Άσκηση 1: Απλή Αναζήτηση DFS σε Δέντρο

Περιγραφή

Δημιουργήστε μια κλάση για έναν κόμβο ενός δέντρου και υλοποιήστε την αναζήτηση DFS για να επισκεφτείτε όλους τους κόμβους του δέντρου.

Δεδομένα

- Δημιουργήστε ένα δέντρο με 7 κόμβους με την εξής διάταξη:

```
    1
   / ¥
  2   3
 / ¥ / ¥
4  5 6  7
```

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

class Program
{
    static void Main()
    {
        // Δημιουργία δέντρου
        var root = new Node(1);
        root.Children.Add(new Node(2));
        root.Children.Add(new Node(3));
        root.Children[0].Children.Add(new Node(4));
        root.Children[0].Children.Add(new Node(5));
        root.Children[1].Children.Add(new Node(6));
        root.Children[1].Children.Add(new Node(7));

        // Εκκίνηση της αναζήτησης
        Console.WriteLine("DFS Traversal:");
        DepthFirstSearch(root);
    }

    // Ορισμός κόμβου δέντρου
    class Node
    {
        public int Value;
```

```
        public List<Node> Children;

        public Node(int value)
        {
            Value = value;
            Children = new List<Node>();
        }
    }

    // Μέθοδος Depth-First Search
    static void DepthFirstSearch(Node node)
    {
        if (node == null) return;

        // Επισκέπτομαι τον τρέχοντα κόμβο
        Console.WriteLine(node.Value);

        // Ανακαλύπτω τα παιδιά του κόμβου
        foreach (var child in node.Children)
        {
            DepthFirstSearch(child);
        }
    }
}
```

Εξήγηση

Αυτή η άσκηση δημιουργεί ένα δέντρο και εφαρμόζει την DFS αναζήτηση για να επισκεφτεί κάθε κόμβο του δέντρου ξεκινώντας από τη ρίζα.

Άσκηση 2: DFS σε Γράφο

Περιγραφή

Δημιουργήστε έναν απλό γραφήμα με κόμβους και ακμές. Υλοποιήστε την αναζήτηση DFS για να επισκεφτείτε όλους τους κόμβους του γράφου.

Δεδομένα

- Δημιουργήστε ένα γράφο με 5 κόμβους με τις εξής ακμές:
    - 1 - 2
    - 1 - 3
    - 2 - 4

- 2-5
- 3-4

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

class Program
{
    static void Main()
    {
        // Δημιουργία γράφου
        var graph = new Graph();
        graph.AddEdge(1, 2);
        graph.AddEdge(1, 3);
        graph.AddEdge(2, 4);
        graph.AddEdge(2, 5);
        graph.AddEdge(3, 4);

        // Εκκίνηση της αναζήτησης
        Console.WriteLine("DFS Traversal:");
        DepthFirstSearch(graph, 1);
    }

    // Ορισμός γράφου
    class Graph
    {
        private Dictionary<int, List<int>> adjList = new Dictionary<int,
List<int>>();

        public void AddEdge(int u, int v)
        {
            if (!adjList.ContainsKey(u)) adjList[u] = new List<int>();
            if (!adjList.ContainsKey(v)) adjList[v] = new List<int>();
            adjList[u].Add(v);
            adjList[v].Add(u); // Για μη κατευθυνόμενο
γράφο
        }
```

```
        public List<int> GetNeighbors(int node)
        {
            return adjList.ContainsKey(node) ? adjList[node] : new List<int>();
        }
    }

    // Μέθοδος Depth-First Search
    static void DepthFirstSearch(Graph graph, int start)
    {
        var visited = new HashSet<int>();
        DFS(graph, start, visited);
    }

    static void DFS(Graph graph, int node, HashSet<int> visited)
    {
        if (visited.Contains(node)) return;

        // Επισκέπτομαι τον τρέχοντα κόμβο
        Console.WriteLine(node);
        visited.Add(node);

        // Ανακαλύπτω τους γείτονες του κόμβου
        foreach (var neighbor in graph.GetNeighbors(node))
        {
            DFS(graph, neighbor, visited);
        }
    }
}
```

Εξήγηση

Αυτή η άσκηση δημιουργεί ένα γράφο και εφαρμόζει την DFS αναζήτηση για να επισκεφτεί κάθε κόμβο ξεκινώντας από έναν δεδομένο κόμβο. Χρησιμοποιούμε μια HashSet για να παρακολουθούμε ποιους κόμβους έχουμε ήδη επισκεφτεί.

Άσκηση 3: DFS για Έλεγχο Σύνδεσης Γράφου

Περιγραφή

Δημιουργήστε έναν γράφο και χρησιμοποιήστε τη DFS για να ελέγξετε αν ο γράφος είναι συνδεδεμένος (δηλαδή, αν υπάρχει διαδρομή μεταξύ κάθε ζεύγους κόμβων).

Δεδομένα

- Δημιουργήστε έναν γράφο με 4 κόμβους και τις εξής ακμές:

- 1 - 2
- 2 - 3
- 3 - 4

Κώδικας

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

class Program
{
    static void Main()
    {
        // Δημιουργία γράφου
        var graph = new Graph();
        graph.AddEdge(1, 2);
        graph.AddEdge(2, 3);
        graph.AddEdge(3, 4);

        // Έλεγχος αν ο γράφος είναι συνδεδεμένος
        Console.WriteLine("Is graph connected?");
        bool isConnected = IsConnected(graph, 1);
        Console.WriteLine(isConnected ? "Yes" : "No");
    }

    // Ορισμός γράφου
    class Graph
    {
        private Dictionary<int, List<int>> adjList = new Dictionary<int,
List<int>>();

        public void AddEdge(int u, int v)
        {
            if (!adjList.ContainsKey(u)) adjList[u] = new List<int>();
            if (!adjList.ContainsKey(v)) adjList[v] = new List<int>();
            adjList[u].Add(v);
            adjList[v].Add(u); // Για μη κατευθυνόμενο
γράφο
        }
```

```
        public List<int> GetNeighbors(int node)
        {
            return adjList.ContainsKey(node) ? adjList[node] : new List<int>();
        }

        public IEnumerable<int> GetAllNodes()
        {
            return adjList.Keys;
        }
    }

    // Έλεγχος αν ο γράφος είναι συνδεδεμένος
    static bool IsConnected(Graph graph, int start)
    {
        var visited = new HashSet<int>();
        DFS(graph, start, visited);

        // Ο γράφος είναι συνδεδεμένος αν επισκεφτεί όλους τους κόμβους
        return visited.Count == graph.GetAllNodes().Count;
    }

    static void DFS(Graph graph, int node, HashSet<int> visited)
    {
        if (visited.Contains(node)) return;

        visited.Add(node);

        foreach (var neighbor in graph.GetNeighbors(node))
        {
            DFS(graph, neighbor, visited);
        }
    }
}
```

Εξήγηση

Αυτή η άσκηση ελέγχει αν ένας γράφος είναι συνδεδεμένος. Ξεκινάμε από έναν κόμβο και χρησιμοποιούμε DFS για να επισκεφτούμε όλους τους συνδεδεμένους κόμβους. Αν καταφέρουμε να επισκεφτούμε όλους τους κόμβους, τότε ο γράφος είναι συνδεδεμένος.

Ελπίζω αυτές οι ασκήσεις να σε βοηθήσουν να κατανοήσεις καλύτερα την αναζήτηση βάθους (DFS) στην C#. Αν χρειάζεσαι περισσότερες πληροφορίες ή άλλες ασκήσεις, μη διστάσεις να ρωτήσεις!