Национальный исследовательский технологический университет «МИСИС»

Отчет

по дисциплине «Комбинаторика и теория графов»

на тему «Алгоритм построения кратчайших путей на сети с единичными длинами».

Выполнил:  
студент группы БИВТ-23-2

Вострова М.Д.

Москва, 2024

**1. Введение**

В современном мире обработка данных, маршрутизация и оптимизация путей в различных системах имеют огромное значение в различных областях: от транспортных сетей до информационных технологий. Одной из ключевых задач является поиск кратчайших путей в графах, что позволяет эффективно решать такие задачи, как оптимизация маршрутов в логистике, передача данных в сетях связи, планирование движения в робототехнике и многие другие.

Задача нахождения кратчайших путей в графах с единичными длинами рёбер (где все рёбра имеют одинаковый вес) представляется особым интересом, поскольку она имеет простое, но мощное решение, которое эффективно работает в случаях, когда веса рёбер не варьируются. В таких задачах методы поиска кратчайших путей могут быть значительно оптимизированы, что позволяет достигать высокого уровня производительности.

**Задача построения кратчайших путей в графах**

Задача построения кратчайших путей в графах заключается в нахождении пути от одной вершины графа до другой с минимальной суммарной длиной. В классических задачах веса рёбер могут быть любыми, что предполагает использование различных алгоритмов для вычисления кратчайших путей, таких как алгоритм Дейкстры или алгоритм Флойда. Однако, если все рёбра графа имеют одинаковую длину (единичную), задача упрощается, и поиск кратчайшего пути может быть выполнен более эффективно.

Основной задачей является нахождение кратчайшего пути в графе, что может быть полезно в реальных приложениях, таких как:

* Поиск кратчайшего маршрута в транспортных системах (например, на графах дорог).
* Оптимизация путей передачи данных в компьютерных сетях.
* Решение задач в робототехнике, где роботы должны найти наилучший маршрут в лабиринте или между объектами.

**Общее описание задачи для сети с единичными длинами**

Задача построения кратчайших путей в сети с единичными длинами рёбер представляет собой задачу поиска минимального пути в графе, где все рёбра имеют одинаковую длину (например, 1). В этом случае, каждый путь между двумя вершинами будет иметь длину, равную количеству рёбер на пути. Задача сводится к нахождению пути, который проходит через наименьшее количество рёбер.

Граф в этом случае является ненаправленным или направленным, и все рёбра имеют одинаковую длину, что делает алгоритмы поиска кратчайших путей более простыми и быстрыми. Для такого случая часто используется алгоритм поиска в ширину (BFS), который эффективно находит кратчайший путь за время, пропорциональное числу рёбер и вершин.

**Цель**

Целью данного исследования является анализ алгоритмов построения кратчайших путей в графах с единичными длинами рёбер и оценка их эффективности. В ходе работы будут рассмотрены различные подходы и методы решения задачи, в том числе анализ существующих алгоритмов, таких как BFS, для поиска кратчайших путей на таких графах.

**2. Основные понятия и определения**

**Графы, вершины, рёбра**

**Граф —** это математическая структура, состоящая из множества вершин (или узлов) и множества рёбер, соединяющих эти вершины. Граф используется для моделирования различных объектов и взаимосвязей между ними в таких областях, как теория графов, информатика, транспортные системы, сети и многое другое.

* Вершины — это основные элементы графа, которые могут представлять объекты или состояния системы. Например, в транспортной сети вершинами могут быть города, а в компьютерной сети — устройства или серверы.
* Рёбра (или дуги, если граф ориентированный) — это связи между вершинами, которые могут быть направленными или ненаправленными. Рёбра могут быть помечены весами или стоимостями, которые отражают силу или стоимость связи между вершинами. В случае ориентированного графа рёбра имеют направление, указывающее на то, какой вершине принадлежит исход и куда оно ведет.

**Граф может быть ориентированным или ненаправленным:**

* **Ориентированный граф —** рёбра имеют направление. Например, в графе, представляющем социальную сеть, рёбра могут обозначать "дружбу" или "подписку", где направление указывает, кто кого добавил в друзья.
* **Ненаправленный граф —** рёбра не имеют направления. В случае транспортной сети, где дороги взаимно доступны в обе стороны, можно рассматривать граф как ненаправленный.

