федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет информационных технологий и программирования
Направление (специальность) Прикладная математика и информатика
Квалификация (степень) Бакалавр прикладной математики и информатики
Кафедра компьютерных технологий Группа М3439

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе

Инкрементальный адаптивный алгоритм для построения маршрутов по заданным критериям в транспортной сети

Автор квалификационной работы Хованский В.С. _____
Научный руководитель Шалыто А.А. _____
Консультанты:
а) По экономике и организации производства _____
б) По безопасности жизнедеятельности и экологии _____
в) _____

К защите допустить
Заведующий кафедрой Васильев В.Н. _____
«____» _____ 2016 г.

Санкт-Петербург, 2016 г.

Квалификационная работа выполнена с оценкой _____

Дата защиты «____» _____ 2016 г.

Секретарь ГАК _____

Листов хранения _____

Чертежей хранения _____

федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

АННОТАЦИЯ
ПО ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Студент Хованский В.С.

Факультет информационных технологий и программирования

Кафедра компьютерных технологий Группа M3439

Направление (специальность) Прикладная математика и информатика

Квалификация (степень) Бакалавр прикладной математики и информатики

Наименование темы Инкрементальный адаптивный алгоритм для построения маршрутов по заданным критериям в транспортной сети

Руководитель Шалыто А.А., канд. техн. наук, доцент

Консультант

КРАТКОЕ СОДЕРЖАНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ И ОСНОВНЫЕ ВЫВОДЫ

объем 23 стр., **графический материал** – стр., **библиография** 0 наим.

Направление и задача исследований

Целью данной работы является иллюстрация стилевого файла LaTeX для оформления бакалаврских работ в ИТМО.

Проектная или исследовательская часть (с указанием основных методов исследований, расчетов и результатов)

Данная работа является примером оформления бакалаврской работы с использованием стилевого файла `itmo-student-thesis.cls`, разработанного Буздаловым М. В. для замены старого комплекта стилевых файлов, имеющего хождение на кафедре «Компьютерные технологии» Университета ИТМО.

Экономическая часть (какие использованы методики, экономическая эффективность результатов)

Данная работа не предполагает извлечения прямой экономической выгоды из полученных результатов.

Характеристика вопросов экологии, техники безопасности и др.

Результатом работы является программный продукт, не нарушающий требования экологической безопасности.

Новизна полученных результатов

Полученные результаты являются новыми, по крайней мере, ранее существующий стилевой файл никоим образом не соответствует ГОСТ, кроме того, он устроен совершенно уродским образом и не генерирует титульных страниц и аннотаций.

Является ли работа продолжением курсовых проектов (работ), есть ли публикации

Работа является продолжением работ над оформлением в LaTeX кандидатской диссертации и отчетов о НИР.

Практическая ценность работы. Рекомендации по внедрению

Результаты, полученные в работе, могут быть использованы как довольно удобный способ получить халявное ГОСТ-образное форматирование в своей бакалаврской работе.

Выпускник _____

Научный руководитель _____

«____» _____ 2016 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В данном разделе размещается введение.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

В данной главе описаны основные определения и определения из области построения маршрутов в теории графов. В первом разделе главы описаны понятия из теории графов. Во втором разделе описаны понятия из формальной области транспортных сетей. В третьем разделе формализуется задача и список требований по поддерживаемым свойствам для построения маршрутов. Четвертый раздел содержит краткое описание основных алгоритмов теории графов для построения путей со сводной таблицей преимуществ и недостатков данных подходов.

1.1. Основные определения

Определение 1. Теория графов – раздел дискретной математики, изучающий свойства графов.

Определение 2. Граф – это множество вершин(узлов), соединенных ребрами. В строгом определении графом называется такая пара множеств. $G = (E, V)$, где V есть подмножество любого счетного множества, а E – подмножество $V \times V$.

Определение 3. Маршрут – это конечная последовательность вершин, в которой каждая вершина (кроме последней) соединена со следующей в последовательности вершиной ребром. Цепью называется маршрут без повторяющихся рёбер. Простой цепью называется маршрут без повторяющихся вершин (откуда следует, что в простой цепи нет повторяющихся рёбер).

