Problem

You are given an integer n. Determine if there is an unconnected graph with n vertices that contains at least two connected components and contains the number of edges that is equal to the number of vertices. Each vertex must follow one of these conditions:

• Its degree is less than or equal to 1.

• It's a cut-vertexLinks to an external site..

Input format:

• First line: n

Output format

• Print Yes if it is an unconnected graph. Otherwise, print No.

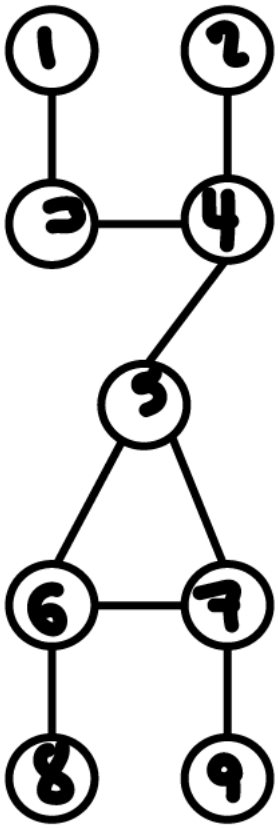
In this graph and code;

• The number vertices is equal to number of edges

• The vertices are either degree 1 or 0 or cut-vertex

• It can print if its an unconnected graph or connected

ONE OF THE POSSIBLE GRAPH ANSWERS

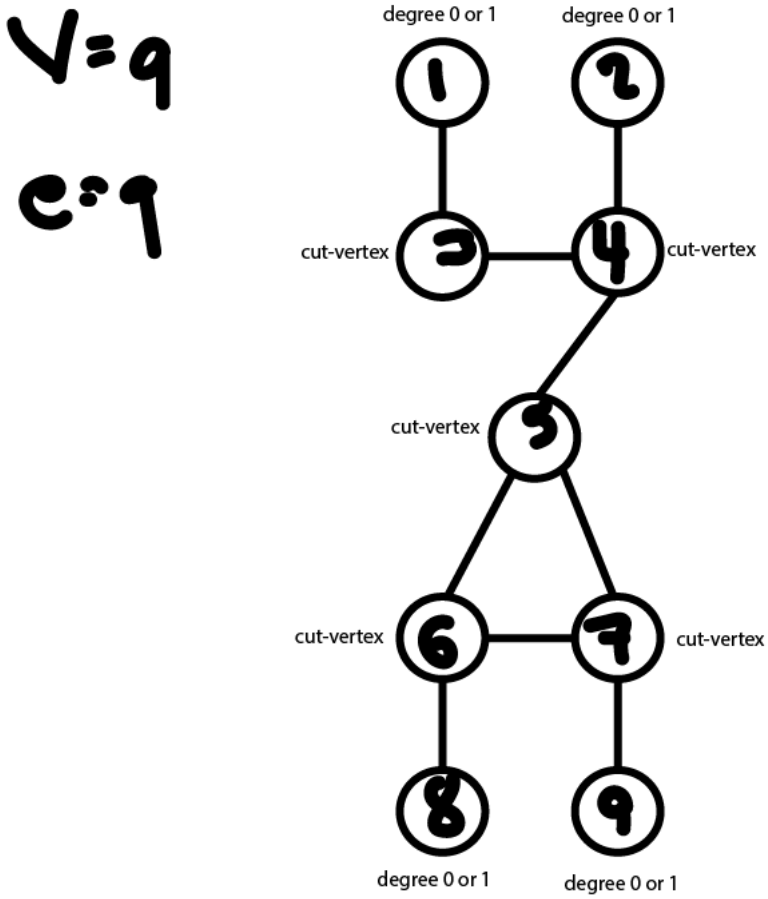


```
class Graph():
    '''initializing the dictionary'''
    def __init__(self, graph_dict=None):
        if graph_dict == None:
            graph_dict = {}
        self._graph_dict = graph_dict

    '''returning the edges of the graph'''
    def edges(self, vertice):
        return self._graph_dict[vertice]

    '''counting the vertices of the graph'''
    def count_vertex(self):
        return len(self._graph_dict.keys())

    '''counting the edges of the graph'''
    def count_edge(self):
        list_val = set()
        for node in self._graph_dict:
            for values in self._graph_dict[node]:
                list_val.add(tuple(sorted([node, values])))
        return len(list_val)

    '''printing all the vertices of the graph'''
    def all_vertices(self):
        for node in self._graph_dict:
            print("Vertex",node)

    '''a function that will allow us to enter a vertex'''
    def add_vertex(self, vertex):
        if vertex not in self._graph_dict:
            self._graph_dict[vertex] = []

    '''a function that will allow us to connect a vertices'''
    def add_edge(self):
        while True:
            edge = input("Enter the starting vertex of the edge: ")
            if edge == "S":
                self.print_graph()
                break
            else:
                edge2 = input("Enter which vertext to connect: ")
                if edge == edge2:
                    print("Loops are not allowed")
                    break
                else:
                    self._graph_dict[edge].append(edge2)
                    self._graph_dict[edge2].append(edge)

    '''a function that will print the graph along with the edges that is connected to the vertices'''
