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 Диницы.**

**По курсу:** Комбинаторика и теория графов

**Ссылка на репозиторий:**

<https://github.com/Dmitry27912/Graphs>

Голощапов Дмитрий Вячеславович

Группа БИВТ-23-6

**Отчёт: Реализация алгоритма Дейкстры с использованием черпаков**

**Содержание**

1. Формальная постановка задачи
2. Теоретическое описание алгоритма и его характеристики
3. Сравнительный анализ с аналогичными алгоритмами
4. Перечень инструментов, используемых для реализации
5. Описание реализации и процесса тестирования
6. Преимущества и ограничения реализации
7. Заключение

**1. Формальная постановка задачи**

**Задача**:  
Для заданного ориентированного графа с неотрицательными весами рёбер требуется найти минимальные расстояния от стартовой вершины s до всех других вершин графа.

**Входные данные**:

* Граф G=(V,E), где:
  + V — множество вершин;
  + E — множество рёбер с весами w(u,v) ≥ 0 для каждого ребра (u,v) ∈ E.
* Стартовая вершина s ∈ V.

**Выходные данные**:  
Минимальные расстояния d(v) от стартовой вершины s до всех других вершин v ∈ V.

**2. Теоретическое описание алгоритма и его характеристики**

**Описание алгоритма Дейкстры с использованием черпаков**:  
Черпак (bucket) — структура данных, которая разделяет диапазон возможных значений расстояний на интервалы (черпаки). Алгоритм эффективно обрабатывает вершины, группируя их по значениям минимальных расстояний.

**Шаги алгоритма:**

1. **Инициализация**:
   * Все расстояния d(v) инициализируются как ∞, кроме стартовой вершины s, для которой d(s) = 0.
   * Черпаки создаются как массив длиной W+1, где W — максимальный вес ребра.
   * Стартовая вершина помещается в черпак с расстоянием 0.
2. **Обработка**:
   * Извлекается вершина из самого "лёгкого" черпака (с минимальным расстоянием).
   * Для всех соседей вершины обновляются минимальные расстояния, а вершины перекладываются в соответствующие черпаки.
3. **Завершение**:
   * Алгоритм заканчивает работу, когда все черпаки становятся пустыми.

**Характеристики:**

* **Временная сложность**:
  + O(W⋅∣V∣+∣E∣), где:
    - W — максимальный вес ребра;
    - ∣|V| — количество вершин;
    - ∣|E| — количество рёбер.
* **Пространственная сложность**:
  + O(∣V∣+W).

**3. Сравнительный анализ с аналогичными алгоритмами**

| **Критерий** | **Дейкстра с очередью** | **Дейкстра с черпаками** |
| --- | --- | --- |
| **Структура данных** | Приоритетная очередь | Массив черпаков |
| **Временная сложность** | O((∣V∣+∣E∣)⋅log∣V∣) | O(W⋅∣V∣+∣E∣) |
| **Пространственная сложность** | O(∣V∣+∣E∣): | O(∣V∣+W) |
| **Применимость** | Универсальная | Графы с небольшими весами рёбер |
| **Скорость на практике** | Средняя | Высокая для графов с ограниченными весами |

**Вывод**:  
Метод с черпаками эффективнее на графах с ограниченными весами рёбер, где W существенно меньше ∣V|.

**4. Перечень инструментов, используемых для реализации**

Для реализации алгоритма использовались следующие инструменты:

* **Язык программирования**: JavaScript.
* **Среда выполнения**: Node.js.
* **Редактор кода**: Visual Studio Code.
* **Модуль fs**: Для чтения данных из файла.
* **Jest**: Для тестирования.

**5. Описание реализации и процесса тестирования**

**Реализация алгоритма**

Код реализован в файле dijkstra\_buckets.js. Основные компоненты:

1. **Класс Graph**:
   * Представляет граф с методами для добавления рёбер и выполнения алгоритма Дейкстры.
2. **Метод dijkstraWithBuckets**:
   * Выполняет алгоритм Дейкстры с использованием черпаков.
3. **Функция main**:
   * Читает входные данные из файла, строит граф и вычисляет минимальные расстояния.

Пример входных данных:

6 7

0 1 1

0 2 4

1 2 4

1 3 2

2 3 3

3 4 2

4 5 1

Пример вывода:

Минимальные расстояния: [0, 1, 4, 3, 5, 6]

**Процесс тестирования**

Тестирование проводилось с использованием библиотеки Jest. Были проверены следующие сценарии:

1. **Пустой граф**:
   * Ожидаемый результат: d=[0,∞,∞,…]
2. **Граф с одним ребром**:
   * Проверка корректности расчёта для минимального входного графа.
3. **Сложный граф**:
   * Проверка корректности на графах с несколькими рёбрами и путями.
4. **Большие графы**:
   * Тестирование производительности и правильности работы на больших графах.

**6. Преимущества и ограничения реализации**

**Преимущества:**

1. **Высокая производительность** для графов с ограниченными весами рёбер.
2. **Простота реализации** по сравнению с приоритетными очередями.

**Ограничения:**

1. Неэффективен для графов с большими значениями весов рёбер.
2. Потребляет больше памяти при большом значении W.

**7. Заключение**

Алгоритм Дейкстры с черпаками демонстрирует высокую эффективность на графах с ограниченными весами рёбер. Реализация на JavaScript позволяет быстро разрабатывать и тестировать алгоритм.

**Основные выводы**:

1. Метод с черпаками значительно упрощает вычисления, когда W относительно невелик.
2. Для универсальных графов с большими весами предпочтительнее использовать приоритетные очереди.