

CPSC 5031 Homework 6

Visualizing Graphs

Name: David Nguyen

Description

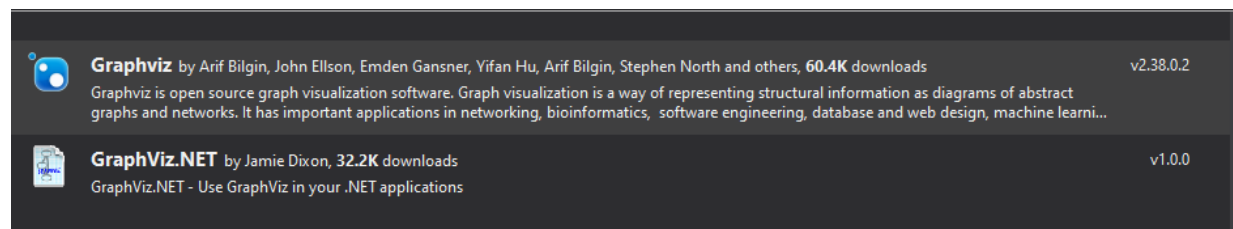
Write a program that can take an adjacency matrix file and generate a GraphViz "dot file". Run the dot files through `dot` (one of the GraphViz command line tools) and generate a PNG file.

Summary after completed the homework

It is a great way to build a graph using GraphViz and it is even more fun to build a tool in my own favorite language to implement this GraphViz API.

I started using MS-Test to add test cases for my program. One lesson I have learned, that the more test cases I added, the more updates I need to add into my function/methods. Clearly, adding test cases is a great way to see what are you missing in your function and discover the bugs at very early stage.

After finishing this homework, I also discovery some develop already built Nuget package that I can use for real application in the future.



Assumption to run this tool

1. Maximum of nodes for a graph is 24 (starting with letter A to Z)
2. Required user to provide text file and its location
3. Tool only run on Windows OS

Source code

https://github.com/davednguyen/cpsc5031_hw6.git

Branch

Develop

Main code project name: cpssc5031_hw6

Test codes project name: GraphVizTestProject

| | | |
|--|-----------------|---------------|
| Total test cases: 20 | | |
| Test framework: MS Test | | |
| Language: C3 | | |
| Test case name | Expected result | Actual result |
| TestCase_4by4Matrix_1_HappyPath_Graph | True | True |
| TestCase_4by4Matrix_2_HappyPath_Graph | True | True |
| TestCase_5by5Matrix_1_HappyPath_Graph | True | True |
| TestCase_6by6Matrix_1_HappyPath_Graph | True | True |
| TestCase_4by4Matrix_1_HappyPath_Digraph | True | True |
| TestCase_4by4Matrix_2_HappyPath_Digraph | True | True |
| TestCase_5by5Matrix_1_HappyPath_Digraph | True | True |
| TestCase_6by6Matrix_1_HappyPath_Digraph | True | True |
| TestCase_Check_EmptyTextFile_Graph | False | False |
| TestCase_Check_EmptyTextFile_digraph | False | False |
| TestCase_Check_NoTextFileFoundInTheFolder_Graph | False | False |
| TestCase_Check_NoTextFileFoundInTheFolder_digraph | False | False |
| TestCase_Check_TextFileHasSpecialCharactersMixWith_0_and_1_Graph | True | True |
| TestCase_Check_TextFileHasSpecialCharactersMixWith_0_and_1_digraph | True | True |
| TestCase_Check_TextFileHasSpecialCharactersOnly_Graph | True | True |
| TestCase_Check_TextFileHasSpecialCharactersOnly_Digraph | True | True |
| TestCase_Check_TextFileHas_1_Only_Graph | True | True |
| TestCase_Check_TextFileHas_1_Only_digraph | True | True |
| TestCase_Check_TextFileHas_0_Only_Graph | True | True |
| TestCase_Check_TextFileHas_0_Only_digraph | True | True |