    def print_graph(self):
        print("--------------------")
        for connect in self._graph_dict:
            connection = self._graph_dict[connect]
            print("Vertex",connect,"is connected to",connection)

    '''finding a path to a certain vertices to identify if it is connected or unconnected'''
    def find_path(self, start_vertex, end_vertex, path=None):

        if path == None:
            path = []
        graph = self._graph_dict
        path = path + [start_vertex]
        if start_vertex == end_vertex:
            return path
        if start_vertex not in graph:
            return None
        for vertex in graph[start_vertex]:
            if vertex not in path:
                extended_path = self.find_path(vertex,
                                               end_vertex,
                                               path)
                if extended_path:
                    return extended_path
        return None

    '''checking if it is connected or unconnected'''
```

```
    def check_graph(self):
        list_path = []
        for vertex in self._graph_dict:
            for next_vertex in self._graph_dict:
                if vertex != next_vertex:
                    path = self.find_path(vertex, next_vertex)
                    if path == None:
                        list_path.append(path)
                    elif path != None:
                        list_path.extend(path)
        if None in list_path:
           print("Yes")
        elif None not in list_path:
            print("No")

    '''checking if the amount of vertex is equal to the amount of edges'''
    def check_quantity(self):
        vert = self.count_vertex()
        edge = self.count_edge()
        if edge == vert:
            print("Yes")
        else:
            print("No")

    '''checking if the vertex is a cut vertex or not'''
    def check_vertex(self):
        for node in self._graph_dict:
            values = self._graph_dict[node]
            if len(values) >= 2:
                print("Vertex", node, "is a cut vertex")
            else:
                print("Vertex", node, "degree 1 or 0")

dictionary = {}

n = int(input("Enter the number of vertex/vertices: "))
for node in range(1,n+1):
    dictionary[str(node)] = []
g = Graph(dictionary)
g.all_vertices()
print("--------------------")
print("Press S to stop connecting/adding edges")
g.add_edge()
print("--------------------")
print("Unconnected Graph")
g.check_graph()
```

