МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО АВТОНОМНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский технологический университет «МИСИС»

Институт Компьютерных Наук

Отчет

Задача построения максимального потока в сети. Алгоритм Форда-Фалкерсона

По курсу: Комбинаторика и теория графов

Ссылка на репозиторий:

https://github.com/ov3rvoid/combinatorics.git

Журавлёв Сергей Романович

Группа БИВТ-23-6

Содержание

---

1. Формальная постановка задачи

Задача:
Построение максимального потока в сети, представленной ориентированным графом. Поток должен быть максимальным и удовлетворять следующим условиям:

1. **Ограничение пропускной способности**: Поток по любому ребру не может превышать его пропускную способность.
2. **Сохранение потока**: Для каждой вершины, кроме истока и стока, сумма входящих потоков должна быть равна сумме исходящих потоков.

Входные данные:

- Ориентированный граф $G = (V, E)$, где:
  - V — множество вершин;
  - E — множество рёбер с пропускными способностями $c(u, v) \geq 0$ для каждого ребра $(u, v) \in E$.
- Две выделенные вершины: исток $s \in V$ и сток $t \in V$.

Выходные данные:
Максимальный поток f, который можно передать из истока s в сток t.

---

2. Теоретическое описание алгоритма и его характеристики

Описание алгоритма Форда-Фалкерсона
Алгоритм Форда-Фалкерсона является жадным методом, использующим поиск путей увеличения потока. Основные шаги:

1. **Инициализация:** Установить начальный поток $f(u, v) = 0$ для всех рёбер $(u, v) \in E$.
2. **Поиск пути увеличения потока:** Используется **поиск в ширину (BFS)** для нахождения пути от истока s до стока t, на котором ещё имеется остаточная пропускная способность.
3. **Увеличение потока:** Найти минимальную остаточную пропускную способность вдоль найденного пути и увеличить поток на этом пути.
4. **Повторение:** Повторять шаги 2 и 3, пока можно найти пути увеличения потока.

Характеристики алгоритма

- **Временная сложность:**
  - O(E * f), где f — величина максимального потока, если используется DFS.
  - O(V * E), если используется BFS (в данном случае алгоритм эквивалентен методу Эдмондса-Карпа).
- **Пространственная сложность:** O(V^2), если граф представлен матрицей смежности, и O(V + E), если представлен списком смежности.
- **Применимость:** Алгоритм подходит для графов с малыми потоками и эффективен для разреженных графов.

---

3. Сравнительный анализ с другими алгоритмами

Критерий	Форда-Фалкерсона	Эдмондса-Карпа	Диница
Метод	BFS + жадный поиск	BFS + улучшение	Уровневый граф + блокирующий поток
Временная сложность	O(E * f)	O(V * E^2)	O(V^2 * E)
Производительность	Медленная для больших f	Быстрая на разреженных графах	Высокая для плотных графов
Сложность реализации	Простая	Средняя	Сложная

Вывод:
Алгоритм Форда-Фалкерсона уступает по производительности методам Эдмондса-Карпа и Диница, но его простота в реализации и понимании делает его удобным для учебных целей.

---

4. Перечень инструментов, используемых для реализации

- **Языки программирования:**
  - Python 3.x — для разработки и тестирования.
- **Среда разработки:**
  - Visual Studio Code для Python.
  - Для тестирования: библиотека pytest.
- **Библиотеки:**
  - **Python:**
    - collections.deque для эффективной реализации BFS.
    - pytest для тестирования.

---

5. Описание реализации и процесса тестирования

Реализация на Python

- **Основные компоненты:**
    1. **Метод add_edge(u, v, capacity)** — добавляет ребро с заданной пропускной способностью и обратное ребро с нулевой пропускной способностью.
    2. **Метод dfs(source, sink, parent)** — выполняет поиск пути увеличения потока с использованием DFS.
    3. **Метод ford_fulkerson(source, sink)** — вычисляет максимальный поток между истоком и стоком.
- **Тестирование:**
  Код тестируется с использованием модуля pytest в файле test_ford_fulkerson.py. Для этого в тестах проверяется корректность работы алгоритма при различных конфигурациях графов.

Пример теста:

Входные данные:
6 10

0 1 16
0 2 13
1 2 10
1 3 12
2 1 4
2 4 14
3 2 9
3 5 20
4 3 7
4 5 4
0 5

Ожидаемый вывод:

Максимальный поток: 23

Тесты выполняются с использованием библиотеки pytest в Python.

---

6. Преимущества реализации на Python

- **Удобство разработки:** Python подходит для быстрой разработки, тестирования и отладки.
- **Легкость тестирования:** Использование библиотеки unittest позволяет легко проверять корректность работы алгоритма на различных тестовых данных.
- **Подходит для небольших графов:** Алгоритм на Python эффективно работает для задач с небольшими и средними графами.

---

7. Заключение

Алгоритм Форда-Фалкерсона представляет собой базовый метод для вычисления максимального потока в сети. Реализация на Python обеспечивает гибкость и удобство при

разработке и тестировании. Алгоритм идеально подходит для образовательных целей, однако для более сложных и крупных графов рекомендуется использовать более оптимизированные алгоритмы, такие как метод Эдмондса-Карпа или алгоритм Диница.

Основные выводы:

1. **Python-реализация:** Идеальна для тестирования и обучения.
2. **Ограничения:** Алгоритм может быть медленным на графах с большими потоками из-за линейной зависимости от величины потока.