Projekt 3

(DFS lub BFS w zależności od wyboru użytkownika - dwie metody muszą być zaimplementowane!). Program powinien zwracać kolejno odwiedzane wierzchołki oraz średni czas działania.

Kod źródłowy:

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

public class Graph
{
   private int V; // liczba wierzchołków
   private List<int>[] adjacencyList; // lista sąsiedztwa

   public Graph(int v)
   {
      V = v;
      adjacencyList = new List<int>[V];
      for (int i = 0; i < V; i++)
      {
         adjacencyList[i] = new List<int>();
      }
   }

   public void AddEdge(int v, int w)
   {
      adjacencyList[v].Add(w);
   }

   // Algorytm DFS (przeszukiwanie w głąb)
   public void DFS(int startVertex)
   {
      bool[] visited = new bool[V];
      List<int> visitedVertices = new List<int>();

      Stopwatch stopwatch = new Stopwatch();
      stopwatch.Start(); // Start pomiaru czasu
```

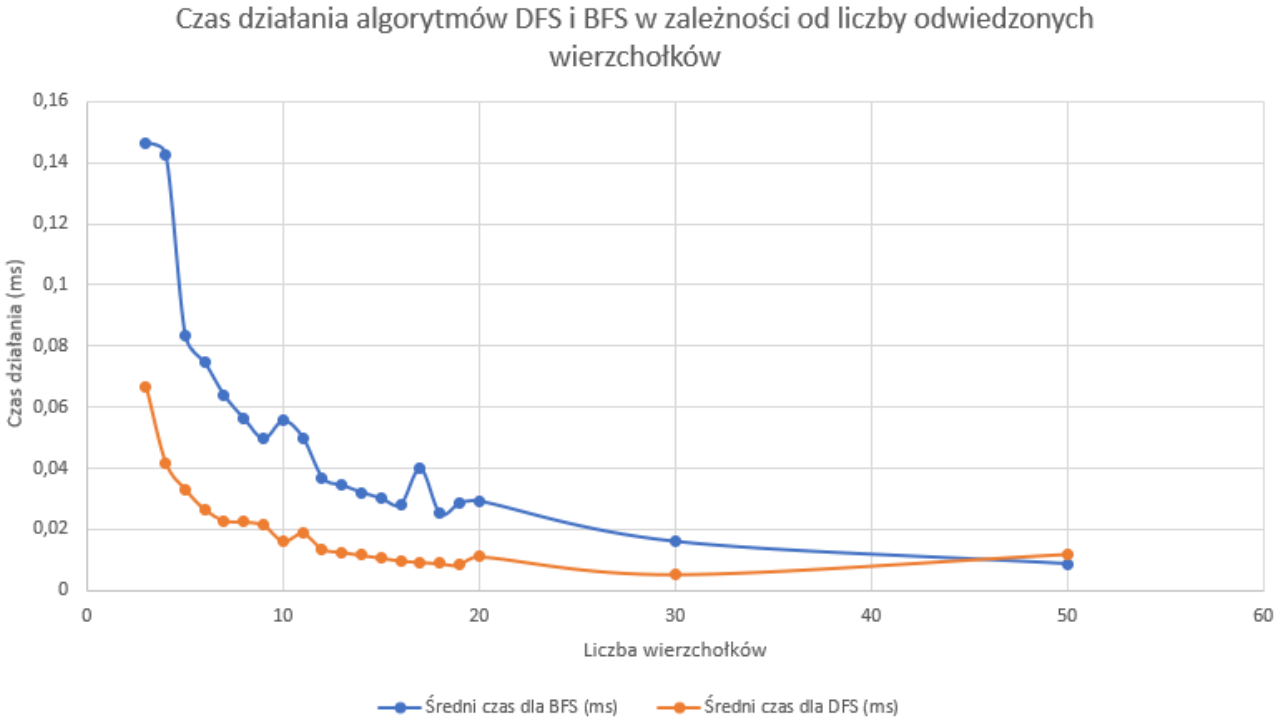
```
        DFSUtil(startVertex, visited, visitedVertices);

        stopwatch.Stop(); // Zatrzymanie pomiaru czasu
        double averageTime = stopwatch.Elapsed.TotalMilliseconds / visitedVertices.Count;
// Obliczenie średniego czasu

        Console.WriteLine("Kolejno odwiedzane wierzchołki (DFS):");
        foreach (int vertex in visitedVertices)
        {
            Console.Write(vertex + " ");
        }
        Console.WriteLine("\nŚredni czas działania (DFS): " + averageTime + " ms");
    }

    private void DFSUtil(int v, bool[] visited, List<int> visitedVertices)
    {
        visited[v] = true;
        visitedVertices.Add(v);

        foreach (int neighbor in adjacencyList[v])
        {
            if (!visited[neighbor])
            {
                DFSUtil(neighbor, visited, visitedVertices);
            }
        }
    }

