МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра информационных систем

ДОПУЩЕНО К ЗАЩИТЕ ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
И.О. заведующего кафедрой
Профессор, д.т.н.
_____ И.Н. Глухих
_____24.06_____ 2016

Магистерская диссертация

ОПРЕДЕЛЕНИЕ ОПТИМАЛЬНОГО МАРШРУТА ПРОКЛАДКИ ТРУБОПРОВОДА

09.04.03 «Прикладная информатика в экономике»

Выполнил работу
Студент 2 курса
Очной формы обучения

Севальнева
(подпись)
Севальнева Екатерина Владимировна

Научный руководитель
Доцент кафедры ИС,
к.т.н.

(подпись)
Полищук Игорь Николаевич

Начальник отдела по
разработке ПО
ООО «ЭК «Восток»

(подпись)
Бобков Илья Евгеньевич

г. Тюмень 2016

Оглавление

ВВЕДЕНИЕ 5
1. СОВРЕМЕННОЕ СОСТОЯНИЕ ПРОБЛЕМЫ 12
1.1. Проектирование трубопроводов 12
1.2. Проблема определения оптимального маршрута 14
1.3. Обзор законодательных и нормативно-правовых документов 16
2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ВЫБОРА МЕТОДОВ И АЛГОРИТМОВ 20
2.1. Основная терминология 20
2.2. Построения графа связности по данным растровой поверхности 21
2.2.1. Шаблон ладьи 23
2.2.2. Шаблон ферзя 23
2.2.3. Шаблон коня 24
2.2.4. Построение оптимального маршрута с использованием разных шаблонов 25
2.3. Анализ сложности алгоритмов 27
2.4. Алгоритмы поиска оптимального пути 35
2.4.1. Алгоритм Дейкстры р 35
2.4.2. Модифицированный метод Дейкстры (изменение целевой функции) 37
2.5. Алгоритм поиска A* 38
2.5.1. Алгоритм поиска с итеративным углублением (Iterative Deeping A*, IDA*, A* с итеративным углублением) 40
2.5.2. Поиск в ширину (BFS, Breadth-First Search) 41
2.5.3. Алгоритм поиска пути Jump Point Search 43
2.5.4. Что используется в ArcGis 44
2.6. Выбор алгоритма для реализации 45
2.7. Выбор структуры данных 46
2.7.1. Hash table (хэш-таблица) 47
2.7.2. Хеширование с цепочками 49

2.7.3. Хеширование с открытой адресацией 50
3. Анализ входных данных и вариантов оптимизации алгоритма A* 52
3.1. Входные данные 52
3.1.1. Векторные данные 52
3.1.2. Растровые данные 52
3.1.3. Анализ векторных и растровых типов данных 53
3.1.4. Анализ геоданных 54
3.2. Анализ алгоритма A* 56
3.2.1. Основная терминология 56
3.2.2. Начало поиска 57
3.2.3. Вычисление стоимости узлов 58
3.2.4. Поиск наилучшего узла 59
3.2.5. Обратная трассировка для нахождения пути 59
3.3. Оптимизация A* 60
3.3.1. Лучевой поиск 60
3.3.2. Алгоритм последовательных приближений 60
3.3.3. Метрика h(n) 60
3.3.4. Использование «запрещенных» областей 61
3.3.5. Параллельное программирование 62
4. Разработка приложения 65
4.1. Проектирование приложения 65
4.2. Последовательность действий 67
4.3. Создание карты 68
4.4. Общие сведения о программной среде ArcGIS 69
4.5. Преобразование векторных объектов в растровые с помощью API ArcGIS 71
4.6. Преобразование матрицы в граф связности 72
4.7. Модификация алгоритма 73
4.8. Основные экраны приложения 77
ЗАКЛЮЧЕНИЕ 81

ПРИЛОЖЕНИЕ 1. 82
Видение продукта..... 82
Введение (Introduction) 82
Позиционирование (Positioning)..... 83
Заинтересованные стороны и пользователи (Stakeholder and User Descriptions) 84
Обзор продукта (Product Overview)..... 86
Функциональные возможности продукта (Product Features)..... 87
Ограничения (Constraints) 87
Диапазон качества (Quality Ranges) 87
Очередность и приоритет (Precedence and Priority)..... 88
Другие требования к продукту (Other Product Requirements)..... 88
Требования к документации (Documentation Requirements)..... 88
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 89

ВВЕДЕНИЕ

На протяжении долгого периода времени Россию можно было назвать страной с ориентацией преимущественно на сырьевую экономику. Но переломный 2014 год заставляет менять стратегию и начинать создавать условия для интенсивного экономического роста, вместо экстенсивного.

Украинские события в 2014 году, Крымский кризис, введенные из-за него санкции и смена основного направления рынков сбыта с запада на восток стали причиной закрытия проекта строительства газопровода «Южный поток» и заключения нового соглашения о строительстве морского газопровода через Черное море по направлению к Турции.

В 2014 году Министерство торговли и президент США под давлением со стороны Евросоюза и американских нефтедобывающих компаний впервые за 40 лет[30] разрешили компаниям Pioneer Natural Resources и Enterprise Product Partners экспортировать сырую нефть и газ не только в Канаду и Мексику, но и в другие страны.

Возможные проблемы с продажей газа в Европу из-за запланированного на 2016 год начала экспорта дешевого сланцевого газа из США [26], грозят снизить Российские газовые доходы на 27% [41], а топливно-энергетические товары, как известно, являются основой Российского экспорта.

Вместе с этим в России очень много неосвоенной нефти в Арктическом шельфе [27] и в неразработанных месторождениях Восточной Сибири, а также большие запасы сланцевой нефти в Баженовской свите. Снижение стоимости на добычу и доставку углеводородов, приведение совокупной себестоимости нефтепродуктов к цене ниже рыночной, может оказать существенное влияние на текущую мировую экономическую и политическую ситуацию.

«Роснефть» — лидер российской нефтяной отрасли и крупнейшая публичная нефтегазовая корпорация мира. Основными видами деятельности ОАО «НК «Роснефть» являются поиск и разведка месторождений

углеводородов, добыча нефти, газа, газового конденсата, реализация проектов по освоению морских месторождений, переработка добытого сырья, реализация нефти, газа и продуктов их переработки на территории России и за ее пределами.

ОАО «НК «Роснефть» занимает лидерские позиции в процессах качественной модернизации и инновационных изменений в российской нефтегазовой отрасли. Усилия Компании направлены на развитие интеллектуального и технологического потенциала отрасли на основе мощного фундамента российской нефтегазовой школы — одной из лучших в мире — и в партнерстве с ведущими компаниями международного нефтяного бизнеса.

Программа инновационного развития НК «Роснефть» — важнейшая часть концепции ее развития. Инновации направлены на модернизацию производственной базы, создание и внедрение новых технологий для решения производственных задач. Это восполнение запасов, увеличение коэффициента извлечения нефти, максимально полное использование попутного нефтяного газа, эффективная реализация шельфовых проектов, увеличение глубины переработки нефти, повышение энергоэффективности, минимизация капитальных и операционных затрат, а также обеспечение экологической и промышленной безопасности.

Существует четыре основных вида транспортировки нефти: водный, автомобильный, железнодорожный и трубопроводный. Каждый вид имеет свои особенности.

Основной объем международных перевозок нефти осуществляют танкеры. Современные танкеры – это гигантские суда. Впечатляющие размеры объясняются экономическим «эффектом масштаба». Стоимость перевозки одного барреля нефти на морских судах обратно пропорциональна их размерам. Кроме того, число членов экипажа большого и среднего танкера

примерно одинаково. Поэтому корабли-гиганты значительно сокращают расходы компаний на транспортировку. Однако не все морские порты в состоянии принять у себя супер-танкер. Для таких гигантов нужны глубоководные порты. Так, например, большинство российских портов из-за ограничений по фарватеру не способно принимать танкеры с дедвейтом более 130-150 тысяч тонн.

Автомобильный транспорт может использоваться для перевозки небольших объемов нефти с удаленных нефтепромыслов до пунктов сбора. Из-за чрезвычайно высокой стоимости автомобильный транспорт используют только на ранней стадии разработки месторождений до ввода в строй нефтепромысловых трубопроводов.

Более широко распространено применение автомобильного транспорта для доставки потребителю конечных нефтепродуктов. Преимущество такого способа транспортировки – его высокая гибкость. В частности, таким способом доставляют автомобильное топливо на АЗС.

Еще один вид транспортировки нефти – по железной дороге. Это быстрый всесезонный способ. В России его используют, чтобы доставить нефть из Западной Сибири на Дальний Восток, Южный Урал и в страны Центральной Азии. Из Урала нефть везут на Запад, на Северный Кавказ и в Новороссийск. Однако для доставки «черного золота» по железной дороге требуется в 10 раз больше трудозатрат, чем для ее транспортировки по нефтепроводам. Поэтому даже в странах с разветвленной железнодорожной сетью этот способ перевозки нефти является второстепенным.

Трубопроводы служат для транспортировки больших количеств нефти, нефтепродуктов и сжиженных газов в одном направлении. Это наиболее дешевый способ транспортировки нефти. Нефтепроводы соединяют добывающие скважины с нефтепромысловой инфраструктурой. По ним нефть поступает на нефтеперерабатывающие и нефтехимические заводы.

Строительство и эксплуатация объектов наземной инфраструктуры входит в 20 приоритетных направлений инновационных исследований НК «Роснефть».

Исходя из текущей экономической ситуации в стране, становится очевидно, что необходимо снизить себестоимость нефти, большая часть которой складывается из расходов на транспортировку.

В России основным способом транспортировки энергоносителей являются трубопроводы. Анализ преимуществ и недостатков возможных видов транспортировки приведен в табл. 1. Основными параметрами были выбраны капитальные и эксплуатационные затраты, а также грузоподъемность, потому что они в конечном итоге больше всего влияют на стоимость транспортировки УВ. Параметр требовательность к погодным условиям был введен чтобы показать зависимость осуществления транспортировки от сезона, что в конечном итоге влияет на бесперебойность работы.

Таблица 1. Анализ способов транспортировки УВ по ключевым параметрам

Параметр	Автомобильный транспорт	Железнодорожный транспорт	Судоходный транспорт	Трубопроводный транспорт
Капитальные затраты	2	3	1	4
Эксплуатационные затраты	4	3	2	1
Грузоподъемность	4	3	2	1
Требовательность к погодным условиям	3	2	4	1
Среднее значение	3,25	2,75	2,25	1,75

Поскольку применение трубопроводов экономически выгодно, а работают они в любую погоду и в любое время года, это средство транспортировки нефти действительно незаменимо – особенно для России, с ее огромными территориями и сезонными ограничениями на использование водного транспорта.

Трубопроводы по своему назначению делятся на 3 типа:

- Внутренние (внутрипромысловые) – соединяют различные объекты и установки на промыслах, НПЗ и нефтебазах.
- Местные (межпромысловые) – по сравнению с внутренними имеют большую протяженность (до нескольких десятков км), соединяют нефтепромыслы или НПЗ с головной станцией магистрального нефтепровода.
- Магистральные характеризуются большой протяженностью (сотни и тысячи км), поэтому перекачка ведется несколькими станциями, расположенными на трассе.

Выбор трассы трубопроводов должен производиться по критериям оптимальности, при этом в качестве таких критериев принимают затраты на сооружение, техобслуживание и ремонт трубопровода при эксплуатации, учитывая и затраты по обеспечению охраны окружающей среды. Кроме того, учитывается металлоёмкость, безопасность, заданное время строительства и наличие дорог.

В ОАО «РОСНЕФТЬ» проектные работы, предшествующие любому строительству, начинаются, как правило, с определения конечных объемов работ и составления сметы на проектно-изыскательские работы (ПИР). В нефтегазовой отрасли можно выделить 4 основных стадии строительных проектных работ:

- Обоснование инвестиций (ОИ) в строительство или реконструкцию, или технико-экономическое обоснование строительства (ТЭО-проект).
- Основные технические решения (ОТР).
- Разработка проектной документации (стадия «П»).
- Разработка рабочей документации (стадия «РД») для строительства (реконструкции) объекта.

Без обоснования инвестиций проект будет приостановлен в самом начале, но стоимость такой работы у подрядных организаций достаточно велика: порядка 5-20% стоимости разработки всей проектной документации, временные затраты при этом составляют от 2 до 6 недель.

Выбор маршрута прокладки — это первый существенный шаг в процессе проектирования и строительства трубопровода, который может оказать значительное влияние на строительство и функционирование трубопровода в целом, поэтому оптимизация этого процесса может существенно повлиять на финансовые и материальные ресурсы, необходимые для строительства трубопровода. Таким образом, эффективный процесс выбора маршрута прокладки важен для минимизации экономических потерь.

В связи с этим становится очевидным, что определение оптимального маршрута прокладки трубопровода является актуальной экономической и научно-технической задачей.

Использование автоматизированных методов для построения маршрутов фактически не распространено. Классические методики основываются на бумажно-аналоговых технологиях. При незначительном изменении входного набора данных (смена начальных и конечных пунктов, появление новой или просто другой информации) все операции по подготовке данных приходится делать заново, и, как правило, вручную.

Цель работы – уменьшение стоимости работ на этапе ТЭО проекта разработки месторождения углеводородов, за счет автоматизации процесса определения оптимального маршрута прокладки внутрипромыслового трубопровода.

Для достижения цели, планируется решить следующие задачи:

- Изучить литературу по предметной области.
- Изучить имеющееся программное обеспечение и инструменты по предметной области.

- Провести анализ существующих методов преобразования массивов в графы;
- Провести сравнительный анализ существующих алгоритмов построения оптимального расстояния;
- Реализовать преобразование векторных объектов в единую растровую карту со стоимостными оценками;
- Реализовать алгоритм оптимального построения маршрута A star;
- Реализовать веб-приложение в котором будет возможность построения и редактирования маршрута;
- Реализовать экспорт полученных данных в табличном и графическом виде;
- Разработать интерфейс приложения;
- Провести апробацию приложения.

В работе планируется рассмотреть построение маршрутов внутрипромысловых трубопроводов на территории Западной Сибири, на основе однокритериального подхода, как наиболее оптимального, так как при ТЭО самым важным критерием является стоимость.

1. СОВРЕМЕННОЕ СОСТОЯНИЕ ПРОБЛЕМЫ

Первым шагом для выбора и последующего анализа литературы стал поиск научно-исследовательских работ на темы, связанные с проектированием и строительством линейных объектов. Тема оптимизации маршрута прокладки трубопровода затрагивается в кандидатских диссертациях Кузнецова Р.М. «Определение оптимального маршрута прокладки трубопровода» [35] и Рыльского И.А. «Геоинформационное обеспечение автоматизированного трассирования трубопроводов».

На основе литературных источников, использованных для написания вышеуказанных диссертаций, а также нескольких других работ [20, 8, 12] и статей [13] было выделено несколько больших групп информационных источников для последующего анализа:

- статьи, касающиеся проектирования, строительства линейных инженерных сооружений или построения их моделей с учетом различных факторов;
- проектно-техническая документация и технические регламенты по проектированию инфраструктуры месторождений углеводородов;
- документы по оценке экономической эффективности проектов;
- литература по теории проектирования и строительства трубопроводов;
- технологии и организация трубопроводного строительства в условиях обводненной и заболоченной местности;
- проектные технологические документы на разработку нефтяных месторождений;
- законодательные и нормативно-правовые документы, регулирующие деятельность предприятий нефтедобывающей промышленности.

1.1. Проектирование трубопроводов

Бородавкина П.П. и Алиева Р.А. можно считать ключевыми деятелями в теории проектирования трубопроводов в России, к тому же Бородавкин П.П. является автором теории проектирования магистральных трубопроводов в

сложных условиях [7, 9, 10]. Большая часть их работ направлена на изучение и создание проектной и рабочей документации для строительства (реконструкции) объектов промыслов.

В книгах Алиева и Бородавкина описывается теория проектирования и строительства трубопроводов. Этими авторами проведена классификация типов трубопроводов и типов местности, через которые они проходят. Так же приведены примеры выбора различных критериев оптимальности: аддитивных и неаддитивных. К аддитивным можно отнести критерии по капитальным или приведенным затратам, а также критерии по времени строительства, к неаддитиным – вероятность завершения строительства в заданный срок.

В книге Алиева Р.А. «Трубопроводный транспорт нефти и газа» [7] обоснован выбор самого выгодного способа транспортировки нефтяных грузов, а также особенности каждого вида транспорта. Транспортировка с помощью трубопроводов имеет ряд неоспоримых преимуществ по сравнению с другими вида транспортировки, к числу которых можно отнести: трасса трубопровода короче трасс других видов транспорта, трубопроводный транспорт, в отличие от других – непрерывный, меньше потери нефти и нефтепродуктов, трубопроводный транспорт наиболее механизированный и легче других поддается автоматизации.

Алиев Р.А. отмечает, что выбор того или иного вида транспорта осуществляется технико-экономическим сравнением вариантов[7]. Важнейшими же показателями являются капитальные затраты и эксплуатационные расходы, так же приведены примеры расчётов этих показателей.

Большинство статей про оптимизацию маршрутов, написанных в последнее время, касаются изучения методологического обеспечения проектирования трасс. В статьях рассматриваются подходы к оптимизации

прокладки различных линейных объектов строительства, а также разработка геоинформационного обеспечения.

Рыльский И.А. в статье «Оптимизация трассы трубопровода с помощью ГИС»[19] предложил метод, основанный на факторном анализе с использованием средств ГИС. Данный метод дает возможность проводить оптимизацию трассы не в узком коридоре вдоль линии, принятой на этапе обоснования инвестиций, а по всей территории региона строительства. При выполнении оптимизации используются данные о рельефе, углах наклона и экспозиции склонов, кривизне поверхности, высотах, характеристиках грунтов, сейсмичности и геологическом строении, растительности, мерзлотных процессах и по 14 прочим характеристикам.

К сожалению, предложенная Рыльским И.А. методика не может использоваться на практике в полной мере. Причиной для этого послужил анализ предложенного в статье списка факторов и объектов, влияющих на стоимость строительства. Например, геологическое строение, не оказывает существенного влияния на выбор маршрута строительства трубопровода. Так же, хочется отметить, что автор статьи в своей работе в качестве примера рассматривал идеализированный вариант – наличие полного набора данных по всем показателям, на практике такой ситуации может не быть.

1.2. Проблема определения оптимального маршрута

В кандидатской диссертации Кузнецова Р.Н. «Определение оптимального маршрута прокладки газопровода» описано построение оптимального маршрута прокладки газопровода с учетом нескольких влияющих факторов.

