МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Автоматической электросвязи

«Допущен к защите»
Заведующий кафедрой АЭС
Чежимбаева К.С., к.т.н., доцент
(Ф.И.О., ученая степень, звание)
_____ «_____»_____**20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Оптимизация транспортной маршрутизации с помощью геоинформационных систем в Бостандыкском районе г. Алматы

Специальность 5В071900-Радиотехника, электроника и телекоммуникации

Выполнил (а) Имангалиева Ф.Д. СТС-10-04
(Фамилия и инициалы) группа

Научный руководитель Чежимбаева К.С. к.т.н., доцент
(Фамилия и инициалы, ученая степень, звание)

Консультанты:
по экономической части:
Бабич А.А.
(Фамилия и инициалы, ученая степень, звание)
_____ « 27 » мая 2014 г.
(подпись)

по безопасности жизнедеятельности:
Санатова Т.С., к.т.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ « 23 » мая 2014 г.
(подпись)

по применению вычислительной техники:
Тулемисова К.Х., к.т.н., профессор
(Фамилия и инициалы, ученая степень, звание)
_____ « 28 » мая 2014 г.
(подпись)

Нормоконтролер: Абаров Д.А. ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ « 04 » 06 2014 г.
(подпись)

Рецензент: _____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

Алматы 2014 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Радиотехники и связи
Специальность 5В071900 – Радиотехника, электроника и телекоммуникации
Кафедра Автоматической электросвязи

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Имангалиева Фариза Дастановна
(фамилия, имя, отчество)

Тема проекта Оптимизация транспортной маршрутизации с помощью геоинформационных систем в Бостандыкском районе г. Алматы

утверждена приказом ректора № ___ от «___» сентября 20___ г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта

Бостандыкский район г. Алматы, южная часть города, улично-дорожная сеть города, место отправления и место назначения, построенный граф дорожной сети города, составленная матрица смежности

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:

1. Анализ географических информационных систем в целом (раскрыть основную идею ГИС, описать задачи, которые решает ГИС и как работает система)
2. Описание используемых средств разработки (подробности программной реализации, основы теории графов, поиск кратчайших путей)
3. Расчетная часть (построение графа дорожной сети города, составление матрицы смежности, программная реализация
4. Применение расчетов в экономической части
5. Применение расчетов по разделу безопасность жизнедеятельности

Перечень графического материала (с точным указанием обязательных чертежей)

Рисунок 2.1 - Общая архитектура системы; рисунок 2.2 - архитектура сервера; рисунок 2.3 - архитектура клиента; рисунок 2.4 - Диаграмма последовательности для вычисления и отображения оптимального маршрута

Рекомендуемая основная литература

1) Жокурин И.М., Гуев О.Я., Шустик Л.Я., Шустик К.Л. Геоинформационная модель прокладки и определения параметров маршрута поездки автомобиля по городу и области. С.-Пб.: Энергоатомиздат, 2002г. - 438с.

2) Майника Э. Алгоритмы оптимизации на сетях и графах. - М.: Мир, 2007г. - 192с.

3) Андриенко В. - ГИС на транспорте, статья журнала Arc View №1 (24) - 2003г.

4) Базылов К.Б., Алибаева С.А., Бабич А.А. Дипломная работа бакалавров. Экономический раздел - Алматы, АИЭС, 2008г. - 20с.

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Сакатова Г.С.	1.04 - 23.05.14г	
Экономич. часть	Бабич А.А	21.04 - 20.05.14г.	
Применение ВТ	Тулменбаева К.Х	04.04 - 28.05	
Нормоконтроль	Абдуллина Ж.М	04.06.14	

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1.	Введение	24.09.13 - 08.10.13г.	Выполнено
2.	Анализ географических информационных систем в целом	08.10.13 - 22.11.13г.	Выполнено
3.	Основная идея ГИС	22.11.13 - 15.12.13г	Выполнено
4.	ГИС среди информационных технологий	15.12.13 - 20.01.14г	Выполнено
5.	Геоинформационная система поиска и оптимизации маршрутов	15.12.13 - 20.01.14г	Выполнено
6.	Используемые средства разработки	21.01.14 - 22.02.14г	Выполнено
7	Программная реализация алгоритма на графах	31.01.14 - 23.03.14г	Выполнено
8.	Представление графов в информационных системах	12.03.14 - 29.03.14г	Выполнено
9.	Составление матрицы смежности	12.03.14 - 29.03.14г	Выполнено
10.	Программная реализация	31.01.14 - 20.04.14г	Выполнено
11.	Определение объема трудоемкости разработки	21.04.14 - 20.05.14г	Выполнено
12.	Расчет цены программного продукта	21.04.14 - 20.05.14г	Выполнено
13.	Анализ условий труда оператора	1.04 - 23.05.14г.	Выполнено
14.	Расчет естественного освещения	1.04. - 23.05.14г	Выполнено
15.	Заключение	13.05. - 02.06.14г	Выполнено

Дата выдачи задания «24» сентября 2013 г.

Заведующий кафедрой _____ _____
(подпись) (Фамилия и инициалы)

Руководитель _____ Чежимбаева К. С.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Имангалиева Э. Д.
(подпись) (Фамилия и инициалы)

Аннотация

В данном дипломном проекте была разработана программа, оптимизирующая транспортную маршрутизацию с помощью геоинформационных систем в городе Алматы Бостандыкского района

В работе был построен граф дорожной сети выбранного района, составлена таблица смежности полученного графа и рассчитан кратчайшее расстояние от одной вершины до другой. Также мною был спрограммирован алгоритм расчета кратчайшего пути и вывода его на карту

В экономической части были рассчитаны затраты на разработку программного продукта и посчитана прогнозируемая отпускная цена. В разделе безопасность жизнедеятельности были рассмотрены вопросы зрительной нагрузки оператора связи при работе в помещении с ЭВМ.

Аңдатпа

Бұл дипломдық жобада Алматы қаласы Бостандық ауданында транспорттық маршрутизацияны үйлестіретін бағдарлама құрылды.

Жобада таңдалған ауданның жолдық желі граф құрылып, алынған графтың шектік кестесі жасалып және бір шыңнан екінші шыңға дейінгі қысқа жол есептелді. Сонымен қатар, қысқа жолдың есептелу алгоритімі бағдарланды да картаға шығарылды.

Экономикалық бөлімінде бағдарламалық өнімнің өңделу шығыны мен жобаланған босату бағасы есептелді. Ал, өмір тіршілік қауіпсіздігі бөлігінде ЭЕМ бар бөлмедегі оператордың көздік ауырлық сұрақтары қарастырылды.

Содержание

Введение

Вопросы построения подходящих маршрутов для транспортных средств исследуются и формируются в течении последних десятков лет. Вводятся ранее не известные постановки проблем, улучшаются методы решения. Решение этих проблем дает возможность осуществлять расчет маршрутов транспортных средств подобным способом, для того чтобы гарантировать транспортную маршрутизацию с наименьшими расходами материальных ресурсов. В этой дипломной работе подобные оптимизационные вопросы находят решение с поддержкой геоинформационных систем.

Основная концепция геоинформационных систем (ГИС) - взаимосвязанность сведений в базе данных и на карте. ГИС - это и аналитические ресурсы для работы с каждыми координатно-привязанными данными. В принципе, ГИС можно анализировать точно как некую развитую концепцию баз данных. В данном значении ГИС практически предполагает из себя другую степень и метод интеграции и структурирования данных. ГИС дает возможность абсолютно другого развития и формирования картографии. Обходятся стороной ключевые минусы обыкновенных карт - их неподвижность и ограниченная вместимость в качестве носителя информации.

ГИС считаются классом информативных режимов, обладающих собственными характерными чертами. Они выстроены с учетом закономерностей геоинформатики и способов, используемых в данной науке. Являясь интегрированными и информативными системами, ГИС предусмотрены с целью решения разных проблем науки и использования на базе применения пространственно - локализованных сведений о предметах и явлениях окружающей среды.

Одним из расширенных функциональностей современной общедоступной ГИС являются возможности разрешения определенного набора задач на графах коммуникации и дорожной сети. В основном такое требование выдвигается пользователями, чья работа напрямую связана с доставкой груза и предоставлением транспортных услуг. Эта работа посвящена рассмотрению возможностей использования ГИС модулей в этой области.

Представление дорожно-транспортной сети в виде графа с использованием картографических данных подразумевает следующие задач при решении проблем транспортной маршрутизации:

- провести вычислительный эксперимент, получив результаты, соответствующие реальным условиям;
- учесть при решении правила дорожного движения;
- в качестве веса ребра задавать физическую длину участка дороги либо среднюю скорость на участке дороги;
- учесть информацию о дорожной обстановке;
- учитывать закрытие участков дорог при построении графа;

- оценить время выполнения тех или иных решений данных алгоритмов и методов;

- провести эксперименты с распараллеливанием задачи на несколько ЭВМ и т.д.

В решение всех обозначенных задач входят следующие подзадачи; построение матрицы смежности с заданием весовых коэффициентов ребер, нахождение маршрутов в графе между заданными пунктами назначения с минимальной суммой весов ребер, входящих в маршруты.

Хотя решения отдельных задач и подзадач достаточно хорошо изучены, не учитываются многие факторы, вследствие чего невозможно их применение в реальных условиях напрямую.

До недавнего времени разработкой картографии занимались только организации, специализирующиеся в данной области. Соответственно, картографические данные были закрыты для широкого круга потребителей и предоставлялись только на коммерческой основе.

1 Введение в ГИС

1.1 Основная идея ГИС

Географическая информационная система - программно-аппаратная совокупность, предназначенная для управления, сбора, рассмотрения и отражения пространственно рассредоточенной информации. ГИС - является не только информационной системой для географии, а информационной системой с географически созданной информацией. В простейшем виде ГИС-комбинация обыкновенных сведений (базисной информации) с электронными картами, с крупными графическими редакторами.

В последнее время бумажные носители информации из-за перегруженности данными становятся нечитабельными. ГИС гарантирует регулирование визуализацией данными. Возникает вероятность извлекать (на экран, на твердую копию) только лишь область или предметы к ним относящиеся, которые интересуют нас в определенный период. Фактически осуществляется переключение с непростых общих карт к комплексу взаимосвязанных отдельных карт. Это помогает улучшить организованность информационных данных, что соответственно повышает эффективность и результативность анализа и обработки. Карта в ГИС оживает и действительно является динамическим объектом, то есть включает в себя:

- изменяемость масштаба карты;
- преобразование картографических проекций;
- пользованием объектным составом карты;
- возможность опрашивать посредством карты в режиме реального времени многие различные базы данных;
- изменения метода отражения объектов (цвет, направление линии и т.п.), также и определение символики посредством значимости атрибутов, то есть синхронизации с изменениями в базе данных;
- легкость внесения различных изменений.

Неотъемлемую часть в ГИС занимают геоинформационные технологии. Это комплекс программно-технических средств направленных на получение все новейших видов информационных данных об окружающем нас мире. Геоинформационные технологии нужны для улучшения эффективности: управленческих процессов, представления и хранения информации, поддержки и обработки принятия решений.

ГИС обладает рядом характерных для него особенностей, которые нужно принимать во внимание при исследовании таких систем. Одним из свойств геоинформационных технологий и ГИС является то, что они являются компонентами информатизации общества. Это проявляется во внедрении геоинформационных технологий и ГИС в производство, науку, образование и использовании в практической деятельности получаемых данных об

окружающей реальности. ГИС как концепция содержит множество взаимосвязанных компонентов, которые связаны прямо или косвенно с любым другим элементом, а две любые области этого элемента не могут являться независимыми, не нарушая единство и целостность системы. Геоинформационные технологии считаются новыми технологиями в сфере информационных систем, направленные на свершение разных целей, в том числе информатизацию производственных и административных действий. Другое отличие ГИС это то, что они являются итогом эволюции информационных систем и соответственно содержат в себе базы функционирования и построения информационной системы.

АИС - это автоматизированная информационная система, которую также называют организационно-технической системой, которая пользуется автоматизированными информационными технологиями для информационно-аналитического обеспечения инженерных работ и процесса управления. В согласовании со сведениями значению ГИС соответствует комплекс различных информационных автоматизированных систем.

Одной из особенностей ГИС является принадлежность ее к интегрированной информационной системе. Интегрированные информационные системы работают на принципах интеграции технологий разных систем. Они чаще всего используются в таких разных областях, что даже название не устанавливает все их возможности и функции. Следовательно объединять ГИС с решением задач только географии или геодезии не стоит. "ГЕО" в названии геоинформационных систем и технологий обозначает объект исследования, а не предметные области, в которых могут использоваться эти системы. Требуется рассмотреть место ГИС в различных других автоматизированных системах. Требуется кратко классифицировать эти системы. Рассматривая различные аспекты автоматизированных информационных систем можно предоставить различные их классификации.

По принадлежности к определенной научной области можно разделить информационные системы на три подкласса: экономические, технические и информационно-аналитические. Экономические системы состоят из автоматизированной системы управления (АСУ), бухгалтерских информационных систем (БУ-ИС), банковских информационных систем (БИС), биржевых информационных систем (БИС), маркетинговых информационных систем (МИС) и др. К техническим относится автоматизированная система научного исследования (АСНИ), система автоматизированных проектирований (САПР), гибкая производственная система (ГПС), робототехнический комплекс (РТК) и др. Информационно-аналитическая автоматизированная система состоит из: автоматизированной справочно-информационной системы (АСИС), базы данных (БД), экспертных систем (ЭС), статистических информационных систем (СТИС) и т.п.

Как интегрированной системы, особенностями ГИС является то, что она интегрирует технологии всех вышеперечисленных классов систем:

экономических, технических и информационно-аналитических. Соответственно, ГИС могут быть использованы как любые из этих трех систем.

1.2 Организация данных в ГИС

Предметные данные хранятся в ГИС в качестве таблиц, следовательно трудностей с их хранением и организацией в базах данных не возникает. Больше проблем предполагает хранение и визуализация пространственно - графических данных. Базой визуального интерпретирования данных с помощью ГИС-технологий являются специальные графические модели. Они подразумевают векторные и растровые виды. В итоге модели координатно - пространственных данных имеют или векторный, или растровый вид, то есть включать или не включать топологические свойства. Этот подход дает возможность классифицировать модели по трем видам: растровые модели; векторные не топологические модели; векторные топологические модели. Все модели между собой связаны и взаимно преобразуемы. Но, при использовании каждой из них необходимо брать во внимание их особенности. В ГИС системе к представлению пространственно - координатных данных относятся два базовых подвида модели - векторная и растровая. Также бывает класс моделей, содержащие характеристики, как векторов, так и мозаик. Их называют гибридными моделями. Между векторными и растровыми моделями имеется отличие, относящееся именно для ГИС. Растровые модели показывают поля данных, иными словами обладают полевыми характеристиками. Векторные модели в ГИС, как заведено, показывают геоинформационные объекты, то есть носят объектный характер.

Если векторное изображение имеет информацию о том, где расположен определенный объект, то растровое изображение - информацию о том, что расположено в определенной точке территории. Это показывает главную функцию растровой модели - непрерывное отражение поверхностей.

1.3 Растровая модель

В растровых моделях дискретизация выполняется наиболее обычным способом - весь предмет (исследуемая территория) отражается в пространственные ячейки, формирующую постоянную сеть. Тем не менее любой ячейке растровых моделей подходит одинаковые по размерам, но разные по характеристикам (плотность, насыщенность) участок поверхности предмета. В ячейке модели находится это значение, которое усредняет характеристики участков поверхностей предметов. В концепции обрабатывания изображений этот процесс известен как пикселизация.

Главное предназначение растровых моделей - непрерывное отражение поверхности. В растровых моделях в роли атомарной модели подходит двухмерные элементы пространства - пиксели (ячейки). Упорядоченный комплекс атомарных моделей формирует растр, который является моделью геообьекта или карты. Растровые модели дают возможность показывать цветовые оттенки или полутона. Для растровых моделей есть ряд особенностей: ориентация, разрешение, место нахождения, значение, зоны. Разрешение - наименьший линейный размер самого маленького участка используемого пространства (поверхности), отражаемый одним пикселем. Пиксели по своему обыкновению, представляют собой квадраты и прямоугольники, реже используются треугольники и многоугольники. Более высоким разрешением владеет растр с минимальным размером ячейки. Высокое разрешение обозначает множество деталей, минимальный размер множества ячеек. Значение - часть информации, которая хранится в элементах растров (пикселей) или в легенде. Место нахождения определяется упорядоченными парами координат (номера строки и номера столбца), однозначно характеризующими положение любого компонента отражаемого пространства в растре.

Растровая модель обладает следующими достоинствами, которые представлены в таблице 1.1.

Т а б л и ц а 1.1 – Достоинства растровой модели

№ п/п	Достоинства растровой модели
1	Растровая модель не требует предварительных знакомств с явлениями, информация собирается с равномерно расположенных точек, что дает возможность в последующем на базе статистического метода анализа получать объективную характеристику исследуемого объекта. Благодаря этой особенности растровая модель может быть использована для изучения нового явления, о котором не собран материал. Будучи простым, этот метод получил самое большое продвижение;
2	Растровые сведения легче для анализа по параллельному алгоритму и тем самым обеспечивает вероятность более высокого быстродействия по сравнению с векторными сведениями;
3	Некоторые проблемы, к примеру формирование буферных зон, намного легче решать в растровых видах;
4	Большинство растровых моделей дают возможность вводить векторную информацию, в то время как обратные процедуры много затруднительны для векторной модели;
5	Процесс растеризации намного легче и алгоритмически, нежели процесс векторизации, который чаще всего требует экспертного решения.

Делая сопоставление растровых и векторных моделей, подчеркнем удобство векторных с целью работы со связанными объектами. Несмотря на это, используя простые приемы, к примеру, включая взаимосвязанность в таблицу атрибутов, возможна организованность взаимосвязи и в растровых

форматах. Нужно отметить вопросы точности отражения в растровой модели. В таких моделях во многих случаях не понятно, принадлежат ли координаты к точке пикселя, которая является наиболее центральной либо к одному из ее углов. Следовательно корректность привязки компонента растра характеризуют точно как половину высоты и ширины ячейки.

1.4 Векторная модель

Векторная модель создается на векторах, занимающих долю пространства в отличии от занимающего все пространство растровой модели. Это устанавливает их главные преимущества - требования меньшего объема памяти для хранения информации и меньшего расхода времени на представление и обработку, а наиболее важное - высокие точности позиционирований и представления информационных данных. При построении векторной модели объект создается с помощью соединения точек прямой линией, дугой окружности, полилинией. Векторная модель применяется в основном в транспортной, коммунальной, маркетинговой ГИС.

Векторная модель (объекта) применяет в качестве элементарных моделей последовательности координат, формирующих линию. Линией именуют границы, сегменты, цепи или дуги. Основной тип координатно представленных данных в комплексе векторной модели обуславливается через основные элементы. Точки определяются как выродившиеся линии нулевых длин, линии - как линии конечных длин, а площадь интерпретируется последовательностями связанных между собою линий. Любой участок линии должен быть границей для двух пересечений (узлов) и для двух ареалов. Отрезки общих границ между двух пересечений (узел) имеют разные наименования, являющихся синонимами в объектной области ГИС. Эксперты по теории графов предпочитают вместо слова линия термин ребро, а для обозначений пересечений используют определение вершина. В другой системе (geodraw, arcinfo) применяется определение дуга. В отличие от обычного вектора в геометрии дуга имеет свой набор атрибуты. Набор атрибутов дуги обозначает полигоны, находящиеся по обе стороны от нее. Относительно последовательного кодирования дуга это полигон, различают левые и правые полигоны. Определение дуги {цепь, ребро) это фундаментальное для векторной ГИС. Векторную модель получают различными методами. Одним из более распространенных является векторизация сканированного (растрового) изображения. Векторизация - процесс выделений векторного объекта с растровых изображений и использование его в векторных форматах. Для векторизации необходимым условием является высочайшее качество (отчетливая линия и контур) растрового образа. Для гарантирования требуемой четкости линии чаще всего требуется улучшать свойства изображений.

При векторизации вероятны различные ошибки. Их исправление происходит в два этапа:

- корректирование растровых изображений до его векторизации;
- корректирование векторных объектов.

Относительно точности векторных сведений, можно отметить преимущество векторной модели по сравнению с растровой, так векторная информация может быть закодирована с произвольными степенями точности, которые ограничиваются лишь возможностью методов внутренних представлений координат. Чаще всего для представлений векторной информации используется восьми или шестнадцати десятичных знаков.

1.5 ГИС среди информационных технологий

Главным вопросом человека, ранее не сталкивавшегося с географической информационной системой (ГИС), естественно будет, "а для чего мне эта необходимо?".

На первый взгляд довольно явным считается лишь использование ГИС в распечатках и подготовках карт и, возможно, в обработке космических снимков. Настоящий же диапазон применения ГИС намного шире, и для того, чтобы дать его оценку, нам нужно взглянуть на использование компьютеров в принципе, тогда роль ГИС сможет представляться значительно четче.

Информацией в нашем представлении следует именовать все, что может быть интерпретировано в виде цифр, изображений и букв. Тогда, все способы, системы, теории, ресурсы, тенденции, техники, приемы, направления и т.д., которые направлены на сбор, анализ и применение информационных данных, вместе именуются информационной технологией. А ГИС – одна из них.

Данная разработка объединяет классические процессы при взаимодействии с базами данных, таковыми, как запрос и статистическое исследование, с достоинствами полной визуализации и географического (пространственного) рассмотрения, которыми обеспечивает карта. Данные возможности различают ГИС от других информационных режимов.

Создание карт и географическое исследование не считаются чем-то совершенно новым. Но разработка ГИС дает новейший, наиболее подходящий современности, наиболее действенный, легкий в понимании и с быстрым доступом подход к рассмотрению вопросов и решению задач, важных для населения в общем, и определенной системой или группой систем, в том числе. Она автоматизирует операцию мониторинга и анализа.

В данный период ГИС - это многомиллионные индустрии, в которые вовлечены бесчисленное множество людей во всем мире. ГИС проходят в университетах, колледжах и школах. Определение решения, таких частных задач, как поиск оптимального маршрута между пунктами - экстремальный и

экскурссионный туризм, подбор оптимального расположения нового офиса, поиск здания по его адресу.

Таким образом, ГИС помогают везде, где используются пространственные данные и данные об объектах, находящихся в определенных местах пространства.

1.6 Составные части ГИС.

Работающие ГИС содержат в себе пять основных элементов: аппаратное средство, программное обеспечение, исполнители, способы и данные (таблица 1.2).