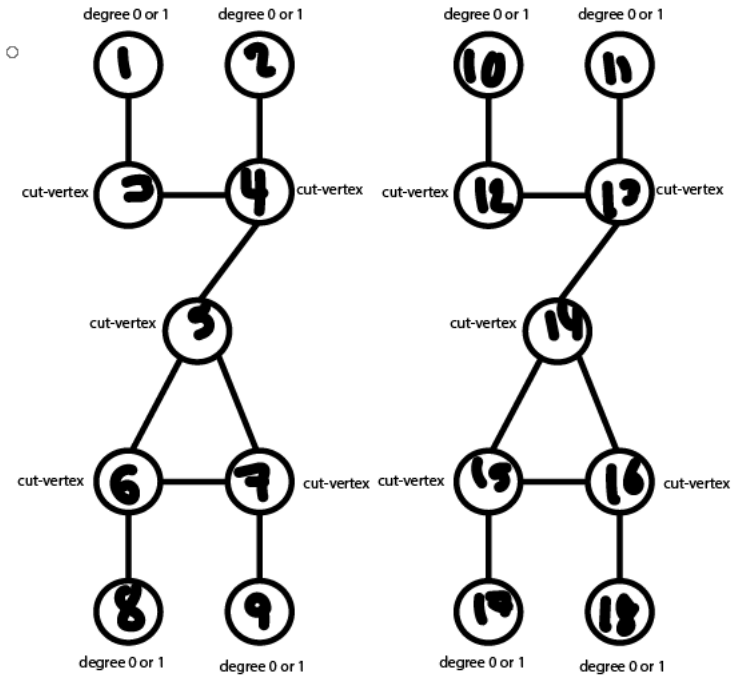
```
Enter the number of vertex/vertices: 9
Vertex 1
Vertex 2
Vertex 3
Vertex 4
Vertex 5
Vertex 6
Vertex 7
Vertex 8
Vertex 9
--------------------
Press S to stop connecting/adding edges
Enter the starting vertex of the edge: 1
Enter which vertext to connect: 3
Enter the starting vertex of the edge: 2
Enter which vertext to connect: 4
Enter the starting vertex of the edge: 3
Enter which vertext to connect: 4
Enter the starting vertex of the edge: 4
Enter which vertext to connect: 5
Enter the starting vertex of the edge: 5
Enter which vertext to connect: 6
Enter the starting vertex of the edge: 5
Enter which vertext to connect: 7
Enter the starting vertex of the edge: 6
Enter which vertext to connect: 7
Enter the starting vertex of the edge: 6
Enter which vertext to connect: 8
```

```
Enter the starting vertex of the edge: 7
Enter which vertext to connect: 9
Enter the starting vertex of the edge: S
-------------------
Vertex 1 is connected to ['3']
Vertex 2 is connected to ['4']
Vertex 3 is connected to ['1', '4']
Vertex 4 is connected to ['2', '3', '5']
Vertex 5 is connected to ['4', '6', '7']
Vertex 6 is connected to ['5', '7', '8']
Vertex 7 is connected to ['5', '6', '9']
Vertex 8 is connected to ['6']
Vertex 9 is connected to ['7']
-------------------
Unconnected Graph
No
```



• In the drawn graph above, all of the vertices is either degree 0 or 1, or cut-vertex, they also have the same no. of vertex and the no. of edges. However, it is not a unconnected graph since we only have one component.