Автор рассматривает две системы представления данных в геоинформационных системах: векторную и растровую. При анализе сетей удобно использовать векторную систему, т.к. это позволяет представлять данные в виде графа. Если же при построении маршрута необходимо

принимать во внимание большое количество факторов, то лучше всего использовать растровую систему.

Кузнецов Р.Н. приводит последовательность действий, которые необходимо сделать, чтобы построить оптимальный маршрут: определить факторы, влияющие на прокладку газопровода, оценить весовые коэффициенты факторов, рассчитать стоимостную карту и проложить путь, соответствующий наименьшей стоимости.

В своей работе Кузнецов строит граф связности ячеек сетки стоимостной карты и приводит алгоритм расчета поверхности накопленной стоимости, разработанный на базе алгоритма Дейкстры.

К числу достоинств этой диссертации можно отнести то, насколько автор хорошо осветил тему, разработанные им алгоритмы и методики. Алгоритм при обходе итоговой поверхности накопленной стоимости одновременно учитывает данные о направлении прокладки и реализует целевую функцию минимальной стоимости прокладки газопровода.

К недостаткам работы относятся отсутствие обоснования выбора алгоритма Дейкстры в качестве базового алгоритма для модификации, а также то, что автор не затронул тему улучшения быстродействия алгоритма.

В настоящее время методы трассирования и выбора оптимального расположения трассы трубопровода (на стадии рабочего проектирования) разработаны достаточно хорошо, однако в этом случае трасса трубопровода не может выходить за пределы коридора, выбранного на стадии технико-экономического обоснования проекта (ТЭО). В то же время методология выбора трассы трубопровода на этапе ТЭО разработана недостаточно. Методики подготовки этих данных к использованию: сбору, анализу и цифровой интерпретации пространственного распределения характеристик окружающей среды были рассмотрены в диссертационной работе Рыльского И.А.

В кандидатской диссертации Рыльского И.А. «Геоинформационное обеспечение автоматизированного трассирования трубопроводов» описаны возможности более широкого применения ГИС в сочетании с методиками, предложенными еще советскими авторами, в частности Бородавкиным П.П. Сделан упор на автоматизацию процесса трассировки с учетом разработанных Рыльским научно-методологических подходов к количественной оценке качества предлагаемых трасс трубопроводов.

В результате анализа был выявлен ряд недостатков диссертации. Во-первых, хочется отметить, что автор приписывает себе разработку уникального метода, который основывается не на жестком классифицировании территории по типам (где стоимость строительства элементарного участка трубопровода одинакова в пределах выделенного типа территории), а на совокупном анализе различных факторов осложнения трассирования и последующей оценке их совокупного влияния для каждой точки территории.

В кандидатской диссертации Басов Е.Д. описаны возможные технологии и организация трубопроводного строительства в условиях обводненной и заболоченной местности, автор проводит анализ строительства переходов через малые преграды, а также оценку распределения малых переходов по природно-климатическим зонам.

1.3. Обзор законодательных и нормативно-правовых документов

Обоснование инвестиций в строительство трубопровода позволяет получить достаточно достоверную информацию о возможности и приблизительной стоимости реализации проектируемого объекта. При этом сроки разработки ТЭО-проекта – от 2 до 6 недель (в зависимости от величины предполагаемого объекта), а стоимость разработки составляет 15-20% от стоимости разработки проектной документации.

ТЭО-проект позволяет определить основную технологическую схему, тип и примерную стоимость основного технологического оборудования, а соответственно и размеры необходимой под строительство площадки, что в свою очередь, позволяет достаточно точно оценить затраты на реализацию объекта целиком.

Основные технические решения разрабатываются до начала работ по проектной документации и на достаточно сложные технологически насыщенные объекты. В составе ОТР обычно разрабатываются основные схемы: технологическая схема, электроснабжения, автоматизации.

ОТР позволяет согласовать основные параметры технологического и вспомогательного оборудования, что значительно сокращает количество переделок при разработке проектной документации. Срок разработки ОТР 2-3 недели и стоимость не более 10% от стоимости проектных работ.

Разработка проектной документации всегда проходит государственную и ведомственную экспертизу, поэтому ее форма должна соответствовать требованиям законодательства. Данная стадии подразумевает разработку всех необходимых разделов проектной документации, согласно Постановлению РФ №87 «О составе разделов проектной документации и требованиях к их содержанию» и Градостроительного кодекса ст.48 п.13. Документация на линейные объекты капитального строительства включает в себя 10 разделов:

1. «Пояснительная записка» (ПЗ);
2. «Проект полосы отвода» (ППО);
3. «Технологические и конструктивные решения линейного объекта. Искусственные сооружения» (ТКР);
4. «Здания, строения и сооружения, входящие в инфраструктуру линейного объекта» (ИЛО);
5. «Проект организации строительства» (ПОС);

6. «Проект организации работ по сносу (демонтажу) линейного объекта» (ПОД);
7. «Мероприятия по охране окружающей среды» (ООС);
8. «Мероприятия по обеспечению пожарной безопасности» (ПБ);
9. «Смета на строительство» (СМ);
10. «Иная документация в случаях, предусмотренных федеральными законами».

Разработка рабочей документации является самым последним этапом. В ГОСТ Р 21.1001-2009 дается определение рабочей документации как совокупности текстовых и графических документов, обеспечивающих реализацию принятых в утвержденной проектной документации технических решений объекта капитального строительства, необходимых для производства строительных и монтажных работ, обеспечения строительства оборудованием, изделиями и материалами и/или изготовление строительных изделий. В состав рабочей документации входят основные комплекты рабочих чертежей, спецификации оборудования, изделий и материалов, сметы, другие прилагаемые документы, разработанные в дополнение к рабочим чертежам основного комплекта.[1]

В СНиП 2.05.06-85 приведены строительные нормы и правила по проектированию и реконструкции магистральных трубопроводов. Нормы не распространяются на проектирование трубопроводов, прокладываемых на территории городов и других населенных пунктов, в морских акваториях и промыслах, а также трубопроводов, предназначенных для транспортирования газа, нефти, нефтепродуктов и сжиженных углеводородных газов, оказывающих коррозионные воздействия на металл труб или охлажденных до температуры ниже минус 40 °С. В тексте норм приведены категории трубопроводов в зависимости участков, по которым они проходят.

В СНиП III-42-80* даны общие положения по строительству трубопроводов. В документе приведена информация о строительстве

переходов через естественные и искусственные препятствия, в особых природных условиях:

- прокладка через болота и обводненные участки;
- прокладка трубопроводов в горных условиях
- прокладка трубопроводов в тоннелях;
- прокладка трубопроводов в просадочных грунтах;
- прокладка трубопроводов в барханных песках, на поливных землях и при пересечении соров;
- прокладка трубопроводов в вечномерзлых грунтах.

Анализ законодательных и нормативно-правовых документов, регулирующих деятельность предприятий нефтедобывающей промышленности, показал, что не все аспекты проектирования и строительства промысловых трубопроводов освещены должным образом, отсутствует четкая классификация всех возможных зон и коэффициентов удорожания по ним.

Однако, можно утверждать, что при прокладке трубопровода на территории Западной Сибири особое внимание следует обратить на следующие участки:

- территории, расположенные в непосредственной близости от населенных пунктов;
- особо охраняемые зоны;
- военные части;
- заповедники;
- болота и обводненные участки;
- территории с просадочными грунтами;
- вечномерзлые грунты.

2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ВЫБОРА МЕТОДОВ И АЛГОРИТМОВ

2.1. Основная терминология

Алгоритм — это точно определённая инструкция, последовательно применяя которую к исходным данным, можно получить решение задачи.

База данных — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, причём такое собрание данных, которое поддерживает одну или более областей применения

Линейный алгоритм — набор команд (указаний), выполняемых последовательно во времени друг за другом.

Неинформированный поиск (также слепой поиск, метод грубой силы, англ. uninformed search, blind search, brute-force search) — стратегия поиска решений в пространстве состояний, в которой не используется дополнительная информация о состояниях, кроме той, которая представлена в определении задачи. Всё, на что способен метод неинформированного поиска, — вырабатывать преемников и отличать целевое состояние от нецелевого

Параллельные вычисления — способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (одновременно).

Поиск пути (англ. Pathfinding) — термин в информатике и искусственном интеллекте, который означает определение компьютерной программой наилучшего, оптимального маршрута между двумя точками по какому-либо критерию, чаще всего это кратчайший путь.

Псевдокод — компактный (зачастую неформальный) язык описания алгоритмов, опускающий несущественные подробности и специфический синтаксис.

Растровая поверхность – это непрерывное поле значений, которые могут меняться в бесконечном количестве точек

Релаксация – процесс уменьшения пути из вершины.

Структура данных (англ. data structure) — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных в вычислительной технике.

Циклический алгоритм — алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными.

Цифровая карта (цифровая карта местности) — цифровая модель местности.

Эвристический алгоритм (эвристика) — алгоритм решения задачи, не имеющий строгого обоснования, но, тем не менее, дающий приемлемое **решение задачи в большинстве практически значимых случаев.**

2.2. Построения графа связности по данным растровой поверхности

Существует два варианта возможных входных данных: геологические данные из базы данных и карты, так как информация на картах будет более полной, то предполагается использовать их с возможностью дальнейшего внедрения работы с данными хранящимися в корпоративной базе данных.

При использовании растровых поверхностей стоимости необходимо определить метод, по которому будет строиться граф связности ячеек сетки. Как правило, граф связности строится исходя из следующих соображений:

1. Центры ячеек сетки служат узлами графа;

2. Связи между соседними ячейками служат дугами графа.[13]

Поскольку для представления растровых данных применяется регулярная сетка, граф связности строится как соединение фиксированного количества соседних ячеек сетки дугами графа. Используются различные вариации этого метода (рис. 7).

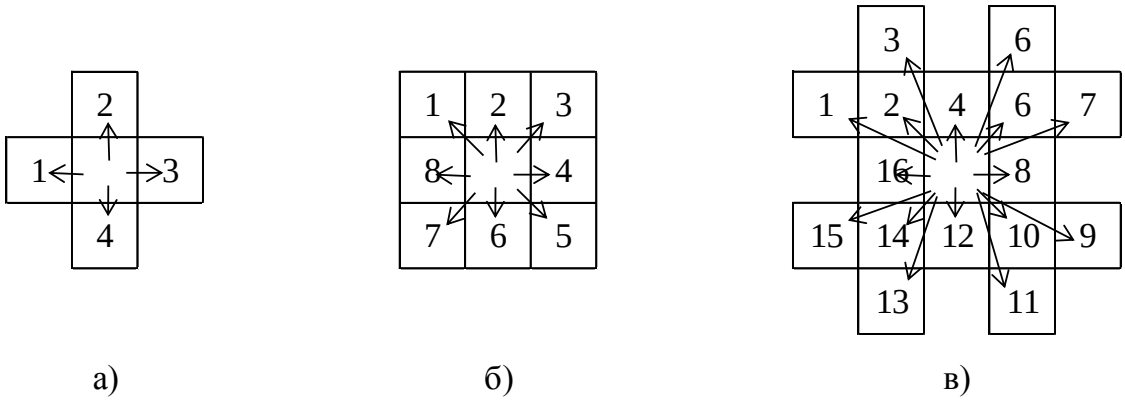

Рисунок 2.1 - Три типа шаблонов соседних ячеек в растровых форматах данных:
а — «шаблон ладьи»; б — «шаблон ферзя»; в — «шаблон коня»

Для того чтобы определиться с выбором шаблона был проведен анализ каждого из них. Анализ проводился на тестовом растре, у которого стоимость прохода по каждой ячейке была равна условной единице.

Расчет веса ребра графа определяется в зависимости от расположения целевой ячейки от центральной. При этом нужно принять во внимание тот факт, что в реальных условиях стоимость прохода у каждой ячейки может быть разной.

Так как, по поставленным выше условиям, стоимость всех ячеек одинакова, то было принято решение перейти от двумерной системы координат к одномерной системе координат, проиндексировав ячейки. Данное преобразование избавит от необходимости вести поиск по 2 значениям координат.

Таким образом, будущие вершины графа получили название в соответствии с индексом ячейки растра. Следующим шагом будет вычисление веса ребер графа.

Вес ребра графа будет изменяться в зависимости от выбранного шаблона.

2.2.1. Шаблон ладьи

Шаблон ладьи является самым простым, максимально возможное количество «соседей» вершины будет равняться четырем, соответственно, ребра будет тоже четыре. Для упрощения, каждое ребро графа можно назвать в соответствии с вершиной, к которой он ведет.

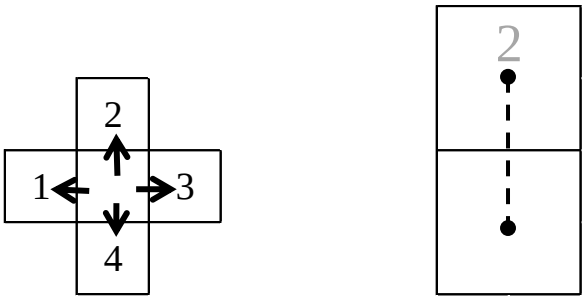

Рисунок 2.2 - Шаблон ладьи

Вес каждого ребра при использовании шаблона ладьи в общем случае будет вычисляться по формуле:

$$E_{ij} = \frac{v_i + v_{ij}}{2}$$

Где i $=\overline{1,n}$, n – количество вершин, j = $\overline{1,4}$ – по количеству ребер, соответственно.

2.2.2. Шаблон ферзя

В шаблоне ферзя часть ребер, а именно 2, 4, 6 и 8, будет считаться по той же формуле, что и в шаблоне ладьи.

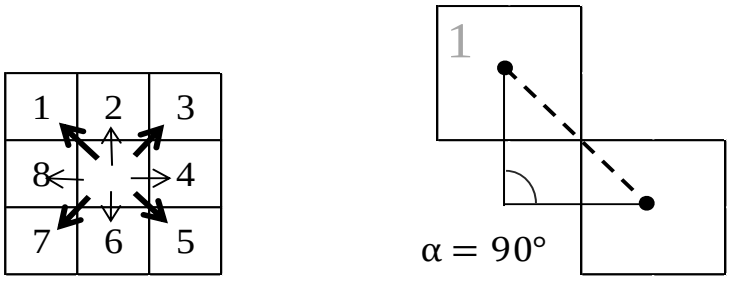

Рисунок 2.3 - Шаблон ферзя

Ребра, выделенные на рисунке (рис. 9) будут считаться как по формуле:

$$E_{ij} = \frac{\sqrt{2}}{2}(v_i + v_{ij})$$

Где v_i – вес центральной ячейки, а v_{ij} – вес соседа.

2.2.3. Шаблон коня

Вычисление веса ребер в шаблоне происходят аналогично вычислению веса ребер в предыдущих шаблонах, веса добавленных ребер (1, 3, 5, 7, 9, 11, 13 и 15) будут вычисляться по формуле:

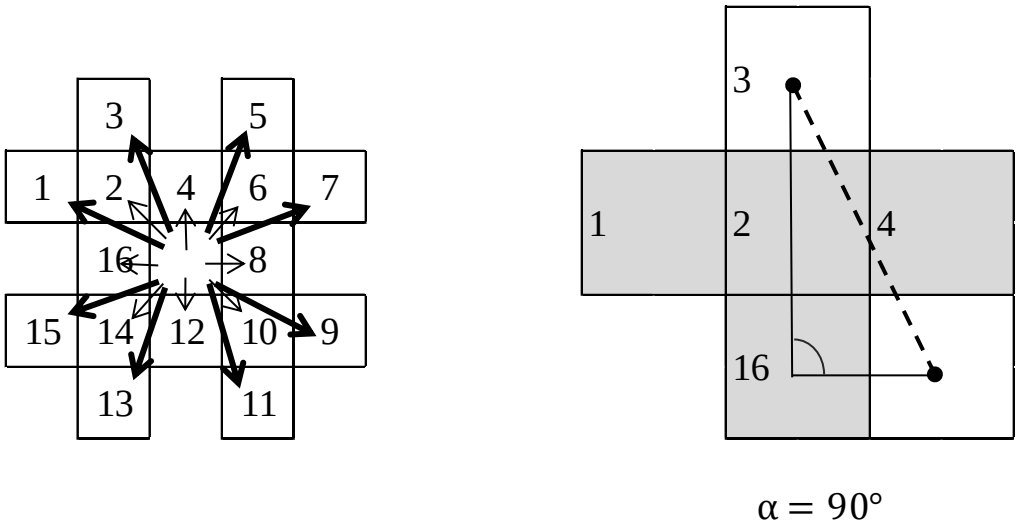

Рисунок 2.4 - Шаблон коня

Так как вершиной каждого графа будет считаться центр растра, то в ситуации, когда стоимость каждой ячейки одинакова, расчет будет сводиться к поиску расстояния от центра одной ячейки до центра другой.

$$E_3 = \sqrt{\left(\frac{a+a}{2}+a\right)^2 + \left(\frac{a+a}{2}\right)^2} = a\sqrt{5}$$

В реальности формула должна учитывать вес всех ячеек, через которые проходит

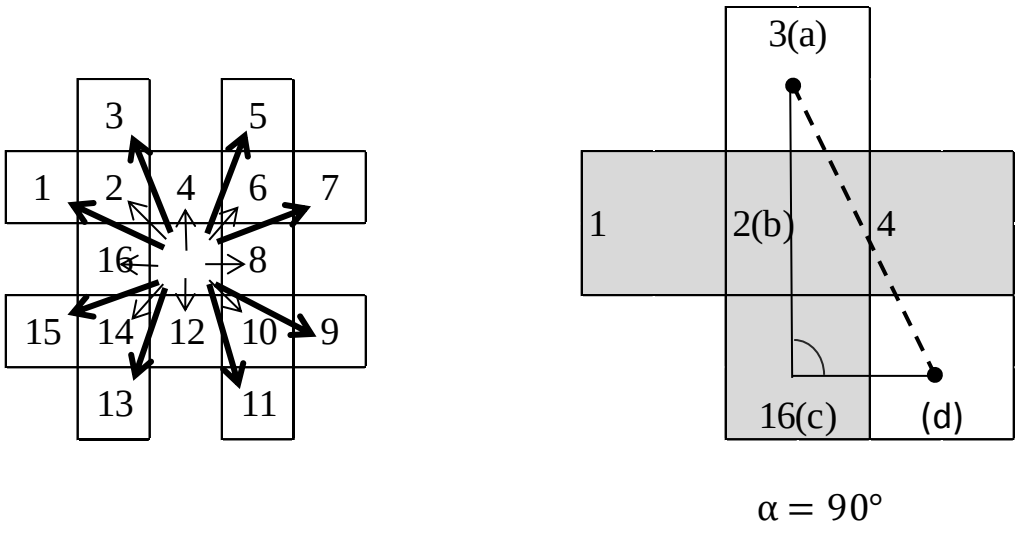

Рисунок 2.5 - Шаблон коня на растре с разными стоимостями ячеек

$$E_3 = \sqrt{\left(\frac{a}{2}+b+\frac{c}{2}\right)^2 + \left(\frac{c+d}{2}\right)^2}$$

2.2.4. Построение оптимального маршрута с использованием разных шаблонов

Рассмотрим несколько случаев поиска оптимального маршрута, стоимость прохода по каждой ячейке равна условной единице

Стоимость пути от начальной до конечной точки в примере а (рис. 12) составляет 10 условных единиц, в примере б 8 условных единиц.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

а)

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

б)

Рисунок 2.6 - Поиск оптимального пути по шаблону ладьи

Стоимость пути от начальной до конечной точки в примере а (рис.) составляет $5\sqrt{2}$ условных единиц, в примере б $3\sqrt{2}+2$ условных единиц.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

а)

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

б)

Рисунок 2.7 - Поиск оптимального пути по шаблону ферзя

Стоимость пути от начальной до конечной точки в примере а (рис. 13) составляет $5\sqrt{2}$ условных единиц, в примере б $2\sqrt{5}+\sqrt{2}$ условных единиц.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

а)

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

б)

Рисунок 2.8 - Поиск оптимального пути по шаблону ладьи

Как можно заметить шаблон ладьи самый простой для построения, но его использование не самое универсальное. Применение шаблона коня более универсально, но вместе с тем построение графа по этому шаблону требует больше ресурсов. Исходя из всего этого, было принято решение использовать шаблон ферзя.

