Problem 158: Read N Characters Given read4 II - Call Multiple Times

Problem Information

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

Problem Description

Given a

file

and assume that you can only read the file using a given method

read4

, implement a method

read

to read

n

characters. Your method

read

may be

called multiple times

.

Method read4:

The API

read4

reads

four consecutive characters

from

file

, then writes those characters into the buffer array

buf4

.

The return value is the number of actual characters read.

Note that

read4()

has its own file pointer, much like

FILE *fp

in C.

Definition of read4:

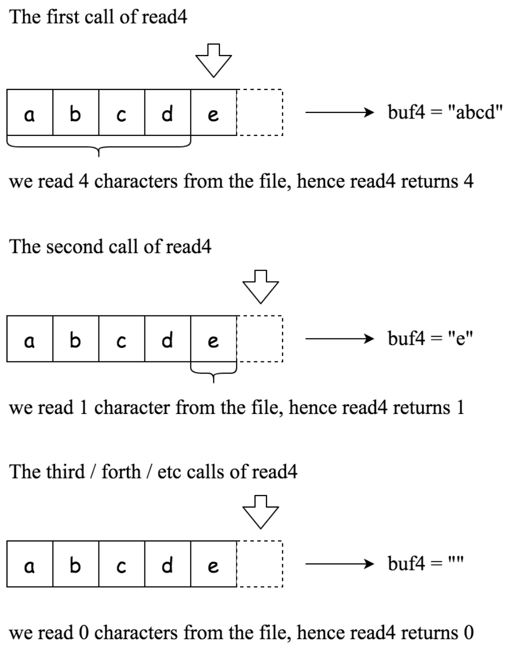
Parameter: char[] buf4 Returns: int

buf4[] is a destination, not a source. The results from read4 will be copied to buf4[].

Below is a high-level example of how

read4

works:


The first call of read4
a b c d e
buf4 = "abcd"
we read 4 characters from the file, hence read4 returns 4
The second call of read4
a b c d e
buf4 = "e"
we read 1 character from the file, hence read4 returns 1
The third / forth / etc calls of read4
a b c d e
buf4 = ""
we read 0 characters from the file, hence read4 returns 0

File file("abcde

"); // File is "

abcde

", initially file pointer (fp) points to 'a' char[] buf4 = new char[4]; // Create buffer with enough space to store characters read4(buf4); // read4 returns 4. Now buf4 = "abcd", fp points to 'e' read4(buf4); // read4 returns 1. Now buf4 = "e", fp points to end of file read4(buf4); // read4 returns 0. Now buf4 = "", fp points to end of file

Method read:

By using the

read4

method, implement the method read that reads n

characters from

file

and store it in the buffer array

buf

. Consider that you cannot manipulate

file

directly.

The return value is the number of actual characters read.

Definition of read:

Parameters: char[] buf, int n Returns: int

buf[] is a destination, not a source. You will need to write the results to buf[].

Note:

Consider that you cannot manipulate the file directly. The file is only accessible for

read4

but not for

read

.

The read function may be

called multiple times

.

Please remember to

RESET

your class variables declared in Solution, as static/class variables are persisted across multiple test cases. Please see

here

for more details.

You may assume the destination buffer array,

buf

, is guaranteed to have enough space for storing

n

characters.

It is guaranteed that in a given test case the same buffer

buf

is called by

read

.

Example 1:

Input:

file = "abc", queries = [1,2,1]

Output:

[1,2,0]

Explanation:

The test case represents the following scenario: File file("abc"); Solution sol; sol.read(buf, 1); // After calling your read method, buf should contain "a". We read a total of 1 character from the file, so return 1. sol.read(buf, 2); // Now buf should contain "bc". We read a total of 2 characters from the file, so return 2. sol.read(buf, 1); // We have reached the end of file, no more characters can be read. So return 0. Assume buf is allocated and guaranteed to have enough space for storing all characters from the file.

Example 2:

Input:

file = "abc", queries = [4,1]

Output:

[3,0]

Explanation:

The test case represents the following scenario: File file("abc"); Solution sol; sol.read(buf, 4); // After calling your read method, buf should contain "abc". We read a total of 3 characters from the file, so return 3. sol.read(buf, 1); // We have reached the end of file, no more characters can be read. So return 0.

Constraints:

1 <= file.length <= 500

file

consist of English letters and digits.

