Άσκηση 1: Αλγόριθμος του Kruskal

Περιγραφή: Δίνεται ένας γράφος με κόμβους και ακμές, όπου κάθε ακμή έχει ένα κόστος. Χρησιμοποίησε τον αλγόριθμο του Kruskal για να βρεις το Minimum Spanning Tree (MST) του γράφου.

Προϋποθέσεις:

- Εισάγονται οι ακμές του γράφου.
- Κάθε ακμή έχει την μορφή (κόμβος_1, κόμβος_2, κόστος).

Κώδικας:

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

public class KruskalAlgorithm
{
    public class Edge
    {
        public int Start { get; }
        public int End { get; }
        public int Weight { get; }

        public Edge(int start, int end, int weight)
        {
            Start = start;
            End = end;
            Weight = weight;
        }
    }

    public class DisjointSet
    {
        private readonly Dictionary<int, int> parent = new();
        private readonly Dictionary<int, int> rank = new();

        public void MakeSet(int x)
        {
            parent[x] = x;
            rank[x] = 0;
```

```
    }

    public int Find(int x)
    {
        if (parent[x] != x)
            parent[x] = Find(parent[x]);
        return parent[x];
    }

    public void Union(int x, int y)
    {
        int rootX = Find(x);
        int rootY = Find(y);

        if (rootX != rootY)
        {
            if (rank[rootX] > rank[rootY])
                parent[rootY] = rootX;
            else if (rank[rootX] < rank[rootY])
                parent[rootX] = rootY;
            else
            {
                parent[rootY] = rootX;
                rank[rootX]++;
            }
        }
    }
}

public static List<Edge> Kruskal(int numberOfNodes, List<Edge> edges)
{
    var result = new List<Edge>();
    var disjointSet = new DisjointSet();

    // Initialize disjoint sets
    for (int i = 1; i <= numberOfNodes; i++)
    {
        disjointSet.MakeSet(i);
    }
```

```
        // Sort edges by weight
        edges = edges.OrderBy(edge => edge.Weight).ToList();

        foreach (var edge in edges)
        {
            int rootStart = disjointSet.Find(edge.Start);
            int rootEnd = disjointSet.Find(edge.End);

            // If the edge does not form a cycle
            if (rootStart != rootEnd)
            {
                result.Add(edge);
                disjointSet.Union(rootStart, rootEnd);
            }
        }

        return result;
    }

    public static void Main()
    {
        var edges = new List<Edge>
        {
            new Edge(1, 2, 4),
            new Edge(1, 3, 1),
            new Edge(2, 3, 2),
            new Edge(2, 4, 5),
            new Edge(3, 4, 8)
        };

        var result = Kruskal(4, edges);

        Console.WriteLine("Edges in the Minimum Spanning Tree:");
        foreach (var edge in result)
        {
            Console.WriteLine($"{edge.Start} - {edge.End}: {edge.Weight}");
        }
    }
}
```

Εξήγηση:

1. **Δημιουργία Κλάσης Edge:** Ορίζει τις ακμές του γράφου με αρχικό κόμβο, τελικό κόμβο και κόστος.
2. **DisjointSet:** Βοηθάει στην ένωση και εύρεση των ομάδων για αποφυγή κύκλων.
3. **Κύριος Αλγόριθμος Kruskal:** Ταξινομεί τις ακμές με βάση το κόστος και επιλέγει τις ακμές για το MST χωρίς να δημιουργεί κύκλους.

Άσκηση 2: Αλγόριθμος του Prim

Περιγραφή: Δίνεται ένας γράφος με κόμβους και ακμές. Χρησιμοποίησε τον αλγόριθμο του Prim για να βρεις το Minimum Spanning Tree (MST) του γράφου ξεκινώντας από έναν καθορισμένο κόμβο.

Προϋποθέσεις:

- Εισάγονται οι ακμές του γράφου.
- Κάθε ακμή έχει την μορφή (κόμβος_1, κόμβος_2, κόστος).
- Δίνεται ο αρχικός κόμβος.

Κώδικας:

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

public class PrimAlgorithm
{
    public class Edge
    {
        public int Start { get; }
        public int End { get; }
        public int Weight { get; }

        public Edge(int start, int end, int weight)
        {
            Start = start;
            End = end;
            Weight = weight;
```

```
        }
    }

    public static List<Edge> Prim(int numberOfNodes, List<Edge> edges, int
startNode)
    {
        var result = new List<Edge>();
        var minHeap = new SortedSet<Edge>(Comparer<Edge>.Create((a, b) =>
a.Weight.CompareTo(b.Weight)));
        var visited = new HashSet<int>();

        void AddEdges(int node)
        {
            visited.Add(node);
            foreach (var edge in edges.Where(e => e.Start == node || e.End ==
node))
            {
                if (!visited.Contains(edge.Start)
|| !visited.Contains(edge.End))
                {
                    minHeap.Add(edge);
                }
            }
        }

        AddEdges(startNode);

        while (minHeap.Count > 0)
        {
            var edge = minHeap.Min;
            minHeap.Remove(edge);

            if (visited.Contains(edge.Start) && visited.Contains(edge.End))
            {
                continue;
            }

            result.Add(edge);
            AddEdges(edge.Start);
            AddEdges(edge.End);
```

```
        }

        return result;
    }

    public static void Main()
    {
        var edges = new List<Edge>
        {
            new Edge(1, 2, 4),
            new Edge(1, 3, 1),
            new Edge(2, 3, 2),
            new Edge(2, 4, 5),
            new Edge(3, 4, 8)
        };

        var result = Prim(4, edges, 1);

        Console.WriteLine("Edges in the Minimum Spanning Tree:");
        foreach (var edge in result)
        {
            Console.WriteLine($"{edge.Start} - {edge.End}: {edge.Weight}");
        }
    }
}
```

Εξήγηση:

1. **Δημιουργία Κλάσης Edge:** Ορίζει τις ακμές του γράφου.
2. **Prim's Algorithm:** Ξεκινά από έναν κόμβο, προσθέτει τις ακμές του γράφου που συνδέουν τον κόμβο αυτόν με μη επισκέφθηκε κόμβους, και επαναλαμβάνει μέχρι να συνδέσει όλους τους κόμβους.

Περίληψη

Οι αλγόριθμοι Kruskal και Prim είναι εργαλεία που χρησιμοποιούμε για να βρούμε το Minimum Spanning Tree ενός γράφου. Ο Kruskal επιλέγει τις πιο φθηνές ακμές πρώτα, ενώ ο Prim επεκτείνεται από έναν αρχικό κόμβο, προσθέτοντας τις πιο φθηνές ακμές που συνδέουν μη επισκέφθηκε κόμβους.

Αυτές οι ασκήσεις θα σε βοηθήσουν να κατανοήσεις πώς λειτουργούν αυτοί οι αλγόριθμοι και πώς να τους εφαρμόσεις στην C#. Αν έχεις οποιαδήποτε απορία ή χρειάζεσαι βοήθεια, μην διστάσεις να ρωτήσεις!