2.3. Анализ сложности алгоритмов

Сложность алгоритма — это способ измерить, насколько быстро программа или алгоритм работают. Анализ сложности позволяет объяснить, как будет вести себя алгоритм при возрастании входного потока данных.

Существует несколько методов анализа сложности алгоритмов по времени, самыми распространенными из них являются:

- Amortized Analysis;
- Analysis of parallel algoritms;
- Asymptotic computational complexity.

Amortized Analysis (амортизационный анализ) – метод анализа алгоритмов, позволяющий осуществлять оценку времени выполнения последовательности из n операций над некоторой структурой данных. Время

выполнения усредняется по всем n операциям и оценивается среднее время выполнения одной операции в худшем случае. [36]

Analysis of parallel algoritms (анализ параллельных алгоритмов). При параллельном выполнении алгоритма разные части программы выполняются на разных вычислительных устройствах, и часто получается, что результат вычисления в одном устройстве является входом для другого устройства. Время, 4 необходимое для обмена данными между вычислительными устройствами, зависит от топологии коммуникационной подсистемы и может оказаться сопоставимым (или даже превышать) с временем, необходимым для выполнения операций алгоритма. Поэтому для параллельных алгоритмов точный анализ сложности требует учета как количества шагов алгоритма (с учетом возможности параллельного выполнения), так и необходимых пересылок. [21]

Asymptotic computational complexity (асимптотический анализ сложности алгоритмов) – метод анализа алгоритмов, в котором рассматривается поведение алгоритма при $n \to \infty$ (рис. 2.9). При асимптотическом анализе вычислительной сложности алгоритма оценивают количество операций для следующих случаев: наихудший случай (worst case) – максимальное количество операций, требуемых для обработки набора входных данных; средний случай (average case) – среднее количество операций, требуемых для обработки набора входных данных; наилучший случай (best case) – минимальное количество операций, требуемых для обработки набора входных данных. Для записи асимптотической сложности алгоритмов используются разные асимптотические обозначения, например, O-нотация (Big O).

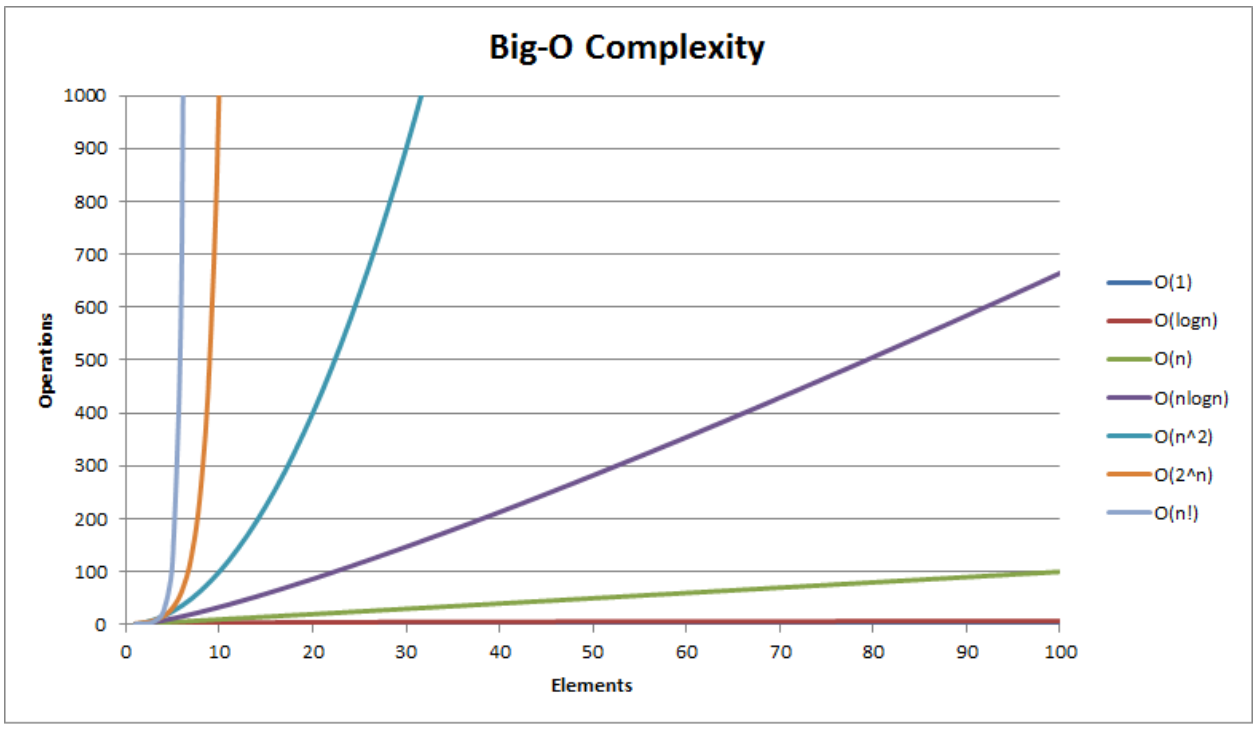

Рисунок 2.9 - График диаграммы сложности нотации big-O

Как можно заметить на рисунке 2.15 количество операций возрастает в зависимости от количества элементов. Скорость возрастания зависит от алгоритма.

Традиционно принято оценивать степень сложности алгоритма по объему используемых им основных ресурсов компьютера: процессорного времени и оперативной памяти. В связи с этим вводятся такие понятия, как time complexity (временная сложность алгоритма) и space complexity (объемная сложность алгоритма).

Параметр временной сложности становится особенно важным для задач, предусматривающих интерактивный режим работы программы, или для задач управления в режиме реального времени. Как правило, повышение точности вычислений ведет к увеличению времени.

Объемная сложность программы становится критической, когда объем обрабатываемых данных оказывается на пределе объема оперативной памяти ЭВМ. На современных компьютерах эта проблема не стоит так остро.

На время выполнения алгоритма (программы) влияют:

- Размер входных данных
- Качество реализации алгоритма на языке программирования
- Качество скомпилированного кода
- Производительность вычислительной машины

У каждого алгоритма есть параметры, определяющие размер его входных данных:

- Поиск наименьшего элемента в массиве: n–количество элементов в массиве
- Алгоритм умножения двух матриц: количества строк m и столбцов n в матрицах
- Сравнения двух строк: s_1, s_2–длина первой и второй строк
- Поиск кратчайшего пути в графе между парой вершин: n, m–количество вершин и ребер в графе

Количество операций алгоритма можно выразить как функцию от размера его входных данных: $T(n)$, $T(s_1, s_2)$, $T(n, m)$

В качестве исполнителя используется модель однопроцессорной вычислительной машины с произвольным доступом к памяти (Random Access Machine–RAM)

- Машина обладает неограниченной памятью
- Для выполнения арифметических и логических операций (+, –, *, /, %) требуется один временной шаг –такт процессора
- Обращение к оперативной памяти для чтения или записи занимает один временной шаг
- Выполнение условного перехода (if-then-else) требует вычисления логического выражения и выполнения одной из ветвей if-then-else
- Выполнение цикла (for, while, do) подразумевает выполнение всех его итераций

Существует три возможных случая:

- Лучший случай (best case)
- Худший случай (worst case)
- Средний случай (average case)

Лучший случай (best case) –это экземпляр задачи (набор входных данных), на котором алгоритм выполняет наименьшее число операций.

$$T_{best}(n) = 1$$

Худший случай (worst case) –это экземпляр задачи, на котором алгоритм выполняет наибольшее число операций.

$$T_{worst}(n) = 1$$

Средний случай (average case) –это “средний” экземпляр задачи, набор “усреднённых” входных данных.

В среднем случае оценивается математическое ожидание количества операций, выполняемых алгоритмом. Не всегда очевидно, какие входные данные считать «усреднёнными» для задачи

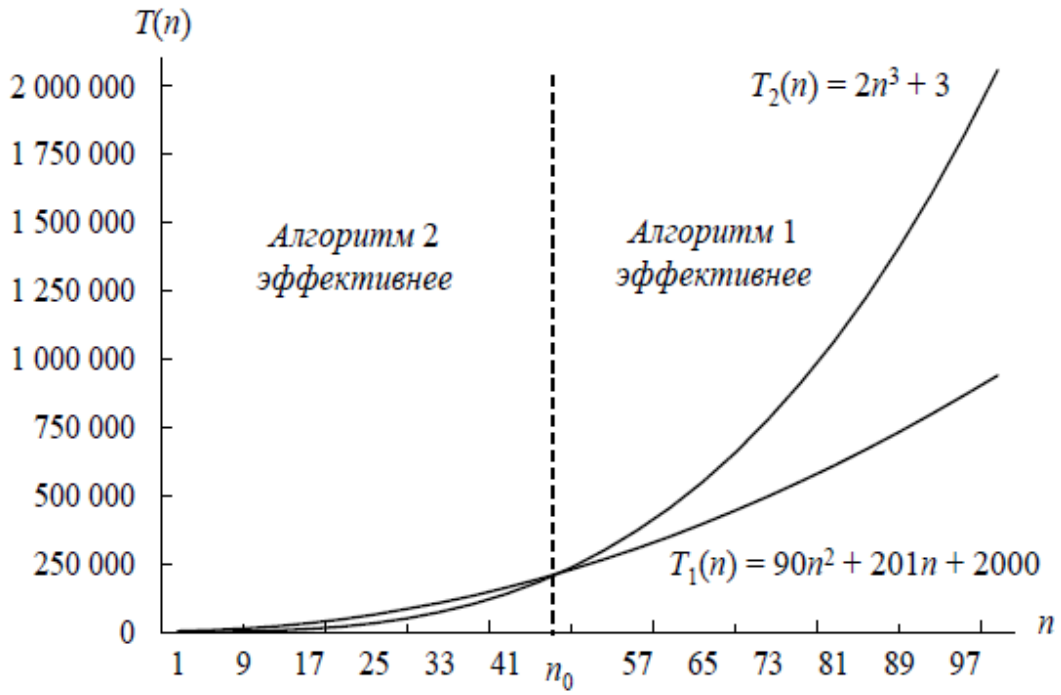

Рисунок 2.10 - Пример анализа алгоритмов

Если известно, что на вход будут поступать данные небольших размеров, то вопрос о выборе эффективного алгоритма является первостепенным – можно использовать самый “простой” алгоритм. Вопросы, связанные с эффективностью алгоритмов, приобретают смысл при больших размерах входных данных – при $n \to \infty$.

Асимптотический анализ (asymptotic analysis), который позволяет оценивать скорость роста функций T(n) при стремлении размера входных данных к бесконечности (при n→∞).

Мы можем найти такое значение n_0 при котором происходит пересечение функций $T_1(n)$ и $T_2(n)$, и на основе n_0 отдавать предпочтение тому или иному алгоритму.

Как правило, функция времени T(n) выполнения алгоритма имеет большое количество локальных экстремумов – неровный график с выпуклостями и впадинами

Проще работать с верхней и нижней оценками (границами) времени выполнения алгоритма (рис. 2.11.)

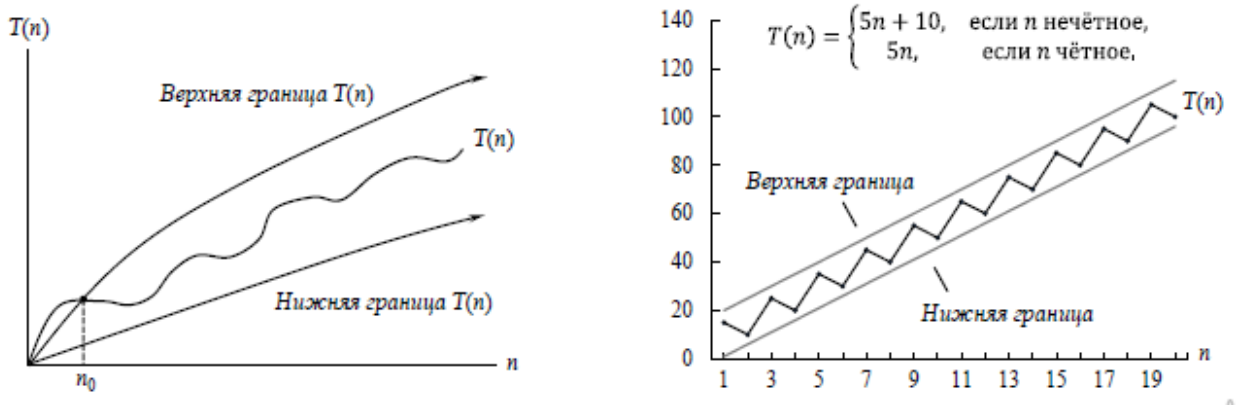

Рисунок 2.11 - Верхняя и нижняя оценка времени выполнения алгоритма

В теории вычислительной сложности алгоритмов (computational complexity theory) для указания границ функций T(n) используют

асимптотические обозначения: O (о большое), Ω (омега большое), Θ (тета большое), а также o (о малое), ω (омега малое)

Далее будем считать, что областью определения функции f(n) и g(n), которые выражают число операций алгоритма, является множество неотрицательных целых чисел: n ∈ {0, 1, 2, …}

Функции f(n) и g(n) являются асимптотически неотрицательными–при больших значениях n они принимают значения большие или равные нулю

O-обозначение (о большое)

Используется чтобы показать, что время работы алгоритма растёт не быстрее чем функция g(n). Асимптотическая верхняя граница (asymptotic upperbound) для функции f(n) показана на рис.2.12.

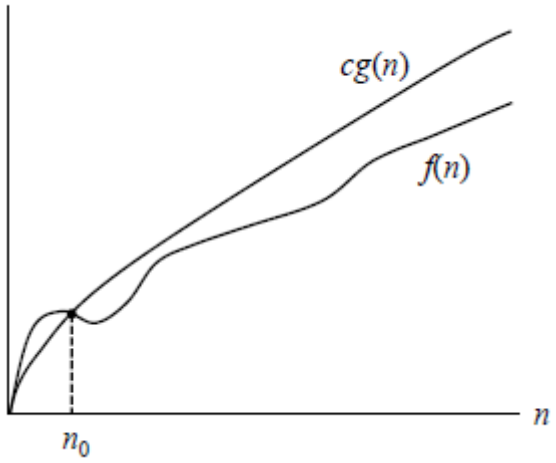

Рисунок 2.12 - Оценка O-обозначение

Ω-обозначение (омега большое)

Используется чтобы показать, что функция (время работы алгоритма) растет не медленнее чем функция g(n). Асимптотическая нижняя граница (asymptotic lower bound) для функции f(n) показана на рис.2.13.

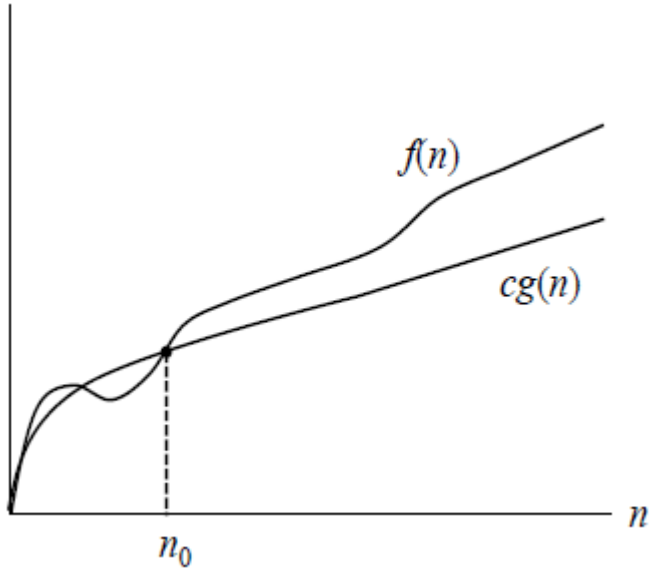

Рисунок 2.13 - Оценка Ω-обозначение

Θ-обозначения (тета большое)

Функция f(n) ограничена снизу и сверху функцией g(n) с точность до постоянного множителя (рис.2.14).

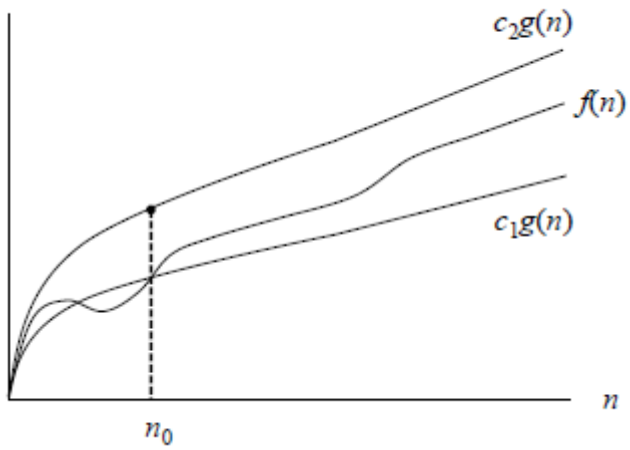

Рисунок 2.14 -Оценка Θ-обозначения

Классификация алгоритмов по трудоемкости в порядке возрастания трудоемкости представлена в таблице 2.