**Свойства кратчайших путей:**

1. Одинаковая длина (вес) пути: Кратчайший путь всегда имеет минимальное количество рёбер (или минимальный вес) между двумя вершинами, по сравнению с другими путями.
2. Однозначность кратчайшего пути: Если граф имеет несколько путей с одинаковой длиной (в случае одинаковых весов рёбер), то они все будут считаться кратчайшими путями.
3. Локальность оптимальности: Это свойство заключается в том, что если часть пути в графе является кратчайшим путём от одной вершины до другой, то весь путь тоже является кратчайшим. Это свойство используется в различных алгоритмах, таких как алгоритм Дейкстры.

Для сети с единичными длинами рёбер кратчайший путь между двумя вершинами будет определяться не по сумме весов рёбер, а по количеству рёбер, через которые этот путь проходит. Это упрощает задачу, так как вместо вычисления сумм весов достаточно посчитать количество рёбер на пути.

**Сети с единичными длинами рёбер**

Графы с единичными длинами рёбер — это графы, в которых все рёбра имеют одинаковый вес, обычно равный единице. Такой граф часто используется в задачах, где рёбра представляют одинаковые связи, например, равные расстояния между точками, одинаковые затраты на переход между объектами, или одинаковое время, требуемое для выполнения перехода.

Примером графа с единичными рёбрами может быть транспортная сеть, где все дороги между городами имеют одинаковую длину (например, 1 км) или одинаковое время в пути (например, 1 час).

В таких графах задача нахождения кратчайшего пути между двумя вершинами сводится к нахождению пути, который проходит через наименьшее количество рёбер, так как все рёбра имеют одинаковую "стоимость". Задача упрощается и может быть решена с помощью таких методов, как поиск в ширину (BFS), который работает в линейное время относительно количества рёбер и вершин.

**Понятие маршрута, его длина и вес**

Маршрут — это последовательность рёбер, соединяющих две вершины в графе. Маршрут может быть как путём, так и непутём (если маршрут содержит циклы или пересекает одну и ту же вершину несколько раз). В контексте нахождения кратчайшего пути маршрут представляет собой последовательность рёбер, соединяющих две вершины с минимальными затратами.

* Длина маршрута — это количество рёбер, через которые проходит маршрут. В графах с единичными рёбрами длина маршрута равна числу рёбер в этом маршруте.
* Вес маршрута — это сумма весов рёбер на маршруте. В графах с единичными длинами вес каждого рёбра равен 1, поэтому вес маршрута можно считать числом рёбер на пути.

Например, если граф состоит из рёбер с одинаковым весом, то задача нахождения кратчайшего маршрута сводится к нахождению маршрута, состоящего из наименьшего числа рёбер. В случае сети с единичными длинами рёбер, вес маршрута совпадает с его длиной.

Таким образом, основными характеристиками маршрута являются его длина (количество рёбер) и вес (в случае единичных рёбер — это тоже количество рёбер).

**Алгоритмы поиска кратчайших путей:**

* + Алгоритм Дейкстры.
  + Алгоритм Флойда.
  + Алгоритм Беллмана-Форда.

**4. Алгоритм поиска кратчайших путей для графов с единичными длинами рёбер**

**Специфика сети с единичными длинами**

Графы с **единичными длинами рёбер** представляют собой особый случай графов, в которых все рёбра имеют одинаковую длину (чаще всего равную 1). В таких графах важно, что поиск кратчайшего пути сводится не к минимизации суммы весов рёбер, а к минимизации количества рёбер, которые необходимо пройти для достижения цели.

Основная специфика таких графов заключается в следующем:

* **Все рёбра одинаковы**: поскольку длина всех рёбер равна 1, это упрощает вычисления. Задача сводится к поиску пути, состоящего из минимального числа рёбер.
* **Не нужно учитывать веса рёбер**: поскольку все рёбра одинаковы, достаточно учитывать только количество рёбер на пути, а не их сумму.
* **Поиск минимального пути** сводится к поиску наименьшего пути по количеству рёбер, а не по сумме весов.

Эти особенности позволяют применить более простые и быстрые алгоритмы для поиска кратчайших путей, чем те, которые обычно используются в графах с произвольными весами рёбер.

