# Lab: Trees Representation and Traversal (BFS, DFS)

You can check your solutions here: <https://judge.softuni.bg/Contests/3189/Additional-Exercises>.

## Build a Tree

In this task, you need to **build** trees, using the TreeNode<int> class. When ready, **print** the tree.

Note: In order to use the class, install the SimpleTreeNode NuGet package.

Graphical user interface, application

Description automatically generated

Look at the TreeNode<int> class implementation and examples at <https://github.com/nakov/SimpleTreeNode> for more understanding of the data structure.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Application  Description automatically generated with medium confidence | A  B  D  H  I  E  C  F  J  G |
| A picture containing scatter chart  Description automatically generated | 25  20  10  5  1  8  12  15  22  36  30  28  40  38  48  45  50 |

Build the trees and **print** them **separately** on the console. **Submit** each of them to the corresponding task in Judge.

### Hints

For the first example, it is a good idea to create the **root node** (A), then all its **child nodes** (B,C), then their child nodes, etc. This way you can have more clearance of the **tree structure** and the place of each node in it. You can start like this:

Text

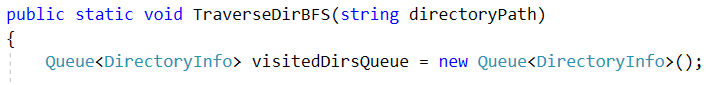
Description automatically generated

## BFS Traverse Folders

As you already know, **Breadth-first search (BFS)** is an algorithm for traversing or searching **tree** data structures. It starts at the **tree root** and explores all of the **neighbor nodes** at the present depth prior to moving on to the nodes at the next depth level.

We want to implement the algorithm to **traverse** our **file system**, as it has a **tree structure**.

First, write the TraverseDirBFS() method, which should accept **directory path**. Add a Queue<DirectoryInfo>, which will hold **visited** folders:



Note that the DirectoryInfo class allows us to **access directories** and **subdirectories** of our **file system**. Then, get the first DirectoryInfo element from the **queue** like this:



Next, create a loop to **traverse** through folders until the queue is **empty**.

A picture containing graphical user interface

Description automatically generated

In the loop, get the **current folder**, remove it from the queue and print its name.

Text

Description automatically generated

Then, use the GetDirectories() method to get **sub-folders** of the current folder. Add each of them to the queue:

A screenshot of a computer

Description automatically generated with low confidence

Finally, as our TraverseDirBFS() method is ready, invoke it from Main(). Manually, point out the directory **path** to the method. In our example, we will traverse "C:\Windows\assembly":

Logo

Description automatically generated with medium confidence

This is an example of part of the file system. This is the **order**, in which folders are traversed, using the **BFS algorithm** we wrote.

Diagram

Description automatically generated

In this case, with the file structure from the picture, the result should be like this:

Text

Description automatically generated

**Test** the program with your **file system**, as the result might differ. It is a good idea to **debug** the code to see how the **BFS algorithms** traverses the folders.

**Submit** your solution to the Judge system. Note that you should first **comment** the TraverseDirBFS() invocation in the Main():

Logo

Description automatically generated

Add the .cs and .csproj files to a .zip archive and submit it to Judge.

## DFS Traverse Folders

**Depth-first search (DFS)** is an algorithm for traversing or searching tree data structures. The algorithm starts at the **root node** and explores **as far as possible** along each branch before **backtracking**.

Let’s use it to traverse our folders, as we did using the BFS algorithm.

First, create TraverseDirDFS() method, which accepts **directory path**. Then, create an overload of the TraverseDirDFS() method to accept DirectoryInfo and string (we will use it for keeping spaces, which will help us create and print the output result with a tree-like structure):

Graphical user interface, text, application, email

Description automatically generated

In the TraverseDirDFS(string directoryPath) method, invoke the other TraverseDirDFS method like this:

A picture containing text

Description automatically generated

Now, let’s write the other method, as it is the one, which actually holds the **DFS algorithm implementation**. First, get the **sub-directories** of the current directory and use a **loop** to print each of the sub-folders. Note that you need to use **recursion** to go back to previous folders when you reach a leaf of the file system tree.

A screenshot of a computer

Description automatically generated with medium confidence

Also, it is important that we **print** the directory name at each invoke of the method before it is recursively invoked again:

A picture containing text

Description automatically generated

Finally, we should invoke the TraverseDirDFS(string directoryPath) method from the Main() with the directory path we want to traverse in. Let’s use "C:\Windows\assembly" again:

Logo

Description automatically generated with medium confidence

In this case, our **file system tree** should be traversed like this:

Diagram

Description automatically generated

With the above structure, the result looks like this:

Text

Description automatically generated

**Test** the program with your **file system**, as the result might differ. It is a good idea to **debug** the code to see how the **DFS algorithms** traverses the folders. Note that **spaces** helped us create a tree-like structure of output.

**Submit** your solution to Judge. Do not forget to comment the TraverseDirDFS() method in the Main().

## DFS Traverse Folders and Files

Now we will use the code from the **previous task** and expand it to **print files** in our folders, as well. The only thing you should do is to use the FileInfo class, as well as the GetFiles() method to get **files**. Do it like this:

Graphical user interface, text, application, email

Description automatically generated

The result should be something like this:

Text

Description automatically generated

**Test** your program. You should see your **files** printed on the console.

**Submit** your solution to Judge. Do not forget to comment the TraverseDirDFS() method in the Main().