Таблица 2. Классификация алгоритмов по трудоемкости

Сложность	Нотация	Описание
Постоянная	O(1)	Постоянное количество операций, не зависящее от количества входящих данных
Логарифмическая	O(logn)	Количество операций пропорционально $\log_2 n$
Линейная	O(n)	Количество операций пропорционально количеству входящих данных
Квадратичная	$O(n^2)$	Количество операций пропорционально квадрату входящих данных
Кубическая	$O(n^3)$	Количество операций пропорционально кубу входящих данных
Экспоненциальная	$O(2^n)$, $O(k^n)$, O (n!)	Количество операций очень быстро возрастает

2.4. Алгоритмы поиска оптимального пути

Существует большое количество алгоритмов для построения оптимального маршрута:

- Алгоритм Дейкстры
- Поиск в ширину (BFS, Breadth-First Search)
- Поиск по первому наилучшему совпадению (Best-First Search)
- Алгоритм поиска A*
- Алгоритм поиска с итеративным углублением IDA* (Iterative-Deepening A*)
- Jump Point Search

2.4.1. Алгоритм Дейкстры р

Алгоритм Дейкстры представляет собой n итераций, на каждой из которых выбирается непомеченная вершина с наименьшей величиной d[v], эта вершина помечается, и затем просматриваются все рёбра, исходящие из данной вершины, и вдоль каждого ребра делается попытка улучшить значение d[v] на другом конце ребра.

Время работы алгоритма складывается из следующих действий:

- n раз поиск вершины с наименьшей величиной d[v] среди всех непомеченных вершин, т.е. среди O(n) вершин;
- m раз производится попытка релаксаций.

При простейшей реализации этих операций на поиск вершины будет затрачиваться O(n) операций, а на одну релаксацию — O(1) операций, и итоговая асимптотика алгоритма составляет: $O(n^2+m)$.

В ориентированном взвешенном графе $G = (V, E)$, вес рёбер которого неотрицателен и определяется весовой функцией $w : E \to R$, алгоритм Дейкстры находит длины кратчайших путей из заданной вершины s до всех остальных.

В алгоритме поддерживается множество вершин U, для которых уже вычислены длины кратчайших путей до них из s. На каждой итерации основного цикла выбирается вершина u $u \notin U$, которой на текущий момент соответствует минимальная оценка кратчайшего пути. Вершина u добавляется в множество U и производится релаксация всех исходящих из неё рёбер.

Псевдокод

```
func dijkstra(s):
   for v in V
      d[v] = ∞
      used[v] = false
   d[s] = 0
   for i in V
      v = null
      for j in V     // найдем вершину с минимальным расстоянием
         if !used[j] and (v == null or d[j] < d[v])
            v = j
      if d[v] == ∞
         break
      used[v] = true
      for e : исходящие из v рёбра     // произведём релаксацию по всем рёбрам, исходящим
из v
         if d[v] + e.len < d[e.to]
            d[e.to] = d[v] + e.len
```

Скорость работы алгоритма Дейкстры сильно зависит от скорости операций с приоритетной очередью.[47]

Существуют способы ускорения работы алгоритма, как правило, это делается с помощью использования разных структур данных: двоичных куч, 4 – куча, 8 – куча, MLB, MLB + калибровка.

Структура данных	Время работы (сек)
Двоичная куча	12,38
4-куча	11,53
8-куча	11,52
MLB	9,36
MLB + калибровка	8,04

Таблица 3. Сравнение различных структур данных для поиска кратчайшего пути на карте Европы (CPU 2,4GHz, 16MB RAM) [47]

Так же можно реализовать хранение данных в хеш-таблице, тогда нужно будет использовать метод цепочек (открытое хеширование) или метод открытой адресации (закрытое хеширование). [50] Данные, хранящиеся в виде такой структуры, удобны в обработке: хеш-таблица позволяет за минимальное время (O(1)) выполнять операции поиска, вставки и удаления элементов. [49]

2.4.2. Модифицированный метод Дейкстры (изменение целевой функции)

- n – число узлов распределительной сети;
- $D(k)$ – вес пути (сумма весов рёбер вдоль данного пути) от корневого узла 1 до узла k;
- $\mathrm{l}(\mathrm{m}, \mathrm{k})$ – вес ребра между m-м и k-м узлами;
- N – множество, элементами которого являются номера узлов, добавляемые на каждом шаге алгоритма оптимизации на основе вычисления путей с минимальным весом.

Модификация алгоритма Дейкстры состоит в изменении целевой функции и выражений для вычисления обновляемых весов путей.

Сравнительная характеристика модифицированного алгоритма и алгоритма Дейкстры представлена в табл. 4.

Отличие модифицированного алгоритма от алгоритма Дейкстры при вычислении обновляемых весов путей заключается в следующем. Выбор минимальных путей при их обновлении производится не только от корневого узла, но и от уже построенного к данному шагу, то есть от произвольного узла, уже включенного в распределительную сеть. В общем случае данные алгоритмы дают различные варианты сетей, хотя при некоторых наборах значений исходных данных (весов ребер) результирующие сети могут совпадать.

Таблица 4. Сравнительная характеристика модифицированного алгоритма и алгоритма Дейкстры. [33]

Модифицированный алгоритм	Алгоритм Дейкстры
1. Целевая функция	
Определяется суммарным весом ребер всех путей, составляющих распределительную сеть	Определяется весом каждого из путей, начинающихся в корневом узле
2. Критерий оптимальности	
Минимум суммарного веса ребер всех путей, составляющих распределительную сеть	Минимум веса пути из корневого узла в каждый узел распределительной сети
3. Обновление весов путей	
$D_i(k) = D_{i-1}(m_k)+l(m_k, k)$, $m \subset N$, $k \not\subset N$, где m_k=argmin[l(m, k)] –номер узла сети, при котором функция l(m, k) принимает минимальное значение	$D_i(k) = \min[D_{i-1}(k), D_{i-1}(m)+l(m, k)]$, где $D_i(k)$ – вес пути из корневого узла в k-й узел на i-ом шаге процедуры оптимизации

2.5. Алгоритм поиска A*

Алгоритм A* ("A star", "A звёздочка") — алгоритм поиска, который находит во взвешенном графе маршрут наименьшей стоимости от начальной вершины до выбранной конечной. Этот алгоритм был впервые описан в 1968 году Питером Хартом, Нильсом Нильсоном и Бертрамом Рафаэлем, и по сути является расширением алгоритма Дейкстры, созданного в 1959 году.

В процессе работы алгоритма для вершин рассчитывается функция $f(v) = g(v) + h(v)$, где

- $g(v)$ — наименьшая стоимость пути в v из стартовой вершины;
- $h(v)$ —приближение стоимости пути от v до конечной цели. $h(v)$должна быть допустимой, то есть не должна переоценивать рассояние до цели. Например, для задачи маршрутизации $h(v)$ может представлять собой расстояние до цели по прямой линии, так как это физически наименьшее возможное расстояние между двумя точками.

Чем меньше h, тем раньше вершина будет открыта и исследована алгоритмом. Таким образом, открытые алгоритмом вершины хранятся в очереди с приоритетом по значению $f(v)$. A* действует подобно алгоритму Дейкстры и просматривает среди всех маршрутов, ведущих к цели сначала те, которые благодаря имеющейся информации в данный момент являются наилучшими.

В приведённой реализации алгоритма в псевдокоде:

- Q — множество вершин, которые требуется рассмотреть,
- U — множество рассмотренных вершин,
- $f[x]$ — значение функции "расстояние + стоимость" для вершины x,
- $g[x]$ — стоимость пути от начальной вершины до x,
- $h(x)$ —оценка расстояния от вершины x до конечной вершины.

На каждом этапе работы алгоритма из множества Q выбирается вершина с наименьшим значением функции и просматриваются её соседи. Для каждого из соседей обновляется расстояние, значение функции и он добавляется в множество. Псевдокод:

```
bool A*(start, goal):
   U = ∅
   Q = ∅
   Q.push(start)
   g[start] = 0
```

```
f[start] = g[start] + h(start)
while Q.size() != 0
    current = вершина из Q с минимальным значением f
    if current == goal
        return true // нашли путь до нужной вершины
    Q.remove(current)
    U.push(current)
    for v : смежные с current вершины
        tentativeScore = g[current] + d(current, v) // d(current, v) — стоимость пути между current и v
        if v ∈ U and tentativeScore >= g[v]
            continue
        if v ∉ U or tentativeScore < g[v]
            parent[v] = current
            g[v] = tentativeScore
            f[v] = g[v] + h(v)
            if v ∉ Q
                Q.push(v)
return false
```

2.5.1. Алгоритм поиска с итеративным углублением (Iterative Deeping A*, IDA*, A* с итеративным углублением)

Алгоритм поиска с итеративным углублением это – измененная версия A*, использующая меньше памяти за счет меньшего количества развертываемых узлов. Работает быстрее A* в случае удачного выбора метрики. Результат работы – кратчайший путь.

Основное различие между алгоритмом IDA* и стандартным алгоритмом итеративного углубления состоит в том, что применяемым условием останова развертывания служит f-стоимость (g+h), а не глубина; на каждой итерации этим остановочным значением является минимальная f-стоимость любого узла, превышающая остановочное значение, достигнутое в предыдущей итерации. Алгоритм IDA* является практически применимым для решения многих задач с единичными стоимостями этапов и позволяет избежать существенных издержек, связанных с поддержкой отсортированной очереди узлов.

node – текущий узел
g – стоимость начала решения root..node
f – оценка стоимости минимального пути через node
h(node) –оценка стоимости остатка пути node..goal
cost(node, succ) – функция стоимости пути
is_goal(node) – функция проверки цели
successors(node) – функция развёртывания узла node

```
procedure ida_star(root, cost(), is_goal(), h())
  bound := h(root)
  loop
    t := search(root, 0, bound)
    if t = FOUND then return FOUND
    if t = ∞ then return NOT_FOUND
    bound := t
  end loop
end procedure

function search(node, g, bound)
  f := g + h(node)
  if f > bound then return f
  if is_goal(node) then return FOUND
  min := ∞
  for succ in successors(node) do
    t := search(succ, g + cost(node, succ), bound)
    if t = FOUND then return FOUND
    if t < min then min := t
  end for
  return min
end function
```

2.5.2. Поиск в ширину (BFS, Breadth-First Search)

Поиск в ширину (англ. breadth-first search, BFS) — метод обхода графа и поиска пути в графе. Поиск в ширину является одним из неинформированных алгоритмов поиска.

Алгоритм был разработан независимо Муром и Ли для разных приложений (поиск пути в лабиринте и разводка проводников соответственно) в 1959 и 1961 годах. Этот алгоритм можно сравнить с поджиганием соседних вершин графа: сначала мы зажигаем одну вершину (ту, из которой начинаем путь), а затем огонь за один элементарный промежуток времени

перекидывается на все соседние с ней не горящие вершины. В последствие то же происходит со всеми подожженными вершинами. Таким образом, огонь распространяется "в ширину". В результате его работы будет найден кратчайший путь до нужной клетки.

Поиск в ширину работает путём последовательного просмотра отдельных уровней графа, начиная с узла-источника u.

Рассмотрим все рёбра (u, v), выходящие из узла u. Если очередной узел v является целевым узлом, то поиск завершается; в противном случае узел v добавляется в очередь. После того, как будут проверены все рёбра, выходящие из узла u, из очереди извлекается следующий узел u, и процесс повторяется.

Ниже приведён псевдокод алгоритма для случая, когда необходимо лишь найти целевой узел. В зависимости от конкретного применения алгоритма, может потребоваться дополнительный код, обеспечивающий сохранение нужной информации (расстояние от начального узла, узел-родитель и т. п.)

Рекурсивная формулировка:

```
BFS(start_node, goal_node) {
  return BFS'({start_node}, ∅, goal_node);
}
BFS'(fringe, visited, goal_node) {
  if(fringe == ∅) {     // Целевой узел не найден
    return false;
  }
  if (goal_node ∈ fringe) {
    return true;
  }
  return BFS'({child | x ∈ fringe, child ∈ expand(x)} \ visited, visited ∪ fringe,
goal_node);
}
```

Итеративная формулировка:

```
BFS(start_node, goal_node) {
```

```
  for(all nodes i) visited[i] = false; // изначально список посещённых узлов пуст
  queue.push(start_node);              // начиная с узла-источника
  visited[start_node] = true;
  while(! queue.empty() ) {            // пока очередь не пуста
    node = queue.pop();                // извлечь первый элемент в очереди
    if(node == goal_node) {
      return true;                     // проверить, не является ли текущий узел целевым
    }
    foreach(child in expand(node)) {   // все преемники текущего узла, ...
      if(visited[child] == false) {    // ... которые ещё не были посещены ...
        queue.push(child);             // ... добавить в конец очереди...
        visited[child] = true;         // ... и пометить как посещённые
      }
    }
  }
  return false;                        // Целевой узел недостижим
}
```

2.5.3. Алгоритм поиска пути Jump Point Search

Jump Point Search это алгоритм поиска пути, представленный в 2011 году Дэниэлом Харбором в серии статей «Fast Pathfinding via Symmetry Breaking», «Rectangular Symmetry Reduction», «Jump Point Search». Алгоритм является улучшением алгоритма A*. JPS ускоряет поиск пути, “перепрыгивая” многие места, которые должны быть просмотрены. В отличие от подобных алгоритмов JPS не требует предварительной обработки и дополнительных затрат памяти.

Алгоритм работает на гридах, у которых стоимость прохода везде одинакова, но существуют «препятствия» - ячейки с бесконечно большой стоимостью. Каждое поле карты имеет <= 8 соседей, которые могут быть проходимы или же нет. Каждый шаг по направлению (по вертикали или по горизонтали) имеет стоимость 1; шаг по диагонали имеет стоимость √2. Движения через препятствия запрещены. “Прыжковые точки” позволяют ускорить алгоритм поиска пути, рассматривая только “необходимые” точки. Такие точки могут быть описаны двумя простыми правилами выбора соседей при рекурсивном поиске: одно правило для прямолинейного движения и другое – для диагонального. В обоих случаях необходимо доказать, что, после

исключения полей из набора ближайших соседей точки, всегда найдётся оптимальный путь из предка текущей точки до каждого из соседей, и этот путь не будет содержать в себе посещенную точку.

2.5.4. Что используется в ArcGis

Механизмы расчета задач выбора маршрута в дополнительном модуле Дополнительный модуль ArcGIS Network Analyst – Маршрут (Route), Ближайший пункт обслуживания (Closest Facility) и Матрица Источник-Назначение (OD Cost Matrix) – основаны на известном алгоритме поиска кратчайших путей Дейкстры. Каждый из этих трех механизмов расчета реализует два типа алгоритма поиска пути. Первый тип – поиск точного кратчайшего пути, а второй – иерархический поиск пути для повышения быстродействия. Классический алгоритм Дейкстры решает задачу поиска кратчайшего пути при помощи неориентированного, неотрицательного, взвешенного графа. Для использования в контексте реальных данных о транспортных перевозках этот алгоритм модифицируется с учетом пользовательских параметров, таких как ограничения, связанные с односторонним движением и поворотами, импеданс узлов, барьеры и ограничения для стороны улицы, при этом минимизируется заданный пользователем стоимостный атрибут. Быстродействие алгоритма Дейкстры улучшается при использовании более эффективных структур данных, таких как d-кучи. Кроме того, алгоритм должен иметь возможность моделировать местоположения в любой точке ребра, а не только на узлах пересечения.

Классический алгоритм Дейкстры решает задачу поиска кратчайшего пути с одной исходной точкой на взвешенном графе. Для поиска кратчайшего пути из начального местоположения s до конечного местоположения d алгоритм Дейкстры использует набор узлов S, окончательный кратчайший путь от которых до s уже вычислен. Алгоритм многократно выполняет в наборе поиск узла с минимальной оценкой кратчайшего пути, добавляет его в набор узлов S и обновляет оценки кратчайшего пути всех соседей этого узла,

которые не входят в S. Алгоритм повторяется, пока узел назначения не будет добавлен в набор S.

В ArcGis используется модифицированный алгоритм Дейкстры[63] для двух направлений с целью вычисления маршрута между исходной и конечной точками, а также применяются иерархии по предпочтительности путей. Поскольку поиск ограничен иерархией верхнего уровня, он затрагивает меньшее количество ребер, что повышает быстродействие. Следует отметить, что это эвристический алгоритм; его цель – высокое быстродействие и хорошие решения, но он не гарантирует, что будет найден кратчайший путь. Для успешной работы эвристического алгоритма необходимо подключить иерархию высшего уровня, поскольку он не спускается на уровень ниже при достижении тупикового конца.

2.6. Выбор алгоритма для реализации

Выбор алгоритма для реализации можно осуществить, проанализировав таблицу, в которой рассматриваются ключевые факторы.

Стоимость прохода по ячейкам определяет подходит данный алгоритм для задачи поиска оптимального маршрута по карте или нет. Некоторые алгоритмы предназначены для поиска пути в лабиринте и стоимость прохода в них может быть равна 1 или ∞, например, в Breadth-First Search, и такой алгоритм не подходит для данной задачи, в силу своей узкой направленности.

Таблица 5. Таблица сравнения алгоритмов

Алгоритм	Стоимость прохода по ячейкам	Worst time (худшее время)	Worst space (наибольший объем памяти)
Алгоритм Дейкстры (Dijkstra)	любая	$O(n^2)$	-
Поиск в ширину (BFS, Breadth-First Search)	1 или ∞	$O(b^{d+1})$	$O(b^d)$
A*	любая	$O(b^{d+1})$, зависит от метрики	$O(b^d)$
IDA* (Iterative Deeping A*, A* с	1 или ∞	$O(b^d)$	O(bd)

итеративным углублением)			
Jump Point Search	1 или ∞	-	-

Анализ показал, что только два алгоритма из рассмотренных можно использовать: алгоритм Дейкстры и алгоритм A*. Из этих двух алгоритмов самым оптимальным по времени является A*, поэтому дальнейшие исследования будут посвящены его оптимизации.

2.7. Выбор структуры данных

Существует большое количество различных структур данных, самые распространенные из них представлены на рис. 2.15.

