МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО АВТОНОМНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский технологический университет «МИСИС»

Институт Компьютерных Наук

Отчет

Алгоритм Беллмана-Форда построения кратчайших расстояний.

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 кратчайших расстояний от одной вершины (истока) до всех остальных вершин в графе. Алгоритм должен работать в графах с возможными отрицательными весами рёбер, но не поддерживает наличие отрицательных циклов.

Входные данные:

- Ориентированный граф $G = (V, E)$, где:
  - V — множество вершин;
  - E — множество рёбер с весами $w(u, v)$ для каждого рёбра $(u, v) \in E$
- Вершина-исток $s \in V$.

Выходные данные:

- Кратчайшие расстояния от истока s до всех остальных вершин $d(u)$, где $u \in V$.

---

2. Теоретическое описание алгоритма и его характеристики

Описание алгоритма Беллмана-Форда: Алгоритм Беллмана-Форда используется для поиска кратчайших путей от одной вершины ко всем остальным в графе, который может содержать рёбра с отрицательными весами. Основная идея алгоритма заключается в релаксации рёбер. На каждом шаге алгоритм обновляет кратчайшее расстояние до вершины через её соседей.

Алгоритм состоит из следующих шагов:

1. **Инициализация**: Установить кратчайшее расстояние от истока s до всех вершин как бесконечность, за исключением истока, которому присваивается значение 0.
2. **Релаксация рёбер**: Для каждого ребра графа (u, v) обновляется расстояние до вершины v, если найден более короткий путь через вершину u.
3. **Повторение**: Шаг 2 повторяется $V-1$ раз, где V — количество вершин в графе. Это гарантирует, что кратчайшие пути будут найдены.
4. **Проверка на отрицательные циклы**: После $V-1$ итераций проверяется наличие отрицательных циклов. Если расстояние до вершины можно ещё улучшить, это означает наличие отрицательного цикла.

Характеристики алгоритма:

- **Временная сложность**: O(V·E), где V — количество вершин, E — количество рёбер.
- **Пространственная сложность**: O(V), так как необходимо хранить расстояния до всех вершин.
- **Применимость**: Алгоритм работает с графами, содержащими рёбра с отрицательными весами, но не поддерживает графы с отрицательными циклами.

---

3. Сравнительный анализ с другими алгоритмами

1. Алгоритм Беллмана-Форда vs Алгоритм Дейкстры

Критерий	Алгоритм Беллмана-Форда	Алгоритм Дейкстры
Тип графа	Работает с графами с отрицательными рёбрами, но не поддерживает отрицательные циклы.	Работает только с графами, у которых все рёбра имеют неотрицательные веса.
Время работы	O(V·E), где V — количество вершин, E — количество рёбер.	O(V^2) для неорганизованного списка рёбер, O(E + V log V) с использованием кучи.
Алгоритм	Динамическое программирование, релаксация рёбер.	Жадный алгоритм.
Поддержка отрицательных рёбер	Да, но нельзя иметь отрицательных циклов.	Нет, требует всех рёбер с неотрицательными весами.
Применимость	Подходит для графов с отрицательными весами рёбер.	Подходит для графов с неотрицательными весами, особенно с плотными графами.
Пространственная сложность	O(V)	O(V)

Вывод: Алгоритм Дейкстры более эффективен для графов с неотрицательными весами рёбер, однако для графов с отрицательными рёбрами необходимо использовать алгоритм Беллмана-Форда.

2. Алгоритм Беллмана-Форда vs Алгоритм Флойда-Уоршелла

Критерий	Алгоритм Беллмана-Форда	Алгоритм Флойда-Уоршелла
Тип графа	Работает с графами с отрицательными рёбрами, но не поддерживает отрицательные циклы.	Работает с графами с отрицательными рёбрами, но не поддерживает отрицательные циклы.
Время работы	O(V·E), где V — количество вершин, E — количество рёбер.	O(V^3), где V — количество вершин.
Алгоритм	Динамическое программирование, релаксация рёбер.	Динамическое программирование, основанное на обновлении кратчайших путей для каждой пары вершин.

Критерий	Алгоритм Беллмана-Форда	Алгоритм Флойда-Уоршелла
Поддержка отрицательных рёбер	Да, но нельзя иметь отрицательных циклов.	Да, но нельзя иметь отрицательных циклов.
Применимость	Подходит для графов с одним источником, когда необходимо найти кратчайшие пути от одной вершины ко всем остальным.	Подходит для графов с несколькими источниками, когда необходимо вычислить кратчайшие пути между всеми парами вершин.
Пространственная сложность	O(V)	O(V^2)

Вывод: Алгоритм Флойда-Уоршелла имеет большую временную сложность, чем Беллмана-Форда, и используется, когда требуется вычислить кратчайшие пути между всеми парами вершин.

---

4. Перечень инструментов, используемых для реализации

- **Языки программирования**:
  - Python 3.x для реализации алгоритма и тестирования.
- **Среда разработки**:
  - Visual Studio Code (Python).
- **Библиотеки**:
  - Python:
    - pytest для тестирования.
    - collections для работы с очередями и графами.

---

5. Описание реализации и процесса тестирования

Реализация на Python: Основной код реализован в файле bellman_ford.py. Главные компоненты:

1. Метод add_edge(u, v, weight) — добавляет ребро с заданным весом.
2. Метод bellman_ford(source) — реализует алгоритм Беллмана-Форда для поиска кратчайших путей от истока ко всем вершинам.

Тестирование: Тестирование выполнено с использованием pytest в файле test_bellman_ford.py.

Пример теста:

```
def test_bellman_ford():
    graph = Graph(5)
    graph.add_edge(0, 1, -1)
    graph.add_edge(0, 2, 4)
    graph.add_edge(1, 2, 3)
    graph.add_edge(1, 3, 2)
```

```
graph.add_edge(1, 4, 2)
graph.add_edge(3, 2, 5)
graph.add_edge(3, 1, 1)
graph.add_edge(4, 3, -3)

dist = graph.bellman_ford(0)
assert dist == [0, -1, 2, -2, 1]
```

---

6. Преимущества реализации на Python

- Простота и наглядность кода,

что делает его удобным для учебных целей.

- Использование встроенных библиотек Python для тестирования и обработки данных.
- Легкость в изменении и улучшении алгоритма.

---

7. Заключение

Алгоритм Беллмана-Форда является важным методом поиска кратчайших путей в графах с отрицательными рёбрами. Несмотря на свою вычислительную сложность, он остаётся незаменимым инструментом для работы с такими графами, особенно когда важно выявить отрицательные циклы. Сравнение с другими алгоритмами, такими как Дейкстра и Флойд-Уоршелл, показывает, что выбор алгоритма зависит от специфики задачи и структуры графа.