| Test | Duration | Traits |
|--|----------|--------|
| ▲ ✓ GraphVizTestProject (20) | 1.8 sec | |
| ▲ ✓ GraphVizTestProject (20) | 1.8 sec | |
| ▲ ✓ Main (20) | 1.8 sec | |
| ✓ TestCase_4by4Matrix_1_HappyPath_Digraph | 74 ms | |
| ✓ TestCase_4by4Matrix_1_HappyPath_Graph | 60 ms | |
| ✓ TestCase_4by4Matrix_2_HappyPath_Digraph | 65 ms | |
| ✓ TestCase_4by4Matrix_2_HappyPath_Graph | 73 ms | |
| ✓ TestCase_5by5Matrix_1_HappyPath_Digraph | 83 ms | |
| ✓ TestCase_5by5Matrix_1_HappyPath_Graph | 174 ms | |
| ✓ TestCase_6by6Matrix_1_HappyPath_Digraph | 88 ms | |
| ✓ TestCase_6by6Matrix_1_HappyPath_Graph | 92 ms | |
| ✓ TestCase_Check_EmptyTextFile_digraph | 39 ms | |
| ✓ TestCase_Check_EmptyTextFile_Graph | 41 ms | |
| ✓ TestCase_Check_NoTextFileFoundInTheFolder_digraph | 165 ms | |
| ✓ TestCase_Check_NoTextFileFoundInTheFolder_Graph | 39 ms | |
| ✓ TestCase_Check_TextFileHas_0_Only_digraph | 66 ms | |
| ✓ TestCase_Check_TextFileHas_0_Only_Graph | 172 ms | |
| ✓ TestCase_Check_TextFileHas_1_Only_digraph | 73 ms | |
| ✓ TestCase_Check_TextFileHas_1_Only_Graph | 226 ms | |
| ✓ TestCase_Check_TextFileHasSpecialCharactersMixWith_0_and_1_digraph | 70 ms | |
| ✓ TestCase_Check_TextFileHasSpecialCharactersMixWith_0_and_1_Graph | 75 ms | |
| ✓ TestCase_Check_TextFileHasSpecialCharactersOnly_Digraph | 42 ms | |
| ✓ TestCase_Check_TextFileHasSpecialCharactersOnly_Graph | 43 ms | |

| CPSC5031_02 > week8 > homework6 > files | | | |
|---|--------------------|---------------------|-------|
| Name | Date modified | Type | Size |
| folder1 | 5/26/2021 9:03 PM | File folder | |
| folder2 | 5/26/2021 9:03 PM | File folder | |
| folder3 | 5/26/2021 9:04 PM | File folder | |
| folder4 | 5/26/2021 9:04 PM | File folder | |
| adj1.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj2.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj3.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj4.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj5.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj6.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj7.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj8.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj11.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj12.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj13.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj14.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj15.dot | 5/27/2021 4:17 PM | Microsoft Word 9... | 1 KB |
| adj1.png | 5/27/2021 4:17 PM | PNG File | 8 KB |
| adj2.png | 5/27/2021 4:17 PM | PNG File | 13 KB |
| adj3.png | 5/27/2021 4:17 PM | PNG File | 15 KB |
| adj4.png | 5/27/2021 4:17 PM | PNG File | 52 KB |
| adj5.png | 5/27/2021 4:17 PM | PNG File | 8 KB |
| adj6.png | 5/27/2021 4:17 PM | PNG File | 13 KB |
| adj7.png | 5/27/2021 4:17 PM | PNG File | 17 KB |
| adj8.png | 5/27/2021 4:17 PM | PNG File | 55 KB |
| adj11.png | 5/27/2021 4:17 PM | PNG File | 8 KB |
| adj12.png | 5/27/2021 4:17 PM | PNG File | 1 KB |
| adj13.png | 5/27/2021 4:17 PM | PNG File | 21 KB |
| adj14.png | 5/27/2021 4:17 PM | PNG File | 22 KB |
| adj15.png | 5/27/2021 4:17 PM | PNG File | 4 KB |
| adj1.txt | 5/26/2021 11:41 AM | Text Document | 1 KB |
| adj2.txt | 5/26/2021 7:54 PM | Text Document | 1 KB |
| adj3.txt | 5/26/2021 8:06 PM | Text Document | 1 KB |
| adj4.txt | 5/26/2021 8:09 PM | Text Document | 1 KB |
| adj5.txt | 5/27/2021 2:48 PM | Text Document | 0 KB |
| adj6.txt | 5/27/2021 3:02 PM | Text Document | 1 KB |
| adj7.txt | 5/27/2021 3:35 PM | Text Document | 1 KB |
| adj8.txt | 5/27/2021 3:47 PM | Text Document | 1 KB |
| adj9.txt | 5/27/2021 3:48 PM | Text Document | 1 KB |