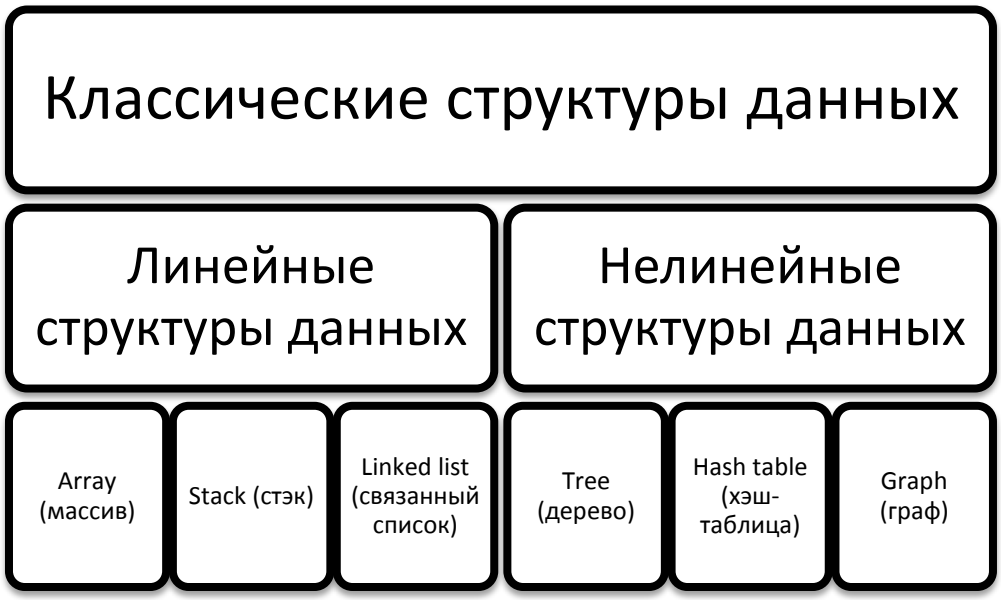

Рисунок 2.15 - Классические структуры данных

Каждая из представленных структур служит для своих целей, в табл. 6. показаны результаты time complexity (временной сложности) выполнения операций: доступа, поиска, вставки и удаления в среднем и в худшем случае.

Таблица 6. Time complexity выполнения стандартных операций

Data Structure	Time complexity (время выполнения)							
	Average (в среднем)				Worst (в худшем случае)			
	Access (доступ)	Search (поиск)	Insertion (вставка)	Deletion (удаление)	Access (доступ)	Search (поиск)	Insertion (вставка)	Deletion (удаление)
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)

Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked list	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)
Binary tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)
Hash table	-	O(1)	O(1)	O(1)	-	O(n)	O(n)	O(n)

В табл. 7 показаны результаты анализа структур данных по space complexity (объемной сложности).

Таблица 7. Space complexity

Data Structure	Space complexity
Array	O(n)
Stack	O(n)
Linked list	O(n)
Binary tree	O(n)
Hash table	O(n)

По результатам анализа структур данных, приведенных выше, было принято решение использовать Hash table, как структуру данных, требующую наименьшего среднего времени для выполнения стандартных операций.

2.7.1. Hash table (хэш-таблица)

Хеш-таблица представляет собой обобщение обычного массива. Однако в то время как ключом массива может быть только число, для хеш-таблицы им может быть любой объект, для которого можно вычислить хеш-код. Интерфейс хеш-таблиц предоставляет следующие операции:

- Добавление новой пары ключ-значение
- Поиск значения по ключу
- Удаление пары ключ-значение по ключу

Среднее время поиска значения по ключу в хеш-таблице равно O(1). Это значит, что в среднем, поиск не будет зависеть от количества элементов в хеш-таблице, и будет равен некоторому константному значению. В зависимости от самой внутренней реализации хеш-таблицы, время поиска для наихудшего

случая может быть O(n), то есть линейно зависеть от количества элементов в таблице, либо же оставаться O(1).

Идея хеширования основана на распределении ключей в обычном массиве H[0..m-1]. Распределение осуществляется вычислением для каждого ключа элемента некоторой хеш-функции h. Эта функция на основе ключа вычисляет целое число n, которое служит индексом для массива H. Сложность заключается в том, что необходимо придумать такую хеш-функцию, которая бы давала различный хеш-код для различных объектов. На рис. 2.16 изображена хеш-таблица, индексами ключей в ней является результат хеш-функции h, применённой к ключу.

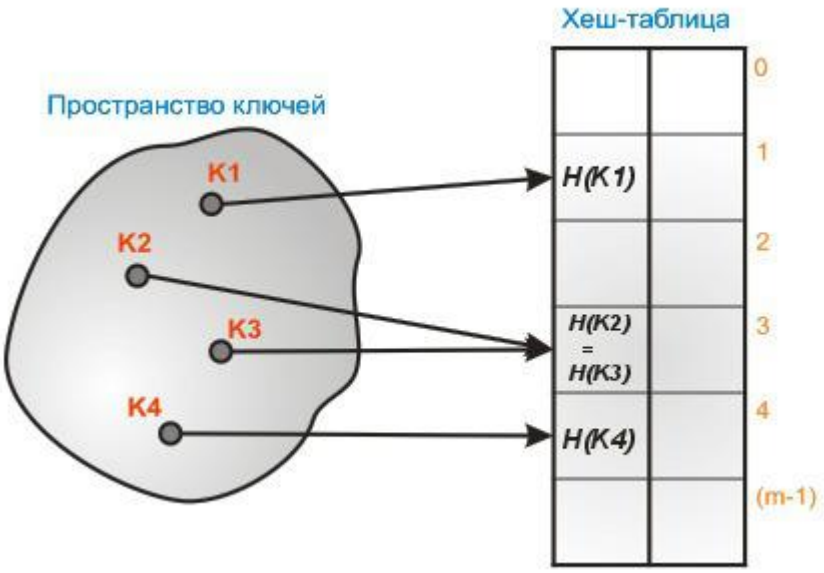

Рисунок 2.16 - Хеш-таблица

Так же этот рисунок иллюстрирует одну из основных проблем. При достаточно маленьком значении m (размера хеш-таблицы) по отношению к n (количеству ключей) или при плохой хеш-функции, может случиться так, что два ключа будут помещены в одну и ту же ячейку массива H. Такая ситуация называется коллизией.

При выборе хеш-функции необходимо минимизировать вероятность коллизий, однако, учитывая то, что пространство всех возможных ключей

может быть больше размера нашей хеш-таблицы H, всё же избежать их вряд ли удастся. На этот случай имеются несколько технологий для разрешения коллизий. Основные из них представлены далее.

2.7.2. Хеширование с цепочками

В случае открытого хеширования (другое название хеширования цепочками), происходит объединение элементов, хешированных в одну и ту же ячейку, в связный список (рис. 2.17).

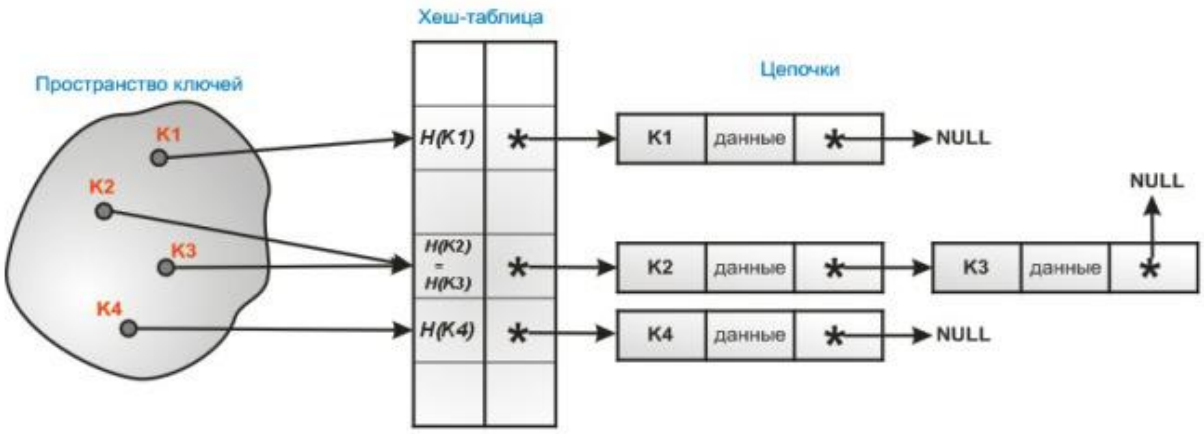

Рисунок 2.17 - Пример открытого хеширования

Так, идея достаточно проста. Если при добавлении в хеш-таблицу в заданную ячейку встречается ссылка на элемент связного списка, то случается коллизия. Поэтому можно просто вставить новый элемент как узел в список. При поиске программа прохолит по цепочкам, сравнивая ключи между собой на эквивалентность, пока не доберётся до нужного. При удалении ситуация такая же.

Процедура вставки выполняется даже в наихудшем случае за $O(1)$, при отсутствие вставляемого элемента в таблице. Время поиска зависит от длины списка, и в худшем случае равно $O(n)$. Эта ситуация, когда все элементы хешируются в единственную ячейку. Если функция распределяем n ключей по m ячейкам таблицы равномерно, то в каждом списке будет содержаться порядка n/m ключей. Это число называется коэффициентом заполнения хеш-таблицы. Математический анализ хеширования с цепочками показывает, что

в среднем случае все операции в такой хеш-таблице в среднем выполняются за время O(1).

2.7.3. Хеширование с открытой адресацией

В случае метода открытой адресации (или по-другому: закрытого хеширования) все элементы хранятся непосредственно в хеш-таблице, без использования связанных списков. В отличии от хеширования с цепочками, при использовании метода открытой адресации может возникнуть ситуация, когда хеш-таблица окажется полностью заполненной, так что будет невозможно добавлять в неё новые элементы. Так что при возникновении такой ситуации решением может быть динамическое увеличение размера хеш-таблицы, с одновременной её перестройкой.

Для разрешения же коллизий применяются несколько подходов. Самый простой из них – это метод линейного исследования. В этом случае при возникновении коллизии следующие за текущей ячейки проверяются одна за другой, пока не найдётся пустая ячейка, куда и помещается элемент. Так, при достижении последнего индекса таблицы, программа возвращается в ее начало, рассматривая таблицу как «цикличный» массив. Иллюстрация этого способа представлена на рисунке 2.18:

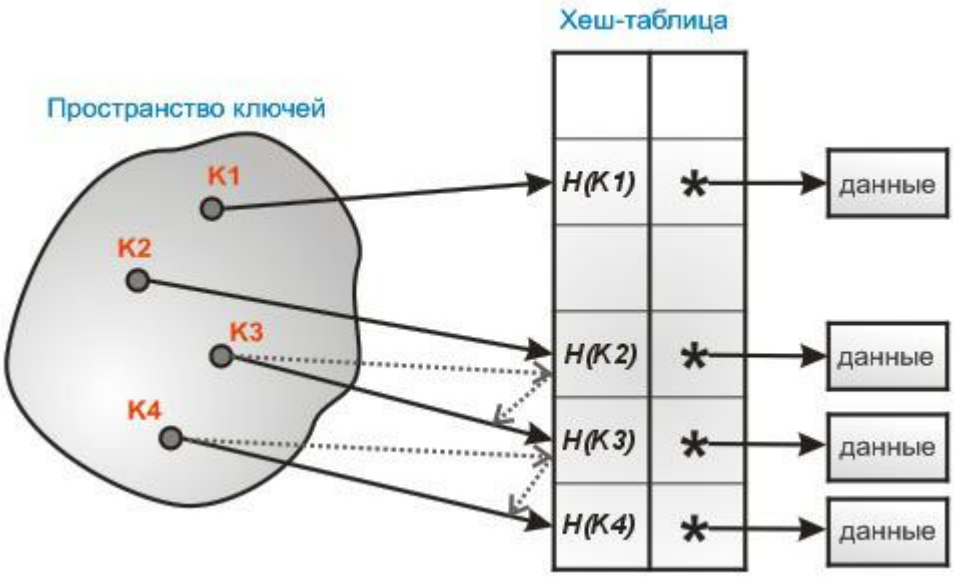

Рисунок 2.18 - Пример разрешения коллизий

Линейное хеширование достаточно просто реализуется, однако с ним связана существенная проблема – кластеризация. Это явление создания длинных последовательностей занятых ячеек, которое увеличивает среднее время поиска в таблице. Для снижения эффекта кластеризации используется другая стратегия разрешения коллизий – двойное хеширование. Основная идея заключается в том, что для определения шага смещения исследований при коллизии в ячейке используется другая хеш-функция, вместо линейного смещения на одну позицию.

Одной из сложных вопросов реализации хеширования с открытой адресацией – это операция удаления элемента. При удалении некого элемента их хеш-таблицы, становится невозможным поиск ключа, в процессе вставки которого текущая ячейка оказалась заполненной. Этого можно избежать, если помечать очищенные ячейки, чтобы впоследствии это учитывать.

3. Анализ входных данных и вариантов оптимизации алгоритма A*

3.1. Входные данные

Пространственные данные бывают 2 основных типов: векторные и растровые.

3.1.1. Векторные данные

К векторным относятся: точки, полинии, многоугольники.

1. Точки

Используются для обозначения географических объектов, для которых важно местоположение, а не их форма или размеры. Возможность обозначения объекта точкой зависит от масштаба карты. В то время как на карте мира города целесообразно обозначать точечными объектами, то на карте города сам город представляется в виде множества объектов.

2. Полилинии

Служат для изображения линейных объектов. Полилиния — ломаная линия, составленная из отрезков прямых. Полилиниями изображаются дороги, железнодорожные пути, реки, улицы. Допустимость изображения объектов полилиниями также зависит от масштаба карты. Например, крупная река в масштабах континента вполне может изображаться линейным объектом, тогда как уже в масштабах города требуется её изображение площадным объектом. Характеристикой линейного объекта является длина.

3. Многоугольники (полигоны)

Служат для обозначения площадных объектов с четкими границами. Примерами могут служить озера, парки, здания, страны, континенты. Характеризуются площадью и длиной периметра.

3.1.2. Растровые данные

К растровым данным относятся называемые гриды - изображения поверхностей или данные по пространственным свойствам. В то время как

векторные объекты используют геометрию (точки, полилинии и полигоны) для представления объектов реального мира, растровые данные основаны на другом подходе. Растры являются матрицами пикселей (также называемых ячейками), каждый из которых хранит определенное значение для области, соответствующую этому пикселу (рис.3.1).

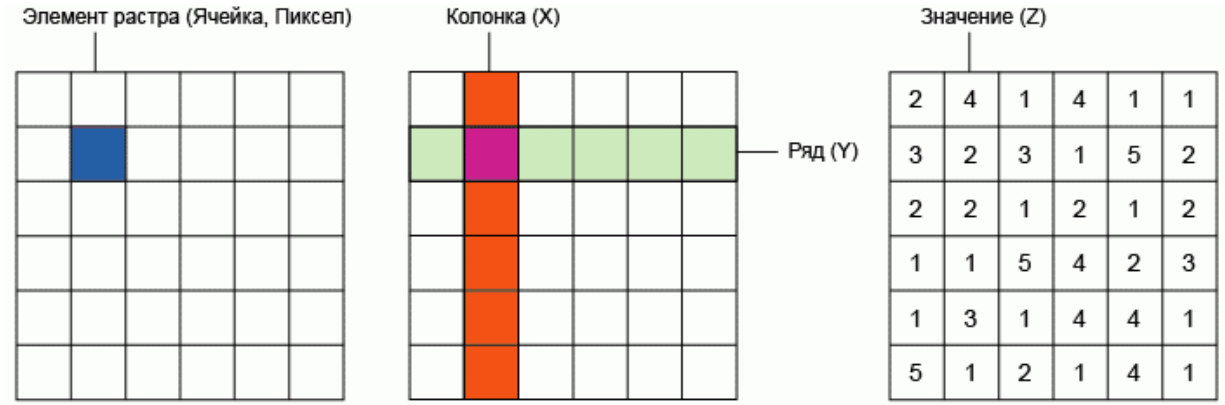

Рисунок 3.1 - Растровые данные

Иногда растровые данные создаются на основе векторных данных, т.к. обладатели этих данных хотят опубликовать их в общедоступном формате. Если пользователю нужно иметь возможность просматривать или анализировать атрибутивную таблицу, растровые форматы не подойдут для такой задачи, т.к. большинство из них не имеет связанной атрибутивной таблицы.

3.1.3. Анализ векторных и растровых типов данных

Растровая и линейная модель данных по-разному ведет себя при масштабировании (рис. 3.2), поэтому для корректной работы используется масштаб 1:50000, утвержденный

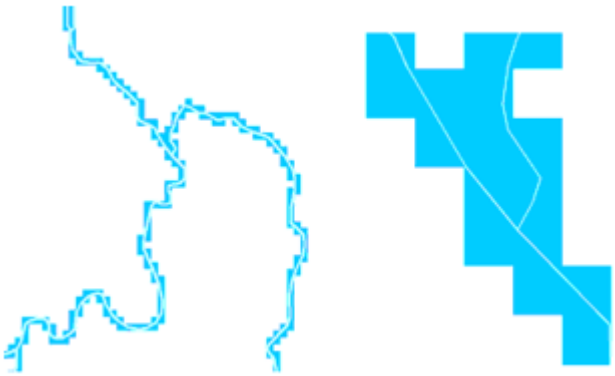

Рисунок 3.2 - Масштабирование

Сравнение растровой и векторной модели данных представлено в таблице 8.

Таблица 8. Сравнение растровой и векторной модели данных, плюсы и минусы

Свойство / Модель данных	Растровая	Векторная
Масштабируемость	-	+
Избыточность (объем данных)	-	+
Передача непрерывных свойств	+	-
Передача дискретных объектов	-	+
Легкость создания	+	-

По таблице видно, что разные модели данных подходят для разных целей.

3.1.4. Анализ геоданных

Проекты по обустройству месторождений углеводородов, как правило, имеют недостаточную обеспеченность актуальной пространственной информацией, но для этапа ТЭО этого достаточно. В качестве картографической основы используются топографические карты масштаба от 1:25 000 и мельче.

Данные, которые должны учитываться при проектировании маршрута:

- населенные пункты;

- промышленные предприятия;
- сельскохозяйственные предприятия;
- особо охраняемые зоны;
- военные части;
- заповедники;
- болота и обводненные участки;
- территории с просадочными грунтами;
- вечномерзлые грунты;
- дороги;
- реки;
- озера;
- уклон местности;
- растительность.

