

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Лабораторна робота №5
з курсу «Паралельні та розподілені обчислення»

Виконав:
Гуменюк Станіслав
Група Пмі-33с

Оцінка ____
Перевірив:
Пасічник
Т.В.

2024

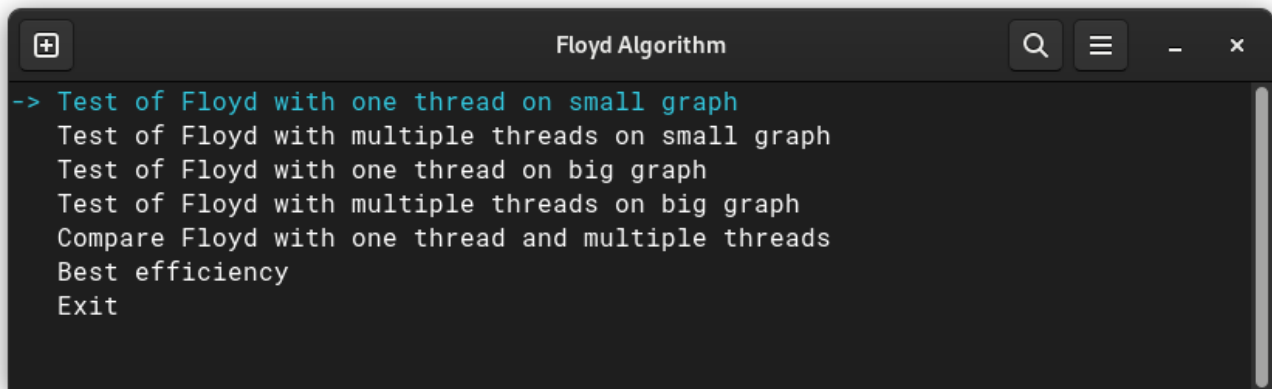
Завдання:

Для орієнтованого зваженого графа $G(V, F)$, де $V = \{a_0, a_1, \dots, a_n\}$ – множина вершин (n – велике число), а F множина орієнтованих ребер (шляхів) між вершинами, використовуючи алгоритм Флойда, знайти найкоротший шлях між заданими вузлами a та b .

Для різної розмірності графів та довільних вузлів a та b порахувати час виконання програми без потоків та при заданих k потоках розпаралелення.

Програмна реалізація:

Програма написана на пакеті .NET з використанням мови C#.
В меню є вибір функціоналу

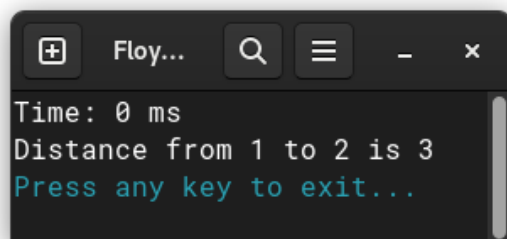


```
Floyd Algorithm
-> Test of Floyd with one thread on small graph
    Test of Floyd with multiple threads on small graph
    Test of Floyd with one thread on big graph
    Test of Floyd with multiple threads on big graph
    Compare Floyd with one thread and multiple threads
    Best efficiency
    Exit
```

Тестовий граф для перевірки роботи алгоритму:

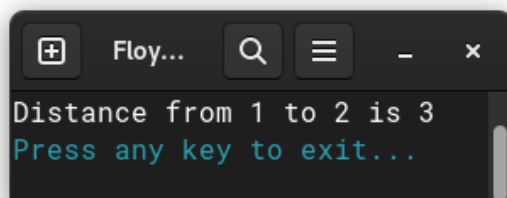
```
private static int[,] graph_test = {
    {0, 5, Graphs.INF, 10},
    {Graphs.INF, 0, 3, Graphs.INF},
    {Graphs.INF, Graphs.INF, 0, 1},
    {Graphs.INF, Graphs.INF, Graphs.INF, 0}
};
```

Результат при одному потоці відстані від 1 до 3:



```
Floy...
Time: 0 ms
Distance from 1 to 2 is 3
Press any key to exit...
```

Результат при декількох потоках на тому самому тестовому графі:



```
Floy...
Distance from 1 to 2 is 3
Press any key to exit...
```

Тест одного потоку на графі з розміром 1000:

```
Floyd A...
Enter number of vertices: 1000
Time: 16504 ms
Distance from 1 to 2 is 2
Press any key to exit...
```

Тест 12-ти потоків на графі розміром 1000:

```
Floyd A...
Enter number of vertices: 1000
Enter number of threads: 12
Time: 3926 ms
Distance from 1 to 2 is 2
Press any key to exit...
```

