Прізвище: Долінський
Ім'я: Олег
Група: КН-406
Варіант: 8
Кафедра: САПР
Дисципліна: Дискретні моделі в САПР
Прийняв: Кривий Р.З.
Посилання: https://github.com/olehdol/labs.git

ЗВІТ

до лабораторної роботи №4

на тему «**Потокові алгоритми: Алгоритм Форда-Фалкерсона**»

Мета роботи

Отримати практичні навички роботи з пошуку максимального потоку алгоритмом Форда - Фалкерсона

Короткі теоретичні відомості

Ідея алгоритму полягає в наступному. Ми вибираємо такий шлях від джерела до стоку, щоб для кожного ребра залишкова пропускна здатність була строго більше нуля. При цьому ребра на даному шляху можуть проходитися як у прямому, так і в зворотному напрямку. Вибираємо мінімальне значення серед залишкових пропускних спроможностей ребер даного шляху. Збільшуємо потік на кожному з ребер даного шляху на обране мінімальне значення. Далі шукаємо наступний аналогічний шлях. Робота алгоритму продовжується до тих пір, поки вдається знаходити дані шляхи. Відразу відзначимо, що даний алгоритм відноситься до класу недетермінованих, тобто кожен наступний крок алгоритму визначено неоднозначно. І час роботи (кількість кроків) алгоритму залежить від того, як будуть вибиратися кроки.

Важливо те, що алгоритм не конкретизує, який саме шлях ми шукаємо на кроці 2 або як ми це робимо. З цієї причини алгоритм гарантовано сходиться тільки для цілих пропускних спроможностей, але навіть для них при великих значеннях пропускних спроможностей він може працювати дуже довго або зовсім не привести до оптимального рішення.

Індивідуальне завдання:

```
8
0 20 20 40 0 0 0 0
0 0 10 0 10 0 0 0
0 0 0 20 20 0 0 0
0 0 0 0 0 20 20 0
0 0 0 0 0 0 0 30
0 0 10 0 20 0 0 20
0 0 0 0 0 10 0 20
0 0 0 0 0 0 0 0
```

Виконання:

Кожне ребро зберігає номер вершини джерела і напрямку, а також пропускну здібність.

Сутність потоку містить булові значення про наповненість даного ребра та чи було уже відвідане при деякому проходженні.

Цей алгоритм містить деякий вічний цикл, так будуть перебиратися шляхи допоки вони не закінчаться.

```
while (currentNode.Id != graph.Nodes.Last().Id)
{
    FlowModel flow = new FlowModel();
    var temp = Flows.Where(x => x.Edge.Source == currentNode.Id
                                && !x.Full
                                && !x.Visited)
        .OrderByDescending(x => x.Edge.Weight)
        .FirstOrDefault(x =>
        {
            int destination = x.Edge.Destination;
            return destination
                                    == graph.Nodes.Last().Id;
        });
    if (temp != null)
        flow = temp;
    else
        flow = Flows.OrderByDescending(x => x.Edge.Weight)
                    .FirstOrDefault(x =>
                    {
                        int source = x.Edge.Source;
                        return !x.Full && !x.Visited && source == currentNode.Id;
                    });
    if (flow == null)
    {
        if (currentNode.Id == graph.Nodes.First().Id)
        {
            exit = true;
            break;
        }
        else
        {
            Flows.First(x => x.Edge == previousEdge).Full = true;
            currentNode = startNode;
        }
    }
    else
    {
        var edge = flow.Edge;
        path.Add(edge);
        Flows.First(x => x.Edge == edge).Visited = true;
        currentNode = graph.Nodes
        .FirstOrDefault(x => x.Id == edge.Destination);
        previousEdge = edge;
    }
}
if (!exit)
```

Пускаю через знайдений шлях (він називається збільшувальним шляхом) максимально можливий потік;

```
private void CalculateMinCapacity(List<Edge> path)
{
    int minCapacity = path.Min(e =>
    {
        int flow = Flows.FirstOrDefault(x => x.Edge == e).Flow;
        return e.Weight
                                                    - flow;
    });//min residual capacities, min from stock weight minus already filled capacity
    foreach (var item in path)
    {
        Flows.Where(f => f.Edge == item)
            .Select(x => { x.Flow = x.Flow
                                    + minCapacity; return x; })
            .ToList();//encrease flow for current path by min capacity
    }
}
```

Серед знайденого шляху обирається найменша вага ребра, та наповнюються ребра даного шляху на цю мінімальну вагу.

Результати:

```
Максимальний потік
Початкова матриця
       -1      20      20      20      0       0       0       0
       0       -1      0       0       30      0       0       0
       0       10      -1      0       0       10      20      0
       0       0       0       -1      0       15      0       0
       0       0       10      0       -1      10      0       20
       0       0       0       0       0       -1      10      20
       0       0       0       10      0       0       -1      20
       0       0       0       0       0       0       0       -1

Пот?к

                Джерело: 4      Ціль: 7         (20/20)
                Джерело: 5      Ціль: 7         (15/20)
                Джерело: 6      Ціль: 7         (20/20)
                Величина максимального потоку = 55
```

Висновок:

Вивчено алгоритм розв’язання потокових задач, а саме метод Форда-Фалкерсона. Побудовано програмну реалізацію цього алгоритму для пошуку максимального потоку в мережі.