Main tool codes (for better view, please refer to text file attachment) best application to view C# codes is NodePad++ - (file name: MainToolCodesInCSharp.txt)

```
using System;
using System.Collections.Generic;
```

```

using System.Diagnostics;
using System.IO;

/// <summary>
/// Homework 6
/// developer: David Nguyen
/// </summary>
namespace cpsc5031_hw6
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Homework 6");
            string directory = @"C:\Users\dzzn\Desktop\CPSC5031_02\week8\homework6\files\";
            //string directory =
@"C:\Users\mr4eyesn\Desktop\CPSC5031_2\week8\homework\code\cpsc5031_hw6\files\"
;

            GraphVizGenerator("adj1.txt", "adj1.png", "adj1.dot", directory, false);
            GraphVizGenerator("adj2.txt", "adj2.png", "adj2.dot", directory, false);
            GraphVizGenerator("adj3.txt", "adj3.png", "adj3.dot", directory, false);
            GraphVizGenerator("adj4.txt", "adj4.png", "adj4.dot", directory, false);

            GraphVizGenerator("adj1.txt", "adj5.png", "adj5.dot", directory, true);
            GraphVizGenerator("adj2.txt", "adj6.png", "adj6.dot", directory, true);
            GraphVizGenerator("adj3.txt", "adj7.png", "adj7.dot", directory, true);
            GraphVizGenerator("adj4.txt", "adj8.png", "adj8.dot", directory, true);
        }

        /// <summary>
        /// Generate a graph base on matrix of binary number (0 and 1)
        /// </summary>
        /// <param name="textFileName">matrix text file name provide by user</param>
        /// <param name="imageFileName">image file name provide by user</param>
        /// <param name="dotFileName">dot file name provide by user</param>
        /// <param name="directory">location where to get text file, to save dot file and to save
        image file</param>
        public static bool GraphVizGenerator(string textFileName, string imageFileName, string
        dotFileName, string directory, bool digraph)
        {
            //null check for all required inputs
            if(textFileName != null || imageFileName != null || dotFileName != null || directory !=
            null)
            {

```

```

        //check to make sure user don't provide empty string for any inputs
        if(!textFileName.Equals(string.Empty) || !imageFileName.Equals(string.Empty) ||
!dotFileName.Equals(string.Empty) || !directory.Equals(string.Empty))
        {
            var lines = readTextFile(directory + textFileName);
            var dotFileBody = generateDotFileBody(lines, digraph);
            var dotFilePath = directory + dotFileName;
            var dotFile = dotFileCompose(dotFileBody, dotFilePath);
            generateImage(dotFile, imageFileName, directory);
            if (File.Exists(directory + imageFileName))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}

/// <summary>
///
/// </summary>
/// <param name="textFileName"></param>
/// <param name="imageFileName"></param>
/// <param name="dotFileName"></param>
/// <param name="directory"></param>
/// <returns></returns>
public bool GraphVizGeneratorV2(string textFileName, string imageFileName, string
dotFileName, string directory, bool digraph)
{
    return GraphVizGenerator(textFileName, imageFileName, dotFileName, directory,
digraph);
}
/// <summary>

```

```

/// read text file
/// </summary>
/// <param name="path">file location</param>
/// <returns>lines of text files</returns>
private static string[] readTextFile(string path)
{
    //check if the text file provided by user is
    //existed in the folder
    if (File.Exists(path))
    {
        if (path != null)
        {
            string[] lines;
            lines = File.ReadAllLines(path);
            File.Exists(path);
            if (lines.Length > 0)
            {
                return lines;
            }
            else
            {
                return null;
            }
        }
        else
        {
            return null;
        }
    }
    else
    {
        return null;
    }
}

/// <summary>
/// List of pre-populated Node name for a graph
/// assuming the maximum nodes for a graph is 24
/// </summary>
/// <returns>list of node names</returns>
private static char[] Letters()
{
    char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
        'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };
}