Определение 4. Ориентированный маршрут (или путь) – это конечная последовательность вершин и дуг, в которой каждый элемент инцидентен предыдущему и последующему.

Определение 5. Цикл – это цепь, в которой первая и последняя вершины совпадают. При этом длиной пути (или цикла) называют число составляющих его ребер. Заметим, что если вершины и являются концами некоторого ребра, то согласно данному определению, последовательность является циклом. Чтобы избежать таких «вырожденных» случаев, вводят следующие понятия.

Определение 6. Транспортное средство – это совокупность технических систем, предназначенных для перемещений людей и грузов из одного места в другое.

Определение 7. Транспортный узел – это комплекс транспортных устройств в пункте стыка нескольких видов транспорта, совместно выполняю-

щих операции по обслуживанию транзитных, местных и городских перевозок грузов и пассажиров.

Определение 8. Транспортный рейс –

Определение 9. Транспортная сеть – это совокупность всех транспортных рейсов, представленных в течение интервала продажи билетов.

Определение 10. Остановка — специально отведенное общественное место, предназначенное для посадки/высадки пассажиров рейсового транспортного средства.

Определение 11. Расписание —

Определение 12. Мульмодальный маршрут — это конечная последовательность транспортных рейсов, попав на которые в определенные промежутки времени можно добраться от начального транспортного узла до конечного.

Определение 13. Построитель маршрутов — это программный комплекс для обработки внешних поисковых клиентских запросов, имеющий доступ к полному объему данных о расписаниях на всех транспортных узлах и осуществляющий выдачу определенного количества маршрутов в соответствии с поступившими в запросах требованиями. Также в качестве дополнительных возможностей доступно построение фильтров и различной статистики (активные транспортные узлы, активные транспортные рейсы, проходящие через заданный узел).

Определение 14. Клиентское приложение — это любое приложение, которое осуществляет запросы к построителю маршрутов за результатом (маршрутами и фильтрами).

1.2. Виды транспорта и его особенности

В транспортной сети, в которой будут строиться маршруты, будет существовать только транспорт с конкретным расписанием транспортных рейсов. Таким образом, идет допущение о том, что система сети идеальна и весь транспорт гарантировано совершает остановки в назначенное время. Постановка вспомогательных свойств для построителя маршрутов, которые позволяют сгладить последствия этого допущения будут описаны в следующих главах. Далее идет описание рассматриваемого транспорта.

1.2.1. Железнодорожный

В задаче будут рассматриваться 2 вида железнодорожного транспорта. Во-первых, это будут поезда дальнего следования, у которых небольшое коли-

чество рейсов (около 10^5 в течение интервала продажи билетов). Во-вторых, это будут электрички, которые уже совершают до 10^6 рейсов за аналогичный промежуток времени. Транспортными узлами являются железнодорожные станции и вокзалы.

1.2.2. Воздушный

Воздушный транспорт будет представлен только самолетами. При этом количество рейсов около 10^3, поэтому особый интерес этот случай не представляет. Но стоит отметить, что в большинстве случаев мультимодальный маршрут не будет содержать больше одного воздушного сегмента пути. Транспортными узлами являются аэропорты.

1.2.3. Автомобильный

Автомобильный транспорт состоит из автобусных междугородних рейсов. Около 95% таких рейсов совершаются только между соседними городами, что сильно упрощает задачу, но количество все равно большое — 10^6. Также в эту категорию входит транспорт в пределах города (или любого крупного населенного пункта), например, такси. Стоит отметить, что в этот вид транспорта можно внести любые другие средства передвижения внутри города, так как в конечном счете это не будет влиять на алгоритм. При этом важно, чтобы у нового транспорта в пределах города имелась возможность рассчитать эвристическое времени передвижения между двумя транспортными узлами, которые относятся к одному населенному пункту. Эту задачу следует решать на основе статистики или с помощью сторонних сервисов, которые умеют анализировать дорожную ситуацию, например, такие сервисы, как 2gis или Яндекс.карты, которые могут оценить время движения на основе карты пробок. Транспортными узлами являются автобусные остановки и крупные населенные пункты.