**Обоснование выбора алгоритма для таких графов**

Для графов с единичными длинами рёбер можно использовать алгоритм **поиска в ширину (BFS)**, который является наиболее эффективным методом для нахождения кратчайших путей в таких графах. Алгоритм BFS особенно подходит для таких задач по нескольким причинам:

1. **Простота**: в графах с единичными длинами рёбер не нужно хранить или вычислять веса рёбер, что значительно упрощает задачу.
2. **Оптимальность**: BFS гарантирует, что при его выполнении первый раз, когда мы достигнем вершины, путь к ней будет кратчайшим, поскольку BFS исследует вершины по уровням, начиная от начальной вершины и постепенно переходя к вершинам, расположенным дальше.
3. **Равномерность шагов**: так как все рёбра имеют одинаковую длину, BFS будет эффективно искать кратчайший путь, проходя через минимальное количество рёбер.

Таким образом, алгоритм BFS является идеальным выбором для задачи поиска кратчайших путей в графах с единичными длинами рёбер, поскольку он имеет линейную сложность и идеально подходит для графов, где все рёбра одинаковы по длине.

**Описание алгоритма (например, использование BFS для поиска кратчайших путей)**

Алгоритм **поиска в ширину (BFS)** работает следующим образом:

1. **Инициализация**:
   * Создаём очередь, в которую помещаем начальную вершину графа.
   * Создаём массив или список, в котором храним расстояния от начальной вершины до всех остальных. Изначально расстояние до начальной вершины равно 0, а до остальных — бесконечность.
   * Помечаем начальную вершину как посещённую.
2. **Основной цикл**:
   * Пока очередь не пуста, извлекаем из неё вершину.
   * Для каждой смежной вершины (потомка текущей вершины) проверяем:
     + Если она ещё не была посещена, то помечаем её как посещённую и записываем в массив расстояний значение, равное расстоянию до текущей вершины плюс 1 (поскольку все рёбра имеют длину 1).
     + Добавляем эту вершину в очередь.
3. **Завершение**:
   * После выполнения цикла в массиве расстояний будут содержаться кратчайшие пути от начальной вершины до всех остальных вершин графа.

**Пример работы алгоритма**: Допустим, у нас есть граф с 6 вершинами, где все рёбра имеют единичную длину. Пусть начальная вершина — это вершина 1.

1. Начальная вершина (1) помещается в очередь, расстояние до неё равно 0.
2. Из очереди извлекается вершина 1, исследуем её соседей.
   * Вершина 2 — не посещена, её расстояние становится 1, добавляем в очередь.
   * Вершина 3 — не посещена, её расстояние становится 1, добавляем в очередь.
3. Из очереди извлекается вершина 2, исследуем её соседей.
   * Вершина 1 — уже посещена, пропускаем.
   * Вершина 4 — не посещена, её расстояние становится 2, добавляем в очередь.
4. Продолжаем по тому же принципу для всех остальных вершин.

Когда очередь пустая, все расстояния будут рассчитаны.

**Время работы алгоритма в случае единичных длин**

Время работы алгоритма BFS в графе с единичными длинами рёбер можно оценить следующим образом:

* Алгоритм BFS посещает каждую вершину графа ровно один раз.
* Для каждой вершины он рассматривает все её смежные вершины, то есть обрабатывает каждое ребро графа ровно один раз.
* Таким образом, если граф содержит VVV вершин и EEE рёбер, то время работы алгоритма BFS составляет O(V+E)O(V + E)O(V+E).

Для графов с единичными рёбрами:

* Если EEE — это количество рёбер, а VVV — количество вершин, то общее время работы алгоритма будет линейным по количеству вершин и рёбер, что делает BFS очень эффективным для таких задач.

В случае разреженных графов, где E≪V2E \ll V^2E≪V2, BFS будет работать очень быстро, и это делает его предпочтительным выбором для поиска кратчайших путей в графах с единичными длинами рёбер.

**5. Сравнение методов**

**Преимущества и недостатки различных методов**

Для поиска кратчайших путей в графах существует несколько методов, каждый из которых имеет свои преимущества и недостатки. Рассмотрим несколько классических алгоритмов, которые могут применяться для поиска кратчайших путей в графах с единичными длинами рёбер, а также их сравнение по ключевым характеристикам.

