Graph Analysis

Analyze the edges of the given DIRECTED graph by applying DFS starting from the given "startVertex" and count the occurrence of each type of edges (backward, forward & cross)

**NOTE:** during search, break ties (if any) by selecting the vertices in ASCENDING alpha-numeric order

# **Input:**

* |V| = from 5000 to 8000
* |E| = sparse or dense

# Function to Implement

int[] AnalyzeEdges(string[] vertices, KeyValuePair<string, string>[] edges, string startVertex)

GraphAnalysis.cs includes this method.

* "vertices": array of vertices in the graph
* "edges": array of edges in the graph (where **key: sourceVertex, value: destVertex**)
* "startVertex": name of the start vertex to begin from it

<returns> return array of 3 numbers:

1. outputs[0] number of backward edges,
2. outputs[1] number of forward edges,
3. outputs[2] number of cross edges

# Example

vertices1 = {"A1", "A2", "A3"};

edges1[0] = new KeyValuePair<string, string>("A1", "A2");

edges1[1] = new KeyValuePair<string, string>("A2", "A3");

startVertex = "A1"

output1 = {0, 0, 0}

vertices4 = { "A", "B", "C", "D", "E", "F", "G" };

edges4[0] = new KeyValuePair<string, string>("A", "D");

edges4[1] = new KeyValuePair<string, string>("A", "C");

edges4[2] = new KeyValuePair<string, string>("A", "B");

edges4[3] = new KeyValuePair<string, string>("B", "D");

edges4[4] = new KeyValuePair<string, string>("C", "E");

edges4[5] = new KeyValuePair<string, string>("D", "F");

edges4[6] = new KeyValuePair<string, string>("E", "F");

edges4[7] = new KeyValuePair<string, string>("E", "G");

edges4[8] = new KeyValuePair<string, string>("G", "D");

edges4[9] = new KeyValuePair<string, string>("F", "A");

output1 = { 1, 1, 2 };

# C# Help

## Queues

### Creation

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

Queue<string> myQ = new Queue<string>() //default initial size

Queue<string> myQ = new Queue<string>(**initSize**) //given initial size

### Manipulation

1. myQ.Count 🡺 get actual number of items in the queue
2. myQ.Enqueue(“myString1”)🡺 Add new element to the queue
3. myQ.Dequeue()🡺 return the top element of the queue (FIFO)

## 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);