1.3. Построение маршрутов

Основная задача, ставящаяся перед построителем маршрутов — построение маршрутов по данным, доступным в его памяти и внешних базах данных, доступных для чтения в конкретный момент времени. На алгоритм построения маршрутов в транспортной сети накладываются следующие условия и ограничениями.

1.3.1. Мультимодальность

Маршруты могут быть мультимодальными, то есть проходить через несколько точек-остановок, содержать пересадки, проходить разными видами транспорта со своими особенностями и т.д.; Это нужно для того, чтобы была возможность добраться из любой точки в любую, где есть хотя бы какой-нибудь транспорт. Вариант пройти пешком небольшой кусок пути тоже доступен внутри крупного населенного пункта также должен быть доступен.

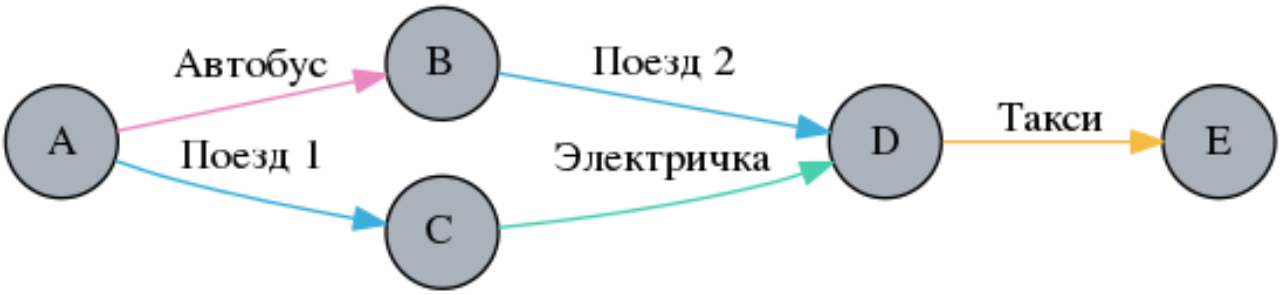

Рисунок 1 – Несколько мультимодальных маршрутов от точки A до точки E.

1.3.2. Временные интервалы

Маршруты можно строить для определенных интервалов времени. Например, хотим выехать в промежуток с 8-00 до 12-00 утра, а приехать в любой день на следующей неделе, но обязательно после 21-00. Это требуется, чтобы иметь возможность бронировать гостиницу, не отходя от кассы.

1.3.3. Инкрементальное построение

Маршруты требуется строить инкрементально (не все сразу, а только небольшую часть из существующих) из-за того, что возможное количество маршрутов может достигать до 109 между парой крупных населенных пунктов с 3 допустимыми пересадками и интервалом времени в пути равным нескольким дням. Это требуется для конечного клиентского приложения, чтобы можно было организовать страничный показ результатов без полного вычисления всех маршрутов на предыдущих страницах.

1.3.4. Адаптивность по времени

Маршруты могут строиться адаптивно по времени из-за того, что важно время отклика алгоритма, то есть в приоритете время выполнения над показом действительно всех требуемых результатов.

1.4. Построение фильтров к доступным маршрутам

Под фильтром в данном случае понимается предикат, который принимает в качестве аргумента построенный маршрут и возвращает ИСТИНА или ЛОЖЬ в зависимости от того, удовлетворяет ли маршрут критериям поиска, которые задает фильтр. Таким образом, помимо построения самих маршрутов требуется построить фильтры по доступным маршрутам со следующими условиями.

1.4.1. Косвенные признаки

Из-за того, что маршруты строятся не все сразу, то кроме непосредственно найденных маршрутов существует огромное количество потенциально доступных маршрутов. При этом мы хотим получить к ним доступ по косвенным признаками. Например, это может быть тип транспорта, номер поезда или тип места в самолете (у окна/у туалета/в хвосте). Формально любой параметр доступный в модели данных может стать доступным для фильтрации. Примечание. Под моделью данных в данном случае подразумевается любой абстрактный объект, который имеет отражение в реальном мире: поезд, самолет, аэропорт и т.д.