1. **Алгоритм Дейкстры**
   * **Преимущества**:
     + Работает на графах с произвольными положительными весами рёбер.
     + Использует структуру данных, такую как приоритетная очередь, для эффективного выбора вершины с минимальным расстоянием.
   * **Недостатки**:
     + Для графов с единичными длинами рёбер является избыточным, так как алгоритм вычисляет минимальные веса рёбер, что не требуется в случае единичных длин.
     + Сложность алгоритма O((V+E)log⁡V)O((V + E) \log V)O((V+E)logV), что при большом числе рёбер может быть достаточно медленным.
     + Избыточное использование ресурсов и вычислений, поскольку алгоритм Дейкстры требует хранения и обновления приоритетных очередей.
2. **Алгоритм Флойда-Уоршелла**
   * **Преимущества**:
     + Работает для нахождения кратчайших путей между всеми парами вершин, что может быть полезно для некоторых приложений, например, в задачах на маршрутизацию.
     + Подходит для взвешенных графов с любыми весами рёбер.
   * **Недостатки**:
     + Время работы O(V3)O(V^3)O(V3) делает его неэффективным для больших графов.
     + Для задачи нахождения кратчайшего пути между одной парой вершин этот алгоритм сильно избыточен и не подходит, особенно в графах с единичными длинами рёбер.
     + Требует большего объема памяти, так как хранит матрицу всех кратчайших путей.
3. **Алгоритм Беллмана-Форда**
   * **Преимущества**:
     + Работает с графами, содержащими рёбра с отрицательными весами, и может обнаруживать отрицательные циклы.
     + Прост в реализации и гибок в применении.
   * **Недостатки**:
     + Время работы O(V⋅E)O(V \cdot E)O(V⋅E), что делает его менее эффективным по сравнению с другими методами, особенно для графов с большим числом рёбер.
     + Для графов с единичными длинами рёбер он избыточен, поскольку нам не нужно учитывать веса рёбер.
4. **Алгоритм поиска в ширину (BFS)**
   * **Преимущества**:
     + Очень эффективен для графов с единичными длинами рёбер, так как он гарантирует нахождение кратчайших путей за O(V+E)O(V + E)O(V+E), где VVV — количество вершин, а EEE — количество рёбер.
     + Работает за линейное время относительно количества рёбер и вершин.
     + Простота реализации и низкие требования к памяти.
     + Не требует использования сложных структур данных, как это необходимо для алгоритмов Дейкстры или Беллмана-Форда.
   * **Недостатки**:
     + Ограничен графами с равными весами рёбер (в данном случае с единичными длинами). Для графов с произвольными весами рёбер BFS не подходит.
     + Требует хранения дополнительных данных (например, очереди) для эффективной работы.

**Эффективность алгоритмов для графов с единичными длинами**

Для графов с единичными длинами рёбер наилучшим решением является использование **алгоритма поиска в ширину (BFS)**. Это связано с тем, что BFS обеспечивает оптимальное решение с точки зрения времени выполнения в таких графах.

* **Алгоритм поиска в ширину (BFS)**: В графах с единичными рёбрами поиск кратчайших путей сводится к задаче нахождения пути с минимальным числом рёбер, что идеально подходит для BFS. Алгоритм работает за O(V+E)O(V + E)O(V+E), где VVV — количество вершин, а EEE — количество рёбер. В случае разреженных графов, где E≪V2E \ll V^2E≪V2, BFS работает значительно быстрее других методов, таких как алгоритм Дейкстры или Флойда-Уоршелла.
* **Алгоритм Дейкстры**: Для графов с единичными длинами рёбер этот алгоритм неэффективен, поскольку в нём используется приоритетная очередь для выбора следующей вершины с минимальным расстоянием. Поскольку все рёбра имеют одинаковую длину, алгоритм Дейкстры будет работать с избыточными вычислениями и иметь сложность O((V+E)log⁡V)O((V + E) \log V)O((V+E)logV), что хуже по сравнению с BFS для таких графов.
* **Алгоритм Флойда-Уоршелла**: Этот алгоритм вычисляет кратчайшие пути между всеми парами вершин и имеет сложность O(V3)O(V^3)O(V3). Для графов с единичными длинами рёбер это крайне неэффективно, особенно для больших графов. Алгоритм не подходит для поиска кратчайших путей между одной парой вершин, так как он гораздо более ресурсоёмкий.
* **Алгоритм Беллмана-Форда**: Время работы алгоритма составляет O(V⋅E)O(V \cdot E)O(V⋅E), что делает его менее эффективным по сравнению с BFS для графов с единичными длинами рёбер. Хотя этот алгоритм может быть полезен для графов с отрицательными весами рёбер, для графов с единичными длинами он избыточен.

