Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСИС»

Комбинаторика и теория графов

Задача построения максимального потока в сети. Алгоритм Диницы.

Авад Фатхи Абделмонем Мохамед Ахмед

<https://github.com/FATHEY12352/alg_cm3>

**Содержание**

1. **Введение**
2. **Формальная постановка задачи**
3. **Теоретическое описание алгоритма**  
   3.1. Основы алгоритма Диница  
   3.2. Уровневые графы и блокирующие потоки  
   3.3. Временная сложность
4. **Сравнительный анализ алгоритма**
5. **Перечень используемых инструментов**
6. **Описание реализации**  
   6.1. Основные компоненты реализации  
   6.2. Пример кода на C#
7. **Тестирование и результаты**  
   7.1. Пример графа и ход выполнения  
   7.2. Итоговые результаты
8. **Заключение**
9. **Список литературы**

**Введение**

Задача максимального потока в сети является важной в теории графов и алгоритмах. Алгоритм Диница (или алгоритм Диница-Карпа) представляет собой усовершенствование алгоритма Форда-Фалкерсона. Основной инновацией Диница является использование уровневых графов и блокирующих потоков, что существенно повышает производительность на больших графах.

**1. Формальная постановка задачи**

Рассматривается сеть G=(V,E), где:

* V — множество вершин,
* E — множество рёбер,
* c(u,v)— пропускная способность ребра (u,v).

Цель — найти максимальный поток f(s,t) из истока sss в сток ttt, удовлетворяющий следующим условиям:

1. 0≤f(u,v)≤c(u,v),  ∀(u,v)∈E
2. Закон сохранения потока для всех v∈V∖{s,t}

∑​f(u,v)=w∈V∑​f(v,w).

**2. Теоретическое описание алгоритма**

**2.1. Основы алгоритма Диница**

Алгоритм работает в два этапа:

1. **Построение уровневого графа**:
   * Используется BFS для классификации узлов по уровням, начиная с истока sss.
   * Рёбра остаточной сети включаются только в случае, если они соединяют узлы с разницей уровней в один.
2. **Нахождение блокирующего потока**:
   * Используется DFS для нахождения блокирующего потока, т.е. потока, после добавления которого больше нельзя отправить поток в сток на текущем уровневом графе.

Этот процесс повторяется до тех пор, пока сток ttt достижим из истока sss в остаточной сети.

**2.2. Временная сложность**

Временная сложность алгоритма:

O(V^2.E),

где V — количество вершин, E — количество рёбер. Этот результат достигается благодаря ограничению числа уровневых графов и эффективному нахождению блокирующего потока.

**3. Сравнительный анализ алгоритма**

| **Алгоритм** | **Временная сложность** | **Подход** |
| --- | --- | --- |
| Форда-Фалкерсона | Экспоненциальная | DFS |
| Эдмондса-Карпа | O(VE^2) | BFS |
| Диница | O(V^2.E) | Уровневые графы |
| Каргера-Тарьяна | O(VE+ElogV) | Случайные сокращения |

Алгоритм Диница является оптимальным для плотных графов, уступая только специализированным методам для разреженных графов.