1.4.2. Осуществление фильтрации

1.4.3. Функциональные зависимости

Не последнюю роль в фильтрах играют функциональные зависимости, потому что в последствии нужно будет их эффективно показывать без противоречий. Например, тип места «у окна» в купе и кающе относятся к разным типам транспорта и их нельзя объединять. Пример «дерева» функциональный зависимостей для поезда:

- Точки отправления и прибытия
- Интервалы отправления и прибытия
- Вид транспорта:
- Поезд:
  - Перевозчик
  - Бренд
  - Номер
  - Тип вагона:
    - Купе:

        - Верхнее/нижнее
        - Не у туалета
    * Сидячий:
        - У окна/у прохода
    * Плацкарт:
        - Верхнее/нижнее
        - Не у туалета
        - Боковое/не боковое

Каждый уровень списка зависит от родительского уровня и строго им определяется для того, чтобы исключить ситуации, когда несколько косвенных признаков совпадают уже разных видов транспорта. Например, признак «Перевозчик» у поезда и самолета. Такое разделение необходимо для корректного с точки зрения логики и удобного вывода результата в клиентском приложении.

1.5. Сортировка маршрутов

Маршруты требуется строить в порядке сортировки. В простейшем варианте можно сортировать только построенные маршруты, что не представляет из себя никакой сложности. В сложном варианте маршруты строятся на основе любого предиката сравнения пары маршрутов и выдаются в результат, гарантируя определенный порядок. В рамках данной работы подходит «средний» вариант. Требуется гарантировать определенный порядок построенных маршрутов без пропусков, но предикаты известны заранее. Всего их основных 4 вида:

1. Количество пересадок
2. Время отправления
3. Время прибытия
4. Время в пути

Рассмотрим каждый подробнее.

1.5.1. Количество пересадок

Самая простая сортировка в рамках данной работы (или просто сортировка по-умолчанию). Несложно заметить, что количество пересадок будет пропорционально количеству транспорта, который будет включен в мультимодальный маршрут. И если представить каждый отрезок пути на конкрет-

ном транспорте отдельным ребром в абстрактном графе, то сортировка будет происходить относительно количества ребер.

1.5.2. Время отправления/прибытия

У каждой остановки любого транспортного рейса существует время прибытия и время отбытия. У первой и последней остановок оно совпадает. Если задан бесконечный или полу-бесконечный интервал, то нижний предел времени отграничен текущим северным временем, а верхний предел – неким разумным пределом, например, датой продаж. Для выбора верхней границы существуют разные стратегии. Например, при заданном интервале отправления можно устанавливать интервал прибытия зависимым от него. Для этого, нужно знать максимальную продолжительность пути в системе, что не представляет сложностей при ограничении максимального количества пересадок.

1.5.3. Время в пути

1.6. Известные алгоритмы

1.6.1. Алгоритм Дейкстры

Классический алгоритм для вычисления кратчайших путей в статическом графе $G = (V, E)$ с функцией веса[1] для ребер называется алгоритмом Дейкстры. В частности, алгоритм, разработанный Э. Дейкстрой, решает проблему нахождения кратчайших маршрутов от единственной стартовой вершины s до всех остальных вершин в графе G. Алгоритм поддерживает для каждой вершины u метку $\delta(u)$ с предварительным расстояние от s до u. Каждая вершина может находится в одном из следующих состояниях: нерассмотренная, рассмотренная, посещенная.

Изначально только вершина s считается рассмотренной с $\delta(s) = 0$ и все остальные вершины u являются нерассмотренными с $\delta(u) = \infty$.

На каждом шаге вершина u с минимальной $\delta(u)$ извлекается и удаляется из приоритетной очереди Q. Будет говорить, что вершина посещена, так как теперь известно, что $\delta(u)$ – кратчайшее расстояние. Все исходящие ребра (u, v) посещенной вершины релаксируются, то есть сравнивается кратчайшее расстояние от s через u до вершины v с предварительным расстоянием $\delta(v)$. Если оно меньше, то мы обновляет $\delta(v)$ и v в приоритетной очереди Q. Стоит заметить, что такое обновление для нее это либо операции вставки, если

[1]Предполагается, что в статическом графе ребра имеют неотрицательные веса. Для общего случая можно использовать алгоритм Беллмана — Форда.