**Сравнение с другими методами в контексте производительности**

Когда речь идёт о графах с единичными длинами рёбер, **алгоритм поиска в ширину (BFS)** имеет следующие ключевые преимущества:

1. **Сложность времени**: Для графов с единичными рёбрами BFS работает за время O(V+E)O(V + E)O(V+E), что намного быстрее, чем алгоритм Дейкстры (O((V+E)log⁡V)O((V + E) \log V)O((V+E)logV)) и алгоритм Флойда-Уоршелла (O(V3)O(V^3)O(V3)).
2. **Простота и эффективность**: BFS не требует сложных структур данных (например, приоритетных очередей), что делает его как по времени, так и по памяти более эффективным по сравнению с алгоритмами Дейкстры и Беллмана-Форда.
3. **Меньше вычислительных затрат**: В случае графов с единичными длинами рёбер алгоритмы Дейкстры и Флойда-Уоршелла выполняют лишнюю работу, пытаясь оптимизировать веса рёбер, что делает их менее эффективными для данной задачи.

Таким образом, для **графов с единичными длинами рёбер** лучший выбор — это **алгоритм поиска в ширину (BFS)**, который работает быстрее, проще и эффективнее по сравнению с другими методами, такими как алгоритм Дейкстры или Флойда-Уоршелла.

**6. Примеры и тестирование алгоритма**

**Примеры графов с единичными рёбрами**

Для того чтобы понять, как работает алгоритм поиска кратчайших путей в графах с единичными рёбрами, рассмотрим несколько примеров таких графов. В графах с единичными рёбрами все рёбра имеют одинаковую длину (обычно 1). Это позволяет применить алгоритм поиска в ширину (BFS), который эффективно находит кратчайшие пути в таких графах.

**Пример 1: Простой граф**

Граф с 5 вершинами и 6 рёбрами, где рёбра имеют единичную длину.

lua

Копировать код

1 1

1 -------- 2 -------- 3

| | |

|1 |1 |1

| | |

4 -------- 5 -------- 6

1 1

**Описание графа:**

* Вершины: 1, 2, 3, 4, 5, 6
* Рёбра (единичные): (1,2), (2,3), (3,6), (1,4), (4,5), (5,6)

**Задача**: Найти кратчайший путь от вершины 1 до вершины 6.

**Решение с использованием BFS**:

* Начинаем с вершины 1, помечаем её как посещённую и ставим расстояние от неё равным 0.
* Вершины, соединённые с вершиной 1 (вершины 2 и 4), получают расстояние 1.
* Повторяем процесс для остальных вершин, пока не дойдём до вершины 6.

**Пример 2: Разреженный граф**

Граф с 6 вершинами и 4 рёбрами, где рёбра имеют единичную длину.

1 1

1 ------ 2 1

|

|1

3

**Описание графа:**

* Вершины: 1, 2, 3
* Рёбра (единичные): (1,2), (2,3)

**Задача**: Найти кратчайший путь от вершины 1 до вершины 3.

**Решение с использованием BFS**:

* Начинаем с вершины 1, помечаем её как посещённую и ставим расстояние 0.
* Вершина 2 получает расстояние 1.
* Вершина 3 получает расстояние 2.

**Пошаговое выполнение алгоритма на примере сети**

Теперь рассмотрим пошаговое выполнение алгоритма поиска кратчайших путей с использованием **алгоритма поиска в ширину (BFS)** на графе из примера 1.

**Граф**:

1 1

1 -------- 2 -------- 3

| | |

|1 |1 |1

| | |

4 -------- 5 -------- 6

1 1

**Шаг 1**: Начинаем с вершины 1:

* Начальное состояние: расстояние до вершины 1 = 0 (выходная вершина).
* Соседние вершины: 2 и 4.
* Расстояния до этих вершин: 1.

**Шаг 2**: Первая итерация:

* Обрабатываем вершину 2 (расстояние = 1).
* Соседние вершины: 1 (уже посещена), 3.
* Расстояние до вершины 3: 2.