    // Algorytm BFS (przeszukiwanie wszerz)
    public void BFS(int startVertex)
    {
        bool[] visited = new bool[V];
        List<int> visitedVertices = new List<int>();

        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start(); // Start pomiaru czasu

        Queue<int> queue = new Queue<int>();
        visited[startVertex] = true;
        queue.Enqueue(startVertex);

        while (queue.Count != 0)
        {
```

```
            int vertex = queue.Dequeue();
            visitedVertices.Add(vertex);

            foreach (int neighbor in adjacencyList[vertex])
            {
                if (!visited[neighbor])
                {
                    visited[neighbor] = true;
                    queue.Enqueue(neighbor);
                }
            }
        }

        stopwatch.Stop(); // Zatrzymanie pomiaru czasu
        double averageTime = stopwatch.Elapsed.TotalMilliseconds / visitedVertices.Count;
// Obliczenie średniego czasu

        Console.WriteLine("Kolejno odwiedzane wierzchołki (BFS):");
        foreach (int vertex in visitedVertices)
        {
            Console.Write(vertex + " ");
        }
        Console.WriteLine("\nŚredni czas działania (BFS): " + averageTime + " ms");
    }
}

public class Program
{
    public static void Main()
    {
        Console.WriteLine("Podaj liczbę wierzchołków grafu:");
        int V = int.Parse(Console.ReadLine());

        Graph graph = new Graph(V);

        Console.WriteLine("Podaj krawędzie w formacie 'wierzchołek1 wierzchołek2'
(oddzielone spacją) wpisz 'koniec' aby zakończyć:");

        while (true)
        {
            string input = Console.ReadLine();
            if (input == "koniec")
                break;
```

```
            string[] vertices = input.Split(' ');
            if (vertices.Length != 2)
            {
                Console.WriteLine("Nieprawidłowy format. Podaj krawędzie w formacie
'wierzchołek1 wierzchołek2'.");
                continue;
            }

            int v, w;
            if (!int.TryParse(vertices[0], out v) || !int.TryParse(vertices[1], out w))
            {
                Console.WriteLine("Nieprawidłowe wartości wierzchołków. Podaj poprawne
liczby całkowite.");
                continue;
            }

            graph.AddEdge(v, w);
        }

        Console.WriteLine("Wybierz algorytm:");
        Console.WriteLine("1. DFS (przeszukiwanie w głąb)");
        Console.WriteLine("2. BFS (przeszukiwanie wszerz)");
        int choice = int.Parse(Console.ReadLine());

        Console.WriteLine("Podaj wierzchołek startowy:");
        int startVertex = int.Parse(Console.ReadLine());

        if (choice == 1)
        {
            graph.DFS(startVertex);
        }
        else if (choice == 2)
        {
            graph.BFS(startVertex);
        }
        else
        {
            Console.WriteLine("Nieprawidłowy wybór algorytmu.");
        }
    }
}
```

Tabela i wykres prezentujące czas działania algorytmów dla różnej liczby wierzchołków w grafie:

ilość wierzchołków	Średni czas dla BFS (ms)	Średni czas dla DFS (ms)
3	0,1465	0,0666
4	0,14265	0,0418
5	0,08336	0,03304
6	0,07453	0,02643
7	0,06371	0,02284
8	0,05622	0,022425
9	0,04963	0,021333
10	0,05554	0,01607
11	0,05003	0,018754
12	0,036916	0,013574
13	0,034423	0,012584
14	0,032078	0,011578
15	0,030119	0,0107
16	0,02818	0,009743
17	0,039841	0,009335
18	0,0251222	0,008883
19	0,028557	0,008336
20	0,029235	0,011125
30	0,016063	0,0053466
50	0,00878	0,0119285



Wnioski:

Dla małych grafów o niewielkiej liczbie wierzchołków, takich jak 3, 4 lub 5, różnica w czasie działania pomiędzy BFS, a DFS jest znikoma. Oba algorytmy wykonują się bardzo szybko.Wraz ze wzrostem liczby wierzchołków w grafie, średni czas wykonania obu algorytmów maleje. Oznacza to, że im większy graf, tym algorytmy działają szybciej. Algorytm DFS wydaje się być nieco szybszy niż BFS dla większości badanych przypadków. Jest to zgodne z oczekiwaniami, ponieważ DFS ma tendencję do wcześniejszego odwiedzania wierzchołków w jednej gałęzi grafu, podczas gdy BFS przeszukuje graf warstwowo, co może wymagać większej ilości operacji. Warto zauważyć, że dla pewnych kombinacji liczby wierzchołków BFS może być czasami szybszy niż DFS, co sugeruje, że złożoność czasowa obu algorytmów może zależeć od specyfiki struktury grafu. Podsumowując, dla większych grafów oba algorytmy (BFS i DFS) są wydajne, ale dla niektórych przypadków DFS może być nieznacznie szybszy niż BFS. Ostateczny wybór algorytmu powinien zależeć od konkretnych wymagań, problemu i struktury grafu.