вершина рассмотрена, либо операция уменьшения ключа в противном случае. Алгоритм заканчивает работу только тогда, когда очередь становится пустой. Это произойдет после n шагов, где n – это количество вершин в графе G.

В приведенном ниже псевдокоде видно, что иногда мы не посещаем все вершины в графе. Это происходит из-за того, что либо такие вершины недостижимы из s, либо очередь становится пустой раньше, чем мы доходим до такой вершины. Для того, чтобы избежать инициализации алгоритма за $O(n)$ при последовательных вызовах, можно сохранять посещенные вершины и обновлять их значения δ после конца алгоритма. Таким образом, асимптотика алгоритма зависит только от количества посещенных вершин и релаксируемых ребер, а не от n.

Листинг 1 – Алгоритм Дейкстры

```
function DIJKSTRA(s)
    δ = {∞, ..., ∞}                          ▷ предварительные расстояния
    δ(s) = 0                                  ▷ поиск начинается из вершины s
    Q.update(0, s)                            ▷ приоритетная очередь
    while Q ≠ ∅ do
        (_, u) = Q.deleteMin()                ▷ посещаем u
        for e = (u, v) ∈ E do                 ▷ релаксируем ребра
            if δ(u) + c(e) < δ(v) then
                delta(v) = δ(u) + c(e)
                Q.update(δ(v), v)
            end if
        end for
    end while
end function
```

Алгоритм Дейкстры использует максимум n операций вставки в приоритетную очередь, n удалений и m операций уменьшения ключа, обеспечивая таким образом асимптотику $O(m+n\log n)$ при использовании Фибоначчиевой кучи.

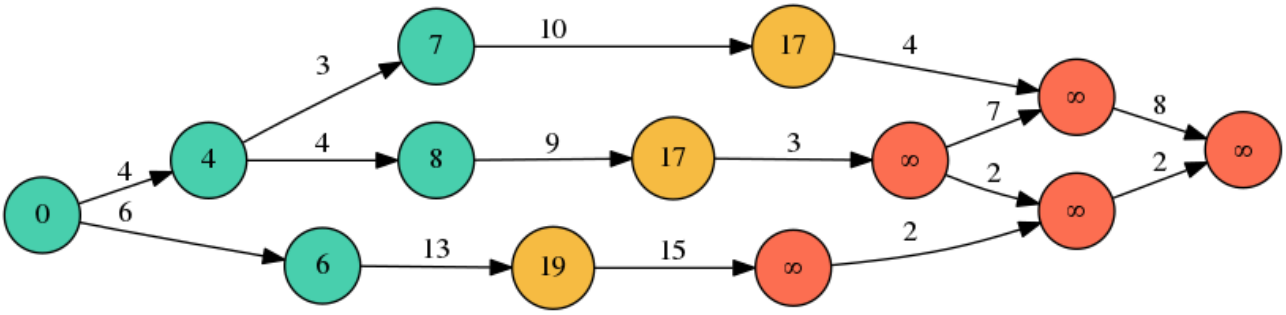

Рисунок 2 – Состояние алгоритма Дейкстры после посещения 5 вершин. Посещенные - зеленые, рассмотренные - желтые, нерассмотренные - красные.

1.6.2. Алгоритм Йена

Алгоритм Йена находит k путей без циклов от единственной стартовой вершины s до конечной вершины t в статичном графе. Разработанный Йеном алгоритм предполагает, что в его основе будет лежать любой другой алгоритм поиска кратчайшего пути, например, его основой может послужить алгоритм Дейкстры. Идея основана на том, что можно построить изначально кратчайший путь и потом на основе этого пути искать отклонения, чтобы построить следующий. Каждая k итерация будет искать отклонения от кратчайших путей, полученных на $k-1$ итерациях.