Т а б л и ц а 1.2 – Составные части ГИС

Аппаратное средство	Это ПК, на котором работает ГИС. В настоящий период ГИС работает на любых видах компьютерных систем, от централизованного сервера до отдельного компьютера и мобильного устройства
Программное обеспечение ГИС	ГИС содержат функции и инструменты, необходимые для анализа, хранения, и визуализации географических (пространственных) данных. основными частями программных продуктов считаются: инструменты для ввода и редактирования географической информацией; системы управления базой данных; инструмент, поддерживающий пространственные запросы, инструмент анализа и визуализации (отображения); графические пользовательские интерфейсы (gui или гип) для простого доступа к функциям и инструментам.
Данные	Это скорее всего наиболее важнейшая часть ГИС. Информация о пространственных положениях (географических данных) и связанных с ними табличных данных может быть собрана и подготавленна непосредственно пользователем, или приобретаться у дистрибьюторов на коммерческой или другой основе. В процессах регулирования пространственных данных, ГИС связывает их с различными источниками и типами данных, а также может пользоваться СУБД, которые применяются различными организациями с целью упорядочивания и поддержки находящихся в их распоряжении данных
Исполнители	Широкое использование технологии ГИС естественно невозможно без людей, работающие с программным продуктом и разрабатывающие план их использования при решении реальных проблем. Пользователем ГИС может быть не только технический специалист, которые разрабатывает и поддерживает систему, но и конечный пользователь, которому ГИС помогает решить текущие ежедневные дела и проблемы
Методы	Эффективность и успешность (в том числе экономическая) применений ГИС во основном зависят от правильно составленных планов и правил работы, которые формируются в соответствии со спецификой задач и работы любой организации

1.7 Как работает ГИС?

ГИС хранит данные о настоящем мире в виде комплекса предметных слоев, которые связаны между собой на базе географического местоположения. Данное простое, однако весьма гибкое отношение показало свою значимость при решении разных действительных задач: для наблюдения за передвижением транспортного средства, подробного отражения действительной обстановки и предполагаемых событий, прогнозирования всемирного циркулирования атмосферы.

Любая географическая сведения содержат информацию о пространственном нахождении, хоть привязка к географическим координатам, или ссылки на почтовые индексы, избирательные округи или округи переписи населения, адрес, наименование земельного или лесного участка, наименование дороги и т.п. При пользовании подобными ссылками с целью автоматического распознавания местонахождение или местоположений объекта (объектов) используется процесс, именуемый геокодированием. С помощью него возможно быстрое определение на карте местоположение интересующего вас объекта или явления, к примеру, по каким маршрутам быстрее и проще достичь нужных вам пунктов или домов.

Растровая форма оптимальна для работы с постоянными свойствами. Растровое изображение предполагает из себя комплект значений для единичных простых элементов (ячеек), оно как отсканированная карта или картинка. Обе модели обладают своими преимуществами и недостатками. Современные ГИС могут содействовать как с векторными, так и с растровыми формами.

Что касается построения и оптимизации маршрутов на имеющейся дорожной сети, можно сказать, что в большом городе это очень важная задача. Многочисленные маршрутов общественного транспорта нужно удержать в памяти и рассмотреть их просто нереально. Данная задача — непростая организационно, так как требует координации огромного числа правящих организаций. Она трудна кроме того и технически, потому что требует систематизации и рассмотрения и сбора огромного объема исходных сведений.

Ресурсы анализа сетей дают возможность создавать оптимальные маршруты на реальных улично-дорожных сетях с их возможностью и ограниченностью (допустимое направление движения, поворот, пропускная способность улиц и т. д.).

Базы данных маршрута пассажирских транспортов с обязательным географическим элементом — прекрасная база и для подготовки классических карт транспорта, и направленная на создание интерактивной информационной системы для населения.

Отметим, что средство анализа, имеющийся в ГИС, позволяет не только планировать маршрут по существующим улично-дорожным сетям, но и давать оценку эффективности этой всей сети, выявлять узкое место, спланировать

развитие. Практически в любых городах можно выявить пример, когда длина даже самых оптимальных маршрутов во десятки раз превышает геометрически кратчайшие расстояния между пунктом отправления и пунктом назначения. А на идеальных сетях превышение не бывает больше 40%. Причиной этого — низкие связности сети, обусловленные препятствиями, а еще неудачные организации движения. Результат всего этого — внушительный перепробег для всех участников дорожных движений: и общественных транспортов, и коммерческих, и личных.

Гис непосредственно связана с рядом других видов информационных систем. Ее главное отличие состоит в возможности управлять и исследовать пространственные данные. Хоть и не существует единой общепринятой систематизации информационных режимов, представленное ниже описание должно посодействовать умению различать ГИС от настольной картографической системы (desktop addmapping), системы САПР (cad), дистанционного зондирования (remote sensing), системы управления базой данных (субд или dbms) и технологией глобального позиционирования (gps).

Пожалуй, основной особенностью ГИС является наиболее "естественное" (для человека) представление как собственно пространственных данных, так и любых других данных, имеющих отношение к объектам, расположенным в пространстве. Способы интерпретирования атрибутивной информации разное: этим может быть и числовые значения с датчика, таблицы из базы данных (как локальной, так и удаленной) о характеристиках объекта, его фотографии, или реальные видеоизображения, наконец, звуковые записи.

2 Используемые системы

2.1 Геоинформационная система поиска и оптимизации маршрутов

Проблемы построения оптимальных маршрутов для транспортных средств исследуются и формируются на протяжении последних десятков лет. Включаются новые постановки проблем, изучаются методы решения. Решение данных проблем дает возможность производить расчет оптимальных маршрутов транспортных средств так, чтобы гарантировать объезд заданных пунктов назначения с минимальными затратами материальных ресурсов за требуемое время

Выделяют следующие наиболее известные задачи:

- TSP (The Travelling Salesman Problem) — классические задачи «коммивояжера»;
- VRP (The Vehicle Routing Problem) — проблема маршрутизации транспортных средств;
- VRPTW (The Vehicle Routing Problem with Time Windows) — проблема маршрутизации транспортного средства с временными разделениями;
- CVRP (Capacitated Vehicle Routing Problem) — задача маршрутизации транспортного средства с учетом грузоподъемности;
- SVRP (The Stochastic Vehicle Routing Problem) — вероятностная задача маршрутизации транспортного средства;
- DVRP (The Dynamic Vehicle Routing Problem) — динамическая задача маршрутизации транспортного средства;

Данные задачи принадлежат к классу NP – полных задач, Это такие задачи, решение которых полиномиально никак не находится в зависимости от входных сведений.

TSP имеет следующую постановку: дан направленный граф G = (V, E) и F - множества гамильтоновых циклов в графе G. Каждому ребру e є E задан вес c_e. Необходимо найти маршрут (гамильтонов цикл) в графе G минимальной суммой весов входящих в него ребер.

OpenStreetAddmap. С развитием открытых технологий появились некоммерческие геоинформационные системы и сервисы, позволяющие копирование, редактирование и использование по свободной лицензии. Одной из таких систем является OpenStreetAddmap (OSM) — бесплатная электронная карта мира, создаваемая и редактируемая сообществом пользователей, насчитывающим в настоящее время более полумиллиона человек. Одно из проведенных исследований показало, что точность картографических данных OpenStreetAddmap не уступает (а иногда и превосходит) таким электронным картам, такие как Bing Addmaps и Google Addmaps.

Таким образом, картографические данные OpenStreetAddmap, являясь открытыми и достаточно точными, могут быть взяты за основу для построения

модели дорожно-транспортной сети и проведения на ее основе исследований задач транспортной оптимизации.

Для работы с данными OpenStreetAddmap предоставляются различные редакторы и утилиты, схемы базы данных для СУБД MySQL и PostgreSQL. На интернет-ресурсах, посвященных OpenStreetAddmap, размещены ежедневно обновляемые экстракции картографических данных для городов и регионов [3] в формате XML. Данные, записанные в XML- формате, могут быть выгружены в локальную базу данных для их последующей обработки.

Схема базы данных составлена так, чтобы хранить определенные примитивы с описанием для последующей их отрисовки, и содержит 31 таблицу. К примитивам, представляющим интерес, относятся вершины (nodes), пути (ways) и отношения (relations).

Вершина (Node) — основной примитив OSM-схемы. Вершинам заданы географические координаты широты и долготы. Заданные отдельно вершины определяют раз-личные объекты, например, светофор или дорожный знак. Также вершины задают границы зданий и сооружений. Вершинам задаются атрибуты (tags) — множество пар ключ/значение, содержащих дополнительные характеристики вершины. Значения тегов перечислены в документации OSM. Примером тега вершины является ее описание, например, сувенирный магазин обозначен тегами shop/gift.

Путь (Way) — упорядоченное множество вершин, представляющее автомобильную дорогу, железнодорожную линию, реку, границы здания и т.д. Для путей задаются множество вершин и множество тегов. Теги путей хранятся в таком же формате, что и для вершин. Теги путей задают такие свойства, как тип дороги (автомагистраль, дороги общего пользования, проселочные дороги, дворовые проезды, пешеходные улицы и т.д.), одностороннее движение и другие.

Отношения задают различные ограничения для примитивов. Для отношений задается множество тегов и множество членов отношения. Теги задают такие значения, как тип отношения (type = restriction — тип отношения «ограничение»), ограничение (restriction = no_left_tum — ограничение «запрет поворота налево»). Членами отношений являются вершины и пути, т.е. отношение ограничения задает различного вида запреты, одним из которых может быть запрет поворота, где членами отношения будут являться два пути (откуда запрещен поворот и куда запрещен поворот) и вершина, через которую они связаны, содержащая данное ограничение.

Таким образом, целесообразно, основываясь на картографических данных OSM, разработать собственную модель дорожно-транспортной сети в виде графа, включив в него только необходимые для проведения расчетов данные и исключив всю избыточную информацию OSM, упростив тем самым последующую работу с моделью.

В модель дорожной сети в виде графа включается следующее:

- вершины графа — точки пересечения или примыкания дорог и улиц;
- ребра графа — отрезки дорог, соединяющие между собой вершины;

- длина ребра — физическая длина участка дороги, описываемого ребром;
- средняя скорость для ребра, выбираемая в зависимости от типа дороги, которую оно описывает (автомагистраль, дорога в городе, проселочная дорога и т.д.);
- ограничения на передвижение между ребрами, связанные с правилами дорожного движения (запрещающие знаки и т.д.);
- при наличии — информация о дорожной обстановке (статическая и/или динамическая).

2.2 Используемые средства разработки

Так как в ходе рассмотрения существующих аналогов для разрабатываемой системы, имеющей интеграцию с картами, пришли к выводу об использовании Google API, было принято решение о разработке веб-сервиса.

Для создания приложения был использован MVC Framework ASP.NET [6], который реализует шаблон проектирования model – view – controller [7].

Система MVC дает возможность разделить информационные данные, представление (View) и обработку действий на три компонента соответственно:
- модель, которая интерпретирует информационные данные и алгоритмы работы с ними,
- представление, отвечающее за отображение данных,
- контроллер, обеспечивающий взаимосвязь между системой и пользователем.

Для доступа к данным используется Entity Framework – объектно-ориентированная технология доступа к данным от Microsoft. Позволяет строить приложения, реализуя шаблон проектирования MVC и избегать прямой работы с базой данных в коде. Entity Framework автоматически сохраняет изменения в базе данных.

Для построения структуры базы данных и доступа к ней используется технология Code First платформы Entity Framework, позволяющая создать собственные классы модели, по которым Entity Framework автоматически генерирует базу данных.

Выбор в пользу MVC Framework ASP.NET был осуществлён из-за строгой типизации и наличия технологии Code First, которая освобождает от необходимости ручного ORM-преобразования.

2.3 Google Addmaps API

Для реализации поставленной цели используемые API карт должны удовлетворять следующим требованиям:

- обозначению мест на карте разными цветами для визуализации процесса заполнения системы данными, а также для отображения имеющихся данных.

- обработке пользовательских событий для обеспечения взаимодействия пользователя с системой, в том числе события нажатия на карту и наведения на маркер

- построению маршрута для отображения полученного решения задачи.

Было принято решение об использовании таких API, удовлетворяющих поставленным требованиям, как Google Addmaps [2].

Google Addmaps API.

2.3.1. Для обозначения мест на карте разными цветами используются маркеры. В API Google Addmaps для маркеров существует класс google.addmaps.Marker. Для добавления маркера событию используется такой метод как addMarker(). При создании маркера важны следующие параметры:

- position (обязательный параметр) – определяет начальное местоположение маркера,

- addmap – задаёт объект Addmap, на который помещается маркер.

Для удаления используется метод setAddmap(), которому передаётся null в качестве аргумента.

2.3.2 Для вычисления матрицы времени перемещения используется служба матрицы расстояний, которая вычисляет расстояние и время в пути между несколькими точками с учётом способа передвижения. Метод DistanceMatrixService.getDistanceMatrix() инициирует отправку запроса службе, передавая ей литеру объекта, содержащий исходные и конечные точки, способ перемещения, а еще способ обратного доступа, который следует выполнить когда получишь ответ. В результате успешной работы запроса можно получить расстояние и время для всех пар точек маршрута, указанных в запросе.

2.3.3. Чтобы обеспечить взаимодействие пользователя с системой, некоторые объекты в API Google Addmaps разработаны для ответа на события пользователя, такие как события мыши и клавиатуры.

2.3.4. Для построения маршрута используется служба маршрутов, которая позволяет получить такие данные о маршруте, как полилинии, текстовые описания маршрутов, а также рассчитать маршрут для разных способов передвижения. Метод DirectionsService.route() направляет запросы в служебный центр маршрутов, передавая ей литеру предмета DirectionsRequest, включающий способ ввода (начальную и конечную точки, способ перемещения и прочие) и способ обратного доступа для результата после получения ответа. Обратный доступ возвращает в ответе DirectionsResult, содержащий результат запроса маршрута и представляющий собой литеру объекта с одним полем: routes[], который включает в себя массив объектов

DirectionsRoute (точный результат для заданной исходной точки и назначенного пункта), и код DirectionsStatus, который возвращает статус запроса маршрута.

Доступ к службам матрицы времени и маршрутов осуществляется асинхронно, так как требуется отправить запрос на внешний сервер. В связи с этим необходимо передавать способ обратного доступа, выполненный по завершении запроса.

2.4 Подробности программной реализации

2.4.1 Общая архитектура системы. Система имеет клиент-серверную архитектуру, ее схема изображена на рисунке 2.1. В качестве клиента выступает набор страниц на html, css, JavaScript, а в качестве сервера – веб-сервер, реализованный на платформе asp.net mvc средствами языка c#. Сервер и клиент общаются между собой по протоколу HTTP.

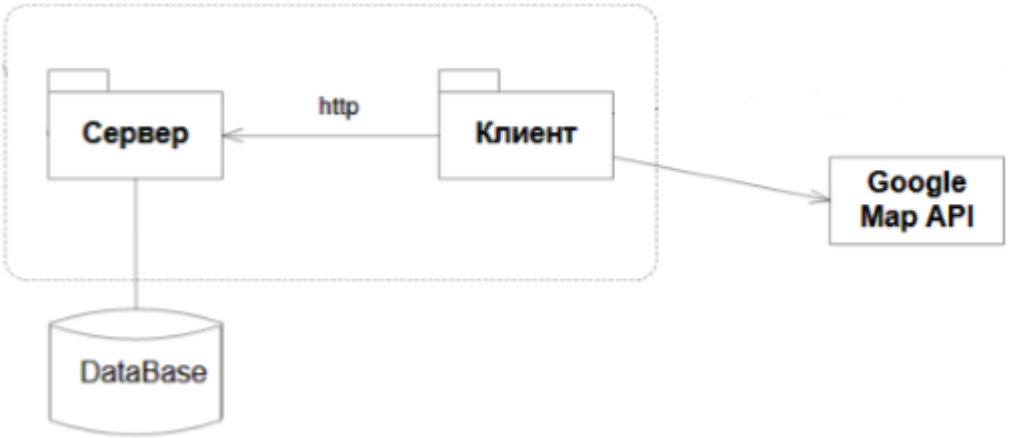


Рисунок 2.1 – Общая архитектура системы.

Сервер обеспечивает доступ к хранению информации, реализует бизнес-логику хранения и редактирования модели, а также обеспечивает инфраструктуру для наполнения веб-страниц и предоставления их клиенту.

Клиент в свою очередь отображает информацию, вычисляет маршрут по модели системы и обеспечивает доступ к внешним API карт, в частности API Google Addmaps.

2.4.2 Архитектура сервера. Модель. На рисунке 2.2 представлена архитектура сервера.

Так как для события необходимо иметь данные обо всех интервалах времени, было принято решение хранить временные интервалы в бинарном

виде и преобразовывать из последовательности битов TimeInterval в модель отображения и наоборот (2-ой способ).

Класс Event предназначен для хранения информации о событии согласно формальному условию задачи.

Класс Location предназначен для хранений географических координат события класса Event или объекта «Дом», который представляет собой начальную и конечную точки маршрута, класса Calendar. Имеет следующие атрибуты:

Latitude – широта,

Longitude – долгота.

Связь между классом Location и классом Event / Calendar осуществляется с помощью Entity Framework.

Класс TimeOfWork предназначен для хранения промежутков времени событий, которые базе данных представлены значениями типа byte[], и имеет атрибуты:

Start – начало временного интервала,

End – конец временного интервала.

Для последующей расширяемости приложения был введён класс AddmapAPI, который хранит информацию об API соответствующих карт, используя атрибуты Id и Name. С календарём связана коллекция событий, а также выбранное пользователем API карт (по идентификатору API). По умолчанию для нового пользователя это Google карты. В настоящее время используются API Google карт, но система расширяема и предусматривает подключение новых API карт.

Для всех элементов модели хранения заведены параллельные классы, которые поддерживают отображение модели на представление (view).

IEditModel – служит интерфейсом для работы с моделью хранения и имеет следующие методы:

ToModel() – сохранить данные,

FromModel() – загрузить данные.

Класс EventEdit имеет те же атрибуты, что и класс Event, но на них наложены некоторые ограничения: часть атрибутов (Id, Latitude, Longitude) не отображается из эстетических соображений и ввиду отсутствия необходимости, другие поля (Name, Duration, Mandatory) являются обязательными для заполнения. Также класс EventEdit реализует методы для работы с моделью.

Класс TimeOfWorkEdit предназначен для редактирования промежутков времени событий.

Класс CalendarEdit имеет те же атрибуты, что и класс Calendar, но так же, как и предыдущий, накладывает на них ограничения: не отображается идентификатор (Id), а имя (Name) является обязательным для заполнения.

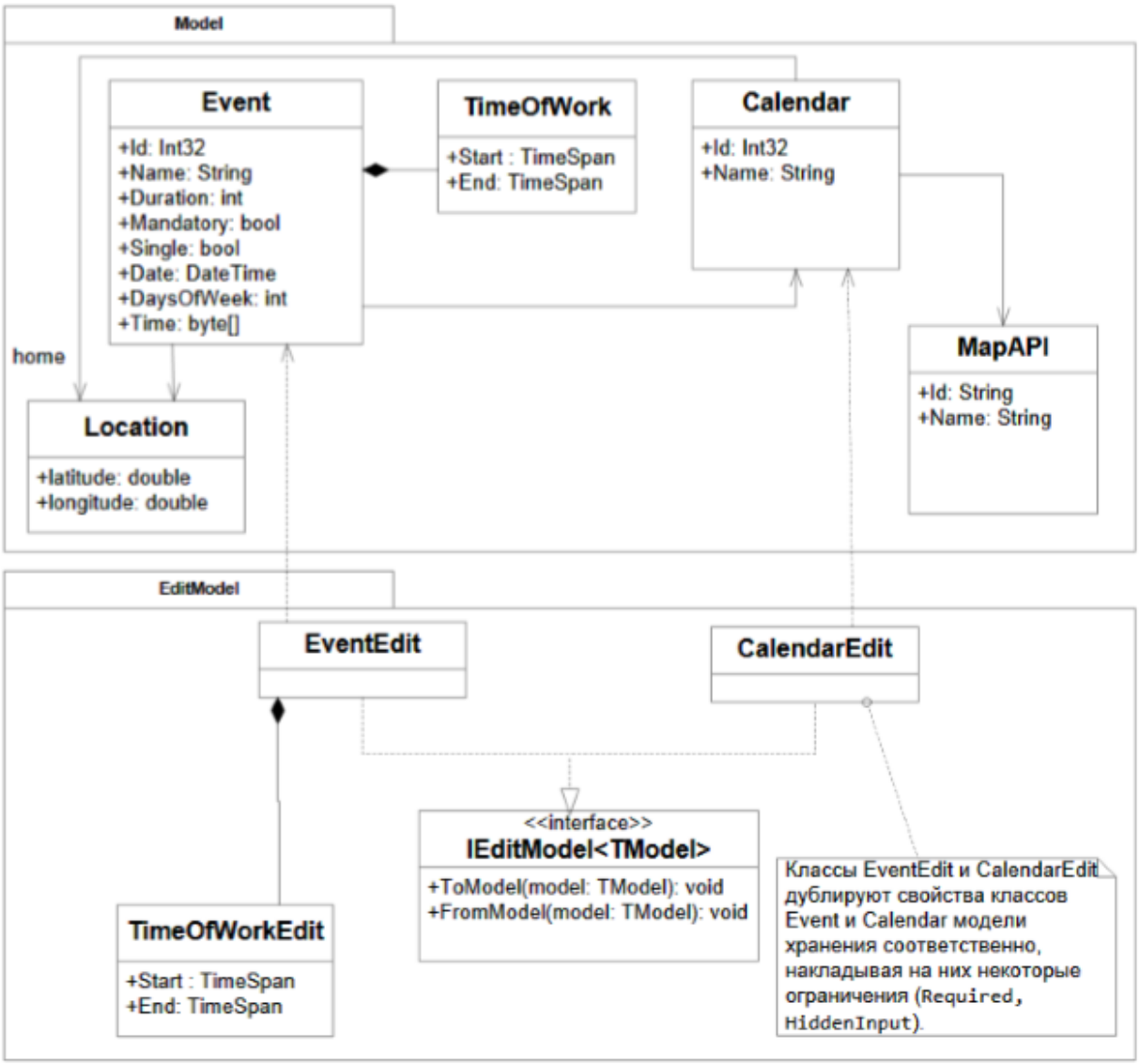


Рисунок 2.2– Архитектура сервера

Класс наследует методы для работы с моделью.

Контроллер.

Реализованы следующие контроллеры:

Контроллер BaseController предоставляет методы для получения и установки текущего календаря.

Также имеется обобщённый контроллер EditController, принимающий параметры:

TModel – модель хранения,

TEditModel – модель редактирования,

TIdentifier – тип идентификатора.