Карты местности всегда используются в проектных работах, при этом, отображаемые на них пространственные данные должны учитываться совместно. Так, при прокладке внутрипромысловых трубопроводов учитываются такие параметры, как уклоны территории, наличие на пути трасы водоёмов, болот, древесной растительности и т.д. Суммирование тематических растровых карт позволяет получить интегральную карту, на которой качественным фоном отображаются благоприятные и неблагоприятные условия для строительства.

В результате суммирования карт получена общая картографическая основа для проектирования. На рис.3.3 оттенками зелёного показаны участки, наиболее благоприятные для прокладки дороги, оттенками красного – менее благоприятные. В данном примере использовались три параметра оценки. Но в каждой конкретной ситуации могут вводиться дополнительные параметры и поправочные коэффициенты, в зависимости от имеющихся данных.

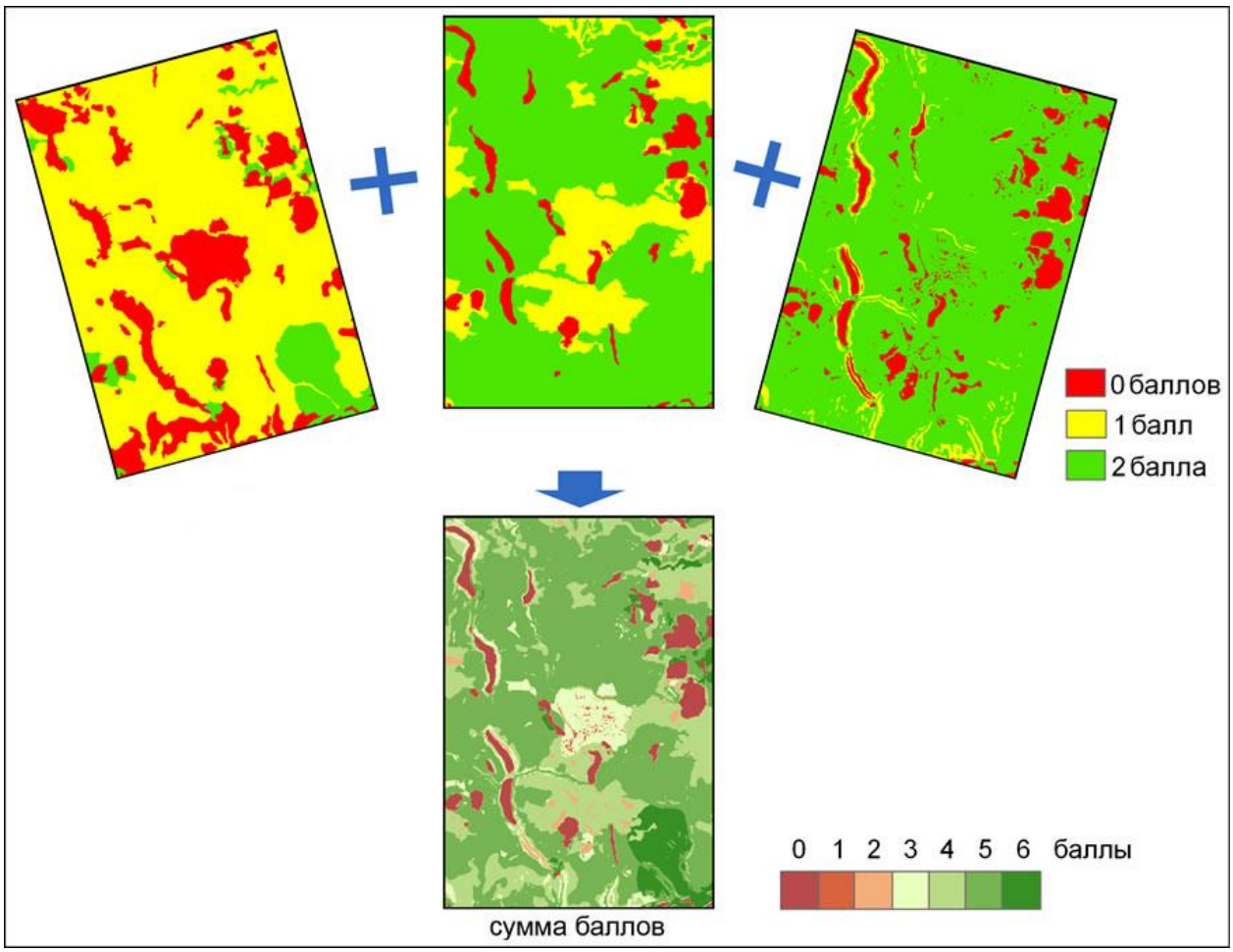

Рисунок 3.3 - Суммирование растровых тематических карт

3.2. Анализ алгоритма A*

По результатам исследования, представленным выше, для поиска оптимального маршрута был выбран алгоритм A*. Его подробный анализ представлен далее.

3.2.1. Основная терминология

Узел — позиция на карте.

Открытый список — список узлов в которые можно переместиться и которые являются смежными с закрытыми узлами.

Закрытый список — список узлов, в которые можно переместиться, но которые уже были пройдены.

На рис 3.4 изображены узлы, составляющие карту. Фактически, узлом является каждый квадрат карты (ячейка). На карте отмечены начальная и конечная позиции, A и B соответственно. Поскольку начальная позиция всегда будет являться частью искомого пути, она автоматически включается в закрытый список пройденных узлов.

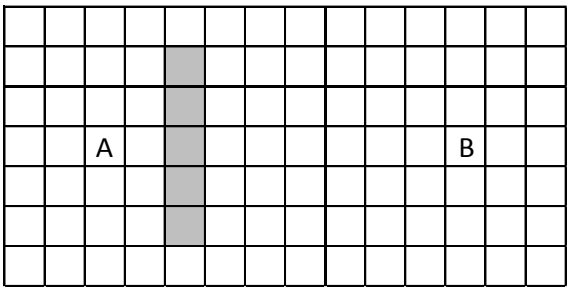

Рисунок 3.4 -Терминология в алгоритме A*

Узлы, соседствующие с единственным узлом из закрытого списка, будут помещены в открытый список. Это показано на рис. 3.5.

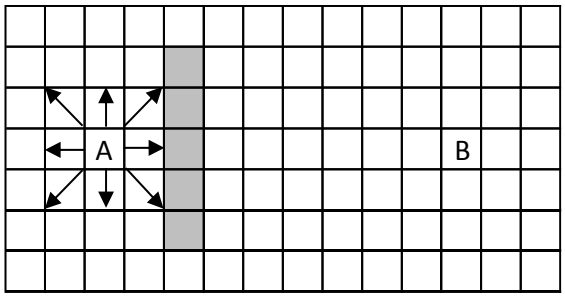

Рисунок 3.5 - Добавление узлов в открытый список

На рис. 3.5. изображены восемь узлов, входящих в открытый список и один узел, входящий в закрытый список. Стрелки показывают направление перемещения из того закрытого узла, к которому относятся данные открытые узлы. Закрытый узел в этом случае называется *родительским узлом* каждого из открытых узлов.

3.2.2. Начало поиска

Первое, что делает алгоритм A* — это добавление начального узла в закрытый список. Это делается потому, что начальный узел всегда будет первым узлом полученного пути. Сделав это необходимо найти все узлы, которые являются смежными с начальным и в которые можно перейти. Если

смежный узел доступен, он добавляется в открытый список. Так как в самом начале нет никаких открытых узлов, перед началом работы алгоритма открытый список пуст.

Этапы поиска:

1) Поместить начальный узел в закрытый список.
2) Поместить доступные смежные узлы в открытый список.

Следующим этапом является вычисление стоимости узлов.

3.2.3. Вычисление стоимости узлов

У каждого узла в закрытом и открытом списках есть его вес, поэтому некоторые из них лучше подходят для создания пути, чем другие. Чтобы выяснить, какой узел является самым лучшим, необходимо каждому узлу в закрытом и открытом списках назначить его итоговую «стоимость». Как только всем узлам назначена стоимость, достаточно простой сортировки, чтобы выяснить какой узел является самым дешевым. Для вычисления стоимости узлов в алгоритме A* необходимы следующие значения:

- Базовая стоимость узла.
- Стоимость возврата к начальному узлу.
- Стоимость достижения цели.

Базовая стоимость узла

Базовой стоимостью узла называется стоимость передвижения через данный узел. В простейшем случае базовая стоимость всех доступных узлов одна и та же. Однако, в случае с поставленной задачей у различных узлов различная стоимость, в зависимости от типа ландшафта и других факторов.

Стоимость относительно цели

Последний компонент стоимости — это стоимость достижения цели из данного узла. Она вычисляется путем сложения количества строк и столбцов на которые текущий узел отстоит от цели.

Общая стоимость

Как только все три упомянутых выше стоимости вычислены, нужно их просуммировать и получить общую стоимость узла.

3.2.4. Поиск наилучшего узла

Зная общую стоимость каждого узла, очень просто найти наилучший узел, для добавления его в закрытый список. Необходимо отсортировать узлы по значению общей стоимости и выбрать тот из них у которого она меньше всего.

После того, как узел с наименьшей общей стоимостью найден, нужно добавить его в закрытый список в качестве кандидата на участие в итоговом пути. Так же необходимо удалить этот узел из открытого списка, чтобы он не был обработан снова.

Продолжение поиска

Если в открытом списке отсутствует конечный пункт маршрута, следует продолжать поиск пути добавив в открытый список те узлы, которые находятся вокруг узла только что добавленного в закрытый список. После этого нужно найти открытый узел с наименьшей стоимостью и добавить его в закрытый список. Эти действия будут повторяться до тех пор, пока конечный пункт маршрута не окажется в открытом списке.

3.2.5. Обратная трассировка для нахождения пути

Как только конечный пункт маршрута окажется в открытом списке, необходимо будет составить путь обратно к исходной точке. Для этого берется родитель открытого узла, в котором расположен конечный пункт. Затем родитель родителя и так далее до тех пор, пока не произойдет возвращение к исходной позиции. В результате будет найден путь от конечного пункта до начального. Теперь необходимо инвертировать полученный путь, чтобы получить итоговый маршрут от исходной точки до цели.

3.3. Оптимизация A*

Существует несколько способов модифицировать алгоритм поиска, чтобы получить хорошие результаты при использовании ограниченных ресурсов:

3.3.1. Лучевой поиск

Одним из способов борьбы с ограничением по памяти является наложение ограничений на количество узлов в списке открытых; когда список полон и необходимо добавить новый узел, просто выбрасывается узел с наихудшим значением. Список закрытых узлов также может быть уничтожен, если каждая ячейка хранит в себе длину наилучшего пути и обратный указатель. Этот алгоритм не гарантирует оптимальности пути, так как узел, ведущий к нему, может быть выброшен, но все равно может позволить найти разумный путь.

3.3.2. Алгоритм последовательных приближений

Для A* может быть использован алгоритм последовательных приближений, использование его полностью избавляет от необходимости хранить списки Open и Closed. Делается простой рекурсивный поиск, собирается наколенная стоимость пути g(x,y), и поиск прекращается при достижении значением f(x,y) = g(x,y) + h(x,y) заданных пределов. Начинать нужно с пределом остановки равным $h(x_1, y_1)$, и в каждой последовательной итерации, устанавливать новый предел остановки равным минимальному значению f(x,y), которое превысило прежнюю границу.

3.3.3. Метрика h(n)

Как обсуждалось выше, если приближение оставшейся стоимости пути слишком мало, то A* может быть очень неэффективным. Но если приближение слишком высоко, то для найденного пути не гарантируется оптимальность. На картах диапазон изменения стоимостей ландшафта широк - от болот до населенных пунктов - можно поэкспериментировать с

различными промежуточными значениями, чтобы найти точный баланс между эффективностью поиска и качеством найденного пути.

Поведение алгоритма напрямую зависит от того, какая метрика используется, выбор метрики зависит от постановки задачи.

- Если мы можем перемещаться в четырех направлениях, в качестве метрики стоит выбрать манхэттенское расстояние. [75]

$$h(x, y) = |x_1 - x_2| + |y_1 - y_2|$$

- Расстояние Чебышева [45] применяется, когда к четырем направлениям добавляются диагонали:

$$h(vx, y) = \max(|x_1 - x_2|, |y_1 - y_2|)$$

- Если передвижение не ограниченно сеткой, то можно использовать евклидово расстояние:

$$h(x, y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

A* работает на любом графе, где можно выделить функцию расстояния (если ее нет, он становится неэффективен), поэтому можно использовать разные варианты сетки, например, не 4-угольной сетки, а 6-угольной.

3.3.4. Использование «запрещенных» областей

В задаче выбора оптимального пути для прокладки трубопровода на карте есть ряд ограничений по выбору маршрута. Например, трубопровод не может быть проложен на территориях заповедников, военных частей и вблизи населенных пунктов. Эти ограничения необходимы с точки зрения нормативных актов и действующего законодательства, но в то же время полезны, так как позволяют заранее отсечь области, в которых строительство не может быть произведено.

Для того, чтобы сократить время выполнения алгоритма, можно уменьшить количество входных данных, предварительно обработав данные и присвоив выбранным «запрещенным областям» значение ∞.

В процессе работы алгоритма A* для вершин рассчитывается функция $f(x, y) = g(x, y) + h(x, y)$, и если есть область, значения которой «бесконечно большие», то алгоритм не будет их рассматривать.

3.3.5. Параллельное программирование

Параллельный алгоритм, противопоставляемый традиционным последовательным алгоритмам, — алгоритм, который может быть реализован по частям на множестве различных вычислительных устройств с последующим объединением полученных результатов и получением корректного результата.

К преимуществам параллельного программирования можно отнести:

- возможность выполнения нескольких задач за меньшее время, т. е. повышенная производительность системы.
- выполнение отдельных задач за меньшее время.

Параллельное программирование включает в себя все черты более традиционного, последовательного программирования, но в параллельном программировании имеются три дополнительных, четко определенных этапа:

1) Определение параллелизма: анализ задачи с целью выделить подзадачи, которые могут выполняться одновременно.
2) Выявление параллелизма: изменение структуры задачи таким образом, чтобы можно было эффективно выполнять подзадачи. Для этого часто требуется найти зависимости между подзадачами и организовать исходный код так, чтобы ими можно было эффективно управлять.
3) Выражение параллелизма: реализация параллельного алгоритма в исходном коде с помощью системы обозначений параллельного программирования.

Распараллелить решение задачи можно на нескольких уровнях. Между этими уровнями нет четкой границы и конкретную технологию распараллеливания бывает сложно отнести к одному из них. Приведенное ниже деление условно и служит для того, чтобы продемонстрировать разнообразие подходов к задаче распараллеливания.

Уровень параллелизма задач.

Часто распараллеливание на этом уровне является самым простым и при этом самым эффективным. Такое распараллеливание возможно в тех случаях, когда решаемая задача естественным образом состоит из независимых подзадач, каждую из которых можно решить отдельно. Распараллеливание на уровне задач нам демонстрирует операционная система, запуская на многоядерной машине программы на разных ядрах.

Уровень параллелизма данных

Название модели «параллелизм данных» происходит оттого, что параллелизм заключается в применении одной и той же операции к множеству элементов данных. Данный вид параллелизма широко используется при решении задач численного моделирования. Счетная область разбивается на геометрические объекты, например, параллелепипеды, и ячейки, вошедшие в эту область, отдаются на обработку определенному ядру. В математической физике такой тип параллелизма называют геометрическим параллелизмом.

Уровень параллелизма алгоритмов

Следующий уровень – это распараллеливание отдельных процедур и алгоритмов. Сюда можно отнести алгоритмы параллельной сортировки, умножения матриц, решения системы линейных уравнений. Задача реализации параллельных алгоритмов достаточно сложна и поэтому существует большое количество библиотек и технологий параллельных алгоритмов (OpenMP, MPI и др.), позволяющих строить программы как из

кубиков, не вдаваясь в устройство реализаций параллельной обработки данных.

Уровень параллелизма инструкций

Наиболее низкий уровень параллелизма, осуществляемый на уровне параллельной обработки процессором нескольких инструкций. На этом же уровне находится пакетная обработка нескольких элементов данных одной командой процессора. Интерес этот уровень распараллеливания может представлять только для узкой группы специалистов или разработчиков компиляторов.

4. Разработка приложения

Приложение для построения оптимального маршрута трубопровода было реализовано на языках java Script и python 2.7, библиотеке ArcGis для java Script (ArcGIS Web API for JavaScript API 4.0), языка гипертекстовой разметки html5 и каскадных таблицах стилей css 3. В приложении к диплому 1 описывается видение продукта.

4.1. Проектирование приложения

1. Проектирование
   a) Функциональные требования: определение цели разработки и исходных требований.

Цель создания ПО – позволить пользователям прокладывать оптимальный маршрут линейных объектов, в частности трубопроводов, используя Web-интерфейс (существующий браузер).

Исходные требования:

– Определение стоимости прохода по конкретному типу участка;
– Создание оптимального маршрута прокладки трубопровода по двум точка;

Таблица 9. Анализ пользователей

Название	Представляет	Роль
Конечный пользователь (User)	Специалистов, занимающихся расчетом стоимости обустройства месторождений	Использует функционал системы для автоматизации своей работы

b) Концептуальное проектирование: моделирование процесса, для которого разрабатывается приложение.

Таблица 10. Концептуальное проектирование

Как сейчас	Как будет

• Специалист получает задание построить оптимальный маршрут на конкретном месторождении с конкретными точками начала и конца, а также карту в масштабе 1:50 000
• Специалист вручную строит маршрут, полагаясь исключительно на свой опыт
• Специалист определяет координаты точек перегиба
• Специалист предоставляет карту с маршрутом и координатами точек перегиба
	• Специалист получает задание построить оптимальный маршрут на конкретном месторождении с координатами точек начала и конца
	• Специалист получает карту в масштабе 1:50 000
	  ○ Программа определяет типы местности
	  ○ Специалист определяет стоимость прокладки трубопровода по типам
	  ○ Программа строит маршрут, основываясь на оценках специалиста
	  ○ Специалист вносит корректировки (при необходимости)
	  ○ Специалист экспортирует карту с маршрутом и координатами точек перегиба
	• Специалист не получает карту
	  ○ Специалист выбирает участок и слои из БД
	  ○ Программа строит маршрут, основываясь на оценках, предусмотренных в БД
	  ○ Специалист вносит корректировки (при необходимости)
	  ○ Специалист экспортирует карту с маршрутом и координатами точек перегиба
	• Специалист предоставляет карту с маршрутом и координатами точек перегиба

c) Физическое проектирование: выбор платформы, на которой будет реализован проект и средств разработки.

