Problem 1184: Distance Between Bus Stops

Problem Information

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

Problem Description

A bus has

n

stops numbered from

0

to

n - 1

that form a circle. We know the distance between all pairs of neighboring stops where

distance[i]

is the distance between the stops number

i

and

(i + 1) % n

.

The bus goes along both directions i.e. clockwise and counterclockwise.

Return the shortest distance between the given

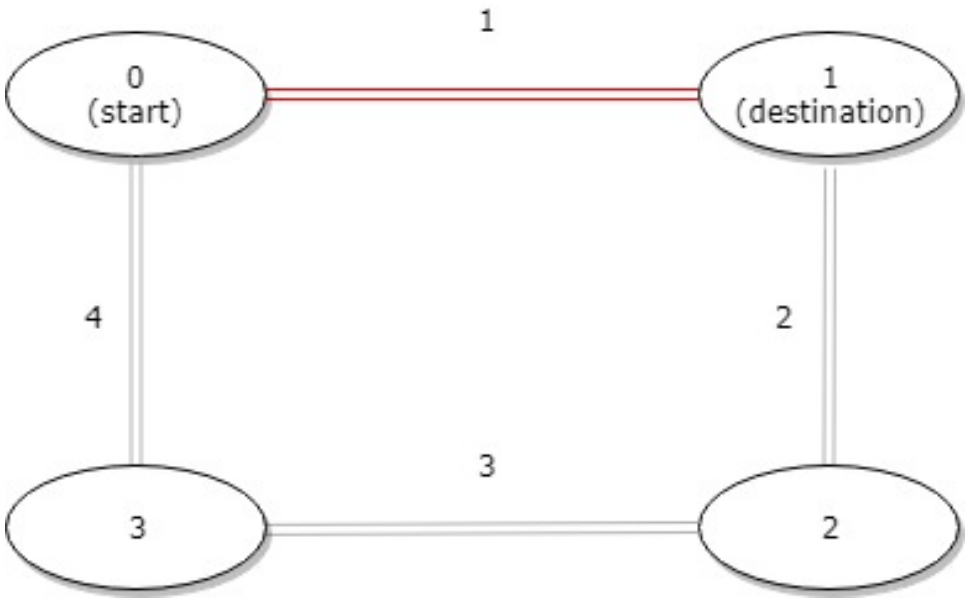
start

and

destination

stops.

Example 1:



Input:

distance = [1,2,3,4], start = 0, destination = 1

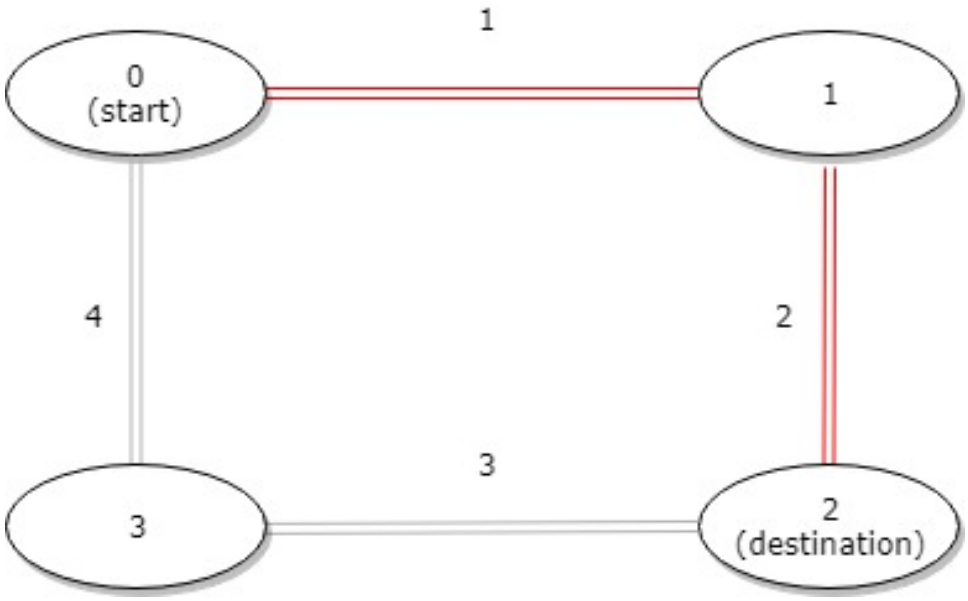
Output:

1

Explanation:

Distance between 0 and 1 is 1 or 9, minimum is 1.

Example 2:



Input:

distance = [1,2,3,4], start = 0, destination = 2

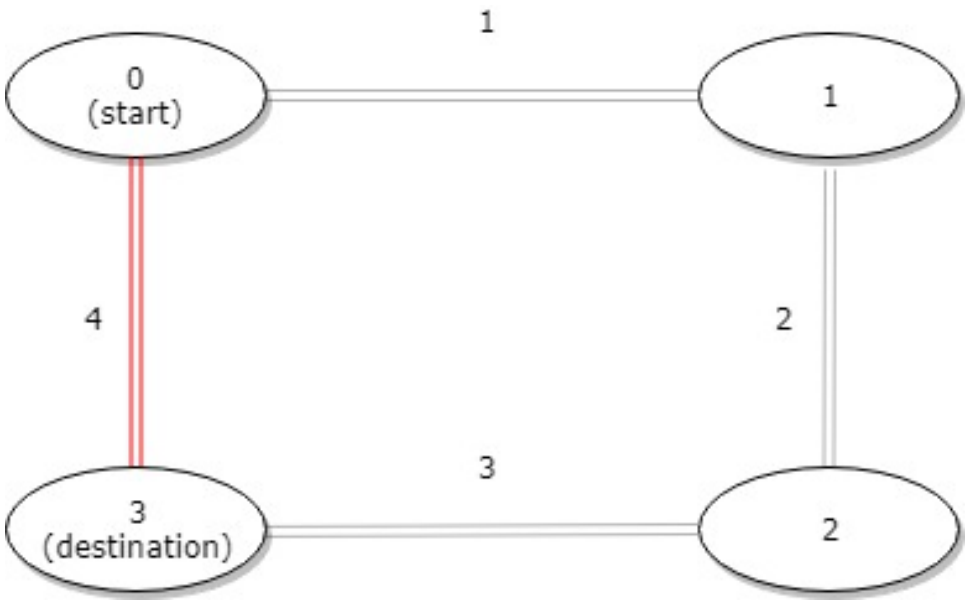
Output:

3

Explanation:

Distance between 0 and 2 is 3 or 7, minimum is 3.

Example 3:



Input:

distance = [1,2,3,4], start = 0, destination = 3

Output:

4

Explanation:

Distance between 0 and 3 is 6 or 4, minimum is 4.

Constraints:

1 <= n <= 10^4

distance.length == n

0 <= start, destination < n

0 <= distance[i] <= 10^4

Code Snippets

C++:

```
class Solution {
public:
int distanceBetweenBusStops(vector<int>& distance, int start, int
destination) {

}
};
```

Java:

```
class Solution {
public int distanceBetweenBusStops(int[] distance, int start, int
destination) {

}
}
```

Python3:

```
class Solution:
def distanceBetweenBusStops(self, distance: List[int], start: int,
destination: int) -> int:
```

Python:

```
class Solution(object):
def distanceBetweenBusStops(self, distance, start, destination):
"""
:type distance: List[int]
:type start: int
:type destination: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} distance
* @param {number} start
* @param {number} destination
* @return {number}
```

```
*/
var distanceBetweenBusStops = function(distance, start, destination) {

};
```

TypeScript:

```
function distanceBetweenBusStops(distance: number[], start: number,
destination: number): number {

};
```