В EditController предусмотрены действия для добавления, редактирования, удаления и некоторые другие. Все Action-ы виртуальные, некоторые из них расписаны подробнее:

Details(TIdentifier id) принимает идентификатор, возвращает представление объекта из базы данных типа TModel, найденного по идентификатору,

Create() возвращает представление модели редактирования, поддерживает GET- версию для отображения формы создания и POST-версию для обработки пользовательского ввода.

Edit(TIdentifier id) принимает идентификатор, редактирует модель хранения и возвращает представление с новыми данными. Поддерживает GET-версию для отображения изменений модели и POST-версию для обработки пользовательского ввода.

Delete(TIdentifier id) принимает идентификатор и удаляет соответствующий объект модели хранения. Поддерживает GET -версию для отображения формы удаления и POST-версию для удаления объекта.

Контроллеры CalendarController, EventController и HomeController являются частными случаями EditController и призваны управлять календарями и событиями; перегружают часть методов EditController и, помимо этого, имеют следующие методы:

2.4.3 Архитектура клиента. На рисунке 2.3 представлена архитектура клиента.

Event – структура, дублирующая структуру Event в модели.

Problem – представляет собой коллекцию событий на день (Event) и две точки (Location): начальное и конечное положение пользователя (как уже было сказано, в пределах данной системы эти точки совпадают). Задача состоит в том, чтобы найти такую последовательность событий, когда результат направленной функции будет наилучшим.

Для каждого события на карте отображается маркер, связанный с координатами события. Маркер представляет собой небольшое изображение.

Для каждого действия есть своё представление. Стили прописаны в Site.css для всех представлений.

Также существует класс Solver, описывающий решение задачи. Возвращает отсортированный список событий со временем их посещения.

В рамках данного класса реализован метод solve(problem, optFunc), решающий задачу с использованием параметров, передаваемых в качестве аргументов: объект задачи (Problem) и целевая функция (OptFunction).

OptFunction – представляет собой абстракцию целевой функции и имеет следующие абстрактные методы, перегружаемые классами-наследниками:

push(event) – принимает в качестве аргумента событие и добавляет его в конец строящегося маршрута,

pop(event) – принимает в качестве аргумента событие и удаляет его из маршрута,

compare(value1, value2) – принимает два значения целевой функции и выбирает из них лучшее по некоторым критериям.

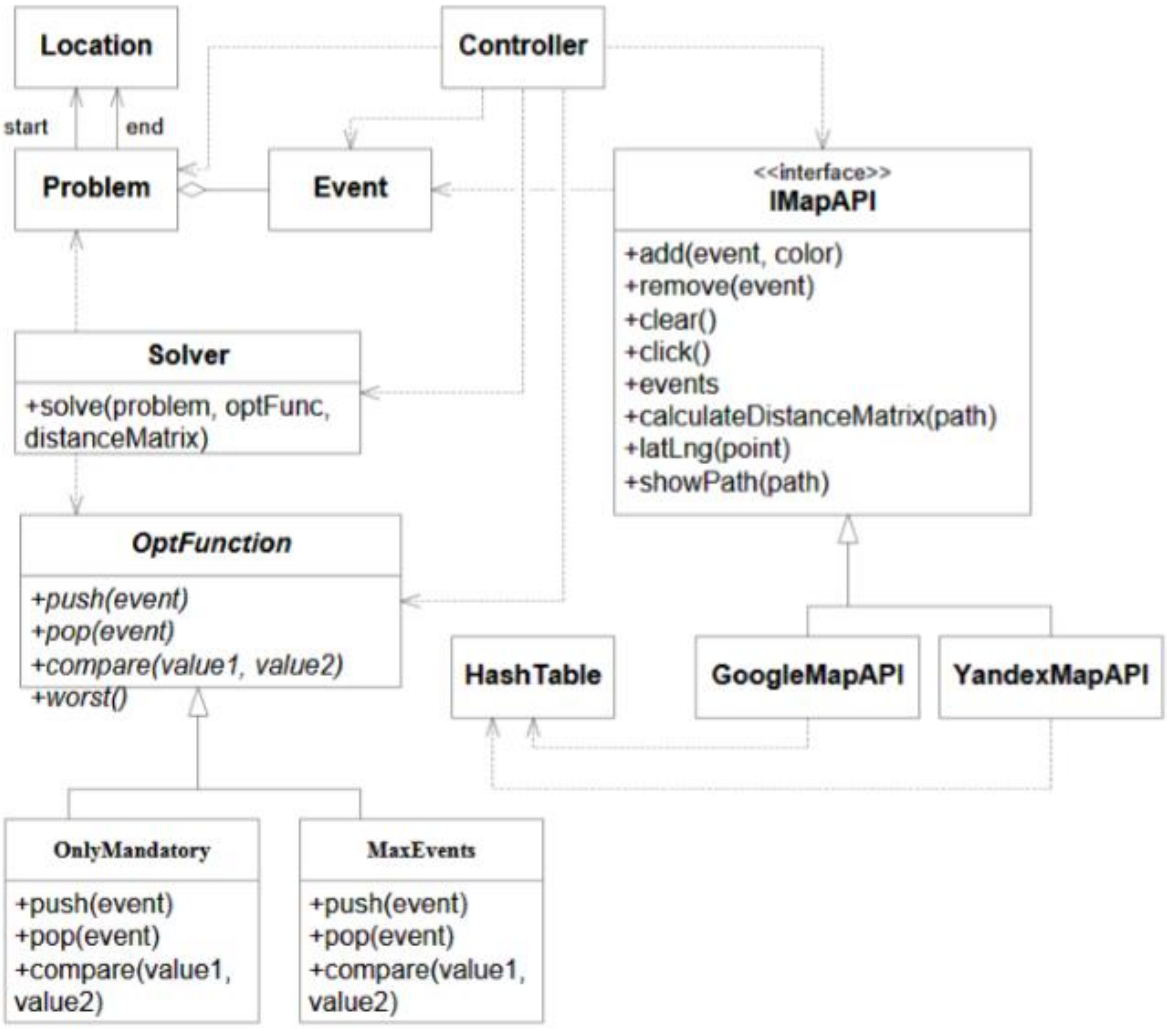


Рисунок 2.3 – Архитектура клиента.

OnlyMandatory – целевая функция для нахождения маршрута, состоящего только из обязательных событий.

MaxEvents – целевая функция для нахождения маршрута, состоящего из всех обязательных и максимального количества необязательных событий.

Поскольку изначально хэш-таблиц нет в JavaScript, для вспомогательных целей был реализован класс Hashtable, который используется для отображения маркеров в события и обратно.

На рисунке 2.4 представлена диаграмма последовательности для вычисления и отображения оптимального маршрута.

Для вычисления оптимального маршрута запрашивается матрица расстояний, которая вместе с объектом задачи и целевой функцией используется для решения задачи. Полученный маршрут отрисовывается на карте.

На рисунке 2.5 представлена диаграмма последовательности для операции поиска оптимальнейшего маршрута.

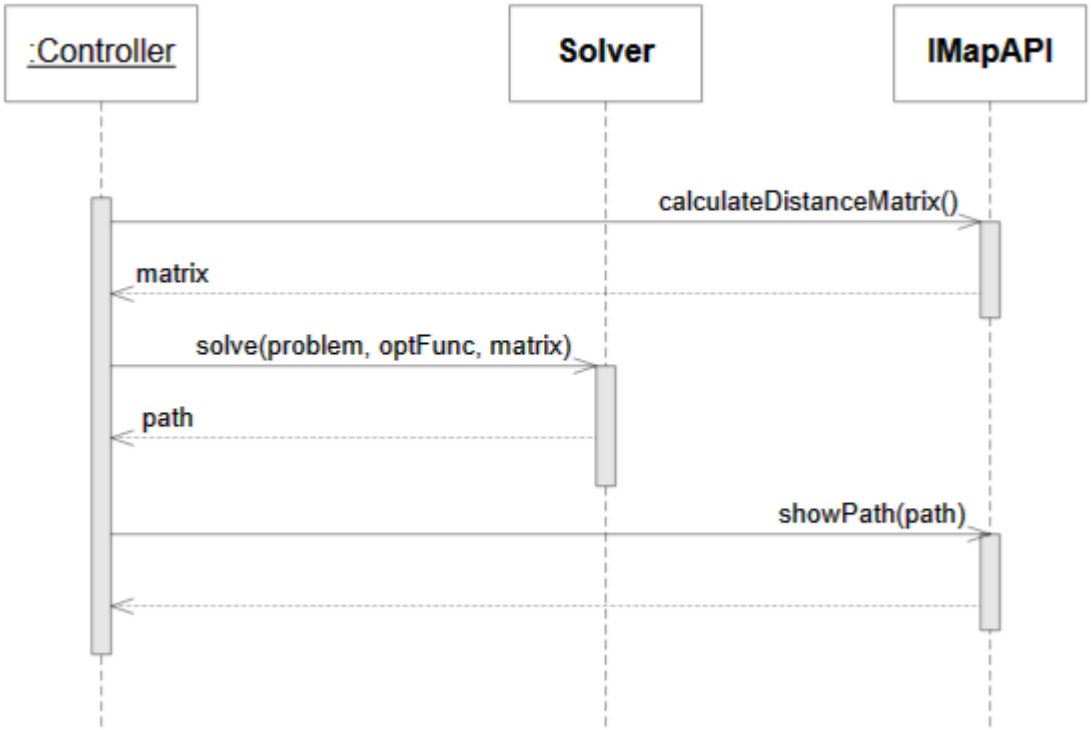


Рисунок 2.4 – Диаграмма последовательности для вычисления и отображения оптимального маршрута.

Для решения задачи последовательно генерируются перестановки. Некоторые из них отличаются между собой только в двух позициях, а большая часть перестановки одинакова для целевой функции, которая считается также для неполных маршрутов и для её уменьшения были реализованы методы Push() и Pop(), позволяющие не пересчитывать результат направленной функции для одинаковой части перестановок.

Для каждой перестановки, где k – длина перестановки, а n – количество событий, вычисляется результат направленной функции следующим образом:

- если $k = n$, при этом значение целевой функции является лучшим, данная перестановка считается текущим решением

- если $k \neq n$, то если событие ещё не использовалось, оно добавляется в перестановку.

При этом если данная перестановка имеет значение целевой функции не хуже, чем текущее решение, то осуществляется переход к следующему, иначе это событие удаляется из перестановки и проверяется другое.

Потом выбирается перестановка с лучшим значением целевой функции, которая и будет возвращена как решение

Для вычисления оптимального маршрута запрашивается матрица расстояний, которая вместе с объектом задачи и целевой функцией используется для решения задачи. Полученный маршрут отрисовывается на карте.

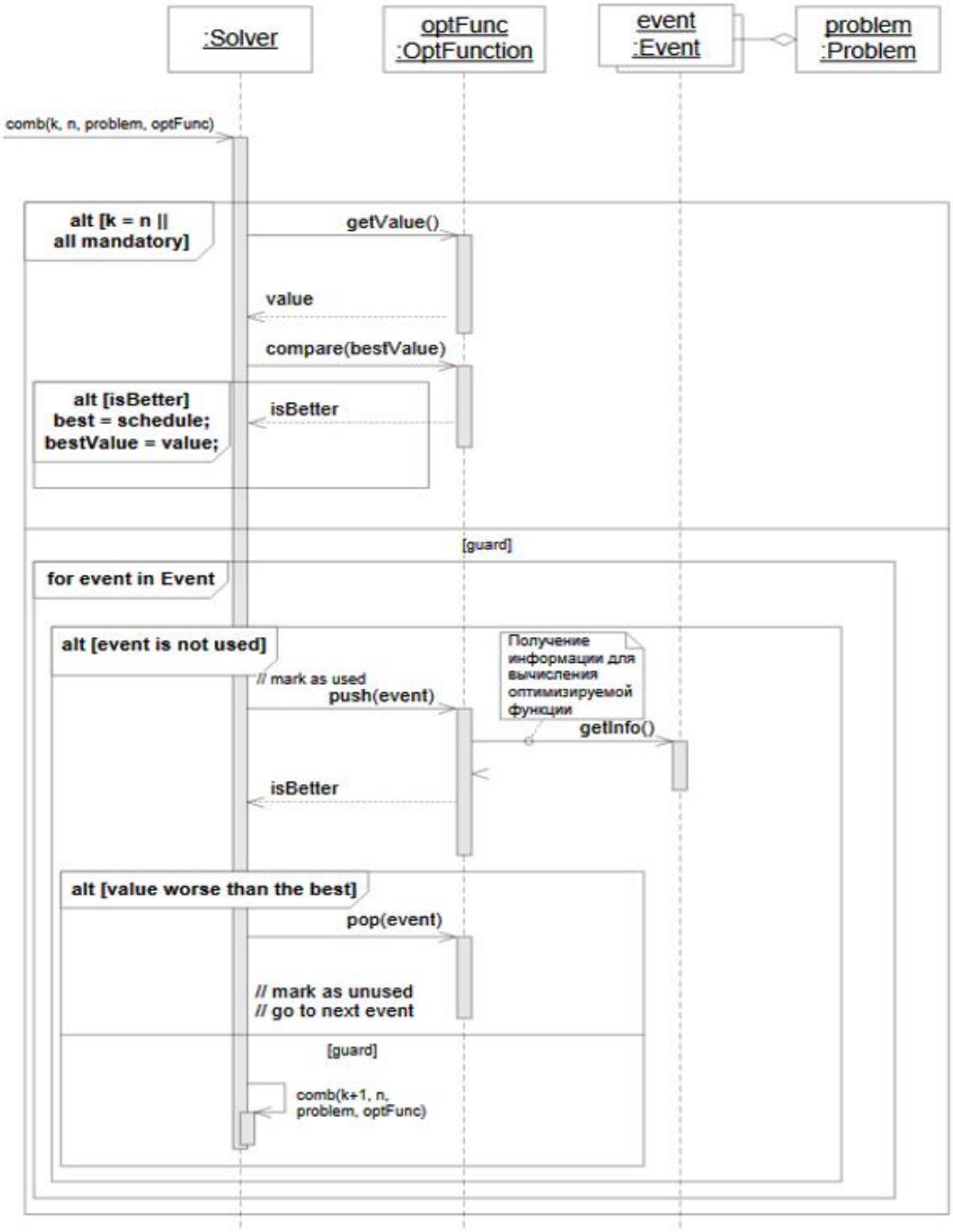


Рисунок 2.5 – Диаграмма последовательности для операции нахождения оптимального маршрута.

2.5 Программная реализация алгоритмов на графах

Работа с графом в ГИС ПИОМ реализована следующем образом. Для работы с вершинами графа (на нашем примере – это перекрестки в уличнодорожной сети) создан класс TCross. В классе инкапсулированы

основные функции для работы с перекрестками: добавление, удаление, изменение и другие. Собственно данные о перекрестках хранятся в переменной этого класса – массиве SourceData. Для работы с ребрами графа (на нашем примере – это участки улиц от перекрестка до перекрестка) создан класс TStreet. В классе инкапсулированы основные функции для работы с улицами: добавление, удаление, изменение и другие. Собственно данные об улицах хранятся в переменной этого класса – массиве SourceData. К этим данным помимо прочего относятся и ссылки на начальный и конечный перекрестки. Таким образом, можно легко перейти от улицы к ее начальному или конечному перекрестку, а вот для перехода от перекрестка к дорогам, которые с него начинаются необходимо полностью просмотреть массив SourceData. Для ускорения перехода от перекрестка к дорогам, которые с него начинаются, создана переменная StreetFromCross. Она представляет собой массив, количество элементов которого совпадает с количеством перекрестков, и каждый элемент StreetFromCross ассоциируется с одним конкретным перекрестом и представляет собой массив ссылок на дороги, начинающиеся с этого перекрестка. Алгоритмы поиска на графе можно описать следующим образом:

- стартовая вершина выбирается как активная;
- запускается цикл перебора всех вершин;
- определяется следующая активная вершина и запускается дальнейшая итерация цикла;
- цикл заканчивается в двух случаях: удачно (маршрут найден), если следующей активной вершина оказалась финальная; и при окончании перебора всех вершин (маршрут не существует). Разница в алгоритмах и методах поиска состоит лишь в правиле выбора следующей активной вершины.

В ГИС ПИОМ реализованы следующие алгоритмы поиска на графах:

- поиск в глубину (для определения существования между двумя пунктами маршрута);
- поиск в ширину (для определения маршрута, содержащего минимальное количество ребер графа);
- алгоритм Дейкстры (осуществляет поиск кратчайшего пути);
- обход препятствий (для скоростного поиска кратчайшего пути);

В методе поиска в глубину в качестве следующей активной вершины выбирается первая попавшаяся вершина, в которую ведет дуга из текущей активной вершины, если эта вершина еще не была добавлена в путь (просмотрена). Этот метод не накладывает никаких ограничений и условий на выбор следующей вершины (кроме того, что она должна быть еще не просмотренной). Это позволяет, пользуясь данным методом, быстро определить, существует ли путь из стартовой вершины в финишную. В оставшихся методах и алгоритмах активная вершина выбирается из всех вершин, являющихся соседними по отношению к уже просмотренным вершинам. Напомним, что вершина a является соседней вершине b, если существует путь из b в a.

В методе поиска в ширину в качестве следующей активной вершины выбирается вершина, расстояние до которой от стартовой вершины минимально. В данном случае под расстоянием понимается количество ребер, находящихся на пути между вершинами. Этот метод никак не принимает во внимание веса ребер – те коэффициенты, которые им сопоставлены. В данной работе в качестве весов используется длина дороги в метрах.

Алгоритм Дейкстры основан на методе поиска в ширину. Его особенность заключается в том, что в нем рассматриваются в качестве критерия выбора следующей активной вершины веса ребер. Иными словами можно сказать, что активной будет выбрана та вершина, расстояние в метрах до которой от стартовой будет минимальным. Расстояние в данном случае рассчитывается как сумма весов ребер, входящих в путь от одной вершины к дугой. Таким образом, данный алгоритм подходит для решения поставленной задачи поиска оптимального пути.

2.6 Основы теории графов

Графы считаются значимым компонентом математических моделей в наиболее различных сферах науки и практики. Они могут помочь четко представить взаимоотношения между объектами или событиями в непростых системах. Термин «граф» неоднозначен. Можно легко заметить, сопоставляя приводимые в разных книгах определения. Но во всех этих определениях имеется кое-что общее. В каждом случае граф состоит из двух множеств – множества вершин и множества ребер, причем для каждого ребра указана пара вершин, которые это ребро соединяет [9]. Графом G мы будем называть пару (V(G), E(G)), где V(G) – непустое конечное множество элементов, называемых вершинами, а E(G) – конечное множество неупорядоченных пар элементов из V(G), называемых ребрами [10]. Мы рассмотрим только конечные графы, у которых оба множества конечны. Вершины и ребра графа называются его элементами. Все вершины графа G будем обозначаются через VG, множество ребер – EG, число вершин – n(G), число ребер – m(G). Ориентированным графом (орграфом) называется пара G=(V, E), где V – конечное множество, E – все упорядоченные пары разных элементов из V [9]. Дальше термин «граф» будем использовать в смысле «обыкновенный граф», а исследуя другие типы графов, будем специально это анализировать. Для задания обыкновенного графа достаточно назвать его вершины и ребра, каждое ребро представлено парой вершин. Предположим, например, VG={a, b, c, d, e, f}, EG={(a,c), (a,f), (b,c), (c,d), (d,f)}. Таким образом дан граф G с n(G)=6, m(G)=5. Если граф не слишком большой, то нагляднее будет представить его с помощью рисунка, на котором вершины изображаются кружками или иными значками, а ребра – линиями, соединяющими вершины (рисунок 2.6).

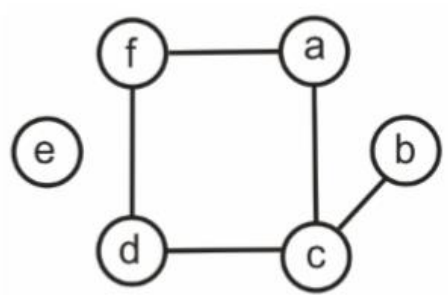


Рисунок 2.6 – Граф

Примеры графов легко найти в наиболее различных сферах науки и практики. Электрическая цепь, линии дорог, трубопроводов, структурный состав химических соединений, блок-схемы программы – в таких случаях графы появляются естественно и видны «невооруженным глазом». Немало возможностей для выявления графов и в математике. Наиболее наглядный пример – любой многогранник в трехмерном пространстве. Вершины и ребра многогранника можно рассматривать как вершины и ребра графа. Но тем не менее мы не обращаем внимание на расположение элементов многогранника в пространстве, используя лишь данные о том, какие вершины соединены ребрами [9] (рисунок 2.7).

В случае если в графе существеут ребро e=(a,b), то в таком случае говорят, что вершины a и b смежные в этом графе, ребро e инцидентно каждой из вершин a, b, а каждая из этих вершин инцидентна ребру e. Множество вершин графа, смежных с данной вершиной a, называются окрестностью этой вершины и обозначается через V(a). Количество вершин, смежных с вершиной a, именуется степенью вершины a и обозначается посредством deg(a).

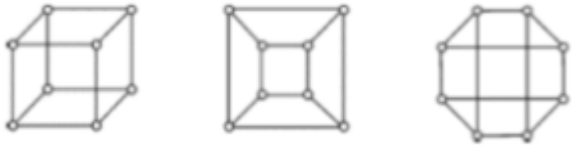


Рисунок 2.7 – Представления куба в виде графа

Вершину уровня 0 называют изолированной. Граф называют постоянным степени d, если степень каждой его вершины равна d.