Проект реализован на java Script, python 2.7, библиотеке ArcGis для java Script (ArcGIS Web API for JavaScript API 4.0), html5, css3.

4.2. Последовательность действий

Последовательность действий представлена на диаграмме деятельности (рис. 4.1)

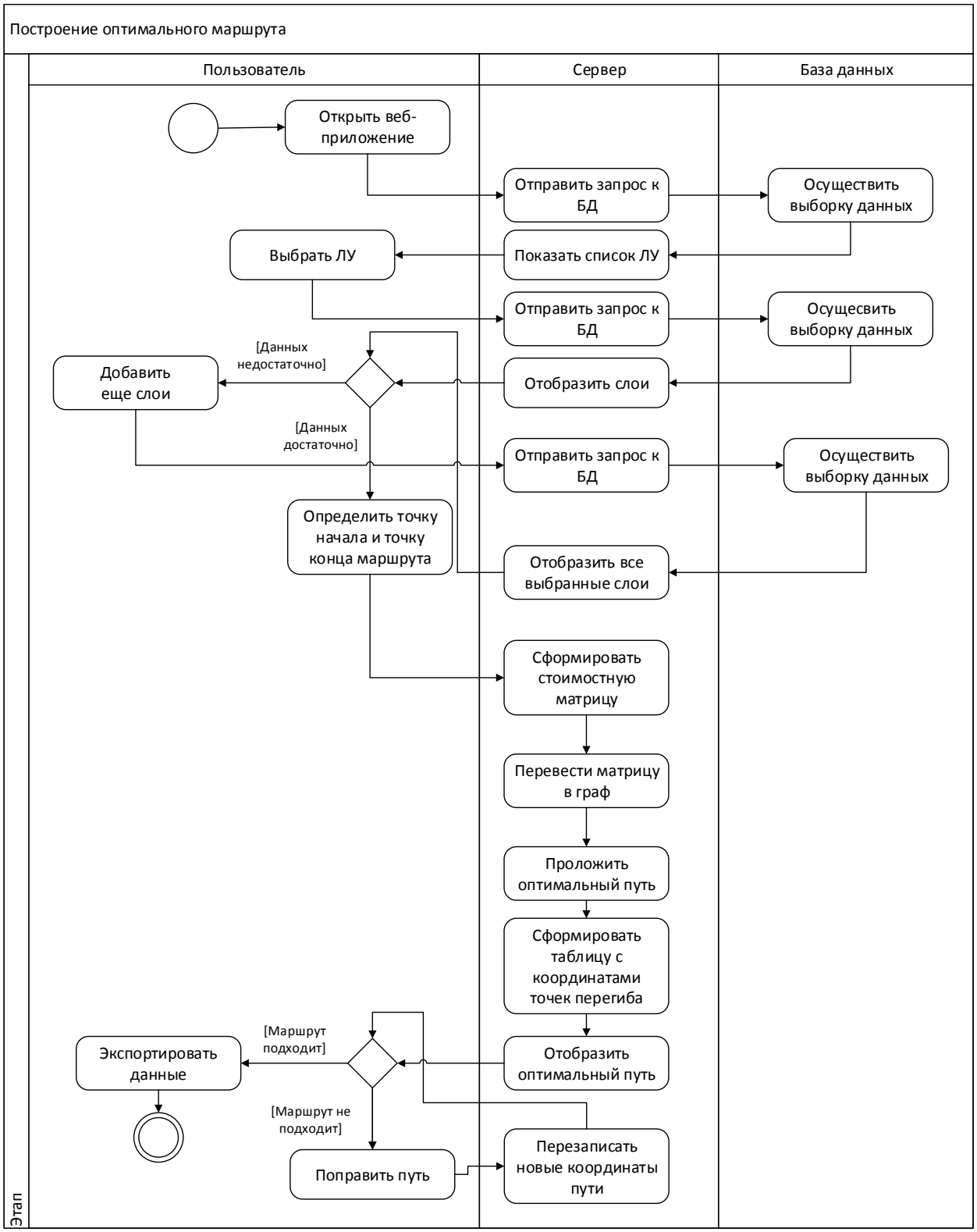

Рисунок 4.1 - Диаграмма деятельности

4.3. Создание карты

Для создания общей карты необходимо было преобразовать все объекты в растр, а затем просуммировать стоимости прохода по каждой ячейке в зависимости от объектов, к которым она относится.

4.4. Общие сведения о программной среде ArcGIS

ArcGIS основана на общей модульной библиотеке отдельных программных ГИС - компонентов. В состав библиотеки входит широкий набор программных модулей, позволяющих описать разные по сложности объекты. В комплексе эти модули предоставляют пользователям богатую функциональность современной ГИС.

В ArcGIS Desktop входят приложения:

- ArcCatalog,
- ArcMap,
- ArcGlobe ArcScene
- ArcToolbox
- ModelBuilder.

С помощью этих приложений и их интерфейсов можно выполнить разные задачи ГИС, включая анализ, редактирование данных их визуализацию и геообработку.

Предусмотрена возможность разработки своих собственных расширений к ArcGIS Desktop, с помощью ArcObjects – API библиотеки программных компонентов ArcGIS.

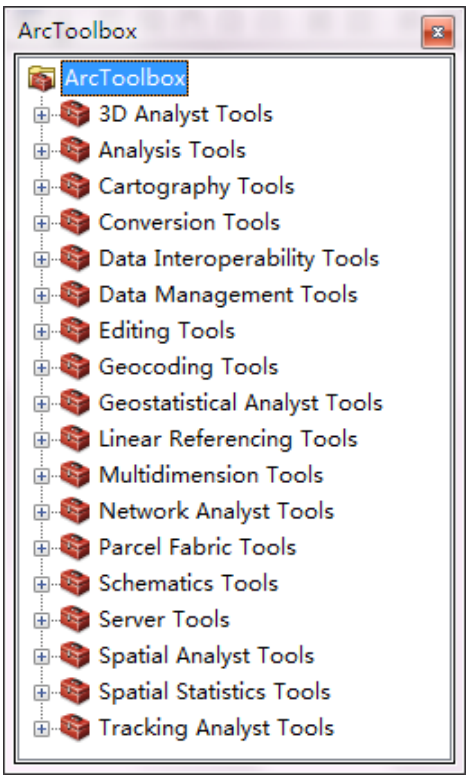

Рисунок 4.2 - Окно ArcToolbox

ArcToolbox (рис. 4.2) содержит обширный набор функций геообработки, включая инструменты для:

- управления данными;
- конвертации данных;
- обработки покрытий;
- векторного анализа;
- геокодирования;
- статистического анализа.

В Arcgis можно компоновать последовательности инструментов геообработки, подавая выход одного инструмента в другой инструмент в качестве входа.

ModelBuilder — это специальное приложение, для создания последовательности инструментов геобработки из встроенных функций и скриптов на языке Python2.7.

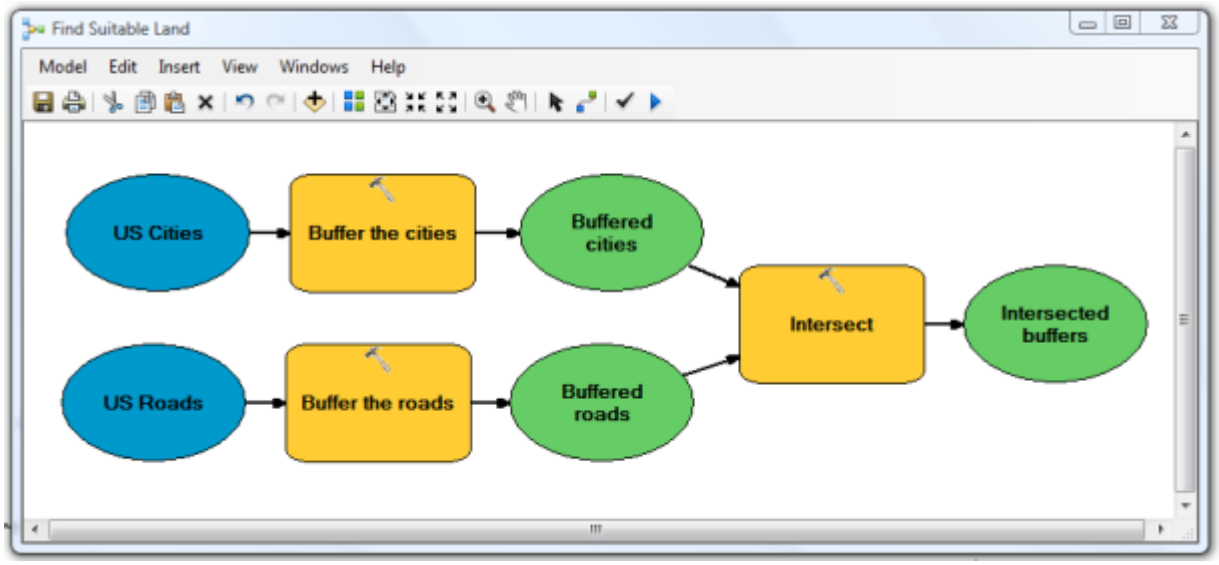

Рисунок 4.3 - Приложение ModelBuilder

4.5. Преобразование векторных объектов в растровые с помощью API ArcGIS

Для того, чтобы можно было создать общую карту для построения оптимального маршрута прокладки трубопровода, необходимо было преобразовать все данные в растровые. Это было осуществлено с помощью API ArcGIS, в частности инструментов для конвертации точек, полилиний и полигонов в растр.

Таблица 11. Входные параметры, необходимые для использования API

Параметр	Объяснение	Тип данных
in_features	Входной набор линейных объектов, который будет конвертирован в растр.	Feature Layer
value_field	Поле, используемое для присвоения значений выходному растру.	Field
out_rasterdataset	Создаваемый выходной набор растровых данных.	Raster Dataset

В качестве входных объектов берутся все слои, добавленные пользователем, для присвоения стоимости по умолчанию используется поле «Cost».

Алгоритм преобразования представлен на рисунке 4.4.

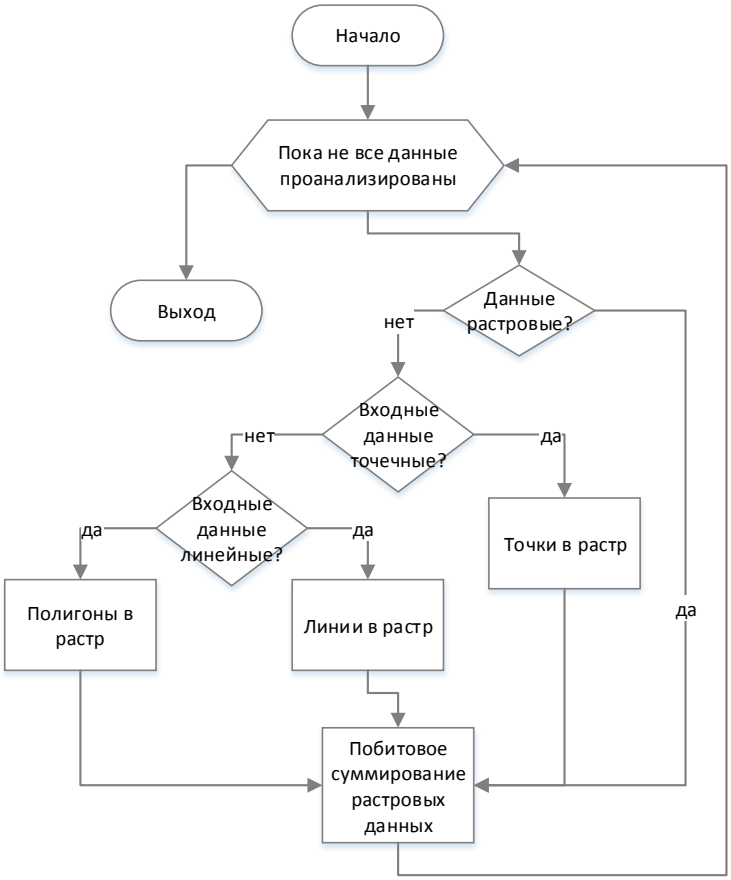

Рисунок 4.4 - Алгоритм перевода векторных объектов в растровые

4.6. Преобразование матрицы в граф связности

Происходит по шаблону ферзя. Алгоритм представлен в блок-схеме на рисунке 4.5.

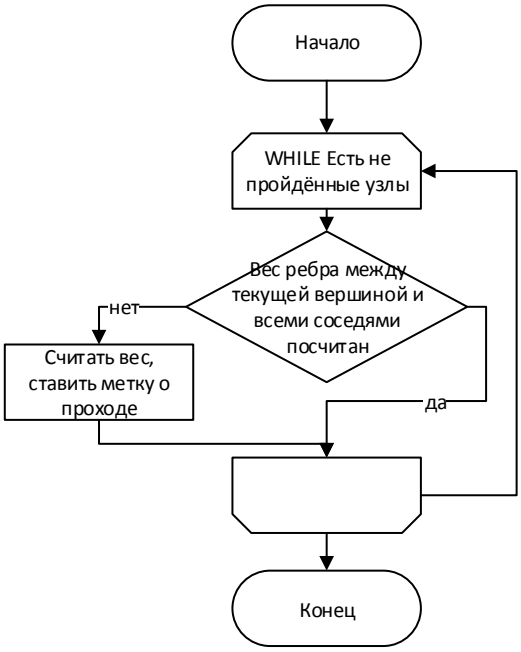

Рисунок 4.5 - Алгоритм построения графа связности

Расчет веса ребра происходит по простым математическим формулам, представленным выше в пункте про построение графа связности по данным растровой поверхности.

4.7. Модификация алгоритма

В классической версии алгоритм A* выполняется в 2 этапа: сначала ищется оптимальный маршрут, а потом выполняется обратная трассировка для нахождения пути. При этом сам алгоритм выглядит так рисунок 4.6.

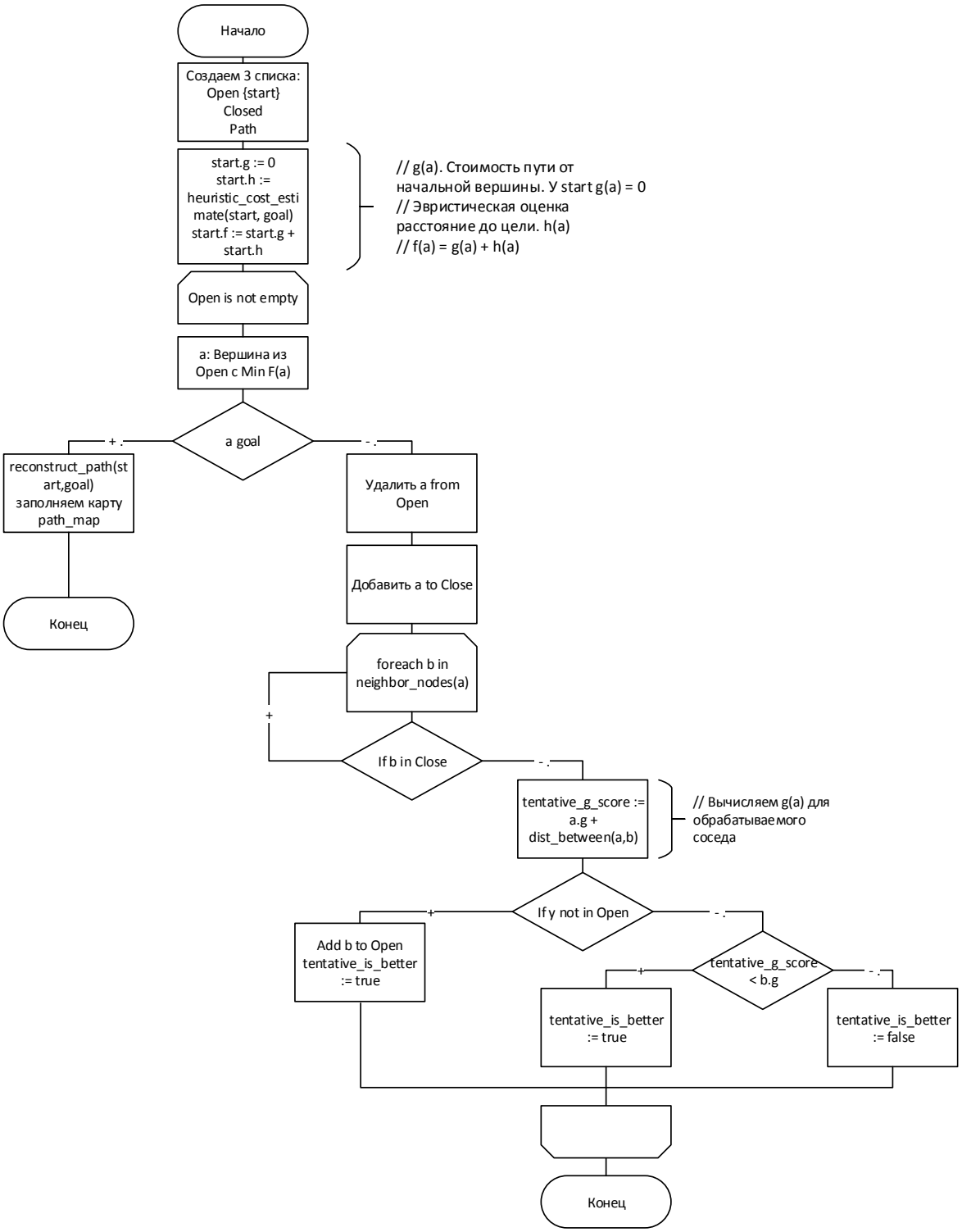

Рисунок 4.6 - Блок-схема алгоритма A*

Start и goal – начальная и конечная ячейки соответственно. A и b обозначают ячейки, просматриваемые алгоритмом, при этом у них есть данные по координатам.

На рисунке 4.7 представлена функция обратной трассировки.

