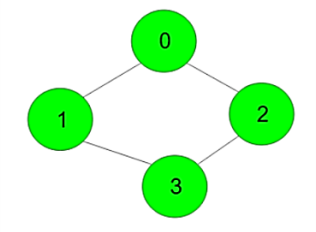
PROBLEM

Euler Circuit in an Undirected Graph

Medium Accuracy: 54.88% Submissions: 19K+ Points: 4

Eulerian Path is a path in a graph that visits **every edge exactly once**. Eulerian Circuit is an Eulerian Path that **starts** and **ends** on the **same vertex**. Given the number of vertices **v** and adjacency list **adj** denoting the graph. Find that there exists the Euler circuit or not. Return **1** if there exist **alteast one** eulerian path else **0.**

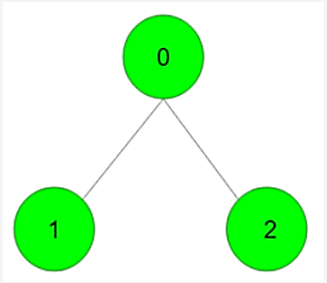
Input:
v = 4
edges[] = {{0, 1},
{0, 2},
{1, 3},
{2, 3}}



Output:
1
Explanation: corresponding adjacency list will be {{1, 2},{0, 3},{0, 3},{1, 2}}
One of the Eularian circuit
starting from vertex 0 is as follows:
0->1->3->2->0

Example 2:

Input:
v = 3
edges[] = {{0, 1},
{0, 2}}



Output:
0
Explanation: corresponding adjacency list will be {{1, 2}}
No Eulerian path is found.

Your Task:
You don't need to read or print anything. Your task is to complete the function **isEulerCircuitExist()** which takes **v** and adjacency list **adj[]** as input parameter and returns boolean value **1** if Eular circuit exists otherwise returns **0**.

Expected Time Complexity: O(v + e)
Expected Space Complexity: O(v)

Constraints:
$1 <= v <= 10^5$
$1 <= edges <= 2*10^5$

CODE

```
#User function Template for python3

class Solution:
    def isEularCircuitExist(self, v, adj):
        #Code here
        arr = [len(i) % 2 == 0 for i in adj]
        return all(arr)
```

EXPLANATION

Class Definition:

class Solution:

Here, we are defining a class named Solution. This class will contain the method to check if an Euler circuit exists.

Method Definition:

def isEularCircuitExist(self, v, adj):

This is a method named isEularCircuitExist. It takes three parameters:

self: This parameter refers to the instance of the class itself. It's required in Python methods.
v: This parameter represents the number of vertices in the graph.
adj: This parameter represents the adjacency list of the graph.

Code Explanation:

arr = [len(i) % 2 == 0 for i in adj]

Here, we're iterating over each list i in the adjacency list adj. We calculate the length of each list (len(i)) and check if it's even (len(i) % 2 == 0). For each list, this expression evaluates to True if the length is even and False if it's odd. We store these boolean values in the list arr.

return all(arr)

Finally, we use the all function to check if all elements in the list arr are True. If all lengths of adjacency lists are even, then arr will contain all True values, indicating that an Euler circuit exists in the graph. If any of the lengths are odd, arr will contain at least one False value, indicating that an Euler circuit doesn't exist.

Visualization:

Imagine you have a graph represented by an adjacency list. Each vertex of the graph is represented by a list, and the length of each list represents the number of edges incident to that vertex. If every vertex in the graph has an even number of edges incident to it, then you can draw a path that traverses each edge exactly once and returns to the starting vertex. This path is called an Euler circuit.

MORE EXPLANATION

let's consider a virtual graph represented by an adjacency list. We'll visualize a simple graph with vertices and edges. Here's an example graph:

```
  A
 / \
B - C
```

Let's represent this graph using an adjacency list:

A: [B, C]
B: [A, C]
C: [A, B]

Now, let's understand how the code works in this context:

arr = [len(i) % 2 == 0 for i in adj]

For each vertex in the adjacency list, we calculate the length of its corresponding list. For vertex A, the length is 2 (indicating two edges), for B and C, the length is 2 as well. Since all lengths are even, the list arr will be [True, True, True].

return all(arr)

The all function checks if all elements in the list arr are True. In this case, they are, so the function returns True. This means that an Euler circuit exists in this graph because every vertex has an even degree, allowing for a path that traverses each edge exactly once and returns to the starting vertex.