C#:

```
public class Solution {
public int DistanceBetweenBusStops(int[] distance, int start, int
destination) {

}
}
```

C:

```
int distanceBetweenBusStops(int* distance, int distanceSize, int start, int
destination){

}
```

Go:

```
func distanceBetweenBusStops(distance []int, start int, destination int) int
{

}
```

Kotlin:

```
class Solution {
fun distanceBetweenBusStops(distance: IntArray, start: Int, destination:
Int): Int {
```

```
}
}
```

Swift:

```
class Solution {
func distanceBetweenBusStops(_ distance: [Int], _ start: Int, _ destination:
Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn distance_between_bus_stops(distance: Vec<i32>, start: i32,
destination: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} distance
# @param {Integer} start
# @param {Integer} destination
# @return {Integer}
def distance_between_bus_stops(distance, start, destination)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $distance
* @param Integer $start
* @param Integer $destination
* @return Integer
*/
```

```
function distanceBetweenBusStops($distance, $start, $destination) {

}
}
```

Scala:

```
object Solution {
def distanceBetweenBusStops(distance: Array[Int], start: Int, destination:
Int): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Distance Between Bus Stops
* Difficulty: Easy
* 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
*/

class Solution {
public:
int distanceBetweenBusStops(vector<int>& distance, int start, int
destination) {

}
};
```

Java Solution:

```
/**
* Problem: Distance Between Bus Stops
```

```
* Difficulty: Easy
* 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
*/

class Solution {
public int distanceBetweenBusStops(int[] distance, int start, int
destination) {

}
}
```

Python3 Solution:

```
"""
Problem: Distance Between Bus Stops
Difficulty: Easy
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
"""

class Solution:
def distanceBetweenBusStops(self, distance: List[int], start: int,
destination: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def distanceBetweenBusStops(self, distance, start, destination):
"""
:type distance: List[int]
:type start: int
:type destination: int
```

```
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Distance Between Bus Stops
* Difficulty: Easy
* 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
*/

/**
* @param {number[]} distance
* @param {number} start
* @param {number} destination
* @return {number}
*/
var distanceBetweenBusStops = function(distance, start, destination) {

};
```

TypeScript Solution:

```
/**
* Problem: Distance Between Bus Stops
* Difficulty: Easy
* 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
*/

function distanceBetweenBusStops(distance: number[], start: number,
destination: number): number {

};
```

C# Solution:

```
/*
* Problem: Distance Between Bus Stops
* Difficulty: Easy
* 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
*/

public class Solution {
public int DistanceBetweenBusStops(int[] distance, int start, int
destination) {

}
}
```

C Solution:

```
/*
* Problem: Distance Between Bus Stops
* Difficulty: Easy
* 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
*/



int distanceBetweenBusStops(int* distance, int distanceSize, int start, int
destination){

}
```

Go Solution:

```
// Problem: Distance Between Bus Stops
// Difficulty: Easy
```

```
// 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

func distanceBetweenBusStops(distance []int, start int, destination int) int
{

}
```

Kotlin Solution:

```
class Solution {
fun distanceBetweenBusStops(distance: IntArray, start: Int, destination:
Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func distanceBetweenBusStops(_ distance: [Int], _ start: Int, _ destination:
Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Distance Between Bus Stops
// Difficulty: Easy
// 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

impl Solution {
pub fn distance_between_bus_stops(distance: Vec<i32>, start: i32,
```

```
destination: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} distance
# @param {Integer} start
# @param {Integer} destination
# @return {Integer}
def distance_between_bus_stops(distance, start, destination)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $distance
* @param Integer $start
* @param Integer $destination
* @return Integer
*/
function distanceBetweenBusStops($distance, $start, $destination) {

}
}
```

Scala Solution:

```
object Solution {
def distanceBetweenBusStops(distance: Array[Int], start: Int, destination:
Int): Int = {

}
}
```