2.6.1 Маршруты, пути, циклы, расстояния. Пути в графе – это последовательности вершин x1, x2… xn, такая, что для каждого i=1, 2… n−1 вершины xi и xi+1 соединены ребром. Эти n−1 ребер называются ребрами маршрута. Говорят, что маршрут проходит через эти ребра, а число n−1 называют длиной пути. Говорят, что, если маршрут соединяет вершины xi и xn,

то они соответственно именуются началом и концом маршрута, вершины x2…xn определяются как промежуточными. Маршрутом именуют замкнутым, в таком случае, если x1=xn. Путь – это маршрут, в котором все ребра различны. Путь называется простым, если и все вершины в нем различны. Цикл – это замкнутый путь. Цикл x1, x2… xn-1, x1 называется простым, если все вершины x1, x2… xn-1 попарно различны [9]. Установим некоторые простые свойства маршрутов. 1) В любом маршруте, соединяющем две различные вершины, содержится простой путь, соединяющий те же вершины. В каком – либо цикле, проходящем через некоторое ребро, содержится простой цикл, проходящий через это ребро. 2) Если в графе степень каждой вершины не меньше 2, то в нем есть цикл. Доказательство этих свойств приведены в [9]. Расстоянием между двумя вершинами графа называется минимальная длина пути, соединяющего эти вершины. Расстояние между вершинами a и b обозначается через d(a,b). Если же в графе нет пути, соединяющего a и b, тогда длина пути между ними считается бесконечным. Функция d(a,b) обладает следующими свойствами: 1) $d(a,b)\geq 0$, причем d(a,b)=0 тогда и только тогда, когда a=b; 2) d(a,b)=d(b,a); 3) $d(a,b)+d(b,c)\geq d(a,c)$ (неравенство треугольника). Расстояние от данной точки до наиболее удаленной от нее точки называется эксцентриситетом вершины a и обозначается через ecc(a). Таким образом, ecc(a)=max d(a,x). Вершину с наименьшим эксцентриситетом называют центральной, а вершину с наибольшим эксцентриситетом – периферийной.

2.7 Представление графов в информационных системах

При формировании разных информационных систем, сопряженных с картой автомобильной дороги, схемой маршрута перемещения, технической схемой устройства, организационной схемой управления, с решением задачи сетевого планирования непременно требуется иметь дело с графами. Графы дают возможность разрешить множество проблем: найти оптимальный маршрут на карте дорог; посчитать максимальное время, потраченное для выполнения проекта (критический путь); выяснить, сколько компонентов должно выйти из строя, для того чтобы отказал весь механизм и т. д. Граф всегда показывают как система точек на плоскости (вершин графа), соединенных линиями (ребрами графа). Несомненно, что это интерпретирование не является подходящим для решения задач с помощью ЭВМ. Предпочтение надлежащей структуры сведений для представления графов оказывает базисное влияние на результативность алгоритмов [11]. Проанализируем несколько способов представления графов. Для определенности отметим, что рассматривать будем только неориентированные графы (для орграфов рассуждения будут аналогичными). суждения будут аналогичными). Еще раз вспомним несколько определений из основ теории графов. Граф есть некоторое множество вершин и некоторое множество ребер,

соединяющих пары различных вершин [12]. Две вершины называются смежными, если есть ребро, соединяющее их. Такое ребро называется инцидентным данным вершинам. Для примера возьмем конкретный граф. Пусть он состоит из 5 вершин (пронумеруем их от 1 до 5), связанных 7 ребрами (обозначим их как « p1» ÷ «p7») (рисунок 2.8).

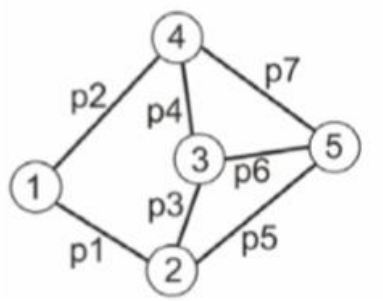


Рисунок 2.8 – Представление графа на плоскости

Рассмотрим различные варианты представления этого графа в информационных системах. В теории графов классическим способом представления служит матрица инциденций. Это матрица A с n строками, соответствующими вершинам, и m столбцами, соответствующими ребрам [11]. Ячейка матрицы содержит 1, если вершина и ребро графа, соответствующие строке и столбцу матрицы инцидентны (таблица 2.1).

Т а б л и ц а 2.1 – Представление графа в виде матрицы инциденций

№	p1	p2	я	p4	p5	p6	p7
1	1	1	0	0	0	0	0
2	1	0	1	0	1	0	0
3	0	0	1	1	0	1	0
4	0	1	0	1	0	0	1
5	0	0	0	0	1	1	1

Матрица инциденций является одним из самых неудачных способов представления графа в информационной системе. Для хранения такой матрицы (размерность n × m) необходимо достаточно много памяти, причем большой объем занят нулями. И ответы на простые вопросы – «Смежны ли вершины x и y?», «Какие вершины смежны с вершиной x?» – требуют, в худшем случае, перебора всех столбцов матрицы.

Лучшим способом представления графа является матрица смежности, определяемая как матрица B=[bij] размера n × n, где bij=1, если существует ребро, идущее из вершины i в вершину j [11] (таблица 2.2).

Преимущество матрицы смежности – это то, что за один шаг можно ответить на вопрос «Существует ли ребро, связывающее вершины x и y?». Недостаток в том, что независимо от числа ребер объем занимаемой памяти пропорционален n2.

Т а б л и ц а 2.2 – Представление графа в виде матрицы смежности

№	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	1
4	1	0	1	0	1
5	0	1	1	1	0

Если в графе нет петель (вершин, соединенных сами с собой), то на главной диагонали всегда будут нули. Матрица смежности является симметричной относительно главной диагонали [12].

Т а б л и ц а 2.3 – Представление графа в виде списка пар, соответствующих его ребрам

1	1	2	2	3	3	4
2	4	3	5	4	5	5

Более экономичным способом представления графа в памяти (особенно в случае неплотных графов) является представление его в виде списка пар, соответствующих ребрам (таблица 2.3). Объем памяти в этом случае пропорционален 2m. Недостатком является большое число шагов, необходимое для определения множества вершин, смежных с данной, и, как следствие, невысокая скорость работы различных алгоритмов поиска. Лучшим решением во многих случаях оказывается структура данных – список инцидентностей [11] – или список смежных вершин [12]. Для неориентированного графа каждое ребро фигурирует в 2 множествах: для каждого ребра x-y вершина x содержится во множестве, соответствующем вершине y и наоборот. Данный способ позволяет быстро решать поисковые задачи, на которые при использовании предыдущих требовалось бы гораздо большее количество шагов. Этот способ представления является одним из экономичных в отношении использования памяти.

У каждого из рассмотренных способов представления графов есть свои недостатки и свои преимущества. Использование того или иного способа зависит от назначения и задач проектируемой информационной системы.

2.8 Поиск на графах, кратчайшие пути

2.8.1 Поиск в глубину. Деятельность каждого алгоритма обхода заключается в поочередном посещении вершин и исследовании ребер. Какие конкретно действия производятся при посещении вершины и исследовании ребра – зависит от определенной задачи, для решения которой выполняется обход. В каждом случае, факт посещения вершины запоминается, так что с поры посещения и вплоть до конца работы алгоритма вершина считается посещенной. Вершину, которая еще не посещена, будем называть новой. В следствии посещения вершина становится открытой и остается такой, пока не будут исследованы все инцидентные ей ребра. После этого она превращается в закрытую [9]. Различают множество алгоритмов на графах, в основе которых лежит систематизированный выбор вершин, такой, что каждая вершина анализируется в точности один раз. Поэтому важной задачей является нахождение хороших методов поиска в графе. Метод поиска замечателен, если он позволяет алгоритму решения интересующей нас задачи легко использовать этот метод и каждое ребро графа анализируется не более одного раза (или количество таких исследований ограничено сверху) [11].

2.8.2 Поиск в ширину Поиск в ширину – это классический метод решения задачи нахождения кратчайшего пути между двумя конкретными вершинами графа. Кратчайший путь – это такой путь, соединяющий вершины графа, который обладает тем свойством, что никакой другой путь, соединяющий эти вершины, не содержит меньшее число ребер. Поиск в глубину мало пригоден для решения этой задачи, поскольку предлагаемый им порядок прохождения графа не имеет отношения к поиску кратчайших путей, а поиск в ширину предназначен как раз для достижения этой цели [12]. Итак, рассмотрим метод поиска в ширину. Отметим, что в рассмотренном в предыдущем разделе методе поиска в глубину чем позднее будет посещена вершина, тем раньше она будет использована – точнее, так будет при допущении, что вторая вершина посещается перед использованием первой. Это прямое следствие того факта, что просмотренные, но еще не использованные вершины скапливаются в стеке. Поиск в ширину, грубо говоря, основывается на замене стека очередью [11]. Концепция поиска в ширину заключается в том, чтобы проходить вершины в порядке их удаленности от определенной заранее указанной начальной вершины a. Другими словами, вначале посещается сама вершина a, потом все вершины, смежные с a, то есть находящиеся от нее на расстоянии 1, затем

вершины, находящиеся от а на расстоянии 2, и далее по удаленности [9]. Рассмотрим способ поиска в ширину с заданной начальной вершиной а. Во-первых, все вершины отмечаются как новые. Вначале посещается вершина "а". Она является единственной открытой вершиной. Во-вторых, каждый определенный шаг начинается с выявления какой-то открытой вершины "x". Эта вершина является активной. В-третьих, анализируются ребра, инцидентные активной вершине. Если это ребро связывает вершину "x" с новой вершиной "y", то вершина "y" посещается и становится открытой. После того, как все, инцидентные активной вершине, ребра проанализированы, она больше не является активной, а становится закрытой. Далее находится новая активная вершина, и проделанные действия повторяются. Процедура заканчивается, если множество открытых вершин становятся пустыми.

Основная отличительная черта поиска в ширину, различающая его от иных вариантов обхода графов, заключается в том, что в качестве активной вершины избирается та из открытых вершин, посещенная ранее других. Именно таким образом обеспечивается главная закономерность поиска в ширину: чем ближе расположена вершина к началу, тем быстрее она будет посещена. Для осуществления этого принципа избирания активной вершины проще применять, для хранения огромного количества открытых вершин, очередь – когда новая вершина становится открытой, она добавляется в конец очереди, а активная выбирается в ее начале

Кратчайшие пути в этом и в последующих разделах будем рассматривать ориентированные графы G=(V,E), дугам которых приписаны веса. Это значит, что каждой дуге (u,v) поставлено в соответствие некоторое вещественное число a(u,v), называемое весом данной дуги. Полагаем $a(u,v)=\infty$, если дуга (u,v) не существует. Под длиной пути будем понимать сумму весов дуг, из которых состоит путь. Длину кратчайшего пути будем обозначать d(s,t) и называть расстоянием от s до t. Если не существует ни одного пути из s в t, то полагаем $d(s,t)=\infty$. Введем еще одно ограничение. Путь веса дуг будут только положительными значениями. Так как при существовании дуг с отрицательными весами длина кратчайшего пути между парой вершин становится неопределенной, исходя из возможности неоднократного включения таких дуг в путь. Зная расстояние между парой вершин, можно легко определить кратчайший путь. Так как для двух произвольных вершин s и t ($s\neq t$) существует такая вершина v, что $d(s,t)=d(s,v)+d(v,t)$. Этим свойством обладает и предпоследняя вершина в кратчайшем пути от s к t. Следовательно, определяя таким образом предпоследнюю вершину, можно пройти от конца кратчайшего пути к его началу [11]. Алгоритм нахождения расстояния от источника до всех вершин, реализующий вышеизложенные действия в общем случае (при отсутствии контуров с отрицательной длиной) называется методом Форда-Беллмана [13].

2.8.3 Алгоритм Дейкстры поиска кратчайшего пути Алгоритм Дейкстры решает задачу нахождения кратчайших путей для единственного источника на

ориентированных графах, имеющих неотрицательные веса. Он основан на методе поиска в ширину. Работу алгоритма Дейкстры можно описать следующей последовательностью действий: 1) начинаем поиск путем помещения источника в просмотренную зону; 2) добавляем на каждом шаге одно ребро, дающее кратчайший путь из источника в вершину, не включенную в просмотренную зону; то есть вершины добавляются в просмотренную зону в порядке возрастания их удаленности от источника. Таким образом, чтобы найти кратчайший путь от s к t с помощью алгоритма Дейкстры, достаточно начать поиск с вершины s и закончить его, когда вершина t добавится в очередь [12].

2.8.4 Метод обхода препятствий A* Представленный метод обхода препятствий базируется на алгоритме Дейкстры. В англоязычной литературе он называется методом A* («а»-звездочка) [14, 15]. Метод реализуется в ряде предположений:

- карта разбита на квадратные части, именуемые в дальнейшем клетками (ячейками);
- любая клетка обладает несколькими характеристиками: • стоимость прохождения по этой клетке,
- предыдущая клетка – клетка, из которой пришли в данную клетку, • статус клетки (нерассмотренная, граничная, отброшенная), • анализ пройденного пути, • анализ оставшегося пути; 3) имеются две клетки – начальная и конечная;
- сосед клетки – клетка, в которую можно попасть из рассматриваемой клетки за 1 шаг.

Общее правило: в любой итерации с абсолютно всех граничных точек избирается та, для которой является минимальной сумма пройденного пути и пути до конца по прямой, и осуществляется дальнейшее продвижение. Алгоритм итерационный. Изобразим реализацию метода пошагово. Начальные данные: Start – изначальная клетка, Finish – конечная клетка. 1 шаг. Помечаем Start как граничную точку. 2 шаг. Среди всех граничных точек находим Клетку1 – клетку с минимальной суммой оценки пройденного пути g и оценки оставшегося пути h. 3 шаг. Для Клетки1 рассматриваем соседей. Если сосед имеет статус нерассмотренного, то мы обозначаем его как граничную клетку, и указываем Клетку1 как предыдущую для него. Оценку g для соседа принимаем равной g+p, где p-стоимость прохождения по клетке-соседу, а g – оценка пройденного пути для Клетки1. Оценка h для любой клетки равна длине кратчайшего пути (количеству ячеек на пути по прямой от рассматриваемой клетки до клетки Finish). Рассматриваемую Клетку1 помечаем как отброшенную. 4 шаг. Если на предыдущем шаге один из соседей оказался равен клетке Finish, то путь найден. Если ни одного нового соседа не существует, то нет и пути. 5 шаг. Переход на шаг 2. [14, 15].

3. Расчетная часть

3.1 Исходные данные

Известные данные для решения задачи это: дорожные сети города, место отправления и место назначения. Нужно выбрать оптимальные маршруты проезда от места отправления к месту назначения

Математические модели. Множества всех возможных трасс поездок по улице города представляется в виде ориентированного графа G= (A, W), где A – область вершин, W – область дуг. Вершине данного графа соответствуют: перекрестки на улицах города, место отправления $i \in A$ и место назначения $j \in A$. Вершина графа - это место дорожной сети, где имеется возможность выбора последующего маршрута поездки по городу. Ребру графа соответствует магистрали и улицы среди двух вершин. Для ребра графа задается матрица расстояний $L= |l_{sd}|$ и матрица возможных скоростей движения $C = |c_{sd}|$, $s, d \in A$.

Для заданной начальной и конечной вершины графа i и j нужно вычислить маршрут проезда R_{ij}, на которое тратится минимальное время, а также множества всех близких к оптимальному маршрутов, которые различаются от оптимального на определенную величину E [7, 8]. Вычисление множества подходящих к оптимальному маршруту дает возможность при окончательном выборе учесть дополнительные требования.

3.2 Построение графа дорожной сети города

Граф G (рисунок 3.1) задается множествами точек (вершин) $x_1, x_2,..., x_n$. (которое обозначается через X) и множеством линий (ребер) $a_1, a_2,...,a_m$. (которое обозначается символом A), соединяющих между собой все или часть этих точек. Таким образом, граф G полностью задается (и обозначается) парой (X, A). Если ребра из множества A ориентированы, что обычно показывается стрелкой, то они называются дугами, и граф с такими ребрами называется ориентированным графом.

К примеру, в случае если дорога содержит никак не 2-ух-, а узкое перемещение в таком случае направленность данного перемещения станет представлено стрелкой.

Если ребра никак не обладают ориентации, в таком случае эрл именуется неориентированным, (двухстороннее перемещение).

В ориентировочность графе классифицируется высокоупорядоченной двойкой, заключающейся с первой и окончательной верхов, её направленность подразумевается установленным с 1-ый верха к 2-ой.

Путем (или ориентированным маршрутом) ориентированного графа называется последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей.

Так, на рисунке 3.1 путями являются последовательности дуг:

a_6, a_5, a_9, a_8, a_4. (1)

a_1, a_6, a_5, a_9. (2)

a_1, a_6, a_5, a_9, a_{10}, a_6, a_4. (3)

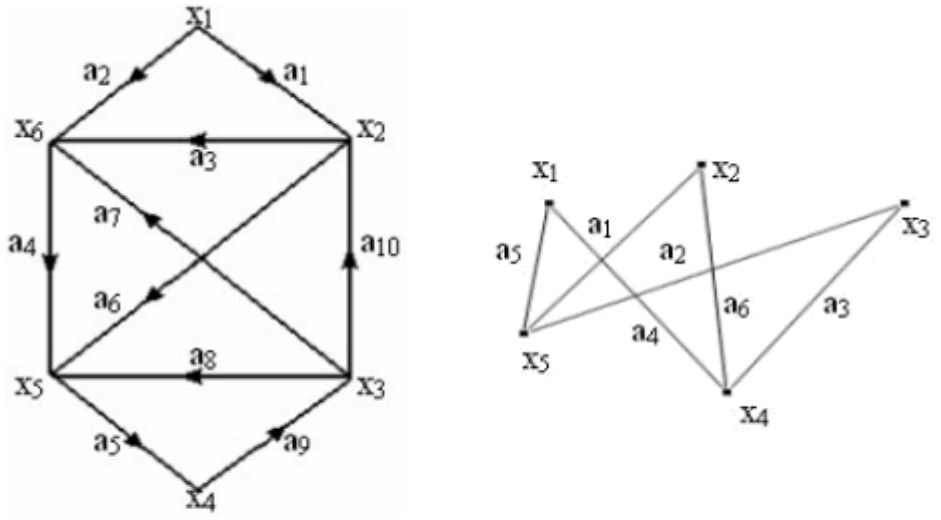


Рисунок 3.1 – Граф G

Ориентированная цепь (ор-цепью) такой путь, которым каждая арка используется не больше однажды, называют как сосредоточенная цепь. Свободное время ор-цепью называют таким путем, которым каждая вершина не используется больше однажды. Например, путь (2).

Маршрут не направлен "дважды" пути, и это понятие рассматривают, когда возможно пренебречь ориентацией арок в колонке. Таким образом маршрут - последовательность краев ä1, ä2... äq, в котором с каждым краем ай, за исключением первых и последних краев, сыграли вничью ai-1 и ai+1 края вершины трейлера.

В количестве, представленном на рис. 1.2, маршруты; два пункта по символу арки означают, что ее пренебрежение ориентации, т.е. арку рассматривают как ненаправленный край. Также путь или маршрут могут быть представленной последовательностью вершин. Например, путь (1) посмотрит после в пути: x2, x5, x4, x3, x5, x6. Иногда к аркам количества числа звонили в развес, стоимость, или длина этой арки приписана. В этом случае количество называют как количество со взвешенными арками. И если вес приписан вершинам количества тогда, количество со взвешенными вершинами оказывается. Если в колонке веса приписаны и аркам и вершинам, это называют просто взвешенным. При рассмотрении пути μ представленного

последовательностью дуг ($ä_1$, $ä_2$,..., $ä_q$), за его вес принимается число l(μ), равное сумме весов всех дуг, входящих в μ, т.е.

$$l(\mu) = \sum_{(Xi, Xj) e \mu} Cij \qquad (3.1)$$

Алгоритм Дейкстры. Алгоритм Дейкстры строит самые короткие способы провести от начальной вершины количества к другим вершинам этого количества (если те доступны).

В ходе работы алгоритма последовательно отмечаются продуманные вершины количества. И вершина, отмеченная в последний раз (в настоящее время), расположена ближе, чтобы подписать вершину, чем все не отмеченные, но далее, чем все отмеченные.

Сначала начальная вершина отмечена; следующий, очевидно, вершина, следующая за начальной буквой и смежная с ним, будет отмечена.

3.3 Составление матрицы смежности

Алгоритм решения задачи состоит из двух этапов. Определения кратчайшего по времени маршрута проезда R_{ij} сводится к решению известной задачи кратчайшего пути на ориентированном графе. Для ее решения применяются известный алгоритм [6], основанный на расстановке пометок на вершинах графа. Для определения множества всех близких к оптимальному маршрутов применяется алгоритм, основанный на методе последовательного анализа вариантов [9] и использовании правила отбраковки бесперспективных вариантов до получения окончательного решения.

Как известно, в алгоритме Дейкстры для поиска кратчайшего пути вершинам графа приписывают временные или постоянные пометки. Пометки определяют для вершины верхнюю границу длины пути оті вершины до текущей. Величины временных пометок вершин постепенно уменьшаются. Значение пометки определяет возможную длину пути от начальной до этой вершины. На каждом шаге алгоритма только одна из пометок с минимальным значением на рассматриваемом уровне выбирается в качестве постоянной. Это значит, что значение пометки является длиной кратчайшего пути из івершины в текущую вершину.

Введем следующие обозначения: V(s) – множество ребер, входящих или выходящих из вершины s (V(s) ⊆ W); |V(s)| – мощность множества; p_{sj} – пометки вершины s ∈ A,j= 0, 2, …,|V(s)|-1; B– множество вершин с постоянными пометками для j = 0.

Задача поиска кратчайшего пути на графе может быть определена для неориентированного, ориентированного или смешанного графа. Далее будет

рассмотрена постановка задачи в самом простом виде для неориентированного графа. Для смешанного и ориентированного графа дополнительно должны учитываться направления ребер.

Граф представляет собой совокупность непустого множества вершин и ребер (наборов пар вершин). Две вершины на графе смежны, если они соединяются общим ребром. Путь в неориентированном графе представляет собой последовательность вершин, таких что смежно. Такой путь называется путем длиной из вершины в (указывает на номер вершины пути и не имеет никакого отношения к нумерации вершин на графе).

Алгоритм первых этапов нахождения оптимальных маршрутов состоят из нескольких шагов:

Шаг 1. Присваиваем пометке изначальной вершины $p_{i0} = 0$ и считать постоянной. Для всех остальных вершин $s \in A\backslash\{i\}$ установить $p_{s0} = \infty$ и считать эти пометки временными. Устанавливаем $d = i$; $B = \{i\}$. Обновляем метки.

Шаг 2. Для любой вершины $s \in V(d)$ вычисляем новые значения

$$p_{sj} = p_{dj} + t_{ds}, \; j = 0, 2, \ldots, |V(s)|-1 \qquad (3.2)$$

Получается, для каждой вершины будем определять время проезда для всех возможных путей от исходных вершины i до вершин s.

Для $j = 0$ временные пометки вычисляем, используя выражение

$$p_{s0} = \min[\, p_{s0}, (p_{d0} + t_{ds})] \qquad (3.3)$$

и превращаем эти пометки в постоянные.

Шаг 3. Среди любых вершин с временной пометкой $s \in A\backslash B$ находим такую вершину k, для которой значение пометки минимально $p_{k0} = \min p_{s0}$.