Листинг 2 – Алгоритм Йена

function YEN(s, t, K)
$A[0] = Dijkstra(s, t)$ ▷ определяем кратчайший путь от s до t
$B = \emptyset$ ▷ приоритетная очередь, хранящая кандидатов на k кратчайший путь
for $k = 1$ to K **do**
for $i = 0$ to $size(A[k-1]) - 1$ **do**
$spurNode = A[k-1].node(i)$ ▷ Вершина ответвления извлекается из полученного на предыдущей итерации пути
$rootPath = A[k-1].nodes(0, i)$ ▷ Последовательность вершин от стартовой до вершины ответвления образуют корневой префикс пути
for $p \in A$ **do**
if $rootPath = p.nodes(0, i)$ **then**
удаляем $p.edge(i, i+1)$ из графа
end if
end for
for $rootPathNode \in rootPath$ **do**
if $rootPathNode \neq spurNode$ **then**
удаляем $rootPathNode$ из графа
end if
end for
$spurPath = Dijkstra(spurNode, t)$ ▷ вычисляем путь-ответвление
$totalPath = rootPath + spurPath$ ▷ Полный путь состоит из корневого префикса и пути-ответвления
$B.append(totalPath)$ ▷ Добавление кандидата на кратчаший k путь
восстанавливаем в графе удаленные на текущей итерации ребра и вершины
end for
if $B = \emptyset$ **then**
break
end if
$(\_, A[k]) = B.deleteMin()$
end for
end function

Выводы по главе 1

ГЛАВА 2. АЛГОРИТМ ПОСТРОЕНИЯ МАРШРУТОВ

2.1. Модели данных

В этом разделе будут описаны 3 способа представления данных о транспортной системе в виде графа, на котором впоследствии будут применятся алгоритмы для построения маршрутов и доступ к которому будет иметь построитель маршрутов.

2.1.1. Статичный граф

В статичном случае каждое ребро взвешенно функцией $c : E \to R$, и не имеет параллельных ребер. Для каждого ребра $e = (u, v)$ будем писать иногда $c(u, v)$ вместо $c(e)$. Будем называть такой граф простым взвешенным. Вес ребра можно интерпретировать как среднее время движения, требуемое на преодоление сегмента дороги, или как физическую длину. Длина пути P в таком случае равна $c(P) = \sum_{i=1}^{k} c(e_i)$. Путь P^* будет является кратчайшим в том случае, если не существует другого пути P' с такой же стартовой и конечной вершинами, что и у пути P^*, такого, что $c(P') < c(P^*)$.

Можно построить граф транспортных рейсов, который будет соответствовать данному случаю. Для этого возьмем на вершину графа транспортный узел, а движение транспорта от одного транспортного узла до другого – за ребро. За вес ребра будет принята длина сегмента пути, так как она не меняется с течением времени. Далее на таком графе можно применить алгоритм Йена и найти k путей.

К сожалению, такой подход имеет ряд существенных минусов. Во-первых, будет доступна только одна естественная сортировка – по количеству пересадок, так как в данном случае это будет просто количество ребер. Во-вторых, поддержка временных интервалов потребует дополнительных вызовов алгоритма для того, чтобы гарантированно получить те маршруты, которых попадают в определенные временные границы. В-третьих, это размер графа, который зависит от количества остановок каждого транспорта за период даты продаж.

