Problem 2374: Node With Highest Edge Score

Problem Information

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

Problem Description

You are given a directed graph with

n

nodes labeled from

0

to

n - 1

, where each node has

exactly one

outgoing edge.

The graph is represented by a given

0-indexed

integer array

edges

of length

n

, where

edges[i]

indicates that there is a

directed

edge from node

i

to node

edges[i]

.

The

edge score

of a node

i

is defined as the sum of the

labels

of all the nodes that have an edge pointing to

i

.

Return

the node with the highest

edge score

. If multiple nodes have the same

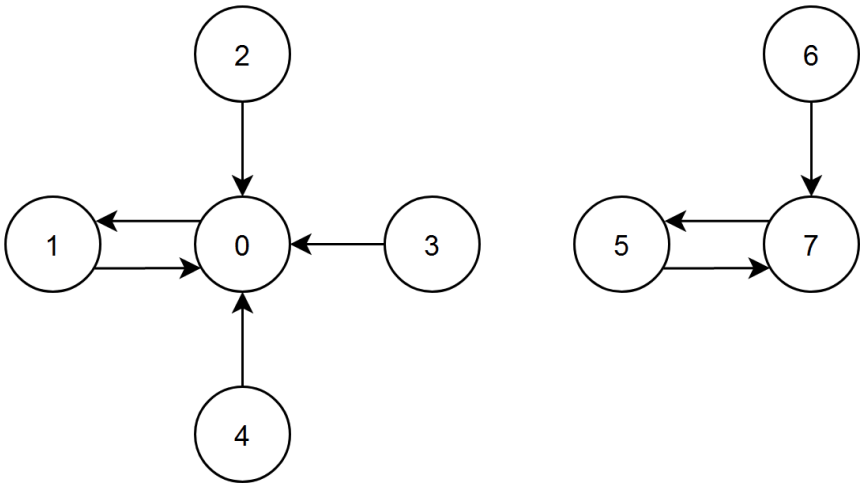
edge score

, return the node with the

smallest

index.

Example 1:



Input:

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

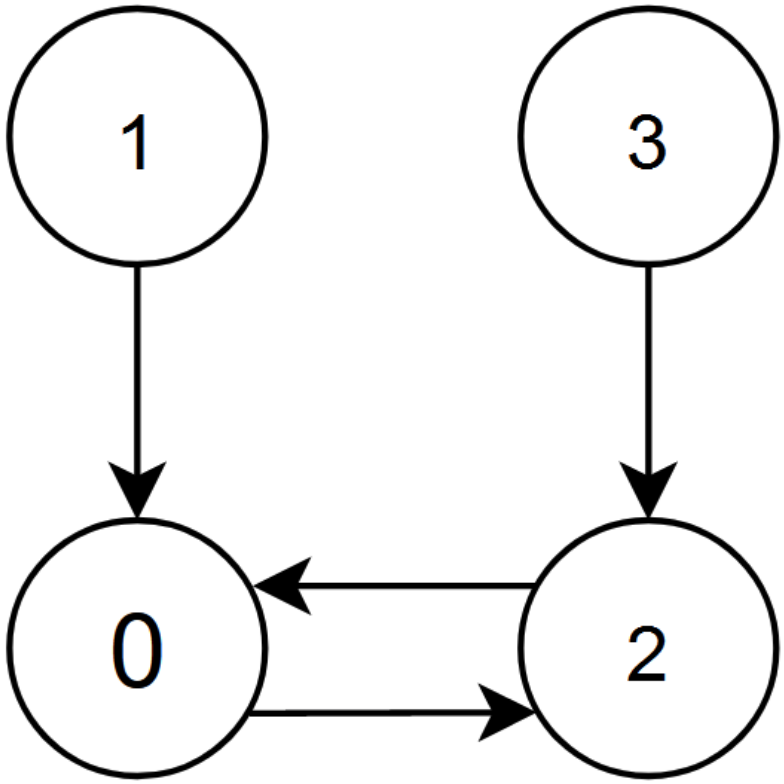
Output:

7

Explanation:

- The nodes 1, 2, 3 and 4 have an edge pointing to node 0. The edge score of node 0 is 1 + 2 + 3 + 4 = 10. - The node 0 has an edge pointing to node 1. The edge score of node 1 is 0. - The node 7 has an edge pointing to node 5. The edge score of node 5 is 7. - The nodes 5 and 6 have an edge pointing to node 7. The edge score of node 7 is 5 + 6 = 11. Node 7 has the highest edge score so return 7.