Шаг 4. Считаем пометку p_{k0} постоянной и устанавливаем $d = k$. В множества B добавляем вершину k.

Шаг 5. Если $d^1 j$, то переходим к шагу 2. Если $d = j$, то p_{k0} автоматически становится длиной кратчайшего пути R_{ij} из вершин i в вершину j.

Шаг 6. Если все пометки всех вершин постоянные, т.е. B= A, то на этом определение оптимального пути завершается.

После этого делается восстановление маршрута проезда.

В случае дальнейшего продолжения получения вариантов путей из таких значений пометок, их значения будут только возрастать.

Карту улиц Бостандыкского района представляем в виде полного взвешенного неориентированного графа, где перекрестки – это вершины графа, а улицы между ними – это ребра их соединяющие. Схема графа представлена в приложении А.

Для оптимизации нахождения кратчайшего пути составим матрицу смежности заданного графа, где за вес ребра возьмем длину участка дороги от

одного перекрестка до другого. Матрица смежности приведена в Приложении Б.

3.4 Расчет кратчайшего пути пошагово

Допустим, найдем схематически кратчайшее расстояние от вершины E5 до вершины E40.

Шаг 1(Рисунок 3.2)

E5-E10 = 310

E5-E7 = 795

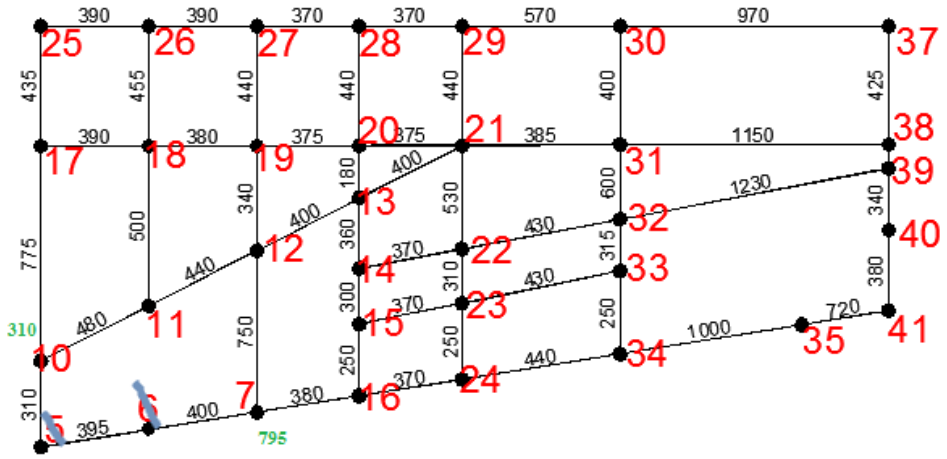


Рисунок 3.2 – Шаг 1

Шаг 2(Рисунок 3.3)

E5-E10-E17 = 1085

E5-E10-E11 = 790

E5-E6-E7-E12 = 1545

E5-E6-E7-E16 = 1175

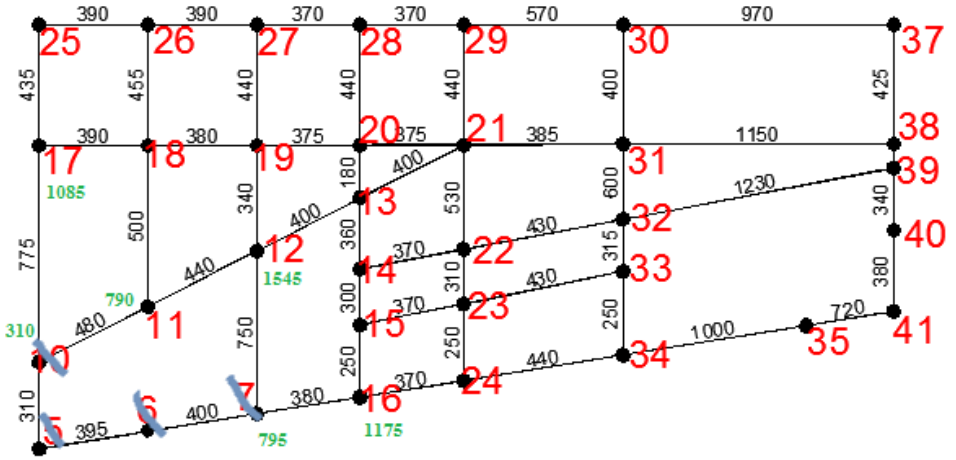


Рисунок 3.3 – Шаг 2

Шаг 3(Рисунок 3.4)
E5-E10-E17-E25 = 1520
E5-E10-E11-E18 = 1290
E5-E10-E11-E12 = 1230
E5-E10-E11-E12-E13 = 1630
E5-E6-E7-E16-E15 = 1425
E5-E6-E7-E16-E24 = 1545

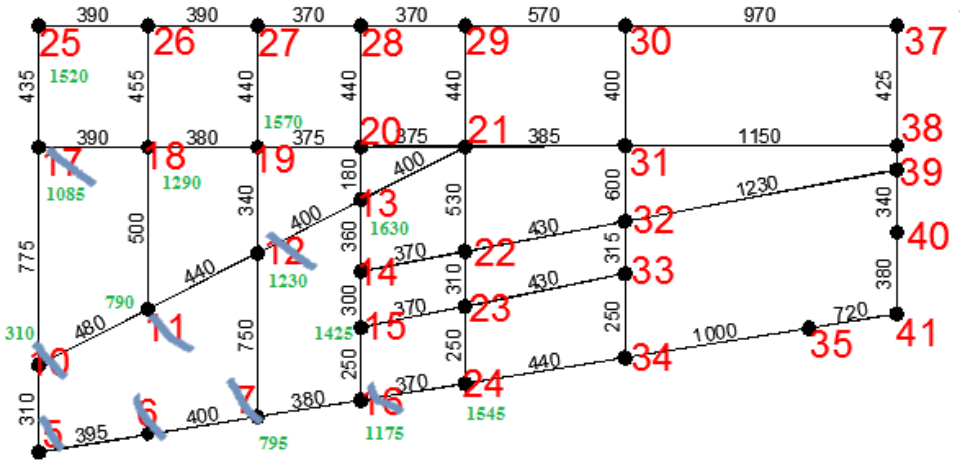


Рисунок 3.4 – Шаг 3

Шаг 4 (Рисунок 3.5)
E5-E10-E11-E18-E26= 1745
E5-E10-E11-E12-E19-E27 = 2010
E5-E10-E11-E12-E13-E20 = 1810
E5-E6-E7-E16-E15-E14 = 1725
E5-E6-E7-E16-E24-E23 = 1795
E5-E6-E7-E16-E24-E34 = 1985

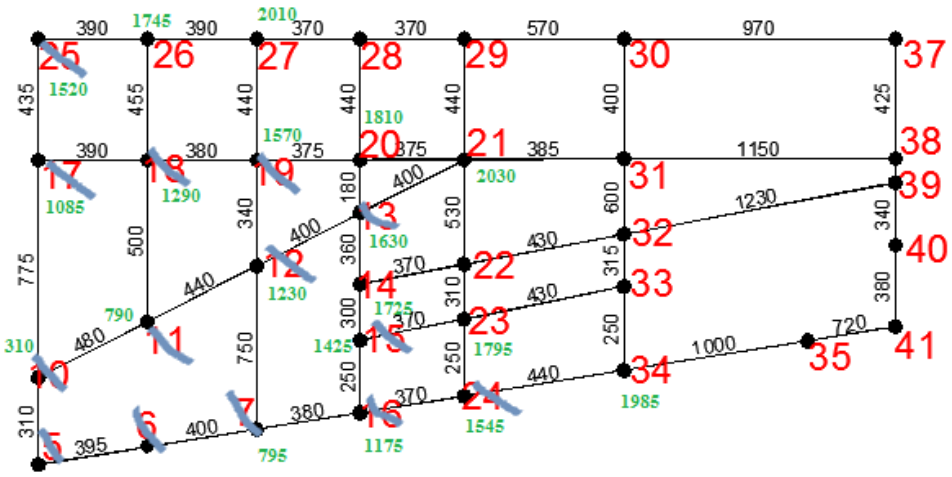


Рисунок 3.5 – Шаг 4

Шаг 5 (Рисунок 3.6)
E5-E10-E11-E12-E13-E20-E28 = 2250
E5-E10-E11-E12-E13-E21-E29 = 2470
E5-E6-E7-E16-E15-E14-E22 = 2095
E5-E6-E7-E16-E24-E23-E33 = 2225
E5-E10-E11-E12-E13-E21-E31 = 2415

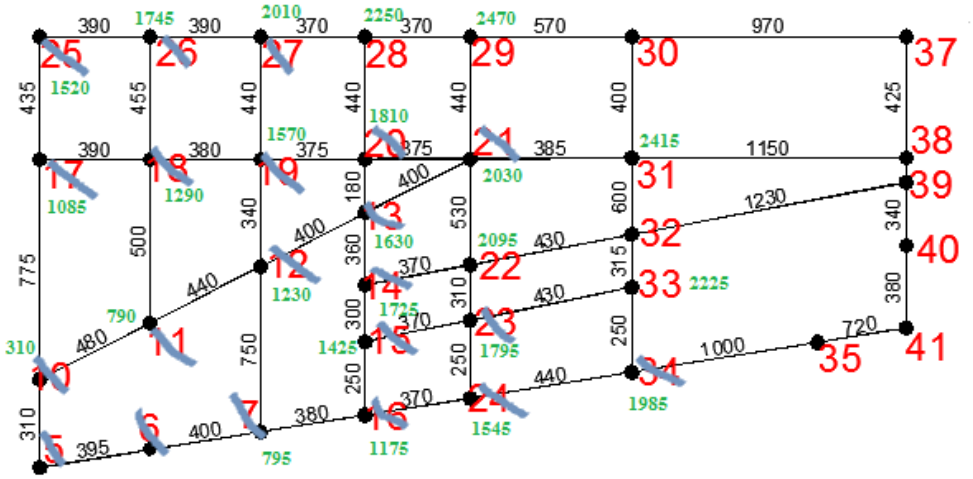


Рисунок 3.6 –Шаг 5

Шаг n (Рисунок 3.7)
E5-E10-E11-E12-E13-E21-E31-E38-E39-E40 = 4005
E5-E6-E7-E16-E24-E34-E35-E41-E40 = 4085

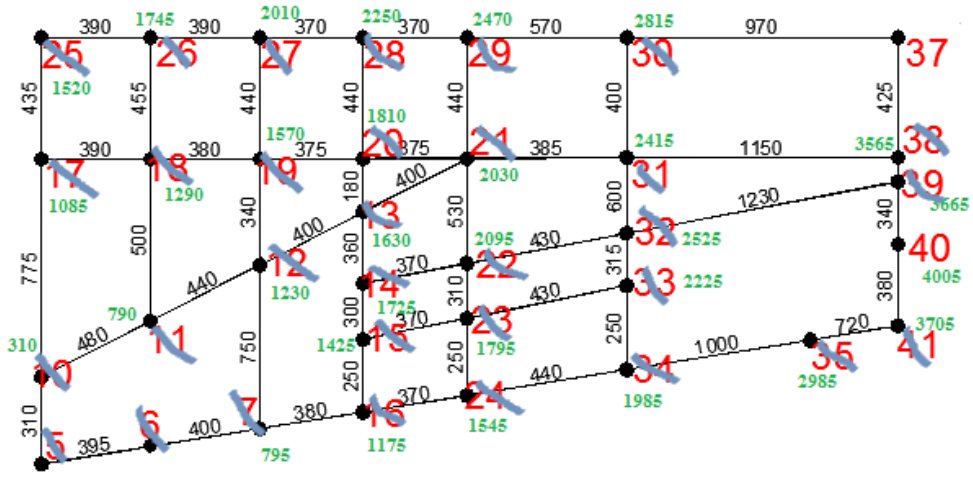


Рисунок 3.7 – Шаг n

Результатом данного алгоритма будет путь E5-E10-E11-E12-E13-E21-E31-E38-E39-E40(Рисунок 3.8)

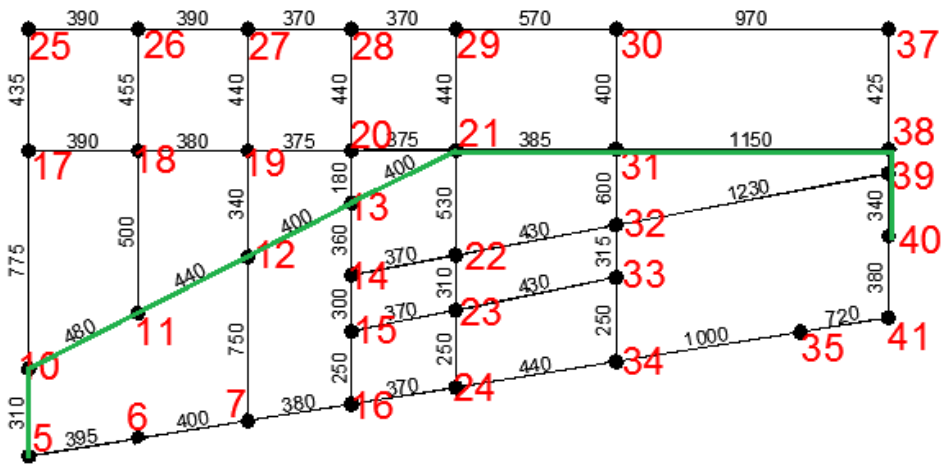


Рисунок 3.8 - Путь E5-E10-E11-E12-E13-E21-E31-E38-E39-E40

3.5 Программная реализация алгоритма

Алгоритм решения данной задачи реализован как программный продукт (ПП) на языке Android.

Карта улиц Бостандыкского района города Алматы загружается из внешнего файла. Далее указываются начальная и конечная точки назначения. Главное окно устройства при запуске программы имеет вид, показанный на рис. 3.2.

В программе реализован алгоритм Дейкстры. Листинг программы смотреть в приложении В.

Выводы:

-предложены математические модели задачи выбора оптимальных маршрутов в виде задач с дискретным программированием и алгоритмом ее решения. При конечном выборе маршрутов учитываем дополнительную неформализованную информацию.

- предлагающийся алгоритм, является модификацией общеизвестного алгоритма Дейкстры, реализованного в виде программного продукта на языке Android. Результат проводимых расчетов представляется в удобном виде на электронных картах города.

- проведенные расчеты доказывают, что метод Дейкстры более эффективен для нахождения оптимального пути. Данный алгоритм нахождения оптимального маршрута указывает на хорошую производительность для графов дорог достаточно больших размеров.

4 Технико-экономическая часть

4.1 Цель проекта

Цель дипломного проекта - написание программных продуктов отражения и обработки картографических данных на цифровых картах местности. Это программное средство обязано выполнять следующую функцию (задачи):

- отражение, загрузка цифровых карт местности (ЦКМ), зуммирование и прокрутка изображения загружаемой цифровой карты местности (с использованием опции имеющейся библиотеки картографических доступов);
- редактирование и создание матрицы корректировки.

Программные средства (ПС) вычислительной техники (ВТ) - это вещественный объект специфических интеллектуальных деятельностей специалистов, состоящих из программных документально оформленных проектов, реализующих свои потребительские свойства и качества в составе функционирующих вычислительных систем или систем обработки данных. По стоимости и срокам службы ПС относятся к основным производственным фондам предприятия.

4.2 Обоснование необходимости разработки программного продукта

Данный программный продукт обязан быть технически совершенным и экономически выгодным. На основах экономических оценок этого программного продукта принимается решение об инвестициях в данный проект. Отсутствие экономического обоснования и расчетов приводит к экономическим ошибкам в проектировании, созданию неэффективных проектов.

В итоге применения и разработки программных модулей обработки и отражения картографических данных на цифровых картах местности экономические эффекты достигаются за счет экономии материальных, финансовых и трудовых ресурсов по сравнению с базовым вариантом, программного модуля.

Актуальностью разработанного программного продукта является его новизна, простота использования, низкая потребляемость программных ресурсов, наличие преимуществ в области картографировании по сравнению с существующими аналогами программного продукта.

4.3 Определение объема трудоемкости разработки программного продукта (ПП)

Объем ПО. Базой для расчета плановой сметы затрат на разработку ПО является объем ПО.

Общий объем (V_0) программного продукта определяется исходя из количества и объема функций, реализуемых программой

$$V_o = \sum_{i=1}^{n} V_i , \qquad (4.1)$$

где V_i – объем отдельной функции ПО;

n – общее число функций.

Расчет общих объемов функции по каталогам представлен в таблице 4.1.

Единицы измерения объема ПО. Оценка объема программного продукта связана с выбором самые подходящие единицы измерения размера продукта. В этом проекте степени, как единица измерения объема ПО, используется количество линий первоначального кодекса (Линии кодекса, местоположения).

Линия первоначального кодекса (местоположение) является универсальными метриками, поскольку это может быть применено при создании любых программных продуктов.

Мы берем ценность указанного объема функций, который равен 2258 местоположений для общей суммы (V0) программного продукта.

Стандартные трудозатраты разработки программного обеспечения. На основе тома (V0), принятого к вычислению и категории трудозатрат стандарта сложности НА (Тн), который определен, принимая во внимание сложность, новинку проекта и степень использования стандартных модулей, когда развитие определено. В этом проекте степени V0 =2258, поэтому Tn=61.

Невозможно вычислить точный объем функций на стадиях технико-экономического обоснования на проекте. Только приблизительные (ожидаемые) оценки на основе доступных фактических данных согласно подобным проектам, которые были выполнены ранее, или применением существующих стандартов, могут быть получены. На основе информации о функциях, развитых НА объеме функций и общей суммы, определен каталогом функций, НА которых определен (исправлен). Указанный объем НА (Vy) окупается

$$V_y = \sum_{i=1}^{n} V_{yi} , \qquad (4.2)$$

где V_{yi} – уточненный объем отдельной функции ПО.

Расчет уточненного объема ПО представлен в таблице 4.1.

Т а б л и ц а 4.1 - Перечень и объем функций программного модуля

	Наименование (содержание)	Объем функции (LOC)	
		по каталогу V_i	уточненный V_{yi}
1	2	3	4
101	Организация ввода информации	150	75
109	Организация ввода / вывода информации в интерактивном режиме	320	250
301	Формирование последовательного файла	290	120
305	Обработка файлов	720	622
309	Формирование файла	1020	571
703	Расчет показателей	460	295
707	Графический вывод результатов	480	325
	Итого:	3440	2258

Из-за использования более прекрасных общих сумм оборудования автоматизации функций были уменьшены. Указанный объем ПО (Vy) сделал 2258 местоположений, общую сумму функций согласно каталогу (v0) 3440 местоположение.

Мы берем ценность указанного объема функций, который равен 2258 местоположений для общей суммы (V0) программного продукта.

Трудозатраты разработки программного обеспечения. Стандартные и общие трудозатраты разработки программного обеспечения определены указанным объемом НА и к стандартам расходов работы за единицу объема.

Стандартные трудозатраты разработки программного обеспечения. На основе тома (V0), принятого к вычислению и категории трудозатрат стандарта сложности НА (Тн), который определен, принимая во внимание сложность, новинку проекта и степень использования стандартных модулей, когда развитие определено. В этом проекте степени V0 =2258, поэтому Tn=61.Общая трудоемкость разработки ПО. Нормативная трудоемкость ($T_н$) служит основой для определения общей трудоемкости ($T_о$), расчет которой осуществляется в зависимости от размера проекта.

Общая трудоемкость в данном проекте рассчитывается по формуле 4.3.

$$T_O = T_н * K_c * K_m * K_н \qquad (4.3)$$

где K_c– коэффициент, учитывающий сложность ПО;

K_m – поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;

K_n– коэффициент, учитывающий степень новизны ПО.

Категория сложности ПО. Все ПО может быть подразделено на три категории сложности, в зависимости от существования (отсутствие) следующих особенностей:

- высокий уровень языка взаимодействует с пользователем;
- рабочий режим в режиме реального времени;
- управление отдаленными объектами;
- машинная графика;
- существенный parallelization вычислений;
- отгрузка ПО;
- и другие.

Влияние фактора сложности на трудозатратах рассматривает умножение стандартных трудозатрат соответствующим коэффициентом сложности. Этот модуль программы обладает 2-й категорией сложности.

Коэффициент сложности (Кс). Посредством коэффициента сложности рассматривают дополнительные расходы работы, связанной со сложностью развитого программного продукта. Коэффициент сложности окупается на формуле 4.4

$$K_c = 1 + \sum_{i=1}^{n} K_i , \qquad (4.4)$$

где K_i – коэффициент, соответствующий степени повышения сложности ПО.

В разрабатываемом дипломном проекте K_i, за счет наличия у программного модуля одновременно двух характеристик:

- машинная графика;
- обеспечение переносимости ПО.

принимаем K_i =0,12.

$K_c = 1 + 0{,}12 = 1{,}12$

Содействующая степень рассмотрения использования при разработке программного обеспечения стандартных модулей (км). Степень использования в развитом ПО стандартных модулях определена их определенным весом в общей сумме спроектированного продукта. В этой степени проекта степени освещения реализованных функций, развитых ПО стандартных модулях, стандартных программах и ПО20%, поэтому км =0,9.

Коэффициент новинки развитой ПО ($К_н$). Сравнение особенностей, развитых ПО с доступными аналогами, позволяет определять опытным путем

степень своей новинки. Развитый модуль программы - развитие определенный параметрический рядя ПО, согласно вышеупомянутому K_H =1,0.

T_o =61*1,12*0,9*1=62 (чел./дн.)

4.4 Численность исполнителей и срок разработки ПО

На основе общей трудоемкости определяется плановое число разработчиков *($Ч_P$)* и плановые сроки, необходимые для реализации проекта в целом (T_P). При этом решаются следующие задачи:

- расчет числа исполнителей при заданных сроках разработки проекта;
- определение сроков разработки проекта при заданной численности исполнителей.

Численность исполнителей проекта ($Ч_P$) рассчитывается по формуле 4.5

$$Ч_p = \frac{T_o}{T_p * Ф_{эф}}, \qquad (4.5)$$

где $Ф_{ЭФ}$ - эффективный фонд времени работы одного работника в течение года (дн.);

T_O - общая трудоемкость разработки проекта (чел./дн.);

T_P - срок разработки проекта (лет).

Срок разработки проекта (T_P) определяется по формуле 4.6.

$$T_p = \frac{T_o}{Ч_p * Ф_{эф}} \qquad (4.6)$$

В данном дипломном проекте срок разработки проекта установлен 1,5 месяца, что составляет 0,125 года, следовательно T_P =0,125 года.

