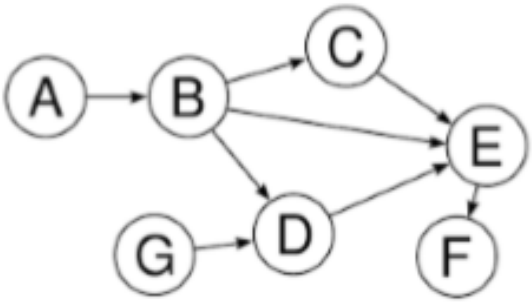
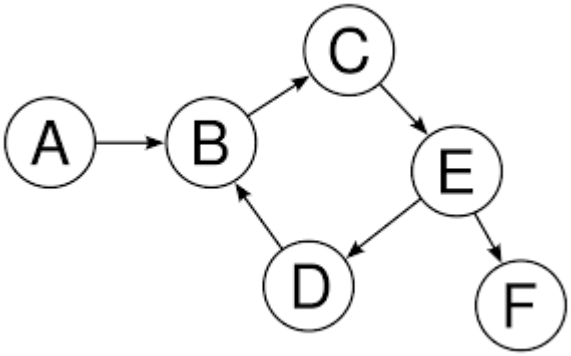
Is it a DAG? (Directed Acyclic Graph)

Directed graph and multiple edges are given as a single component. Determine whether the graph is a Directed Acyclic (DAG) or not. In this case, it is required to determine if the graph is a cyclic graph (Contains Cycles) which contains a path from at least one node back to itself, see **the figures below.**



Is a DAG Graph



Is NOT a DAG

Function to Implement

```
static bool IsDAG(string[] vertices, KeyValuePair<string, string> [] edges)
```

`IsItDAG.cs` includes this method.

Your function has vertices array (1 ≤ size ≤ 100,000) in the graph and edges as a list of `KeyValuePair<string, string>`, where **`key: sourceVertex, value: destVertex`**

Example

```
3               // Vertices Count
2               // Edges Count
A1,A2,A3        // Vertices
A1,A2           // Edges
A2,A3
true            // IsDAG
```

C# Help

Lists

Creation

To create a list of a certain type (e.g. string)

```
List<string> myList1 = new List<string>() //default initial size

List<string> myList2 = new List<string>(initSize) //given initial size
```

Manipulation

1. `myList1.Count` ➔ get actual number of items in the list
2. `myList1.Sort()` ➔ Sort the elements in the list (ascending)
3. `myList1[index]` ➔ Get/Set the elements at the specified index
4. `myList1.Add("myString1")` ➔ Add new element to the list
5. `myList1.Remove("myStr1")` ➔ Remove the 1st occurrence of this element from list
6. `myList1.RemoveAt(index)` ➔ Remove the element at the given index from the list
7. `myList1.Contains("myStr1")` ➔ Check if the element exists in the list

Dictionary (Hash)

Creation

To create a dictionary of a certain key (e.g. string) and value (e.g. array of strings)

```
//default initial size
Dictionary<string, string[]> myDict1 = new  Dictionary<string, string[]>();

//given initial size
Dictionary<string, string[]> myDict2 = new  Dictionary<string, string[]>(size);
```

Manipulation

1. `myDict1.Count` ➔ Get actual number of items in the dictionary
2. `myDict1[key]` ➔ Get/Set the value associated with the given key in the dictionary
3. `myDict1.Add(key, value)` ➔ Add the specified key and value to the dictionary
4. `myDict1.Remove(key)` ➔ Remove the value with the specified key from the dictionary
5. `myDict1.ContainsKey(key)` ➔ Check if the specified key exists in the dictionary

Creating 1D array

```
int [] array = new int [size]
```

Creating 2D array

```
int [,] array = new int [size1, size2]
```

Length of 1D array

```
int arrayLength = my1DArray.Length
```

Length of 2D array

```
int array1stDim = my2DArray.GetLength(0)

int array2ndDim = my2DArray.GetLength(1)
```

Sorting single array

Sort the given array in ascending order

```
Array.Sort(items);
```

Sorting parallel arrays

Sort the first array "master" and re-order the 2nd array "slave" according to this sorting

```
Array.Sort(master, slave);
```