1 <= queries.length <= 10

1 <= queries[i] <= 500

Code Snippets

C++:

```
/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

class Solution {
public:
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
int read(char *buf, int n) {

}
};
```

Java:

```
/**
* The read4 API is defined in the parent class Reader4.
* int read4(char[] buf4);
*/

public class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int read(char[] buf, int n) {

}
}
```

Python3:

```
# The read4 API is already defined for you.
# def read4(buf4: List[str]) -> int:

class Solution:
def read(self, buf: List[str], n: int) -> int:
```

Python:

```
# The read4 API is already defined for you.
# @param buf4, List[str]
# @return an integer
# def read4(buf4):

class Solution(object):
def read(self, buf, n):
"""
:type buf: List[str]
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* Definition for read4()
*
* @param {character[]} buf Destination buffer
* @return {number} The number of characters read
* read4 = function(buf4) {
* ...
* };
*/

/**
* @param {function} read4()
* @return {function}
*/
var solution = function(read4) {
```

```
/**
* @param {character[]} buf Destination buffer
* @param {number} n Number of characters to read
* @return {number} The number of actual characters read
*/
return function(buf, n) {

};
};
```

TypeScript:

```
/**
* Definition for read4()
* read4 = function(buf4: string[]): number {
* ...
* };
*/

var solution = function(read4: any) {

return function(buf: string[], n: number): number {

};
};
```

C#:

```
/**
* The Read4 API is defined in the parent class Reader4.
* int Read4(char[] buf4);
*/

public class Solution : Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int Read(char[] buf, int n) {

}
}
```

C:

```
/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

typedef struct {

} Solution;

/** initialize your data structure here. */
Solution* solutionCreate() {

}

/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
int _read(Solution* obj, char* buf, int n) {

}
```

Go:

```
/**
* The read4 API is already defined for you.
*
* read4 := func(buf4 []byte) int
*
* // Below is an example of how the read4 API can be called.
* file := File("abcdefghijk") // File is "abcdefghijk", initially file
pointer (fp) points to 'a'
* buf4 := make([]byte, 4) // Create buffer with enough space to store
characters
* read4(buf4) // read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
* read4(buf4) // read4 returns 4. Now buf = ['e','f','g','h'], fp points to
```

```
'i'
* read4(buf4) // read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file
*/

var solution = func(read4 func([]byte) int) func([]byte, int) int {
// implement read below.
return func(buf []byte, n int) int {

}
}
```

Kotlin:

```
/**
* The read4 API is defined in the parent class Reader4.
* fun read4(buf4:CharArray): Int {}
*/

class Solution:Reader4() {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
override fun read(buf:CharArray, n:Int): Int {

}
}
```

Swift:

```
/**
* The read4 API is defined in the parent class Reader4.
* func read4(_ buf4: inout [Character]) -> Int;
*/

class Solution : Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
func read(_ buf: inout [Character], _ n: Int) -> Int {

}
}
```

Ruby:

```
# The read4 API is already defined for you.
# Below is an example of how the read4 API can be called.
# file = File.new("abcdefghijk") File is "abcdefghijk", initially file
pointer (fp) points to 'a'
# buf4 = [' '] * 4 Create buffer with enough space to store characters
# read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
# read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to
'i'
# read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file

class Solution
# @param {List[str]} buf
# @param {int} n
# @return {int}
def read(buf, n)

end
end
```

PHP:

```
/* The read4 API is defined in the parent class Reader4.
public function read4(&$buf4){} */

class Solution extends Reader4 {
/**
* @param Char[] &$buf Destination buffer
* @param Integer $n Number of characters to read
* @return Integer The number of actual characters read
*/
function read(&$buf, $n) {

}
}
```

Scala:

```
/**
* The read4 API is defined in the parent class Reader4.
* def read4(buf4: Array[Char]): Int = {}
*/

class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
def read(buf: Array[Char], n: Int): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Read N Characters Given read4 II - Call Multiple Times
* Difficulty: Hard
* Tags: array
*
* 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
*/

/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

class Solution {
```

```
public:
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
int read(char *buf, int n) {

}
};
```