Эффективный фонд времени работы одного работника ($Ф_{ЭФ}$) рассчитывается по формуле:

$$Ф_{эф} = Д_г - Д_п - Д_в - Д_о, \qquad (4.7)$$

где $Д_г$ - количество дней в году;

$Д_п$- количество праздничных дней в году;

$Д_в$ - количество выходных дней в году;

$Д_о$ - количество дней отпуска.

В развитом проекте степени мы примем ценности:

К =32 дням.

Эффективный фонд операционного времени одного рабочего (FEF) окупается на формуле 4.7

(дн)..

Число исполнителей проекта (ChR) окупается на формуле 4.5.

= 2 (люди).

Исходные данные для последующих вычислений, и также результаты вычислений объема и трудозатрат развития программного обеспечения, числа исполнителей и срока развития уменьшены в таблице 4.2.

Т а б л и ц а 4.2 - Исходные данные и результаты расчетов

Наименование показателей	Буквенное обозначение	Единицы измерения	Кол-во
1	2	3	4
Коэффициент новизны	$К_н$		1,0
Категория сложности			2
Дополнительный коэффициент сложности	$К_i$		0,12
Поправочный коэффициент, учитывающий использование типовых программ	$К_т$		0,9
Установленная плановая продолжительность разработки	$Т_р$	год	0,125
Годовой эффективный фонд времени	$Ф_{эф}$	дней	221
Продолжительность рабочего дня	$Т_ч$	ч	8,0
Месячная тарифная ставка 1-го разряда	$Т_{м1}$	тенге	19966
Коэффициент премирования	К		1,5
Норматив дополнительной заработной платы	$Н_д$	%	20,0
Уровень рентабельности ПО	$У_{рп}$	%	20,0
НДС	$Н_{дс}$	%	12,0
Норматив налога на прибыль	$Н_{нп}$	%	20,0

Главный пункт расходов на создание НА является зарплатой разработчиков (исполнители) проекта, в которое число это, как принимают, включает разработчиков программного обеспечения, участвующих в письме кодекса, менеджеров проектов. Зарплату глав организации и разработчиков рассматривают в накладных расходах.

4.5 Расчет основной заработной платы исполнителей.

В разработке данного программного модуля задействован следующий исполнитель:

- руководитель (заработная плата – 100 000 тенге)
- разработчик (заработная плата – 70 000 тенге)

Руководитель:

$З_{осн1}$=100 000 тенге.

$З_{доп1}$=(100000*20%)/100%=20 000 тенге.

$З_{пр1}$=((100000+20000)*15%)/100%=18 000 тенге.

$З_1=З_{осн1}+З_{доп1}+З_{пр1}$

$З_1$=100 000+20 000+18 000=138 000 тенге.

Разработчик:

$З_{осн2}$=70 000 тенге.

$З_{доп2}$=(70000*20%)/100%=14 000 тенге.

$З_{пр2}$=((70000+14000)*15%)/100%=12 600 тенге.

$З_2=З_{осн2}+З_{доп2}+З_{пр2}$

$З_2$=70 000+14 000+12 600=96 600 тенге

Часовая тарифная ставка ($Т_ч$) рассчитывается путем деления месячной тарифной ставки T_M на установленную при 40-часовой недельной норме рабочего времени расчетную среднемесячную норму рабочего времени в часах ($Ф_р$):

$$T_ч = З \ / Ф_р \qquad (4.8)$$

где $Ф_Р$ - составляет 176 часа;

Руководитель:

$T_Ч$ =138 000 / 176 = 785 (тенге).

Разработчик:

$T_Ч$ =96 600 / 176 = 550 (тенге).

Размер фонда оплаты труда разработчиков($З_{фот}$):

$З_{фот}$=89 900(тенге)

Социальный налог составляет 11%(ст.358 п.1 НК РК) от дохода работников, и рассчитывается по формулам 4.9 и 4.10.

$$З_с=(З_{фот}- З_{по})*11\%, \qquad (4.9)$$

$$З_{по}= З_{фот}*10\% \qquad (4.10)$$

$З_с$=(89 900 – 8 990)*11%= 8 900

Расходы определены статьей "Материалы" (М) на основе оценки расходов, развитых на при принятии во внимание существующих стандартов. Под расходами статьи "Материалы" по магнитным перевозчикам отражены избитые карты, бумага, рисуя ленты и другие материалы, необходимые для разработки программного обеспечения. Нормы потребления материалов в полном выражении (Миллимикрон) определены, рассчитывая на 100 линий первоначального кодекса или по стандарту как процент к фонду главной зарплаты разработчиков (Nmz), который установлен организацией (3%).

Т а б л и ц а 4.3 - Расчёт основной заработной платы персонала, задействованного в работе

Наименование содержания работ	Исполнитель	Трудоёмкость, норма - час	Заработная плата за час работы, тг	Сумма заработной платы , тг
Постановка задачи	Руководитель	4	785	3140
Изучение требуемой литературы	Разработчик	20	550	11000
Разработка алгоритмов	Разработчик	40	550	22000
Программирование	Разработчик	60	550	33000
Контроль и проведение испытаний	Руководитель Разработчик	5 5	785 550	3985 2750
Проведение расчетов	Разработчик	20	550	11000
Запись полученных результатов	Руководитель Разработчик	1 4	785 550	785 2200
Итого		160		89 900

Сумма затрат материалов рассчитывается по формуле:

$$M = \frac{Зо * Н_{мз}}{100}, \qquad (4.11)$$

M = (89 900 * 3%)/100% = 2 700 (тенге)

Расходы по статье «Машинное время» (Рм) включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода (Нмв) машинного времени в зависимости от характера решаемых задач и типа ПЭВМ:

$$Рм = Цм * \frac{Vo}{100} * Нмв, \qquad (4.12)$$

где $Ц_м$ – цена одного машино-часа (200 тенге);

$V_о$ – общий объем ПО (строк исходного кода);

$Н_{мв}$ – норматив расхода машинного времени на отладку 100 строк исходного кода (10 машино-часов).

$Рм$ = 200 * (2258/100) * 10 = 44 000 (тенге)

Расходы в соответствии со статьей "Прочие затраты" (Пз) на бетоне НА включают затраты приобретения и подготовки специальной научно-технической информации и специальной литературы. Определены стандартом, развитым в целом на научной организации, как процент к главной зарплате:

$$Пз = \frac{Зо * Н_{пз}}{100}, \qquad (4.13)$$

где $Н_{пз}$ – норматив прочих затрат в целом по научной организации (20%)

$П_З$ =(89 900 * 20%) / 100% = 17 980 (тенге)

Затраты по статье «Накладные расходы» ($Р_н$), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды ($Р_н$), относятся на конкретное ПО по нормативу ($Н_{рн}$) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по научной организации:

$$Рн = \frac{Зо * Н_{рн}}{100}, \qquad (4.14)$$

где $Р_н$ – накладные расходы на конкретную ПО (тыс. тенге);

$Н_{рн}$ – норматив накладных расходов в целом по научной организации (70%).

$Р_н$ =62 930 тенге

Размер фонда оплаты труда разработчиков($З_{фот}$):

$З_{фот}$=89 900(тенге)

Результаты выполненных расчетов сводятся в таблицу 4.4.

Т а б л и ц а 4.4 - Затраты на разработку

Затраты на разработку	Условное обозначение	Значение, тенге	В процентах от общей суммы
Фонд оплаты труда	$З_{фот}$	89 900	40
Социальный налог	$З_с$	8 900	4
Материалы	М	2 700	1,2
Машинное время	$Р_м$	44 000	19,4
Прочие затраты	$П_з$	17 980	7,9
Накладные расходы	$Р_н$	62 930	27,5
Итого:		226 410	100

Социальный налог составляет 11%(ст.358 п.1 НК РК) от дохода работников, и рассчитывается по формулам 4.9 и 4.10.

$$З_с=(З_{фот}- З_{по})*11\%, \quad (4.9)$$
$$З_{по}= З_{фот}*10\% \quad (4.10)$$

$З_с$=(89 900 – 8 990)*11%= 8 900

4.6 Расчет цены программного продукта

Себестоимость ($С_п$) на ПО рассчитывается по формуле 4.15.

$$Сп = Зфот + Зс + М + Рм + Пз + Рн \quad (4.15)$$

Сп = 89 900+8 900+2 700+44 000+17 980+62 930=226 410 тенге

Рентабельность и прибыль по создаваемому ПО определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость и отчисления в местный и республиканский бюджеты. В случае разработки ПО для использования внутри организации оценка программного продукта производится по действующим правилам и показателям внутреннего хозрасчета (по ценам, устанавливаемым для расчета за услуги между подразделениями). Прибыль рассчитывается по формуле 4.16.

$$Пс = \frac{Сп * Урп}{100}, \quad (4.16)$$

Пс = (226 410* 20%)/100% = 45 282 (тенге)

Прогнозируемая цена ПО без налогов ($Ц_п$):

$$Цп = Сп + Пс, \quad (4.17)$$

$Ц_п$ =226 410+45 282=271 692 тенге

Налог на добавленную стоимость (НДС)

$$НДС = \frac{Цп * Ндс}{100\%}, \quad (4.18)$$

где $Н_{дс}$ – норматив НДС (12%).
НДС=(271 692 * 12%) / 100% = 32 603 (тенге)

Прогнозируемая отпускная цена ($Ц_о$):

$$Цо = Цп + НДС, \quad (4.19)$$

$Ц_О$ = 271 692 + 32 603 = 304 295 (тенге)

Вывод: в данном разделе была рассмотрена экономическая часть реализуемого проекта. Для этого были составлены списки дел, которые необходимо выполнить для реализации программного обеспечения. После составления дел была распределена нагрузка в часах и процентах на количество участников, необходимых для написания программного продукта. Таким образом был рассчитан фонд оплаты труда участников. Для составления сметы затрат по всем статьям, использовались затраты по материалам, расходы по статье "машинное время", прочие расходы, накладные расходы. В конечном итоге была рассчитана себестоимость реализации программного продукта, которая составляет 226 410 тенге.

В заключении хочется отметить, что актуальностью разрабатываемого программного продукта является его новизна, простота использования, низкая потребляемость программных ресурсов, наличие преимуществ в области картографировании по сравнению с существующими аналогами программного продукта.

5.Безопасность жизнедеятельности

5.1 Анализ условий труда оператора связи при работе с ЭВМ

В настоящее время геоинформационные системы широко используется практически во всех сферах деятельности. Принято считать, что история развития географических информационных систем насчитывает более тридцати лет со времени создания в середине 60-х годов Канадской ГИС под руководством Р. Томлинсона. Судя по имеющейся литературе, это действительно была первая работающая автоматизированная информационная система, имеющая дело с пространственно распределенной информацией.

Данный дипломный проект посвящен разработке программного продукта, позволяющей при помощи геоинформационных систем оптимизировать транспортную маршрутизацию. Работа проводится на специализированном стенде, оснащенным ЭВМ, ЖК-дисплеем, планшетом, анализатором спектра и навигационным модулем.

Разработанный программный продукт поможет при помощи специальных программных команд, меняющих настройки навигационного модуля, определять координаты местоположения, а так же изменяя чувствительность приемника корректировать точность нахождения данных координат.

Так как программу будет осуществлять оператор ЭВМ, то в этой части диплома будем рассматривать место работы оператора в помещении.

Одним из важнейших условий, при котором будут обеспечиваться нормальные условия работы в помещениях, является хорошая освещенность рабочего места. Рабочие места должны освещаться в такой мере, чтобы работающий имел возможность хорошо видеть процесс работы, не напрягая зрения, не наклонялся к клавиатуре или экрану монитора. Освещенность в помещении оператора, управляющего ЭВМ не должна создавать резких теней и бликов, оказывающих слепящее действие.

Для нормальной работы оператора необходимо предусмотреть рациональное освещение. Освещение рабочего места должно быть таким, чтобы само рабочее место не было слишком ярким, а фон вокруг него слишком тёмным, так как контраст быстро утомляет глаза.

Недостаточное и неправильное освещение рабочего места оператора вызывает переутомление органов зрения, общее утомление организма. Снижение производительности труда, может стать причиной травматизма, профзаболевания.

Так как работа оператора будет приходиться больше на дневное время, в данной части дипломного проекта будем рассматривать естественное освещение, так как оно является основным одним из источником освещения в дневное время. Естественное освещение создается солнечным диском и диффузионным светом небесного излучения.

Исходя из вышенаписанного выполним следующее:

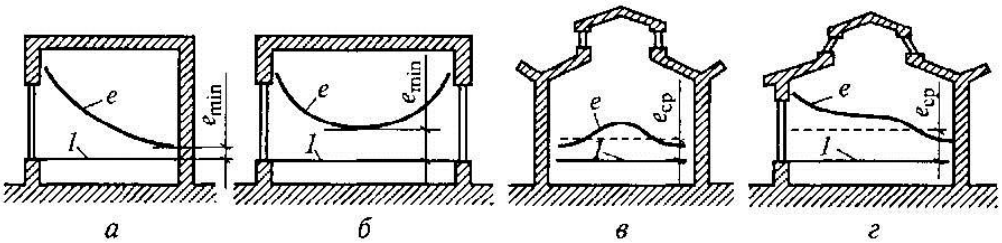
- проведем анализ выполняемой работы с учетом зрительной нагрузки оператора ЭВМ;
- проведем организацию естественного освещения в помещении;
- рассчитаем уровни естественного освещения для данного разряда зрительных работ;
- проведем расчет необходимой площади световых проемов.

5.2 Зрительная нагрузка оператора связи при работе с ЭВМ в помещении

с компьютером главная погрузка - работа акции вида, поскольку во время работы с монитором глаза устали намного более быстрые, чем в любых других типах работы. Груз органов видения в человеке, какая работа связана с компьютерами, возникает должный работать с монитором, на котором маленький текст программ, показаны документы, и также должный контролировать мигание. Мигание монитора возникает, потому что в простых показах частота горизонтального развития обычно не превышает 50¸60 Гц. Чтобы уменьшить груз органов видения от наставника, мы применим показы с довольно большим размером экрана и с частотой горизонтального развития более чем 70 Гц. У используемых мониторов есть гигиеническое свидетельство, которое включает, включая оценку визуальных параметров. Дизайн показа, его дизайн и набор эргономических параметров, чтобы обеспечить надежное и удобное чтение показанной информации. Важное условие высокоэффективной работы оператора - рациональное освещение своего рабочего места. Это объяснено этим посредством визуального устройства людей, получает приблизительно 90% информации от мира вокруг, его квитанция во многих отношениях зависит от качества освещения. Неблагоприятными факторами для профессиональной деятельности оператора, отрицательно влияя на вид, является:-пониженный уровень освещения, приводящего к перенапряжению глаз и, в результате - к быстрому истощению. - чрезмерно высокое освещение также - причина быстрой усталости, приводя к раздражению и схватываниям в глазах; - неправильное направление света способствует появлению острых теней или сильным участкам света, намного более быстрого утомительный глазом. - увеличенная яркость света; - прямая линия и возвращение blestkost; - поднятая пульсация легкого потока (мигание изображения); - отсутствие или недостаток естественного освещения. Уровень освещения должен быть обеспечен, принимая во внимание создание необходимого контраста между экраном показа и окружающей ситуацией, особенностями выполненных работ и требованиями пользователя. Освещение E – плотность области легкого потока F. При однородном распределении легкого потока, перпендикуляр сиявшая поверхность S, освещение [1]: (5.1) Один из важных компонент,

комнаты, необходимые для освещения - естественный свет. Естественное освещение выполнено через окна (освещение ответвления), легкие лампы (вершина) или в то же время через лампы и окна (объединились) [1]. Поскольку нормирование и оценка естественного освещения используют коэффициент естественного освещения (CNI), который выражен как процент: (5.2), где, Ev – освещение пункта в закрытом помещении, lx; ханьцы – одновременное внешнее освещение горизонтальной поверхности рассеянный свет небесного свода (без прямого света), lx. Распределение КЕО в комнатах неравно также будет зависеть от того, как будут расположены легкие апертуры (рисунок 5.1). (и – одностороннее боковое освещение; – двустороннее ответвление это сияется

Распределение КЕО внутри помещений неравномерно и будет зависеть от того, как будут расположены световые проемы (рисунок 5.1).



(а – одностороннее боковое освещение; б – двустороннее боковое освещение; в – верхнее освещение; г – комбинированное освещение; l – уровень рабочей поверхности)

Рисунок 5.1 – Схемы распределения КЕО по характерному разрезу помещения

Хорошо проникающий в помещение естественный свет оказывает благоприятное воздействие на психику человека, вызывая положительные эмоции, обеспечивая хорошие гигиенические условия работы. За счет естественного освещения стимулируется обмен веществ, кровообращение, дыхание, деятельность центральной нервной системы, что в свою очередь, обеспечивает высокую производительность труда.

Таким образом, для поддержания нормального рабочего состояния оператора, необходимо учитывать нормы и правила, регламентирующие оптимальность содержания рабочего места.

Для рабочих мест, оборудованных видео терминалами, в освещении SanPiN рекомендуется 100¸500 роскоши, в то время как для обычного офисного освещения к роскоши 1600 года рекомендуется. Кроме того, согласно гигиеническим нормам, основал SanPiN, освещение на поверхности стола и клавиатуры должно быть не меньше 300-й роскошью и вертикальным освещением экрана - только 100¸250 роскоши. Установка ламп местного освещения для освещения документов позволена. Местное освещение не должно создавать участки света на поверхности экрана и освещении увеличения экрана больше 300-я роскошь. Исследования физиологов и

гигиенистов убедительно доказали как сумерки, и слишком высокое освещение экрана приносит к быстрому визуальному утомлению экран, к которому имеет видеомонитор, от глаз оператора на оптимальном расстоянии 600¸700 мм, но 500 мм, принимающих во внимание размеры в алфавитном порядке – цифровые знаки и символы, не ближе. Уровень глаз в вертикальном расположении экрана PEVM должен упасть на центр или 2/3 высоты экрана. Линия взгляда должна быть перпендикулярна центру экрана, и его оптимальное отклонение от перпендикуляра, проходящего через центр экрана в вертикальном самолете, не должен превышать плюс - минус 5 градусов (допустимый плюс - минус 10градусов). Для защиты от прямых солнцезащитных устройств света (фильмы с детализованным покрытием, жалюзи с вертикальным) должны быть обеспечены. Операторам и пользователям рекомендуют выделить такие средства защиты как компьютерные очки, фильтры "полная защита" прошли тесты в аккредитованных лабораториях и наличии соответствующего гигиенического свидетельства. В случаях появления при работе с PEVM визуального дискомфорта, несмотря на соблюдение – гигиенические, эргономические требования, способы работы и отдыха необходимо применить индивидуальный подход в операционном ограничении времени с PEVM, исправлением продолжительности разрывов, чтобы покоиться или выполнить изменение деятельности с другим. В операционное время в целях обеспечения рабочей способности и сохранения здоровья необходимо сделать отрегулированные разрывы. Во время отрегулированных разрывов в целях уменьшения в давлении истощения визуального анализатора это целесообразно, чтобы выполнить комплексы упражнений для глаз. Упражнения выполнены, сидя или стоя, отворачиваясь от экрана при ритмичном дыхании, с максимальной амплитудой движения глаз.

5.3 Расчет естественного освещения для данного разряда зрительной работы

Для расчета естественного освещения нам необходимы следующие данные:

- длина помещения, L – 10 м;
- ширина помещения, B – 5м;
- высота помещения H = 3 м;
- высота рабочей поверхности h_P = 0,8 м;
- разряд зрительной работы IV (средней точности).

Нормированное значение КЕО e_N для зданий, располагаемых в различных районах будем определять по формуле:

$$e_N = e_н * m_N \qquad (5.3)$$

где N – номер группы обеспеченности естественным светом, в данном случае – 4 группа(Алматы);

$e_н$ – значение КЕО, при IV разряде – 1,5 %;

m_N - коэф. светового климата, с учетом, того световые проемы ориентированы на север, то m_N = 0,75.

Учитывая приведенные данные, по формуле 3 рассчитаем, нормированное значение КЕО:

$e_Н$ = 1,5 *1,125 = 0,9

5.4 Расчет необходимой площади световых проемов.

Для ориентировочных расчетов применяют метод определения требуемой площади светопроемов, которая обеспечивала бы нормированную для данной работы величину КЕО. При боковом освещении помещения используем следующую формулу:

$$S_0 = \frac{S_п e_н \eta_0 k_з k_{зд}}{100 \tau_0 r_1} \qquad (5.4)$$

где, S_0 — площадь окна; $S_п$ — площадь пола помещения; $e_н$ — нормированное значение КЕО; $k_з$ – коэффициент запаса (из таблиц); $k_{зд}$ – коэффициент, учитывающий затенение окон противостоящими зданиями; τ_0 – общий коэффициент светопропускания, r_1 — коэффициент, учитывающий влияние отраженного света при боковом освещении; η_0 — световая характеристика окна.

Формула общего коэффициента светопропускания:

$$\tau_0 = \tau_1 \tau_2 \tau_3 \tau_4 \qquad (5.5)$$

τ_1, τ_2, τ_3– коэффициенты, учитывающие потери света соответственно в материале остекленения, в переплетах светопроема, в слое загрязнения остекленения; τ_4- коэффициент, учитывающий потери света в несущих конструкциях, при боковом освещении равен 1.

Выбираем табличные данные из литературы 3. Подставляем в формулу 5 и вычисляем площадь световых проемов:

$$S_0 = \frac{50 \cdot 1{,}2 \cdot 10{,}5 \cdot 1{,}5 \cdot 1{,}4}{100 \cdot 0{,}48 \cdot 2{,}4} = 11{,}484 \ (м^2)$$

Данный расчет показывает, какая площадь световых проемов, необходима по нормам СНиПа, по замеренным данным окон помещения,

площадь равна 10,5 м^2. Таким образом, световые проемы помещения практически равны приведенным выше вычислениям.

Т а б л и ц а 5.1 - Исходные данные и результаты расчета площади световых проемов

Вид освещения	$S_П$, м^2	e_H^{IV}, %	η_0	$К_З$	$К_{ЗД}$	r_1	τ_0	Площадь световых проемов, м^2
Боковое одностороннее	50	1,5	10,5	1,5	1,4	2,4	0,48	11,484

5.5 Расчет системы вентиляции

Требуемое количество подаваемого воздуха [6] найдено по фактору «тепловыделение». Оно рассчитывается по формуле (5.6):

$$L = \frac{1000 \cdot E_Н \cdot К_З}{C_В \cdot \Delta t \cdot \gamma_В} \quad (5.6)$$

где $\Delta t = t_{УДАЛ} - t_{ПОСТ}$;

$t_{УДАЛ}$ – температура удаляемого воздуха;

$t_{ПОСТ}$ – температура поступающего воздуха;

$C_В$ – теплоемкость воздуха, $C_В$ = 0,24 ккал/кг·^{0}С;

$\gamma_В$– удельная масса приточного воздуха, $\gamma_В$= 1,206 кг/м^3;

$Q_{ИЗБ}$ – избыточное тепло.

