Problem 815: Bus Routes

Problem Information

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

Problem Description

You are given an array

routes

representing bus routes where

routes[i]

is a bus route that the

i

th

bus repeats forever.

For example, if

routes[0] = [1, 5, 7]

, this means that the

0

th

bus travels in the sequence

1 -> 5 -> 7 -> 1 -> 5 -> 7 -> 1 -> ...

forever.

You will start at the bus stop

source

(You are not on any bus initially), and you want to go to the bus stop

target

. You can travel between bus stops by buses only.

Return

the least number of buses you must take to travel from

source

to

target

. Return

-1

if it is not possible.

Example 1:

Input:

routes = [[1,2,7],[3,6,7]], source = 1, target = 6

Output:

2

Explanation:

The best strategy is take the first bus to the bus stop 7, then take the second bus to the bus stop 6.

Example 2:

Input:

routes = [[7,12],[4,5,15],[6],[15,19],[9,12,13]], source = 15, target = 12

Output:

-1

Constraints:

1 <= routes.length <= 500

.

1 <= routes[i].length <= 10

5

All the values of

routes[i]

are

unique

.

sum(routes[i].length) <= 10

5

0 <= routes[i][j] < 10

6

0 <= source, target < 10

6

Code Snippets

C++:

```
class Solution {
public:
int numBusesToDestination(vector<vector<int>>& routes, int source, int
target) {

}
};
```

Java:

```
class Solution {
public int numBusesToDestination(int[][] routes, int source, int target) {

}
}
```

Python3:

```
class Solution:
def numBusesToDestination(self, routes: List[List[int]], source: int, target:
int) -> int:
```

Python:

```
class Solution(object):
def numBusesToDestination(self, routes, source, target):
```

```
"""
:type routes: List[List[int]]
:type source: int
:type target: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[][]} routes
* @param {number} source
* @param {number} target
* @return {number}
*/
var numBusesToDestination = function(routes, source, target) {

};
```

TypeScript:

```
function numBusesToDestination(routes: number[][], source: number, target:
number): number {

};
```

C#:

```
public class Solution {
public int NumBusesToDestination(int[][] routes, int source, int target) {

}
}
```

C:

```
int numBusesToDestination(int** routes, int routesSize, int* routesColSize,
int source, int target) {

}
```

Go:

```
func numBusesToDestination(routes [][]int, source int, target int) int {

}
```

Kotlin:

```
class Solution {
fun numBusesToDestination(routes: Array<IntArray>, source: Int, target: Int):
Int {

}
}
```

Swift:

```
class Solution {
func numBusesToDestination(_ routes: [[Int]], _ source: Int, _ target: Int)
-> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_buses_to_destination(routes: Vec<Vec<i32>>, source: i32, target:
i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} routes
# @param {Integer} source
# @param {Integer} target
# @return {Integer}
def num_buses_to_destination(routes, source, target)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $routes
* @param Integer $source
* @param Integer $target
* @return Integer
*/
function numBusesToDestination($routes, $source, $target) {

}
}
```

Dart:

```
class Solution {
int numBusesToDestination(List<List<int>> routes, int source, int target) {

}
}
```

Scala:

```
object Solution {
def numBusesToDestination(routes: Array[Array[Int]], source: Int, target:
Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_buses_to_destination(routes :: [[integer]], source :: integer,
target :: integer) :: integer
def num_buses_to_destination(routes, source, target) do

end
end
```

Erlang:

```
-spec num_buses_to_destination(Routes :: [[integer()]], Source :: integer(),
Target :: integer()) -> integer().
num_buses_to_destination(Routes, Source, Target) ->
.
```

Racket:

```
(define/contract (num-buses-to-destination routes source target)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?
exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Bus Routes
* Difficulty: Hard
* Tags: array, hash, search
*
* 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:
int numBusesToDestination(vector<vector<int>>& routes, int source, int
target) {

}
};
```

Java Solution:

```
/**
* Problem: Bus Routes
* Difficulty: Hard
* Tags: array, hash, search
*
```

```
* 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 int numBusesToDestination(int[][] routes, int source, int target) {

}
}
```

Python3 Solution:

```
"""
Problem: Bus Routes
Difficulty: Hard
Tags: array, hash, search

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 numBusesToDestination(self, routes: List[List[int]], source: int, target:
int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numBusesToDestination(self, routes, source, target):
"""
:type routes: List[List[int]]
:type source: int
:type target: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Bus Routes
* Difficulty: Hard
* Tags: array, hash, search
*
* 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 {number[][]} routes
* @param {number} source
* @param {number} target
* @return {number}
*/
var numBusesToDestination = function(routes, source, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Bus Routes
* Difficulty: Hard
* Tags: array, hash, search
*
* 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 numBusesToDestination(routes: number[][], source: number, target:
number): number {

};
```

C# Solution:

```
/*
* Problem: Bus Routes
* Difficulty: Hard
```

```
* Tags: array, hash, search
*
* 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 int NumBusesToDestination(int[][] routes, int source, int target) {

}
}
```

C Solution:

```
/*
* Problem: Bus Routes
* Difficulty: Hard
* Tags: array, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int numBusesToDestination(int** routes, int routesSize, int* routesColSize,
int source, int target) {

}
```

Go Solution:

```
// Problem: Bus Routes
// Difficulty: Hard
// Tags: array, hash, search
//
// 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 numBusesToDestination(routes [][]int, source int, target int) int {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun numBusesToDestination(routes: Array<IntArray>, source: Int, target: Int):
Int {

}
}
```

Swift Solution:

```
class Solution {
func numBusesToDestination(_ routes: [[Int]], _ source: Int, _ target: Int)
-> Int {

}
}
```

Rust Solution:

```
// Problem: Bus Routes
// Difficulty: Hard
// Tags: array, hash, search
//
// 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 num_buses_to_destination(routes: Vec<Vec<i32>>, source: i32, target:
i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} routes
# @param {Integer} source
# @param {Integer} target
# @return {Integer}
def num_buses_to_destination(routes, source, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $routes
* @param Integer $source
* @param Integer $target
* @return Integer
*/
function numBusesToDestination($routes, $source, $target) {

}
}
```

Dart Solution:

```
class Solution {
int numBusesToDestination(List<List<int>> routes, int source, int target) {

}
}
```

Scala Solution:

```
object Solution {
def numBusesToDestination(routes: Array[Array[Int]], source: Int, target:
Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_buses_to_destination(routes :: [[integer]], source :: integer,
target :: integer) :: integer
def num_buses_to_destination(routes, source, target) do

end
end
```

Erlang Solution:

```
-spec num_buses_to_destination(Routes :: [[integer()]], Source :: integer(),
Target :: integer()) -> integer().
num_buses_to_destination(Routes, Source, Target) ->
.
```

Racket Solution:

```
(define/contract (num-buses-to-destination routes source target)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?
exact-integer?)
)
```