Java Solution:

```
/**
* Problem: Read N Characters Given read4 II - Call Multiple Times
* Difficulty: Hard
* Tags: array
*
* 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
*/

/**
* The read4 API is defined in the parent class Reader4.
* int read4(char[] buf4);
*/

public class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int read(char[] buf, int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Read N Characters Given read4 II - Call Multiple Times
Difficulty: Hard
Tags: array

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
"""

# The read4 API is already defined for you.
# def read4(buf4: List[str]) -> int:

class Solution:
def read(self, buf: List[str], n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# The read4 API is already defined for you.
# @param buf4, List[str]
# @return an integer
# def read4(buf4):

class Solution(object):
def read(self, buf, n):
"""
:type buf: List[str]
:type n: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Read N Characters Given read4 II - Call Multiple Times
* Difficulty: Hard
* Tags: array
*
* 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
*/

/**
* Definition for read4()
*
* @param {character[]} buf Destination buffer
* @return {number} The number of characters read
* read4 = function(buf4) {
* ...
* };
*/

/**
* @param {function} read4()
* @return {function}
*/
var solution = function(read4) {
/**
* @param {character[]} buf Destination buffer
* @param {number} n Number of characters to read
* @return {number} The number of actual characters read
*/
return function(buf, n) {

};
};
```

TypeScript Solution:

```
/**
* Problem: Read N Characters Given read4 II - Call Multiple Times
* Difficulty: Hard
* Tags: array
*
* 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
*/

/**
```

```
* Definition for read4()
* read4 = function(buf4: string[]): number {
* ...
* };
*/

var solution = function(read4: any) {

return function(buf: string[], n: number): number {

};
};
```

C# Solution:

```
/*
* Problem: Read N Characters Given read4 II - Call Multiple Times
* Difficulty: Hard
* Tags: array
*
* 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
*/

/**
* The Read4 API is defined in the parent class Reader4.
* int Read4(char[] buf4);
*/

public class Solution : Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int Read(char[] buf, int n) {

}
}
```

C Solution:

```c
/*
* Problem: Read N Characters Given read4 II - Call Multiple Times
* Difficulty: Hard
* Tags: array
*
* 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
*/

/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

typedef struct {

} Solution;

/** initialize your data structure here. */
Solution* solutionCreate() {

}

/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
int _read(Solution* obj, char* buf, int n) {

}
```

Go Solution:

```go
// Problem: Read N Characters Given read4 II - Call Multiple Times
// Difficulty: Hard
// Tags: array
//
// 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

/**
* The read4 API is already defined for you.
*
* read4 := func(buf4 []byte) int
*
* // Below is an example of how the read4 API can be called.
* file := File("abcdefghijk") // File is "abcdefghijk", initially file
pointer (fp) points to 'a'
* buf4 := make([]byte, 4) // Create buffer with enough space to store
characters
* read4(buf4) // read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
* read4(buf4) // read4 returns 4. Now buf = ['e','f','g','h'], fp points to
'i'
* read4(buf4) // read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file
*/

var solution = func(read4 func([]byte) int) func([]byte, int) int {
// implement read below.
return func(buf []byte, n int) int {

}
}
```

Kotlin Solution:

```
/**
* The read4 API is defined in the parent class Reader4.
* fun read4(buf4:CharArray): Int {}
*/

class Solution:Reader4() {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
override fun read(buf:CharArray, n:Int): Int {
}
}
```

Swift Solution:

```
/**
* The read4 API is defined in the parent class Reader4.
* func read4(_ buf4: inout [Character]) -> Int;
*/

class Solution : Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
func read(_ buf: inout [Character], _ n: Int) -> Int {

}
}
```

Ruby Solution:

```
# The read4 API is already defined for you.
# Below is an example of how the read4 API can be called.
# file = File.new("abcdefghijk") File is "abcdefghijk", initially file
pointer (fp) points to 'a'
# buf4 = [' '] * 4 Create buffer with enough space to store characters
# read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
# read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to
'i'
# read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file

class Solution
# @param {List[str]} buf
# @param {int} n
# @return {int}
def read(buf, n)
```

```
end
end
```

PHP Solution:

```
/* The read4 API is defined in the parent class Reader4.
public function read4(&$buf4){} */

class Solution extends Reader4 {
/**
* @param Char[] &$buf Destination buffer
* @param Integer $n Number of characters to read
* @return Integer The number of actual characters read
*/
function read(&$buf, $n) {

}
}
```

Scala Solution:

```
/**
* The read4 API is defined in the parent class Reader4.
* def read4(buf4: Array[Char]): Int = {}
*/

class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
def read(buf: Array[Char], n: Int): Int = {

}
}
```