Избыточное тепло найдено из выражения (5.7):

$$Q_{ИЗБ} = Q_{ОБ} + Q_Л + Q_Р - Q_{ОТД}, \quad (5.7)$$

где $Q_{ОБ}$ – тепло, выделяемое офисным оборудованием;

$Q_Л$ – тепло, выделяемое людьми;

$Q_Р$ – тепло, вносимое солнечной радиацией;

$Q_{ОТД}$ – теплоотдача в окружающую среду.

Значения $Q_Р$ и $Q_{ОТД}$ примерно равны и взаимно компенсируются. Поэтому избыточное тепло образуется только за счёт людей и оборудования.

Тепло, выделяемое людьми, найдём по формуле (5.8):

$$Q_Л = К_Л \cdot (q - q_{ПОГЛ}) \quad (5.8)$$

где $К_Л$ – количество людей в помещении, $К_Л$ = 13;

q – тепло, выделяемое одним человеком, q = 250 Ккал/ч;

$q_{ПОГЛ}$ – тепло, поглощаемое одним человеком, $q_{ПОГЛ}$ = 140 Ккал/ч.

Отсюда находим:

$Q_Л = К_Л \cdot (q - q_{ПОГЛ}) = 13 \cdot (250 - 140) = 1430$ Ккал/ч.

Рассчитано количество тепла, выделяемого офисным оборудованием. В помещении расположено 4 персональных компьютера. Каждый компьютер имеет мощность 230 Вт. Общая мощность компьютеров составляет:

$P_{общПК} = 4 \cdot 230 = 920$ Вт = 0,92 кВт.

Тепло, выделяемое офисным оборудованием рассчитаем по формуле (5.9):

$$Q_{ОБ} = 860 \cdot P_{ОБ} \cdot \eta, \qquad (5.9)$$

где 860 – тепловой эквивалент 1 кВт/час;

$P_{ОБ}$ – потребляемая мощность, $P_{ОБ} = 0{,}92$ кВт;

η – коэффициент перехода тепла в помещение, $\eta = 0{,}95$.

Подставив все значения в формулу, найдено:

$Q_{ОБ} = 860 \cdot 0{,}92 \cdot 0{,}95 = 751{,}6$ Ккал/ч.

Рассчитаем тепло напряженности воздуха по формуле (5.10):

$$Q_{ОБ} = \frac{Q_{ИЗБ}}{V_П}, \qquad (5.10)$$

где $V_П$ – объем помещения, $V_П = 150$ м^3.

$Q_{ИЗБ} = Q_Л + Q_{ОБ} = 1430 + 751{,}6 = 2181{,}6$ Ккал/ч.

Таким образом, подставив все данные в формулу, получено:

$$Q_{ОБ} = \frac{2181{,}6}{150} = 14{,}544 \text{ Ккал/м}^3.$$

Т.к. $Q_Н < 20$ Ккал/м^3, то $\Delta t = 8\ ^0$С.

Найдём требуемое количество подаваемого воздуха:

$$L = \frac{2181{,}6}{0{,}24 \cdot 8 \cdot 1{,}206} = 942{,}16 \text{ м}^3/\text{ч}.$$

Рассчитана кратность воздухообмена по формуле (5.11):

$$K = \frac{L}{V}, \qquad (5.11)$$

где, L – требуемое количество подаваемого воздуха, L = 942,16 м^3/ч;

V – объем помещения, V = 150 м^3.

Таким образом, кратность воздухообмена равна:

$$K = \frac{942{,}16}{150} = 6{,}28 \approx 7$$

Таким образом, необходим кондиционер, создающий воздухообмен 942,16 $м^3$/ч.

В помещении установим один настенный кондиционер Tosot Joice T24H-SJ, рассчитанный на 50-65 $м^2$ (расположение кондиционера показано на рисунке 5.2). Данный кондиционер создает воздухообмен 950 $м^3$/ч, что удовлетворяет условию 950 $м^3$/ч >942,16 $м^3$/ч, создает в помещении воздушную среду с температурой 17-26 ^{0}С и влажность 40-70%, удаляет из помещения избыточную влагу и тепло, снабжен таймером, термостатом, бактерицидным фильтром и автоматическим климат контролем. Электропитание кондиционера 220 В, 5 А, 50 Гц; максимальный уровень шума 42 дБ; внутренний блок: длина 1008 мм, высота 319 мм, глубина 221 мм; внешний блок: длина 955 мм, высота 700, глубина 369 мм.

Вывод: в разделе безопасность жизнедеятельности были рассмотрены вопросы зрительной нагрузки оператора связи при работе в помещении с ЭВМ, было рассмотрено, как освещенность рабочего места влияет на работоспособность оператора. Был проведен расчет естественного освещения помещения через боковые проемы (окна), и вычислена необходимая площадь светового проема, которая в итоге составила 11,484 $м^2$. При вычислении площади были учтены такие факторы как толщина оконного покрытия, его светопропускаемость, наличие соседних зданий, затеняющих наши проемы. Световые проемы (окна) используемого помещения соответствуют стандартам СНиПа, тем самым деятельность оператора ЭВМ не будет затруднена и он не будет сильно уставать при работе. Для хорошего освещения так же рекомендуется держать окна в чистоте и проводить регулярное удаление пыли и грязи. Еще одним параметром, влияющим на работоспособность человека, является температура рабочего помещения, и поэтому был проведен расчет вентиляции. При расчете системы вентиляции было найдено необходимое количество подаваемого воздуха L=942,16 $м^3$/ч. Расчет был необходим для установки подходящего кондиционера Tosot Joice T24H-SJ. Данный кондиционер создает воздухообмен 950 $м^3$/ч, что удовлетворяет условию 950 $м^3$/ч >942,16 $м^3$/ч, создает в помещении воздушную среду с температурой 17-26 0С и влажность 40-70%, удаляет из помещения избыточную влагу и тепло, снабжен таймером, термостатом, бактерицидным фильтром и автоматическим климат контролем.

Заключение

В заключении стоит описать задачу визуального отображения информационной системы городской окружающей среды, установленной нами и информационной пользовательской поддержкой. Организация слоя осуществляется со слоем географической и семантической информации по электронным картам была главной идеей. Заключение данных зависит от масштаба, установленного фильтра и параметра настройки. Прототип программы, понимающей нашу идею, развит. Программа – это геоинформационная система с электронной картой части Алма-Аты и инструментов на работе с ним.

ГИС - оптимальная платформа для сложных решений в транспортной сфере. Транспортные системы с их территориальным распределением — идеальные объекты автоматизации с помощью геоинформационных систем. Пространственный компонент - естественное основание интеграции проблем управления транспортной инфраструктурой, задачами урегулирования, проблемами эксплуатационного управления, навигация и т.д.

ГИС - создание зон транспортной доступности и наложение оптимальных маршрутов.

Дальнейшее расширение возможностей геоинформационных системы может быть выполнено для реализации с результатами поиска (формирование и экспортировать в различные форматы найденных маршрутов движения, вычисления расходов для горючих смазок и обслуживания транспортных средств в постоянной дорожной перевозке на найденных маршрутах), добавления дополнительных признаков дорог и формирования на их основе новых критериев поиска (качество дороги и количество переулков; удобство, время и средняя скорость движения).

Список литературы:

1. Сильянов В.В. Глобальный аспект в проблеме обеспечения безопасности дорожного движения. Сб. докладов пятой международной конференции «Организация дорожного движения в крупных городах. СПб, 19-20 сентября 2002 года, СПбГАСУ, 2002. 92 с.
2. Кошкарев А.В. Геоинформатика. Толкования основых терминов. - Программно-аппаратное обеспечение, фонд цифрового материала, услуги и нормативно-правовая-база геоинформатики. Ежегодный обзор. Выпуск 3 (1996-1997). Том 1. Приложение к "Иформационному бюллетеню" ГИС-Ассоциации". - М.: ГИС-Ассоциация, 2008. 190 с.
3. Makarov O.N., Konoplev V.M., Rastoskuev V.V. Environmental information system for rational management in the St. Petersburg region // The International Conference "Index-97". St. Petersburg, Russia, July 7-11, 2007, С.-340
4. Бугаевский JI.M. "Математическая картография". М., Златоуст, 2008, 314 с.
5. Молоденский М.С., Еремеев В.Ф., Юркина М.И. "Методы изучения внешнего гравитационного поля Земли". М., Геодезиздат, 1999, 215 с.
6. Киселев В.П., Токарчук Д.Н. Технология конвертирования данных из формата F1M в обменный формат Addmaplnfo. Материалы второго семинара "Проблемы ввода и обновления пространственной информации" (Москва, 24-27 февраля 2007 г.)
7. А.А. Питенко – Нейросетевой анализ в геоинформационных системах, диссертация, Красноярск, 2010г.
8. Андрианов В.- Гис для бизнеса, статься журнала ArcReview №1 (12) 2000г.
9. Андрианов В. – ГИС на транспорте, статься журнала ArcReview №1 (24) 2003г.
10. Авербух В.Л. – конспект лекций по спецкурсу «Визуализация программного обеспечения», Уральский Государственный Университет им. А.М.Горького, г.Екатеринбург, 2010г.420 с.
11. Reto Meier – Professional Adandroid™ 2 Application development
12. Белков А.В. , Грызлова Т.П. , Шаров В.Г. – Проблемы и методы построения эффективных визуальных информационных систем, статья журнала «Програмные продукты и системы», выпуск №3 за 2002г.
13. Алексеев В.В., Гаврилов Г.П., Сапоженко А.А. (ред.) Теория графов. Покрытия, укладки, турниры. Сборник переводов - М. : Мир, 1974.— 224 с.
14. Асельдеров З.М., Донец Г.А. Представление и восстановление графов - К.:Наукова Думка, 2011, 96 с.
15. Березина Л. Ю. Графы и их применение: Пособие для учителей. — М.: Просвещение, 1979. 160 с.
16. Донец Г.А., Шор Н.З. Алгебраический подход к проблеме раскраски плоских графов - К.: Наукова думка, 1982. — 144 с.

17. Зыков А.А. Основы теории графов. - М.:Наука, 1987, 384 с.

18. Калмыков Г. И. Древесная классификация помеченных графов. - М.: ФИЗМАТЛИТ, 2003. - 192 с.

19. Камерон П., ван Линт Дж. Теория графов, теория кодирования и блок-схемы - М.:Наука, 1980, 140 с.

20. GIS by ESRI. User's Guide. : 1. Understanding GIS. The ARC/INFO Method.

21. PC ARC/INFO. 3. PC ARCPLOT. 4. PC ARCEDIT. 5. SML. - USA, Redlans: Environmental System Research Institute, 2004.

22. ARC/INFO Версии 8: Объектно-компонентная ГИС (по статье Дэвида Дж. Магвайера в ARC News, Vol. 20 № 4) // ARCREVIEW - №3(10).- 2009.

23. Гераськин С., Назаренко Н. Публикация геоданных в Интернете //PC Magazine/RE. - N3. - 2007.

24. Андреев А.М., Березкин Д.В., Куликов Ю.В., Смагин А.Ю., Смелов А.В. Геоинформационные системы, объектно-ориентированный подход к проектированию. Объектные ГИС // «Геодезия и картография» - №9 - 1995.

25. КИБЕРСО, http://www.kiberso.ru/

26. Кокурин И.М., Гусев О.А., Шустик JI.А., Шустик K.JI. Геоинформационная модель прокладки и определения параметров маршрута поездки автомобиля по городу и области. «Организация и безопасностьдорожного движения в крупных городах», Сб. докладов 5-ой межд. Конф. СПб. 2002.

27. Печерский М., Буданов А. Система управления движением транспорта, Открытые системы, № 11-12, 1999 год // Издательство "Открытые Системы" (http://www.osp.ru/).

28. Галазин В.Ф., Базлов Ю.А. и др. "Совместное использование GPS и "Глонасс". Доклад. Май, 1997г.

29. Кристофидес Н. Теория графов. Алгоритмический подход. - М.: Мир, 1997.

30. Майника Э. Алгоритмы оптимизации на сетях и графах. - М.: Мир, 1998.

31. Зыков А.А. Основы теории графов. - М.: Наука, 1997.

32. The Digital Chart of the World for use with ARC/INFO. - ESRI.- 2003.- 89 с.

33. Водов М.А., Пухов Г.Г., Жданов С.Н. и др. Безопасность жизнедеятельности Санкт-Петербурга и ГИС технологии. ARCREVIEW, Спец-выпуск, 2003, Изд-во ООО Data+, - С. 2-3, + www.dataplus.ru

34. М. Де Гроот. Оптимальные статистические решения. -М.: Мир, 2004.- 496 с.

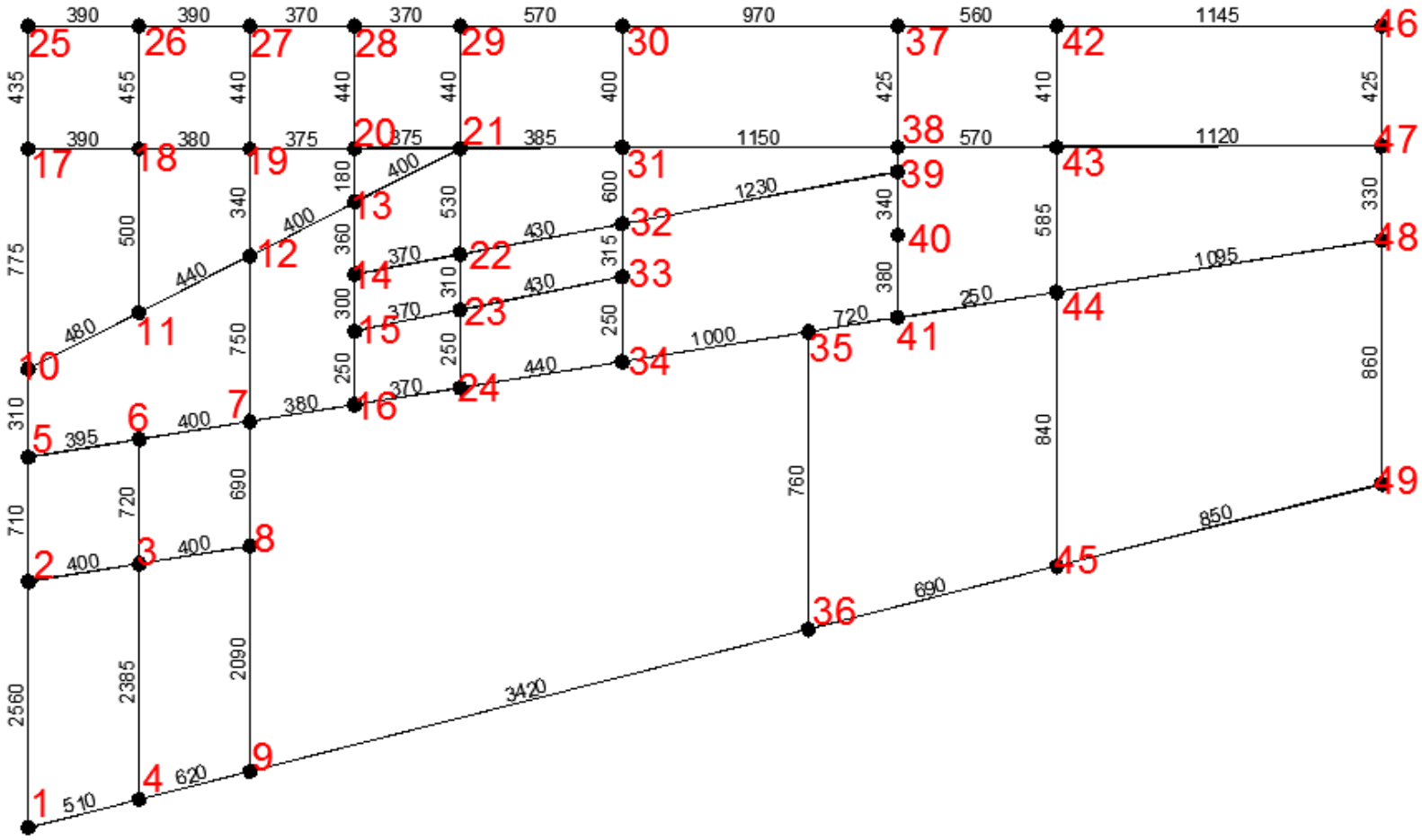
35. Воронцовский А.В. Инвестиции и финансирование: Методы оценки и обоснования. - СПб: Изд-во С.-Петерб. Ун-та, 2009.

36. Завлин П.Н., Васильев А.В. Оценка эффективности инноваций. СПб, Издательский дом «Бизнес-пресса», 2001.

37. Идрисов А.Б., Картышев С.В., Постников А.В. Стратегическое планирование и анализ эффективности инвестиций. Издание 2-е, стереотипное - М.: Информационно-издательский дом «Филинъ», 2008, 215 с.

38. Крушвиц JI. Инвестиционные расчёты. Пер. с нем. под общей редакцией В.В. Ковалёва и З.А. Сабова: Издательство «Питер», 2001.

Приложение А
Граф дорожной сети Бостандыкского района г. Алматы



Приложение Б
Матрица смежности

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	2560	x	510	x	x	x	x	x	x	x	x	x
2	2560	0	400	x	710	x	x	x	x	x	x	x	x
3	x	400	0	2385	x	720	x	400	x	x	x	x	x
4	510	x	2385	0	x	x	x	x	620	x	x	x	x
5	x	710	x	x	0	395	x	x	x	310	x	x	x
6	x	x	720	x	395	0	400	x	x	x	x	x	x
7	x	x	x	x	x	400	0	690	x	x	x	750	x
8	x	x	400	x	x	x	690	0	2090	x	x	x	x
9	x	x	x	620	x	x	x	2090	0	x	x	x	x
10	x	x	x	x	310	x	x	x	x	0	480	x	x
11	x	x	x	x	x	x	x	x	x	480	0	440	x
12	x	x	x	x	x	x	750	x	x	x	440	0	400
13	x	x	x	x	x	x	x	x	x	x	x	400	0
14	x	x	x	x	x	x	x	x	x	x	x	x	360
15	x	x	x	x	x	x	x	x	x	x	x	x	x
16	x	x	x	x	x	x	380	x	x	x	x	x	x
17	x	x	x	x	x	x	x	x	x	775	x	x	x
18	x	x	x	x	x	x	x	x	x	x	500	x	x
19	x	x	x	x	x	x	x	x	x	x	x	340	x
20	x	x	x	x	x	x	x	x	x	x	x	x	180
21	x	x	x	x	x	x	x	x	x	x	x	x	400
22	x	x	x	x	x	x	x	x	x	x	x	x	x
23	x	x	x	x	x	x	x	x	x	x	x	x	x
24	x	x	x	x	x	x	x	x	x	x	x	x	x
25	x	x	x	x	x	x	x	x	x	x	x	x	x
26	x	x	x	x	x	x	x	x	x	x	x	x	x
27	x	x	x	x	x	x	x	x	x	x	x	x	x
28	x	x	x	x	x	x	x	x	x	x	x	x	x
29	x	x	x	x	x	x	x	x	x	x	x	x	x
30	x	x	x	x	x	x	x	x	x	x	x	x	x
31	x	x	x	x	x	x	x	x	x	x	x	x	x
32	x	x	x	x	x	x	x	x	x	x	x	x	x
33	x	x	x	x	x	x	x	x	x	x	x	x	x
34	x	x	x	x	x	x	x	x	x	x	x	x	x
35	x	x	x	x	x	x	x	x	x	x	x	x	x
36	x	x	x	x	x	x	x	x	3420	x	x	x	x
37	x	x	x	x	x	x	x	x	x	x	x	x	x
38	x	x	x	x	x	x	x	x	x	x	x	x	x
39	x	x	x	x	x	x	x	x	x	x	x	x	x
40	x	x	x	x	x	x	x	x	x	x	x	x	x
41	x	x	x	x	x	x	x	x	x	x	x	x	x
42	x	x	x	x	x	x	x	x	x	x	x	x	x
43	x	x	x	x	x	x	x	x	x	x	x	x	x
44	x	x	x	x	x	x	x	x	x	x	x	x	x
45	x	x	x	x	x	x	x	x	x	x	x	x	x
46	x	x	x	x	x	x	x	x	x	x	x	x	x
47	x	x	x	x	x	x	x	x	x	x	x	x	x
48	x	x	x	x	x	x	x	x	x	x	x	x	x
49	x	x	x	x	x	x	x	x	x	x	x	x	x