```

```

    return letters;
}

/// <summary>
/// Take array of string and generate a file body for a dot file
/// list of connect between nodes within a graph
/// </summary>
/// <param name="lines">list of lines between two nodes</param>
/// <returns>string body for a dot file</returns>
private static string generateDotFileBody(string[] lines, bool digraph)
{
    if(lines != null && lines.Length > 0)
    {
        string graph = "graph matrix {";
        string dgraph = "digraph matrix {";
        string lastLine = "}";
        string gconnector = "--";
        string dgconnector = "->";
        string connector = "";
        //assign name for each node in the graph
        var nodes = Letters();
        string dotFileBody;
        if (digraph)
        {
            dotFileBody = dgraph + "\n";
            connector = dgconnector;
        }
        else
        {
            dotFileBody = graph + "\n";
            connector = gconnector;
        }

        //to keep track of all the nodes
        List<string> completedNodes = new List<string>();
        for (int i = 0; i < lines.Length; i++)
        {
            var list = lines[i].Trim().Replace(" ", string.Empty);
            for (int j = 0; j < list.Length; j++)
            {
                if (list[j].Equals('1'))
                {
                    string part1 = nodes[i] + connector + nodes[j];
                    string part2 = nodes[j] + connector + nodes[i];

```



```

        if (!completedNodes.Contains(part1) && !completedNodes.Contains(part2))
        {
            dotFileoBody = dotFileoBody + nodes[i] + connector + nodes[j] + "\n";
            completedNodes.Add(part1);
            completedNodes.Add(part2);
        }
    }
    else if (list[j].Equals('0'))
    {
        string part1 = nodes[i].ToString();
        string part2 = nodes[j].ToString();
        if (!completedNodes.Contains(part1) && !completedNodes.Contains(part2))
        {
            dotFileoBody = dotFileoBody + nodes[j] + "\n";
            completedNodes.Add(part1);
            completedNodes.Add(part2);
        }
    }
}
dotFileoBody = dotFileoBody + lastLine;
return dotFileoBody;
}
else
{
    return null;
}
}

/// <summary>
/// Build a dot file for graph
/// </summary>
/// <param name="stringbody">Dot file string body</param>
/// <param name="path">location and file name for the dot file</param>
private static string dotFileCompose(string stringbody, string path)
{
    //delete the file if it already exsited in the foler
    if (File.Exists(path))
    {
        File.Delete(path);
    }
    //write text into dot file
    if(stringbody != null && !stringbody.Equals(string.Empty))
    {

```

```

        using (StreamWriter writer = File.CreateText(path))
        {
            writer.Write(stringbody);
            writer.Flush();
            writer.Dispose();
            writer.Close();
        }
        File.Exists(path);
        return path;
    }
    else
    {
        return null;
    }
}

/// <summary>
/// Generate Graph based on dot file
/// </summary>
/// <param name="dotFile">dot file name</param>
/// <param name="imageFile">image file name</param>
/// <param name="directory"></param>
private static void generateImage(string dotFile, string imageFile, string directory)
{
    //delete the image file if it already exsited in the foler
    string exisitingImageFile = directory + imageFile;
    if (File.Exists(exisitingImageFile))
    {
        File.Delete(exisitingImageFile);
    }
    //command to generage image file
    string commandTemplate = "dot -Tpng {0} -o {1}";
    //where to run the command
    string application = "cmd.exe";
    //complete command
    string command = String.Format(commandTemplate, dotFile, imageFile);
    using(Process process = new Process())
    {
        process.StartInfo = new ProcessStartInfo(application)
        {
            RedirectStandardInput = true,
            UseShellExecute = false,
            WorkingDirectory = directory
        };
    }
};

```

```

        process.Start();
        process.StandardInput.WriteLine(command);
        process.StandardInput.Close();
        process.WaitForExit();
        process.CloseMainWindow();
        process.Close();
    }
}
}
}

```

Test cases in codes (for better view, please refer to text file attachment) best application to view C# codes is NodePad++ - (file name: MainToolCodesInCSharp.txt)

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using cpsc5031_hw6;

namespace GraphVizTestProject
{
    [TestClass]
    public class Main
    {
        //set initial directory for testing
        //string directory =
        @"C:\Users\mr4eyesn\Desktop\CPSC5031_2\week8\homework\code\cpsc5031_hw6\files\";
        string directory = @"C:\Users\dzzn\Desktop\CPSC5031_02\week8\homework6\files\";
        [TestMethod]
        public void TestCase_4by4Matrix_1_HappyPath_Graph()
        {
            Program graph = new Program();
            var check = graph.GraphVizGeneratorV2("adj1.txt", "adj1.png", "adj1.dot",
directory, false);
            Assert.AreEqual(true, check);
        }