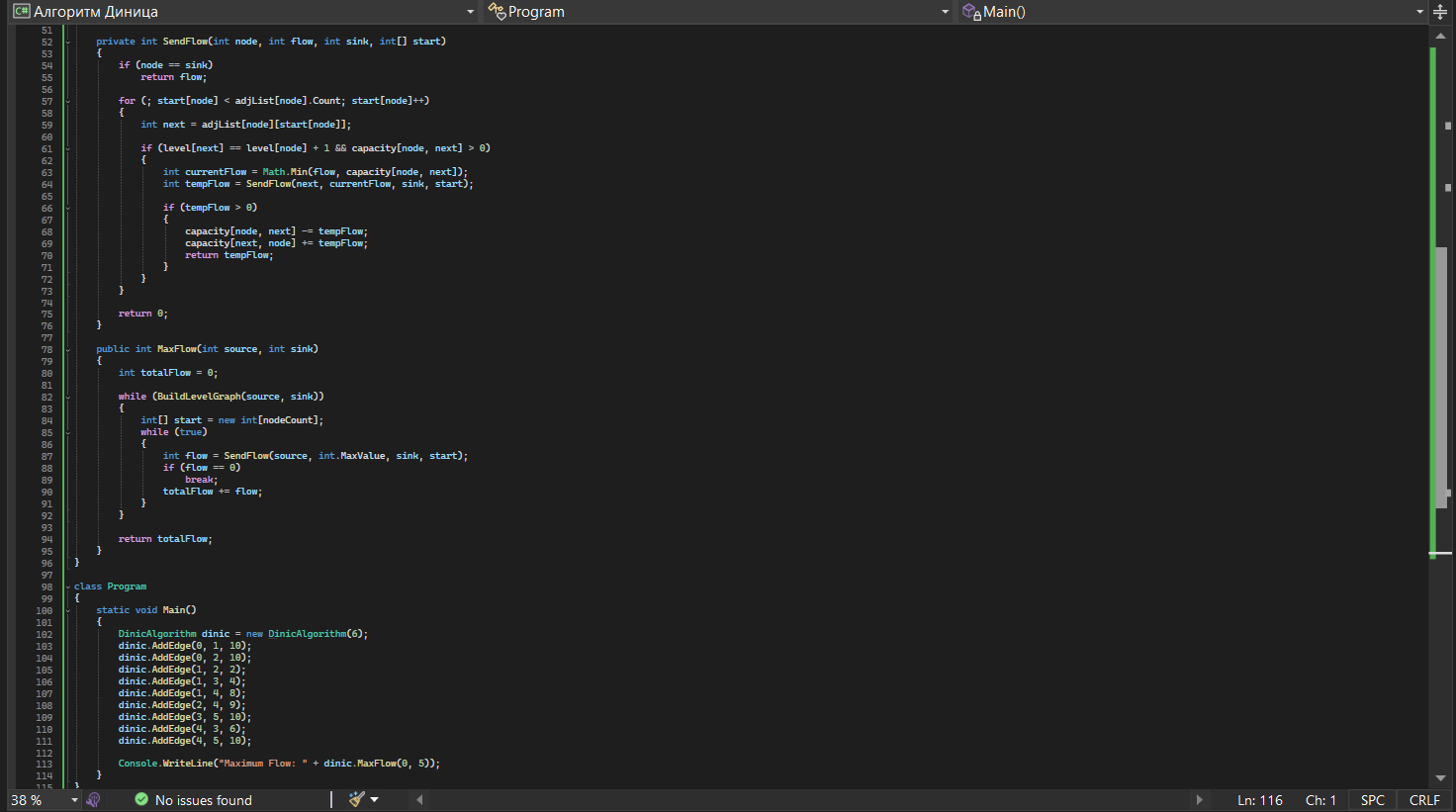
**4 Перечень используемых инструментов**