Продолжение приложения Б

	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6	x	x	x	x	x	x	x	x	x	x	x	x	x	x
7	x	x	380	x	x	x	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x	x	x	x	x	x
9	x	x	x	x	x	x	x	x	x	x	x	x	x	x
10	x	x	x	775	x	x	x	x	x	x	x	x	x	x
11	x	x	x	x	500	x	x	x	x	x	x	x	x	x
12	x	x	x	x	x	340	x	x	x	x	x	x	x	x
13	360	x	x	x	x	x	180	400	x	x	x	x	x	x
14	0	300	x	x	x	x	x	x	370	x	x	x	x	x
15	300	0	250	x	x	x	x	x	x	370	x	x	x	x
16	x	250	0	x	x	x	x	x	x	x	370	x	x	x
17	x	x	x	0	390	x	x	x	x	x	x	435	x	x
18	x	x	x	390	0	380	x	x	x	x	x	x	455	x
19	x	x	x	x	380	0	375	x	x	x	x	x	x	440
20	x	x	x	x	x	375	0	375	x	x	x	x	x	x
21	x	x	x	x	x	x	375	0	530	x	x	x	x	x
22	370	x	x	x	x	x	x	530	0	310	x	x	x	x
23	x	370	x	x	x	x	x	x	310	0	250	x	x	x
24	x	x	370	x	x	x	x	x	x	250	0	x	x	x
25	x	x	x	435	x	x	x	x	x	x	x	0	390	x
26	x	x	x	x	455	x	x	x	x	x	x	390	0	390
27	x	x	x	x	x	440	x	x	x	x	x	x	390	0
28	x	x	x	x	x	x	440	x	x	x	x	x	x	370
29	x	x	x	x	x	x	x	440	x	x	x	x	x	x
30	x	x	x	x	x	x	x	x	x	x	x	x	x	x
31	x	x	x	x	x	x	x	385	x	x	x	x	x	x
32	x	x	x	x	x	x	x	x	430	x	x	x	x	x
33	x	x	x	x	x	x	x	x	x	430	x	x	x	x
34	x	x	x	x	x	x	x	x	x	x	440	x	x	x
35	x	x	x	x	x	x	x	x	x	x	x	x	x	x
36	x	x	x	x	x	x	x	x	x	x	x	x	x	x
37	x	x	x	x	x	x	x	x	x	x	x	x	x	x
38	x	x	x	x	x	x	x	x	x	x	x	x	x	x
39	x	x	x	x	x	x	x	x	x	x	x	x	x	x
40	x	x	x	x	x	x	x	x	x	x	x	x	x	x
41	x	x	x	x	x	x	x	x	x	x	x	x	x	x
42	x	x	x	x	x	x	x	x	x	x	x	x	x	x
43	x	x	x	x	x	x	x	x	x	x	x	x	x	x
44	x	x	x	x	x	x	x	x	x	x	x	x	x	x
45	x	x	x	x	x	x	x	x	x	x	x	x	x	x
46	x	x	x	x	x	x	x	x	x	x	x	x	x	x
47	x	x	x	x	x	x	x	x	x	x	x	x	x	x
48	x	x	x	x	x	x	x	x	x	x	x	x	x	x
49	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	28	29	30	31	32	33	34	35	36	37	38	39
1	x	x	x	x	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x	x	x	x	x
3	x	x	x	x	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x	x	x	x	x
6	x	x	x	x	x	x	x	x	x	x	x	x
7	x	x	x	x	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x	x	x	x
9	x	x	x	x	x	x	x	x	3420	x	x	x
10	x	x	x	x	x	x	x	x	x	x	x	x
11	x	x	x	x	x	x	x	x	x	x	x	x
12	x	x	x	x	x	x	x	x	x	x	x	x
13	x	x	x	x	x	x	x	x	x	x	x	x
14	x	x	x	x	x	x	x	x	x	x	x	x
15	x	x	x	x	x	x	x	x	x	x	x	x
16	x	x	x	x	x	x	x	x	x	x	x	x
17	x	x	x	x	x	x	x	x	x	x	x	x
18	x	x	x	x	x	x	x	x	x	x	x	x
19	x	x	x	x	x	x	x	x	x	x	x	x
20	440	x	x	x	x	x	x	x	x	x	x	x
21	x	440	x	385	x	x	x	x	x	x	x	x
22	x	x	x	x	430	x	x	x	x	x	x	x
23	x	x	x	x	x	430	x	x	x	x	x	x
24	x	x	x	x	x	x	440	x	x	x	x	x
25	x	x	x	x	x	x	x	x	x	x	x	x
26	x	x	x	x	x	x	x	x	x	x	x	x
27	370	x	x	x	x	x	x	x	x	x	x	x
28	0	370	x	x	x	x	x	x	x	x	x	x
29	370	0	570	x	x	x	x	x	x	x	x	x
30	x	570	0	400	x	x	x	x	x	970	x	x
31	x	x	400	0	600	x	x	x	x	x	1150	x
32	x	x	x	600	0	315	x	x	x	x	x	1230
33	x	x	x	x	315	0	250	x	x	x	x	x
34	x	x	x	x	x	250	0	1000	x	x	x	x
35	x	x	x	x	x	x	1000	0	760	x	x	x
36	x	x	x	x	x	x	x	760	0	x	x	x
37	x	x	970	x	x	x	x	x	x	0	425	x
38	x	x	x	1150	x	x	x	x	x	425	0	110
39	x	x	x	x	1230	x	x	x	x	x	110	0
40	x	x	x	x	x	x	x	x	x	x	x	340
41	x	x	x	x	x	x	x	720	x	x	x	x
42	x	x	x	x	x	x	x	x	x	560	x	x
43	x	x	x	x	x	x	x	x	x	x	570	x
44	x	x	x	x	x	x	x	x	x	x	x	x
45	x	x	x	x	x	x	x	x	690	x	x	x
46	x	x	x	x	x	x	x	x	x	x	x	x
47	x	x	x	x	x	x	x	x	x	x	x	x
48	x	x	x	x	x	x	x	x	x	x	x	x
49	x	x	x	x	x	x	x	x	x	x	x	x

	40	41	42	43	44	45	46	47	48	49
1	x	x	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x	x	x
3	x	x	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x	x	x
6	x	x	x	x	x	x	x	x	x	x
7	x	x	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x	x
9	x	x	x	x	x	x	x	x	x	x
10	x	x	x	x	x	x	x	x	x	x
11	x	x	x	x	x	x	x	x	x	x
12	x	x	x	x	x	x	x	x	x	x
13	x	x	x	x	x	x	x	x	x	x
14	x	x	x	x	x	x	x	x	x	x
15	x	x	x	x	x	x	x	x	x	x
16	x	x	x	x	x	x	x	x	x	x
17	x	x	x	x	x	x	x	x	x	x
18	x	x	x	x	x	x	x	x	x	x
19	x	x	x	x	x	x	x	x	x	x
20	x	x	x	x	x	x	x	x	x	x
21	x	x	x	x	x	x	x	x	x	x
22	x	x	x	x	x	x	x	x	x	x
23	x	x	x	x	x	x	x	x	x	x
24	x	x	x	x	x	x	x	x	x	x
25	x	x	x	x	x	x	x	x	x	x
26	x	x	x	x	x	x	x	x	x	x
27	x	x	x	x	x	x	x	x	x	x
28	x	x	x	x	x	x	x	x	x	x
29	x	x	x	x	x	x	x	x	x	x
30	x	x	x	x	x	x	x	x	x	x
31	x	x	x	x	x	x	x	x	x	x
32	x	x	x	x	x	x	x	x	x	x
33	x	x	x	x	x	x	x	x	x	x
34	x	x	x	x	x	x	x	x	x	x
35	x	720	x	x	x	x	x	x	x	x
36	x	x	x	x	x	690	x	x	x	x
37	x	x	560	x	x	x	x	x	x	x
38	x	x	x	570	x	x	x	x	x	x
39	340	x	x	x	x	x	x	x	x	x
40	0	380	x	x	x	x	x	x	x	x
41	380	0	x	x	250	x	x	x	x	x
42	x	x	0	410	x	x	1145	x	x	x
43	x	x	410	0	585	x	x	1120	x	x
44	x	250	x	585	0	840	x	x	1095	x
45	x	x	x	x	840	0	x	x	x	850
46	x	x	1145	x	x	x	0	425	x	x
47	x	x	x	1120	x	x	425	0	330	x
48	x	x	x	x	1095	x	x	330	0	860
49	x	x	x	x	x	850	x	x	860	0

Приложение В
Листинг программы Java

The main Java file “RoutePath” which load the Google Addmap, and search the way from two points.

```
package com.agarwal.routepath;
Limport java.io.IOException;
Limport java.net.HttpURLConnection;
Limport java.net.MalformedURLException;
Limport java.net.URL;
Limport java.util.ArrayList;
Limport java.util.List;
Limport javax.xml.parsers.DocBuilder;
Limport javax.xml.parsers.DocBuilderFactory;
Limport javax.xml.parsers.ParserConfigurationException;
Limport org.w3c.dom.Doc;
Limport org.w3c.dom.Element;
Limport org.w3c.dom.Node;
Limport org.w3c.dom.NodeList;
Limport org.xml.sax.SAXException;
Limport adandroid.app.AlertDialg;
Limport adandroid.app.ProgressDilog;
Limport adandroid.content.DilogInterface;
Limport adandroid.graphic.Bitaddaddmap;
Limport adandroid.graphic.BitaddaddmapFactory;
Limport adandroid.graphic.Canvas;
Limport adandroid.graphic.Color;
Limport adandroid.graphic.Point;
Limport adandroid.graphic.drawable.Drawable;
Limport adandroid.os.AsyncTask;
Limport adandroid.os.Bundle;
Limport adandroid.util.Log;
Limport adandroid.view.Menu;
Limport adandroid.view.MenuItem;
Limport adandroid.view.MotionEvent;
Limport adandroid.view.View;
Limport adandroid.view.View.OnClickListener;
Limport adandroid.widget.Button;
Limport adandroid.widget.Toast;
Limport com.google.adandroid.addaddmaps.GeoPoint;
Limport com.google.adandroid.addaddmaps.AddaddmapActivity;
Limport com.google.adandroid.addaddmaps.AddaddmapView;
```

```
Limport com.google.adandroid.addaddmaps.Overlay;
Limport com.google.adandroid.addaddmaps.OverlayItem;
public class RoutePath extends AddaddmapActivity implements OnClickListener{
    /** Called when the activity is first created. */
    Button fromBtn;
    Button toBtn;
    Button searchBtn;
    Boolean isFrom=false, isTo=false;
        AddaddmapView addaddmapView;
    private RoutePath _activity;
    GeoPoint srcGeoPoint = null,destGeoPoint=null;
    private static List<Overlay> mOverlays;

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      addaddmapView = (AddaddmapView) findViewById(R.id.addaddmapView);
      addaddmapView.setClickable(true);
      toBtn = (Button) findViewById(R.id.to);
      fromBtn = (Button) findViewById(R.id.from);
      searchBtn = (Button) findViewById(R.id.search);
      toBtn.setOnClickListener(this);
      fromBtn.setOnClickListener(this);
      searchBtn.setOnClickListener(this);
      _activity = this;
        List<Overlay> addaddmapOverlays = addaddmapView.getOverlays();
GeoPointpoint=newGeoPoint((int)(43.228603*1E6),(int)(76.918500*1E6));
      addaddmapView.getController().animateTo(point);
      addaddmapView.getController().setCenter(point);
      addaddmapView.getController().setZoom(16);

      addaddmapView.setBuiltInZoomControls(true);
      addaddmapView.displayZoomControls(true);
      mOverlays = addaddmapView.getOverlays();
      AddaddmapOverlay addaddmapOverlay = new AddaddmapOverlay();
      addaddmapOverlays.add(addaddmapOverlay);

      addaddmapView.invalidate();
```

```
class AddaddmapOverlay extends Overlay
{    @Override
  public boolean onTouchEvent(MotionEvent event, AddaddmapView addaddmapView)
  {
    //---when user lifts his finger---
    if (event.getAction() == MotionEvent.ACTION_UP) {
      GeoPoint p = addmapView.getProjection().fromPixels(
        (int) event.getX(),
        (int) event.getY());
        //Toast.makeText(getBaseContext(),
        //    p.getLatitudeE6() / 1E6 + "," +
        //    p.getLongitudeE6() /1E6 ,
        //    Toast.LENGTH_SHORT).show();
      if(isFrom){
        List<Overlay> addmapOverlays = addmapView.getOverlays();
        if(addmapOverlays.size()>2){
        addmapOverlays.remove(2);
        addmapOverlays.remove(1);
        }else if(addmapOverlays.size()>1){
        addmapOverlays.remove(1);    }
        addmapOverlays.add(new MarkerOverlay(p));
        Toast.makeText(getBaseContext(),
              "Начальная точка" ,
              Toast.LENGTH_SHORT).show();
        srcGeoPoint = new GeoPoint((int) (p.getLatitudeE6()),(int) (p.getLongitudeE6()));
        addmapView.invalidate();}
      else if(isTo){
        List<Overlay> addmapOverlays = addmapView.getOverlays();
        if(addmapOverlays.size()>2)
              addmapOverlays.remove(2);
        addmapOverlays.add(new MarkerOverlay2(p));
        Toast.makeText(getBaseContext(),
              "Конечная точка" ,
              Toast.LENGTH_SHORT).show();

        destGeoPoint = new GeoPoint((int) (p.getLatitudeE6()),(int) (p.getLongitudeE6()));
        addmapView.invalidate();  }   }
    return false;}       }
```

```
class MarkerOverlay extends Overlay{
    private GeoPoint p;
    public MarkerOverlay(GeoPoint p){
        this.p = p;}

    @Override
    public boolean draw(Canvas canvas, AddmapView addmapView,
            boolean shadow, long when){
        super.draw(canvas, addmapView, shadow);

        //---translate the GeoPoint to screen pixels---
        Point screenPts = new Point();
        addmapView.getProjection().toPixels(p, screenPts);

        //---add the marker---
        Bitaddmap bmp = BitaddmapFactory.decodeResource(getResources(),
R.drawable.pin_green);
        canvas.drawBitaddmap(bmp, screenPts.x, screenPts.y-50, null);
        return true;}}

class MarkerOverlay2 extends Overlay{
    private GeoPoint p;
    public MarkerOverlay2(GeoPoint p){
        this.p = p;}

    @Override
    public boolean draw(Canvas canvas, AddmapView addmapView,
            boolean shadow, long when){
        super.draw(canvas, addmapView, shadow);

        //---translate the GeoPoint to screen pixels---
        Point screenPts = new Point();
        addmapView.getProjection().toPixels(p, screenPts);

        //---add the marker---
        Bitaddmap bmp = BitaddmapFactory.decodeResource(getResources(),
R.drawable.pin_red);
        canvas.drawBitaddmap(bmp, screenPts.x, screenPts.y-50, null);
        return true;}}
```

```
@Override
protected boolean isRouteDisplayed() {
   // TODO Auto-generated method stub
   return false;
}

@Override
     public boolean onCreateOptionsMenu(Menu menu) {
     menu.add(0, 1, 1, "StreetView");
             menu.add(0, 2, 2, "Satellite");
             return super.onCreateOptionsMenu(menu);
     }

     @Override
     public boolean onOptionsItemSelected(MenuItem item) {
             switch(item.getItemId()){
                     case 2:
                             addmapView.setTraffic(false);
                             addmapView.setStreetView(false);
                             addmapView.setSatellite(true);
                             break;
                     case 1:
                             addmapView.setTraffic(false);
                             addmapView.setSatellite(false);
                             addmapView.setStreetView(true);
                             break;
                     case 3:
                             break;
             }
             return super.onOptionsItemSelected(item);}

private class connectAsyncTask extends AsyncTask<Void, Void, Void>{
   private ProgressDialg progressDialg;
   @Override
   protected void onPreExecute() {
      // TODO Auto-generated method stub
      super.onPreExecute();
      progressDialg = new ProgressDialg(_activity);
      progressDialg.setMessage("Fetching route, Please wait...");
      progressDialg.setIndeterminate(true);
      progressDialg.show();}
```

```
@Override
protected Void doInBackground(Void... params) {
    // TODO Auto-generated method stub
    fetchData();
    return null;
}
@Override
protected void onPostExecute(Void result) {
    // TODO Auto-generated method stub
    super.onPostExecute(result);
    if(doc!=null){
        Overlay ol = new MyOverLay(_activity,srcGeoPoint,srcGeoPoint,1);
        mOverlays.add(ol);
        NodeList _nodelist = doc.getElementsByTagName("status");
        Node node1 = _nodelist.item(0);
        String _status1  = node1.getChildNodes().item(0).getNodeValue();
        if(_status1.equalsIgnoreCase("OK")){
            NodeList _nodelist_path =
doc.getElementsByTagName("overview_polyline");
            Node node_path = _nodelist_path.item(0);
            Element _status_path = (Element)node_path;
            NodeList _nodelist_destination_path =
_status_path.getElementsByTagName("points");
            Node _nodelist_dest = _nodelist_destination_path.item(0);
            String _path =
_nodelist_dest.getChildNodes().item(0).getNodeValue();
            List<GeoPoint> _geopoints = decodePoly(_path);
            GeoPoint gp1;
            GeoPoint gp2;
            gp2 = _geopoints.get(0);
            Log.d("_geopoints",":::"+_geopoints.size());
            for(int i=1;i<_geopoints.size();i++) // the last one would be crash

            {gp1 = gp2;
                gp2 = _geopoints.get(i);
                Overlay ol1 = new MyOverLay(gp1,gp2,2,Color.BLUE);
                mOverlays.add(ol1);}
            Overlay ol2 = new
MyOverLay(_activity,destGeoPoint,destGeoPoint,3);
            mOverlays.add(ol2);
```

```
                    progressDialg.dismiss();
                }else{
                    showAlert("Unable to find the route");}

                Overlay ol2 = new
MyOverLay(_activity,destGeoPoint,destGeoPoint,3);
                mOverlays.add(ol2);
                progressDialg.dismiss();
                addmapView.scrollBy(-1,-1);
                addmapView.scrollBy(1,1);}else{
                showAlert("Unable to find the route");}}}
        Doc doc = null;
        private void fetchData()

        {StringBuilder urlString = new StringBuilder();

urlString.append("http://addmaps.google.com/addmaps/api/directions/xml?origin=");
            urlString.append(
Double.toString((double)srcGeoPoint.getLatitudeE6()/1.0E6 ));
            urlString.append(",");
            urlString.append(
Double.toString((double)srcGeoPoint.getLongitudeE6()/1.0E6 ));
            urlString.append("&destination=");//to
            urlString.append(
Double.toString((double)destGeoPoint.getLatitudeE6()/1.0E6 ));
            urlString.append(",");
            urlString.append(
Double.toString((double)destGeoPoint.getLongitudeE6()/1.0E6 ));
            urlString.append("&sensor=true&mode=driving");
            Log.d("url","::"+urlString.toString());
            HttpURLConnection urlConnection= null;
            URL url = null;
            Try

            {url = new URL(urlString.toString());
                urlConnection=(HttpURLConnection)url.openConnection();
                urlConnection.setRequestMethod("GET");
                urlConnection.setDoOutput(true);
                urlConnection.setDoInput(true);
                urlConnection.connect();
```

```
            DocBuilderFactory dbf = DocBuilderFactory.newInstance();
            DocBuilder db = dbf.newDocBuilder();
            doc                                    =                                    (Doc)
db.parse(urlConnection.getInputStream());//Util.XMLfromString(response);
        }catch (MalformedURLException e){
            e.printStackTrace();
        }catch (IOException e){
            e.printStackTrace();
        }catch (ParserConfigurationException e){
            e.printStackTrace(); }
        catch (SAXException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();    }  }
    private List<GeoPoint> decodePoly(String encoded) {

        List<GeoPoint> poly = new ArrayList<GeoPoint>();
        int index = 0, len = encoded.length();
        int lat = 0, lng = 0;

        while (index < len) {
            int b, shift = 0, result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lat += dlat;

            shift = 0;
            result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lng += dlng;

            GeoPoint p = new GeoPoint((int) (((double) lat / 1E5) * 1E6),
```

```
                    (int) (((double) lng / 1E5) * 1E6));
                poly.add(p);}
            return poly;
            }
            private void showAlert(String message){
                AlertDialg.Builder alert = new AlertDialg.Builder(_activity);
                alert.setTitle("Error");
                alert.setCancelable(false);
                alert.setMessage(message);
                alert.setPositiveButton("Ok",new DialgInterface.OnClickListener() {
                    @Override
                    public void onClick(DialgInterface Dialg, int which) {
                        // TODO Auto-generated method stub}});
                alert.show();}
            private Drawable getDrawable(String fileName){
                return
Drawable.createFromStream(_activity.getClass().getClassLoader().getResourceAsStr
eam(fileName), "pin");}

                @Override
                public void onClick(View v) {
                        switch(v.getId()){
                        case R.id.from:
                                if(mOverlays.size()>1){
                                        for(int i=mOverlays.size();i>1;i--){
                                                mOverlays.remove(i-1);}}
                                isFrom = true;
                                isTo = false;
                                break;
                        case R.id.to:
                                if(mOverlays.size()>2){
                                        for(int i=mOverlays.size();i>1;i--){
                                                mOverlays.remove(i-1);}}
                                isFrom = false;
                                isTo = true;
                                break;
                        case R.id.search:
```

Продолжение приложения В

```
                                isFrom = false;
```

```
                    isTo = false;
                    if(destGeoPoint==null || srcGeoPoint==null ){
                        Toast.makeText(getBaseContext(),
                "Выберите начальную и конечную точку" ,
                Toast.LENGTH_SHORT).show();}
            connectAsyncTask _connectAsyncTask = new
connectAsyncTask();
        _connectAsyncTask.execute();
                    break;}}}
```

The main xml file which show views on the adandroid devices. It consist of three buttons and addmapView.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:adandroid="http://schemas.adandroid.com/apk/res/adandroid"
    xmlns:tools="http://schemas.adandroid.com/tools"
    adandroid:layout_width="match_parent"
    adandroid:layout_height="match_parent"
    tools:context=".Activity_addmap"
    adandroid:orientation="vertical">
        <LinearLayout
            adandroid:id="@+id/forSomething"
            adandroid:layout_height="40dp"
            adandroid:layout_width="fill_parent"
            adandroid:weightSum="1.0"
            adandroid:orientation="horizontal"
            adandroid:background="@drawable/top_layout">
             <Button
                adandroid:id="@+id/from"
                adandroid:padding="2dip"
                adandroid:layout_margin="4dip"
                adandroid:layout_width="30dp"
                adandroid:layout_height="30dp"
                  adandroid:layout_gravity="center_vertical|center_horizontal"
                adandroid:layout_weight="0.1"
                adandroid:text="From"
                adandroid:textColor="#ffffff"
                adandroid:textSize="14sp"
```

Продолжение приложения В

```
                adandroid:background="@drawable/black_button"
                adandroid:onClick="backFunction"/>
            <Button
                adandroid:id="@+id/to"
                adandroid:padding="2dip"
                adandroid:layout_margin="4dip"
                adandroid:layout_width="30dp"
                adandroid:layout_height="30dp"
                  adandroid:layout_gravity="center_vertical|center_horizontal"
                adandroid:layout_weight="0.1"
                adandroid:text="To"
                adandroid:textColor="#ffffff"
                adandroid:textSize="14sp"
                adandroid:background="@drawable/black_button"
                adandroid:onClick="backFunction"/>
            <Button
                adandroid:id="@+id/search"
                adandroid:padding="2dip"
                adandroid:layout_margin="4dip"
                adandroid:layout_width="wrap_content"
                adandroid:layout_height="30dp"
                adandroid:layout_gravity="center_vertical|center_horizontal"
                adandroid:layout_weight="0.1"
                adandroid:text="Поиск"
                adandroid:textColor="#ffffff"
                adandroid:textSize="14sp"
                adandroid:background="@drawable/black_button"
                adandroid:onClick="backFunction"/>
          </LinearLayout>
          <com.google.adandroid.addmaps.AddmapView
        adandroid:id="@+id/addmapView"
        adandroid:layout_width="fill_parent"
        adandroid:layout_height="fill_parent"

        adandroid:enabled="true"
        adandroid:clickable="true"
        adandroid:apiKey="0_Qbc7C7LvOUt-1Zqvng0gB8TAZxUa-L0-
5sl6A"/>

    </LinearLayout>
```