        [TestMethod]
        public void TestCase_4by4Matrix_2_HappyPath_Graph()
        {
            Program graph = new Program();
            var check = graph.GraphVizGeneratorV2("adj2.txt", "adj2.png", "adj2.dot",
directory, false);
            Assert.AreEqual(true, check);
        }

        [TestMethod]
        public void TestCase_5by5Matrix_1_HappyPath_Graph()
        {
            Program graph = new Program();
            var check = graph.GraphVizGeneratorV2("adj3.txt", "adj3.png", "adj3.dot",
directory, false);
            Assert.AreEqual(true, check);
        }

        [TestMethod]

```

```

    public void TestCase_6by6Matrix_1_HappyPath_Graph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj4.txt", "adj4.png", "adj4.dot",
directory, false);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_4by4Matrix_1_HappyPath_Digraph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj1.txt", "adj5.png", "adj5.dot",
directory, true);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_4by4Matrix_2_HappyPath_Digraph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj2.txt", "adj6.png", "adj6.dot",
directory, true);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_5by5Matrix_1_HappyPath_Digraph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj3.txt", "adj7.png", "adj7.dot",
directory, true);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_6by6Matrix_1_HappyPath_Digraph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj4.txt", "adj8.png", "adj8.dot",
directory, true);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_Check_EmptyTextFile_Graph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj5.txt", "adj9.png", "adj9.dot",
directory, false);
        Assert.AreEqual(false, check);
    }

    [TestMethod]
    public void TestCase_Check_EmptyTextFile_digraph()
    {
        Program graph = new Program();

```

```

        var check = graph.GraphVizGeneratorV2("adj5.txt", "adj10.png", "adj10.dot",
directory, true);
        Assert.AreEqual(false, check);
    }

    [TestMethod]
    public void TestCase_Check_NoTextFileFoundInTheFolder_Graph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj20.txt", "adj9.png", "adj9.dot",
directory, false);
        Assert.AreEqual(false, check);
    }

    [TestMethod]
    public void TestCase_Check_NoTextFileFoundInTheFolder_dgraph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj20.txt", "adj10.png",
"adj10.dot", directory, true);
        Assert.AreEqual(false, check);
    }

    [TestMethod]
    public void TestCase_Check_TextFileHasSpecialCharactersMixWith_0_and_1_Graph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj6.txt", "adj11.png", "adj11.dot",
directory, false);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_Check_TextFileHasSpecialCharactersMixWith_0_and_1_dgraph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj6.txt", "adj12.png", "adj12.dot",
directory, true);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_Check_TextFileHasSpecialCharactersOnly_Graph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj7.txt", "adj12.png", "adj12.dot",
directory, false);
        Assert.AreEqual(true, check);
    }

    [TestMethod]
    public void TestCase_Check_TextFileHasSpecialCharactersOnly_Dgraph()
    {
        Program graph = new Program();
        var check = graph.GraphVizGeneratorV2("adj7.txt", "adj13.png", "adj13.dot",
directory, true);
        Assert.AreEqual(true, check);
    }
}

```

```

[TestMethod]
public void TestCase_Check_TextFileHas_1_Only_Graph()
{
    Program graph = new Program();
    var check = graph.GraphVizGeneratorV2("adj8.txt", "adj13.png", "adj13.dot",
directory, false);
    Assert.AreEqual(true, check);
}

[TestMethod]
public void TestCase_Check_TextFileHas_1_Only_dgraph()
{
    Program graph = new Program();
    var check = graph.GraphVizGeneratorV2("adj8.txt", "adj14.png", "adj14.dot",
directory, true);
    Assert.AreEqual(true, check);
}

[TestMethod]
public void TestCase_Check_TextFileHas_0_Only_Graph()
{
    Program graph = new Program();
    var check = graph.GraphVizGeneratorV2("adj9.txt", "adj14.png", "adj14.dot",
directory, false);
    Assert.AreEqual(true, check);
}

[TestMethod]
public void TestCase_Check_TextFileHas_0_Only_dgraph()
{
    Program graph = new Program();
    var check = graph.GraphVizGeneratorV2("adj9.txt", "adj15.png", "adj15.dot",
directory, true);
    Assert.AreEqual(true, check);
}
}
}

```