Example 2:



Input:

edges = [2,0,0,2]

Output:

0

Explanation:

- The nodes 1 and 2 have an edge pointing to node 0. The edge score of node 0 is 1 + 2 = 3. - The nodes 0 and 3 have an edge pointing to node 2. The edge score of node 2 is 0 + 3 = 3. Nodes 0 and 2 both have an edge score of 3. Since node 0 has a smaller index, we return 0.

Constraints:

n == edges.length

2 <= n <= 10

5

0 <= edges[i] < n

edges[i] != i

Code Snippets

C++:

```
class Solution {
public:
int edgeScore(vector<int>& edges) {

}
};
```

Java:

```
class Solution {
public int edgeScore(int[] edges) {

}
}
```

Python3:

```
class Solution:
def edgeScore(self, edges: List[int]) -> int:
```

Python:

```
class Solution(object):
def edgeScore(self, edges):
"""
:type edges: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} edges
* @return {number}
*/
var edgeScore = function(edges) {

};
```

TypeScript:

```
function edgeScore(edges: number[]): number {

};
```

C#:

```
public class Solution {
public int EdgeScore(int[] edges) {

}
}
```

C:

```
int edgeScore(int* edges, int edgesSize) {

}
```

Go:

```
func edgeScore(edges []int) int {

}
```

Kotlin:

```
class Solution {
fun edgeScore(edges: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func edgeScore(_ edges: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn edge_score(edges: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} edges
# @return {Integer}
def edge_score(edges)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $edges
* @return Integer
```

```
*/
function edgeScore($edges) {

}
}
```

Dart:

```
class Solution {
int edgeScore(List<int> edges) {

}
}
```

Scala:

```
object Solution {
def edgeScore(edges: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec edge_score(edges :: [integer]) :: integer
def edge_score(edges) do

end
end
```

Erlang:

```
-spec edge_score(Edges :: [integer()]) -> integer().
edge_score(Edges) ->
.
```

Racket:

```
(define/contract (edge-score edges)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Node With Highest Edge Score
* Difficulty: Medium
* Tags: array, graph, hash
*
* 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 edgeScore(vector<int>& edges) {

}
};
```

Java Solution:

```
/**
* Problem: Node With Highest Edge Score
* Difficulty: Medium
* Tags: array, graph, hash
*
* 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 edgeScore(int[] edges) {

}
}
```

Python3 Solution:

```
"""
Problem: Node With Highest Edge Score
Difficulty: Medium
Tags: array, graph, hash

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 edgeScore(self, edges: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def edgeScore(self, edges):
"""
:type edges: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Node With Highest Edge Score
* Difficulty: Medium
* Tags: array, graph, hash
*
* 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[]} edges
* @return {number}
*/
var edgeScore = function(edges) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Node With Highest Edge Score
* Difficulty: Medium
* Tags: array, graph, hash
*
* 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 edgeScore(edges: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Node With Highest Edge Score
* Difficulty: Medium
* Tags: array, graph, hash
*
* 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 EdgeScore(int[] edges) {

}
}
```

C Solution:

```
/*
* Problem: Node With Highest Edge Score
* Difficulty: Medium
```

```
* Tags: array, graph, hash
*
* 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 edgeScore(int* edges, int edgesSize) {

}
```

Go Solution:

```
// Problem: Node With Highest Edge Score
// Difficulty: Medium
// Tags: array, graph, hash
//
// 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 edgeScore(edges []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun edgeScore(edges: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func edgeScore(_ edges: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Node With Highest Edge Score
// Difficulty: Medium
// Tags: array, graph, hash
//
// 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 edge_score(edges: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} edges
# @return {Integer}
def edge_score(edges)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $edges
* @return Integer
*/
function edgeScore($edges) {

}
}
```

Dart Solution:

```
class Solution {
int edgeScore(List<int> edges) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def edgeScore(edges: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec edge_score(edges :: [integer]) :: integer
def edge_score(edges) do

end
end
```

Erlang Solution:

```
-spec edge_score(Edges :: [integer()]) -> integer().
edge_score(Edges) ->
.
```

Racket Solution:

```
(define/contract (edge-score edges)
(-> (listof exact-integer?) exact-integer?)
)
```