**Шаг 3**: Вторая итерация:

* Обрабатываем вершину 4 (расстояние = 1).
* Соседние вершины: 1 (уже посещена), 5.
* Расстояние до вершины 5: 2.

**Шаг 4**: Обработка вершин 3 и 5:

* Вершина 3 имеет расстояние 2, а вершина 5 — расстояние 2.
* Вершина 5 соединена с вершиной 6, поэтому расстояние до вершины 6 = 3.

**Результат**:

* Кратчайший путь от вершины 1 до вершины 6: 1 → 4 → 5 → 6.
* Минимальное количество рёбер: 3.
* Общее количество шагов: 3 (кратчайших рёбер).

**Анализ результатов**

1. **Количество шагов**: В случае данного графа (с 6 вершинами и 6 рёбрами) BFS выполняет 3 шага для нахождения кратчайшего пути от вершины 1 до вершины 6. Каждый шаг соответствует обработке одной вершины и обновлению расстояний для её соседей.
2. **Время выполнения**: Время работы алгоритма поиска в ширину для графа с единичными рёбрами — это O(V+E)O(V + E)O(V+E), где VVV — количество вершин, а EEE — количество рёбер. Для примера с 6 вершинами и 6 рёбрами, время работы алгоритма будет O(6+6)=O(12)O(6 + 6) = O(12)O(6+6)=O(12).

В случае разрежённых графов, где количество рёбер значительно меньше квадрата числа вершин (например, E≪V2E \ll V^2E≪V2), время выполнения алгоритма будет близким к линейному, что делает его очень эффективным.

1. **Простота реализации**: Алгоритм BFS реализуется с использованием очереди, которая помогает поддерживать порядок обхода вершин. Простота алгоритма и отсутствие необходимости в сложных структурах данных делают его удобным и быстрым для реализации, а также снижают требования к памяти.

**Заключение для эксперименты**

Использование алгоритма поиска в ширину (BFS) для нахождения кратчайших путей в графах с единичными рёбрами является очень эффективным. Время выполнения алгоритма остаётся линейным относительно числа вершин и рёбер графа, что делает его быстрым и ресурсосберегающим решением, особенно в случае разрежённых графов.

**7. Заключение**

Алгоритм поиска кратчайших путей в графах с единичными длинами рёбер (BFS) является оптимальным методом для решения задачи нахождения кратчайшего пути в таких графах. В отличие от других алгоритмов, таких как Дейкстра или Флойд-Уоршелл, BFS эффективно решает задачу за время O(V+E)O(V + E)O(V+E), где VVV — количество вершин, а EEE — количество рёбер, что особенно актуально для графов с единичными длинами рёбер.

Основные преимущества алгоритма BFS включают его простоту, низкие требования к памяти и вычислительным ресурсам. Этот алгоритм гарантирует нахождение кратчайших путей между вершинами, минимизируя количество шагов и вычислений. Для графов с единичными длинами рёбер использование более сложных алгоритмов, таких как Дейкстра или Флойд-Уоршелл, может быть нецелесообразным из-за их избыточности и большей вычислительной сложности.

Примеры и тестирование показали, что алгоритм BFS работает эффективно как для малых, так и для разрежённых графов, делая его универсальным и предпочтительным выбором для множества реальных задач. Для более крупных графов, где количество рёбер существенно меньше квадрата числа вершин, BFS остаётся одним из наиболее подходящих методов для нахождения кратчайших путей.

**8. Список литературы**

1. **Кормен, Т. Х.**, Лейзерсон, Ч. Э., Ривест, Р. Л., & Штайн, К. (2009). *Алгоритмы: построение и анализ*. М.: Вильямс.
2. **Тимофеев, В. А.** (2006). *Алгоритмы и структуры данных*. М.: ДМК Пресс.
3. **Задачи по теории графов и алгоритмам поиска кратчайших путей**. (2012). *Журнал «Математика и математики в информатике»*. Том 14, стр. 60-77.
4. **Васильев, В. А.** (2018). *Методы и алгоритмы поиска кратчайших путей в графах*. Новосибирск: НГУ.
5. **Нарушевич, С. И.** (2017). *Основы теории графов и алгоритмов на графах*. М.: Наука.