Порівняння на одному і тому самому графі Одного потоку і 12-ти

```
Floyd Algorithm
Enter number of vertices: 1000
Enter number of threads: 12
Time elapsed with one thread: 14854 ms
Time elapsed with multiple threads: 3778 ms
Time difference: 3.93
Press any key to exit...
```

Найкраща ефективність досягається при 16 потоках

```
Floyd Algorithm
Enter number of vertices: 1000
Time elapsed with one thread: 15444 ms
Time elapsed with 2 threads: 10116 ms
Time difference: 1.53
Time elapsed with 3 threads: 7025 ms
Time difference: 2.2
Time elapsed with 4 threads: 5967 ms
Time difference: 2.59
Time elapsed with 5 threads: 4909 ms
Time difference: 3.15
Time elapsed with 6 threads: 4391 ms
Time difference: 3.52
Time elapsed with 7 threads: 4104 ms
Time difference: 3.76
Time elapsed with 8 threads: 3719 ms
Time difference: 4.15
Time elapsed with 9 threads: 3758 ms
Time difference: 4.11
Time elapsed with 10 threads: 3799 ms
Time difference: 4.07
Time elapsed with 11 threads: 3821 ms
Time difference: 4.04
Time elapsed with 12 threads: 3758 ms
Time difference: 4.11
Time elapsed with 13 threads: 3820 ms
Time difference: 4.04
Time elapsed with 14 threads: 3752 ms
Time difference: 4.12
Time elapsed with 15 threads: 3672 ms
Time difference: 4.21
Time elapsed with 16 threads: 3750 ms
Time difference: 4.12
Best number of threads: 16
Best efficiency: 4.21
Press any key to exit...
```

Код для генерації графу

```
public static int[,] GenerateGraph(int n)
{
    Random random = new Random();
    int[,] graph = new int[n, n];
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
        {
            graph[i, j] = random.Next(0, 2) == 0 ? INF : random.Next(1, 10);
            if (i == j)
                graph[i, j] = 0;
        }
    return graph;
}
```

Код алгоритму на одному потоці

```
public static long FloydOneThread(int[,] graph, int a, int b, bool debug = false)
{
    Stopwatch sw = new Stopwatch();

    int n = graph.GetLength(0);
    int[,] d = new int[n, n];
    Array.Copy(graph, d, n * n);

    sw.Start();

    for (int k = 0; k < n; k++) // перебираємо проміжні вершини
        for (int i = 0; i < n; i++) // перебираємо початкові вершини
            for (int j = 0; j < n; j++) // перебираємо кінцеві вершини
                if (d[i, k] < INF && d[k, j] < INF)
                    d[i, j] = Math.Min(d[i, j], d[i, k] + d[k, j]);

    sw.Stop();
    if (debug)
    {
        Console.WriteLine("Time: " + sw.ElapsedMilliseconds + " ms");
        Console.WriteLine("Distance from " + a + " to " + b + " is " + d[a, b]);
    }
    return sw.ElapsedMilliseconds;
}
```

Код алгоритму на багатьох потоках

```
public static long FloydMultiThread(int[,] graph, int a, int b, int k_threads, bool debug = false)
{
    Stopwatch sw = new Stopwatch();

    int n = graph.GetLength(0);
    int[,] d = new int[n, n];
    Array.Copy(graph, d, n * n);

    sw.Start();

    for (int k = 0; k < n; k++) // перебираємо проміжні вершини
        Parallel.For(0, n, new ParallelOptions { MaxDegreeOfParallelism = k_threads }, i => // перебираємо початкові вершини
        {
            for (int j = 0; j < n; j++) // перебираємо кінцеві вершини
                if (d[i, k] < INF && d[k, j] < INF)
                    d[i, j] = Math.Min(d[i, j], d[i, k] + d[k, j]);
        });

    sw.Stop();
    if (debug)
    {
        Console.WriteLine("Time: " + sw.ElapsedMilliseconds + " ms");
        Console.WriteLine("Distance from " + a + " to " + b + " is " + d[a, b]);
    }
    return sw.ElapsedMilliseconds;
}
```

Висновок:

Обрахунок алгоритму Флойда є ресурсозатратним процесом для великих розмірів, тому паралельне обчислення допомагає скоротити час на виконання таких операцій. В залежності від потужності комп'ютера, можна використовувати різну кількість потоків. Для мого 16-ти поточного процесору, враховуючи що система забирає декілька потоків собі, найефективнішою конфігурацією виявилась 16 потоків