• If we want to have a graph that can have an unconnected graph, we can delete one of the cut vertices in the graph above, or add another vertex/vertices that is not connected on the first component




```
class Graph():
    '''initializing the dictionary'''
    def __init__(self, graph_dict=None):
        if graph_dict == None:
            graph_dict = {}
        self._graph_dict = graph_dict

    '''returning the edges of the graph'''
    def edges(self, vertice):
        return self._graph_dict[vertice]

    '''counting the vertices of the graph'''
    def count_vertex(self):
        return len(self._graph_dict.keys())

    '''counting the edges of the graph'''
    def count_edge(self):
        list_val = set()
        for node in self._graph_dict:
            for values in self._graph_dict[node]:
                list_val.add(tuple(sorted([node, values])))
        return len(list_val)

    '''printing all the vertices of the graph'''
    def all_vertices(self):
        for node in self._graph_dict:
            print("Vertex",node)

    '''a function that will allow us to enter a vertex'''
    def add_vertex(self, vertex):
        if vertex not in self._graph_dict:
            self._graph_dict[vertex] = []

    '''a function that will allow us to connect a vertices'''
    def add_edge(self):
        while True:
            edge = input("Enter the starting vertex of the edge: ")
            if edge == "S":
                self.print_graph()
                break
            else:
                edge2 = input("Enter which vertext to connect: ")
                if edge == edge2:
                    print("Loops are not allowed")
                    break
```

```
                else:
                    self._graph_dict[edge].append(edge2)
                    self._graph_dict[edge2].append(edge)

    '''a function that will print the graph along with the edges that is connected to the vertices'''
    def print_graph(self):
        print("--------------------")
        for connect in self._graph_dict:
            connection = self._graph_dict[connect]
            print("Vertex",connect,"is connected to",connection)

    '''finding a path to a certain vertices to identify if it is connected or unconnected'''
    def find_path(self, start_vertex, end_vertex, path=None):

        if path == None:
            path = []
        graph = self._graph_dict
        path = path + [start_vertex]
        if start_vertex == end_vertex:
            return path
        if start_vertex not in graph:
            return None
        for vertex in graph[start_vertex]:
            if vertex not in path:
                extended_path = self.find_path(vertex,
                                               end_vertex,
                                               path)
                if extended_path:
                    return extended_path
        return None

    '''checking if it is connected or unconnected'''
    def check_graph(self):
        list_path = []
        for vertex in self._graph_dict:
            for next_vertex in self._graph_dict:
                if vertex != next_vertex:
                    path = self.find_path(vertex, next_vertex)
                    if path == None:
                        list_path.append(path)
                    elif path != None:
                        list_path.extend(path)
        if None in list_path:
           print("Yes")
        elif None not in list_path:
            print("No")

    '''checking if the amount of vertex is equal to the amount of edges'''
    def check_quantity(self):
        vert = self.count_vertex()
        edge = self.count_edge()
        if edge == vert:
            print("Yes")
        else:
            print("No")

    '''checking if the vertex is a cut vertex or not'''
    def check_vertex(self):
        for node in self._graph_dict:
            values = self._graph_dict[node]
            if len(values) >= 2:
                print("Vertex", node, "is a cut vertex")
            else:
                print("Vertex", node, "degree 1 or 0")

dictionary = {}

n = int(input("Enter the number of vertex/vertices: "))
for node in range(1,n+1):
    dictionary[str(node)] = []
g = Graph(dictionary)
g.all_vertices()
print("--------------------")
print("Press S to stop connecting/adding edges")
g.add_edge()
print("--------------------")
print("Unconnected Graph")
```

```
g.check_graph()
```

```
Enter which vertext to connect: 3
Enter the starting vertex of the edge: 2
Enter which vertext to connect: 4
Enter the starting vertex of the edge: 3
Enter which vertext to connect: 4
Enter the starting vertex of the edge: 4
Enter which vertext to connect: 5
Enter the starting vertex of the edge: 5
Enter which vertext to connect: 6
Enter the starting vertex of the edge: 5
Enter which vertext to connect: 7
Enter the starting vertex of the edge: 6
Enter which vertext to connect: 8
Enter the starting vertex of the edge: 6
Enter which vertext to connect: 7
Enter the starting vertex of the edge: 7
Enter which vertext to connect: 9
Enter the starting vertex of the edge: 10
Enter which vertext to connect: 12
Enter the starting vertex of the edge: 11
Enter which vertext to connect: 13
Enter the starting vertex of the edge: 12
Enter which vertext to connect: 13
Enter the starting vertex of the edge: 13
Enter which vertext to connect: 14
Enter the starting vertex of the edge: 14
Enter which vertext to connect: 15
Enter the starting vertex of the edge: 14
Enter which vertext to connect: 16
Enter the starting vertex of the edge: 15
Enter which vertext to connect: 17
Enter the starting vertex of the edge: 15
Enter which vertext to connect: 16
Enter the starting vertex of the edge: 16
Enter which vertext to connect: 18
Enter the starting vertex of the edge: S
--------------------
Vertex 1 is connected to ['3']
Vertex 2 is connected to ['4']
Vertex 3 is connected to ['1', '4']
Vertex 4 is connected to ['2', '3', '5']
Vertex 5 is connected to ['4', '6', '7']
Vertex 6 is connected to ['5', '8', '7']
Vertex 7 is connected to ['5', '6', '9']
Vertex 8 is connected to ['6']
Vertex 9 is connected to ['7']
Vertex 10 is connected to ['12']
Vertex 11 is connected to ['13']
Vertex 12 is connected to ['10', '13']
Vertex 13 is connected to ['11', '12', '14']
Vertex 14 is connected to ['13', '15', '16']
Vertex 15 is connected to ['14', '17', '16']
Vertex 16 is connected to ['14', '15', '18']
Vertex 17 is connected to ['15']
Vertex 18 is connected to ['16']
--------------------
Unconnected Graph
```