ::
[String.t], report :: [String.t], student_id :: [integer], k :: integer) ::
[integer]
def top_students(positive_feedback, negative_feedback, report, student_id, k)
do

end
end
```

Erlang:

```
-spec top_students(Positive_feedback :: [unicode:unicode_binary()],
Negative_feedback :: [unicode:unicode_binary()], Report ::
[unicode:unicode_binary()], Student_id :: [integer()], K :: integer()) ->
[integer()].
top_students(Positive_feedback, Negative_feedback, Report, Student_id, K) ->
.
```

Racket:

```
(define/contract (top-students positive_feedback negative_feedback report
student_id k)
(-> (listof string?) (listof string?) (listof string?) (listof
exact-integer?) exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reward Top K Students
* Difficulty: Medium
* Tags: array, string, hash, sort, queue, heap
*
```

```
* 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:
vector<int> topStudents(vector<string>& positive_feedback, vector<string>&
negative_feedback, vector<string>& report, vector<int>& student_id, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Reward Top K Students
* Difficulty: Medium
* Tags: array, string, hash, sort, queue, heap
*
* 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 List<Integer> topStudents(String[] positive_feedback, String[]
negative_feedback, String[] report, int[] student_id, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Reward Top K Students
Difficulty: Medium
Tags: array, string, hash, sort, queue, heap

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 topStudents(self, positive_feedback: List[str], negative_feedback:
List[str], report: List[str], student_id: List[int], k: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def topStudents(self, positive_feedback, negative_feedback, report,
student_id, k):
"""
:type positive_feedback: List[str]
:type negative_feedback: List[str]
:type report: List[str]
:type student_id: List[int]
:type k: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Reward Top K Students
* Difficulty: Medium
* Tags: array, string, hash, sort, queue, heap
*
* 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[]} positive_feedback
* @param {string[]} negative_feedback
* @param {string[]} report
* @param {number[]} student_id
* @param {number} k
* @return {number[]}
*/
var topStudents = function(positive_feedback, negative_feedback, report,
student_id, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Reward Top K Students
* Difficulty: Medium
* Tags: array, string, hash, sort, queue, heap
*
* 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 topStudents(positive_feedback: string[], negative_feedback:
string[], report: string[], student_id: number[], k: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Reward Top K Students
* Difficulty: Medium
* Tags: array, string, hash, sort, queue, heap
*
* 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 IList<int> TopStudents(string[] positive_feedback, string[]
negative_feedback, string[] report, int[] student_id, int k) {

}
}
```

C Solution:

```
/*
* Problem: Reward Top K Students
* Difficulty: Medium
* Tags: array, string, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* topStudents(char** positive_feedback, int positive_feedbackSize, char**
negative_feedback, int negative_feedbackSize, char** report, int reportSize,
int* student_id, int student_idSize, int k, int* returnSize) {

}
```

Go Solution:

```
// Problem: Reward Top K Students
// Difficulty: Medium
// Tags: array, string, hash, sort, queue, heap
//
// 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 topStudents(positive_feedback []string, negative_feedback []string,
report []string, student_id []int, k int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun topStudents(positive_feedback: Array<String>, negative_feedback:
Array<String>, report: Array<String>, student_id: IntArray, k: Int):
List<Int> {

}
}
```

Swift Solution:

```
class Solution {
func topStudents(_ positive_feedback: [String], _ negative_feedback:
[String], _ report: [String], _ student_id: [Int], _ k: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Reward Top K Students
// Difficulty: Medium
// Tags: array, string, hash, sort, queue, heap
//
// 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 top_students(positive_feedback: Vec<String>, negative_feedback:
Vec<String>, report: Vec<String>, student_id: Vec<i32>, k: i32) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {String[]} positive_feedback
# @param {String[]} negative_feedback
# @param {String[]} report
# @param {Integer[]} student_id
# @param {Integer} k
# @return {Integer[]}
```

```
def top_students(positive_feedback, negative_feedback, report, student_id, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $positive_feedback
* @param String[] $negative_feedback
* @param String[] $report
* @param Integer[] $student_id
* @param Integer $k
* @return Integer[]
*/
function topStudents($positive_feedback, $negative_feedback, $report,
$student_id, $k) {

}
}
```

Dart Solution:

```
class Solution {
List<int> topStudents(List<String> positive_feedback, List<String>
negative_feedback, List<String> report, List<int> student_id, int k) {

}
}
```

Scala Solution:

```
object Solution {
def topStudents(positive_feedback: Array[String], negative_feedback:
Array[String], report: Array[String], student_id: Array[Int], k: Int):
List[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec top_students(positive_feedback :: [String.t], negative_feedback ::
[String.t], report :: [String.t], student_id :: [integer], k :: integer) ::
[integer]
def top_students(positive_feedback, negative_feedback, report, student_id, k)
do

end
end
```

Erlang Solution:

```
-spec top_students(Positive_feedback :: [unicode:unicode_binary()],
Negative_feedback :: [unicode:unicode_binary()], Report ::
[unicode:unicode_binary()], Student_id :: [integer()], K :: integer()) ->
[integer()].
top_students(Positive_feedback, Negative_feedback, Report, Student_id, K) ->
.
```

Racket Solution:

```
(define/contract (top-students positive_feedback negative_feedback report
student_id k)
(-> (listof string?) (listof string?) (listof string?) (listof
exact-integer?) exact-integer? (listof exact-integer?))
)
```