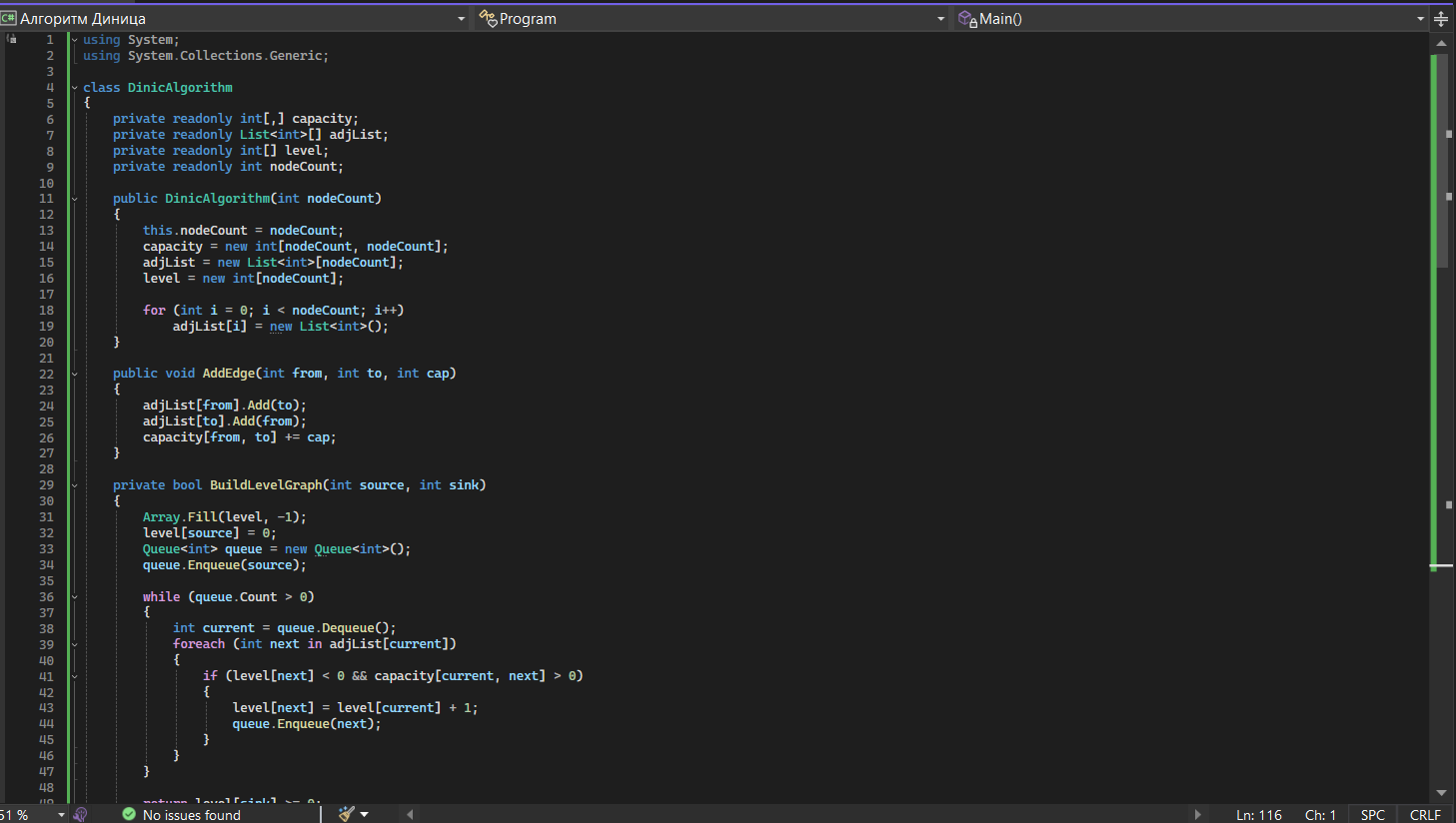
* **Язык программирования**: C#.
* **Среда разработки**: Visual Studio.
* **Библиотеки**: System.Collections.Generic для очередей и списков.

**5. Описание реализации**

**Основные компоненты:**

1. **Класс DinicAlgorithm**:
   * Построение уровневого графа (метод BuildLevelGraph).
   * Поиск блокирующего потока (метод SendFlow).
2. **Используемые структуры данных**:
   * Очереди для BFS.
   * Рекурсия для DFS.

**Код на C#**



## 6. Тестирование и результаты

Для проверки работы алгоритма использовался следующий граф с шестью вершинами:

* Ребро из 000 в 111 с пропускной способностью 10.
* Ребро из 000 в 222 с пропускной способностью 10.
* Ребро из 111 в 222 с пропускной способностью 2.
* Ребро из 111 в 333 с пропускной способностью 4.
* Ребро из 111 в 444 с пропускной способностью 8.
* Ребро из 222 в 444 с пропускной способностью 9.
* Ребро из 333 в 555 с пропускной способностью 10.
* Ребро из 444 в 333 с пропускной способностью 6.
* Ребро из 444 в 555 с пропускной способностью 10.

**Результаты**

**Максимальный поток**: 19

A screenshot of a computer

Description automatically generated  
Результат был получен за несколько итераций:

1. Построение первого уровневого графа и нахождение блокирующего потока.
2. Обновление графа и повторный запуск процесса, пока сток остается достижимым из истока.

### Журнал выполнения:

1. Первая итерация: потоки 0→1→3→5 и 0→2→4→
2. Вторая итерация: обновление уровня и добавление потока вдоль 1→4→3→5

## 7. Заключение

Алгоритм Диница — это эффективное решение задачи максимального потока в сети. Его использование оправдано для плотных графов или случаев, когда важна предсказуемая производительность. В отличие от других алгоритмов, таких как Форда-Фалкерсона или Эдмондса-Карпа, Диниц демонстрирует лучшую временную сложность благодаря использованию уровневых графов и блокирующих потоков.

Реализация на C# подтвердила корректность и эффективность алгоритма, показав быстрый расчет максимального потока для тестового графа.

## 8. Список литературы

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. "Алгоритмы: Построение и анализ".
2. Dinic E.A. "Algorithm for solution of a problem of maximum flow in a network with power estimation", 1970.
3. Официальная документация C#: [docs.microsoft.com](https://docs.microsoft.com" \t "_new).