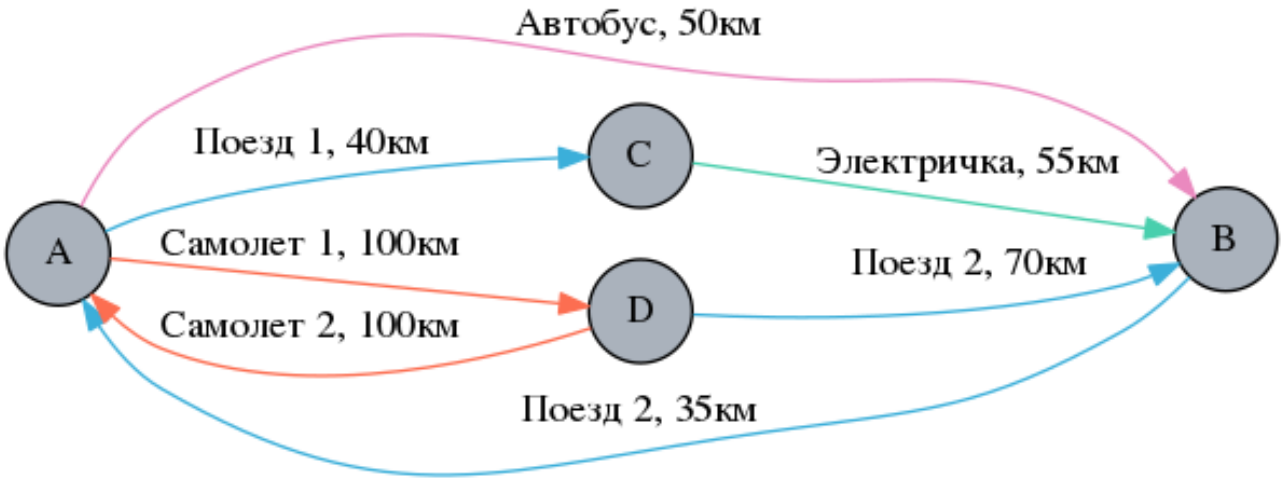

Рисунок 3 – Статичный граф с 4 городами и 6 транспортами

2.1.2. Граф расписаний

Традиционно расписания представляются множеством поездов (автобусов, самолетов и т.д.). Каждый поезд посещает последовательность станций (автобусных остановок, аэропортов и т.д.). Для каждой станции, за исключением последней, расписание включает в себя время отбытия, и для каждой станции, за исключением первой, включает в себя время прибытия, как показано в таблице:

Для того, чтобы была возможность математически определить связи, состоящие из нескольких поездов, мы разделим их в элементарные связи. Более формально, мы имеет множество станций B, множество остановок Z_S на каждой станции $S \in B$ и множество элементарных связей C, чьи элементы c являются кортежами вида $c = \{Z_d, Z_a, S_d, S_a, \tau_d, \tau_d\}$. Такой кортеж интерпретируется как поезд, который отправляется со станции S_d со временем отбытия τ_d после остановки Z_d и затем следующую остановку Z_a на станции S_a с временем прибытия τ_a.

Определение 15. Длительность элементарной связи c определяется как $d(c) = \tau_a(c) - \tau_d(c)$.

Для моделирования более реалистичных маршрутов в данном графе вес ребра будет зависеть от длительности сегмента пути

2.1.3. Граф рейсов

2.2. Построение маршрутов

2.2.1. Дополнение временных интервалов

2.3. Построение фильтров

2.3.1. Косвенные признаки

2.3.2. Осуществление фильтрации

2.3.3. Функциональные зависимости

2.3.4. "Белые"и "черные"фильтры

2.4. Сортировка маршрутов

2.4.1. Количество пересадок

2.4.2. Время прибытия

2.4.3. Время отправления

2.4.4. Время в пути

Листинг 3 должен иметь номер 4.

Листинг 3 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Рисунок 4 должен иметь номер 2.

Рисунок 4 – Пример рисунка

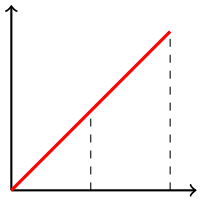

Таблица 1 должна иметь номер 3.

Выводы по главе 2

В конце каждой главы желательно делать выводы. Вывод по данной главе — нумерация работает корректно, ура!

Таблица 1 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

ГЛАВА 3. ДЕТАЛИ РЕАЛИЗАЦИИ И ТЕСТИРОВАНИЕ

3.1. Работа с базой данных

3.1.1. Особенности базы данных

3.1.2. Персистентные модели данных

3.1.2.1. Транспорт

3.1.2.2. Остановки

3.1.2.3. Пересадки

3.1.3. Кэши

3.1.3.1. Для моделей данных

3.1.3.2. Для запросов

3.2. Модуль для генерации карт транспортных рейсов

3.2.1. Генерация транспортных узлов

3.2.2. Генерация транспортных рейсов

3.2.3. Генерация центральных узлов

3.3. Результаты тестирования

Выводы по главе 3

ЗАКЛЮЧЕНИЕ

В данном разделе размещается заключение.

ПРИЛОЖЕНИЕ А. ПРИМЕР ПРИЛОЖЕНИЯ