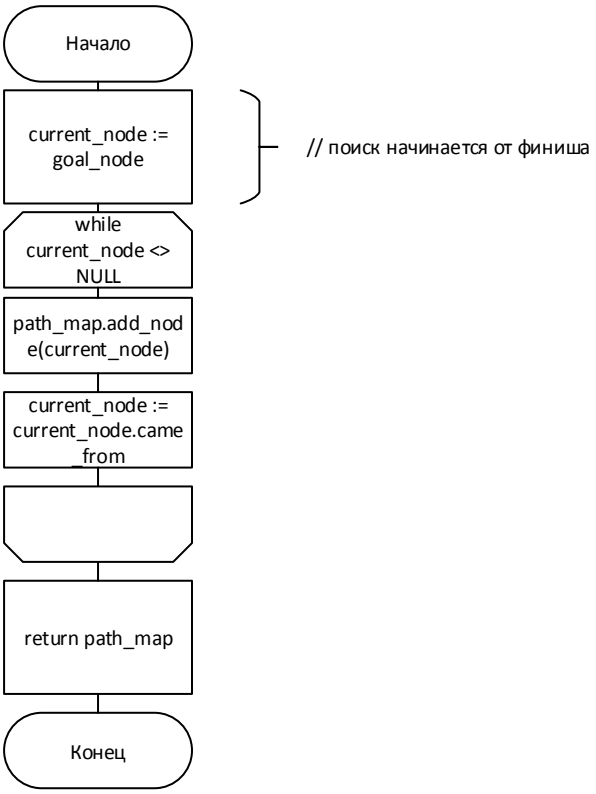

Рисунок 4.7 - Блок-схема алгоритма A*. Функция обратной трассировки

В данной работе алгоритм был модифицирован с помощью последовательных приближений и «запрещенных области», для хранения и возможности быстрого доступа к данным использовались хэш-таблицы. Лучевой поиск не был реализован, потому что в этом случае алгоритм не гарантирует оптимальности пути, хотя и может позволить найти разумный путь.

Как был реализован алгоритм в этой работе представлено на рисунках 4.8 и 4.9.

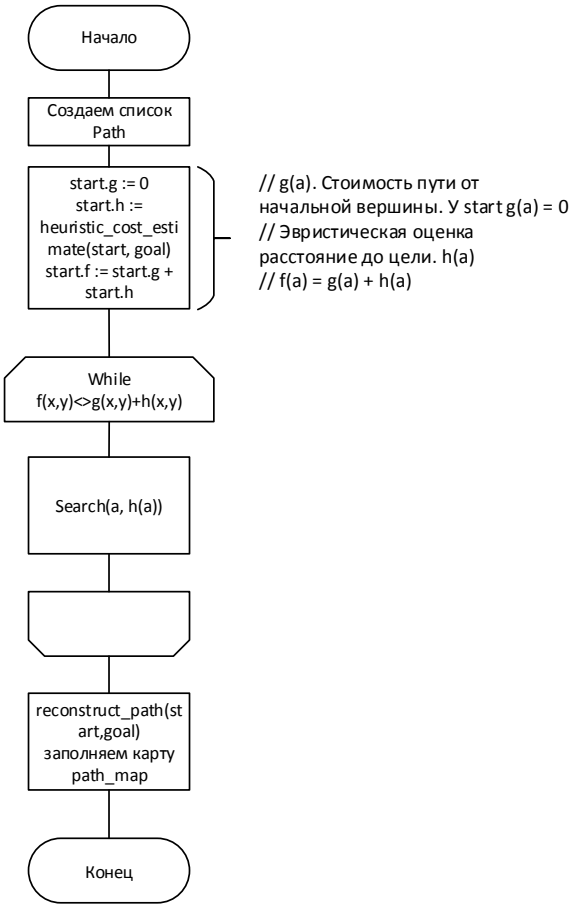

Рисунок 4.8. Блок-схема алгоритма A* после модификации.

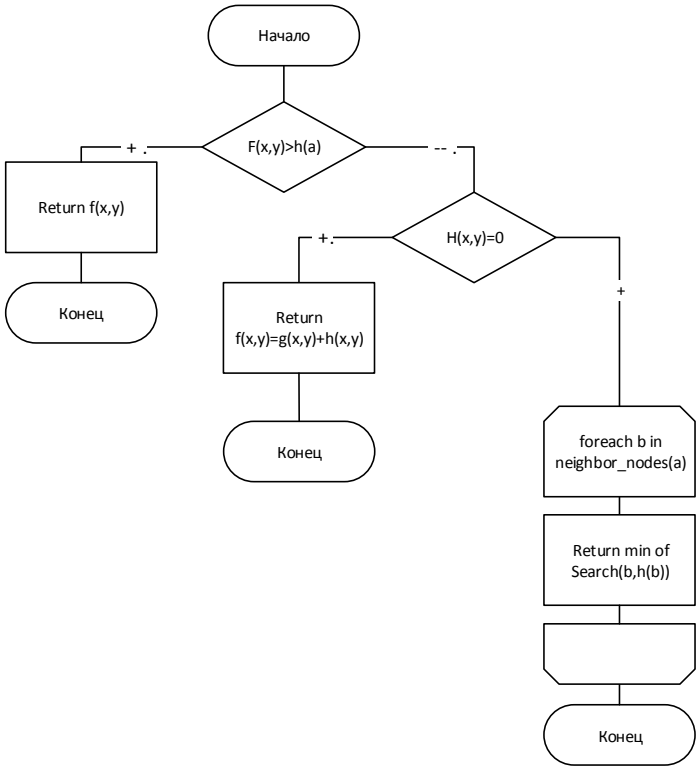

Рисунок 4.9. Функция для алгоритма последовательных приближений.

4.8. Основные экраны приложения

При запуске приложения появляется модальное окно для выбора ЛУ, оно показано на рисунке 4.10.

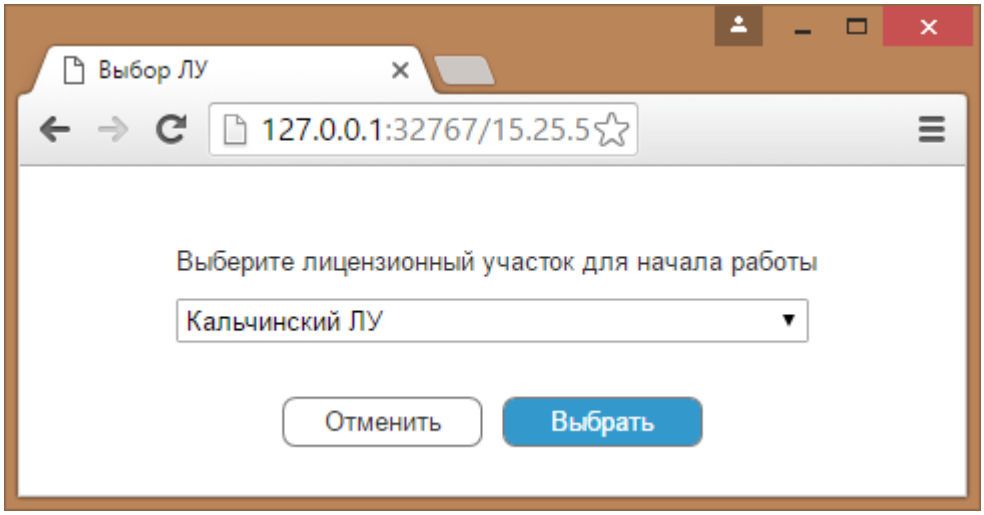

Рисунок 4.10 - Окно запуска приложения

После того, как пользователь выбрал лицензионный участок, автоматически подгружается информация о границах лицензионного участка и структурная карта (рис.4.11), если он, конечно, есть в базе. Если ее нет, то пользователь может подгрузить ее самостоятельно. Для этого существуют ограничения в виде масштаба (1:50 000) и системы координат (Pulkovo 42). Проверка на систему координат выполняется в автоматическом режиме.

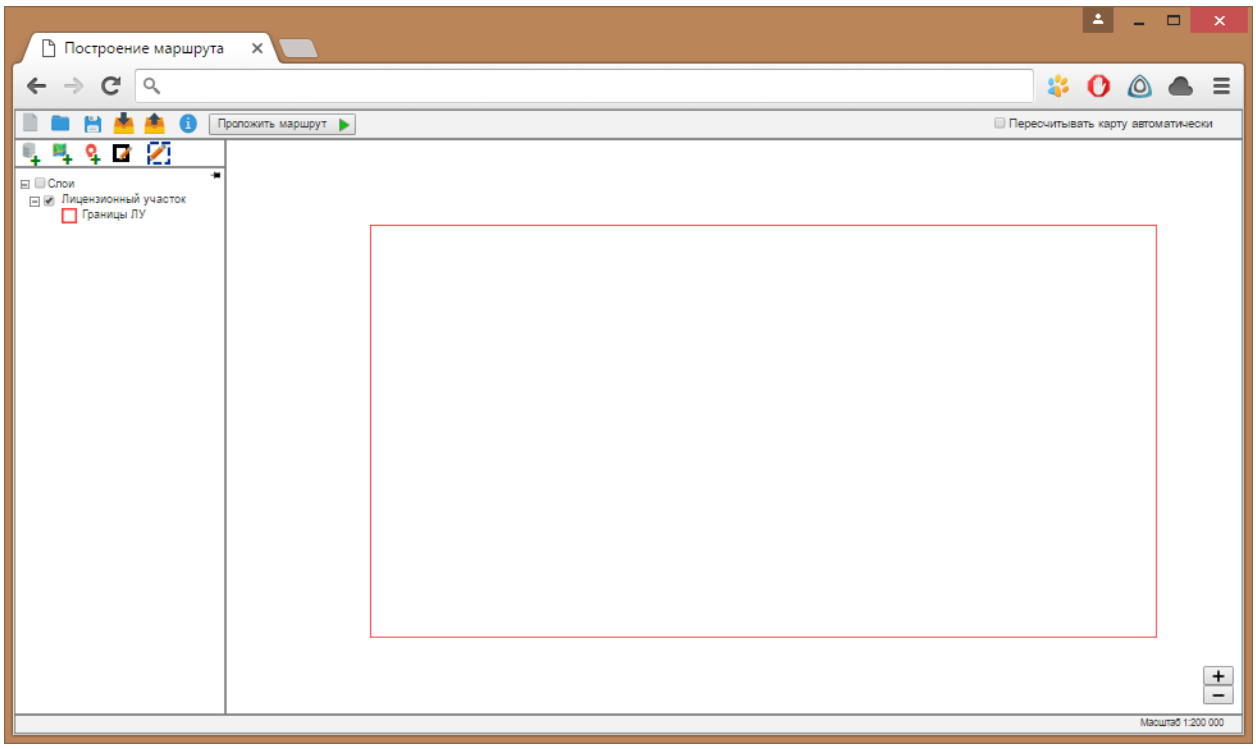

Рисунок 4.11 - Основное окно приложения

Пользователь может делать масштаб крупнее или мельче, текущей масштаб отображения показывается снизу. При этом все расчеты, как уже отмечалось ранее, проводятся на картах в масштабе 1:50 000.

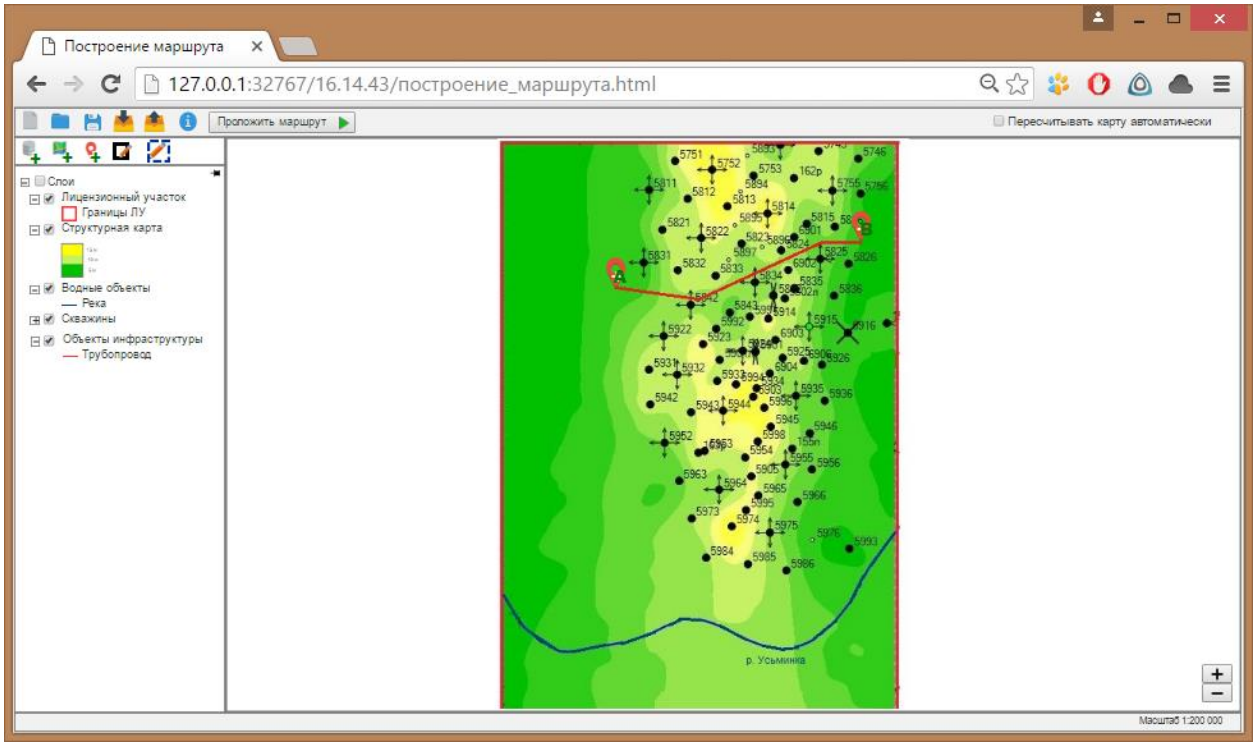

Рисунок 4.12 - Основное окно приложения

Так же в приложении реализована возможность изменять маршрут трубопровода, для этого необходимо выбрать инструмент редактирование. Новые точки перегиба можно добавлять, зажав клавишу <ctrl> и щелкнув по трубопроводу в нужной точке. Для удаления нужно выделить нужную точку и нажать <delete>.

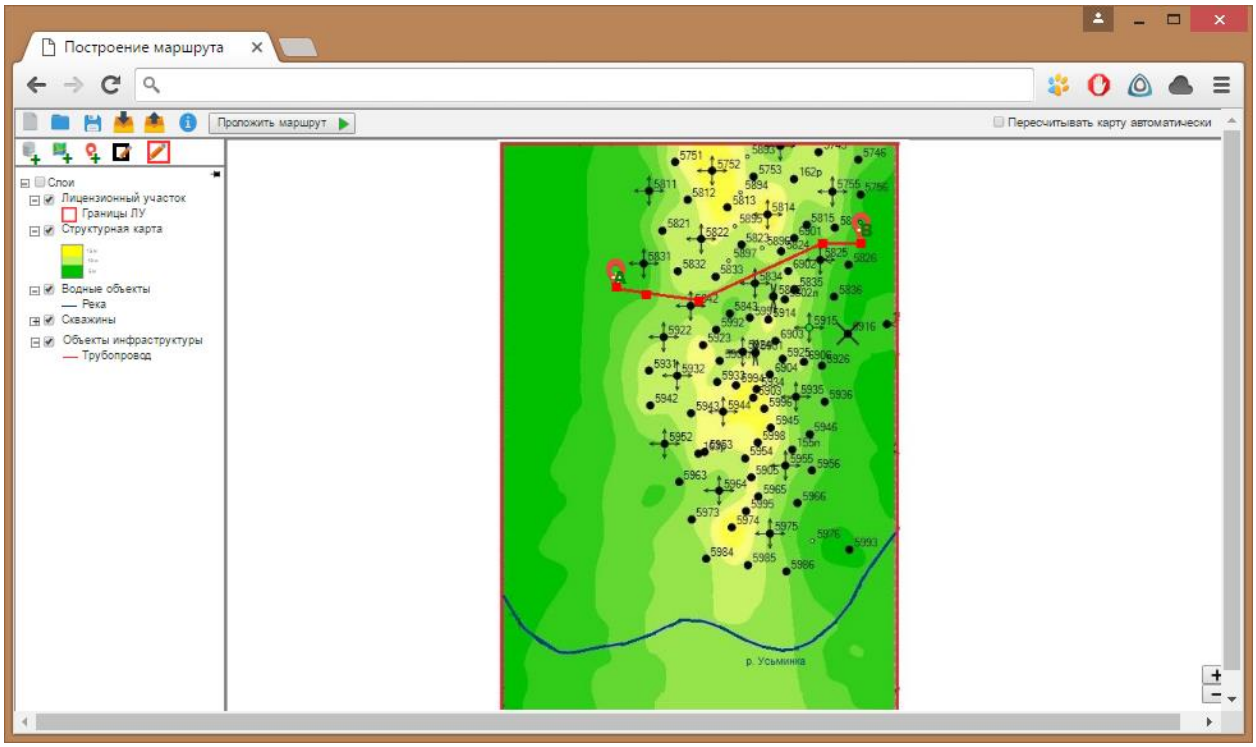

Рисунок 4.13 - Основное окно приложения. Редактирование трубопровода

Из приложения можно экспортировать точки перегибов трубопровода, построенного автоматически, в Excel. Экспортированные данные выгружаются в формате, представленном на рисунке 4.14.

Points2.xlsx

	A	B	C	D
1	id	X	Y	
2	1	520000	6000000	
3	2	540000	6000000	
4	3	560000	6000000	
5	4	580000	6000000	
6	5	600000	6000000	
7	6	620000	6000000	
8	7	520000	6100000	
9	8	540000	6100000	
10	9	560000	6100000	
11	10	580000	6100000	
12	11	600000	6100000	
13	12	620000	6100000	
14	13	640000	6000000	
15	14	660000	6000000	
16	15	680000	6000000	
17	16	700000	6000000	
18	17	620000	6100000	
19	18	640000	6100000	
20	19	660000	6100000	
21	20	680000	6100000	
22				

Рисунок 4.14 – Пример выгрузки координат точек перегибов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была достигнута основная цель, решены частные задачи и получены следующие результаты и выводы:

- Проведен анализ методов преобразования массивов в графы;
- Проведен сравнительный анализ существующих алгоритмов построения оптимального расстояния;
- Реализовано преобразование векторных объектов в единую растровую карту со стоимостными оценками;
- Реализован алгоритм оптимального построения маршрута A star;
- Разработан интерфейс приложения;
- Реализовано веб-приложение с возможностью построения и редактирования маршрута, а также экспорта полученных данных в табличном и графическом виде.

В данный момент приложение находится на апробации в ООО «ТННЦ».

ПРИЛОЖЕНИЕ 1.

Видение продукта

Введение (Introduction)

Цель данного документа заключается в том, чтобы собрать, проанализировать и определить общие задачи и особенности системы RuteMaker. Центральное место в нем занимают возможности, которые необходимы заинтересованным сторонам и целевым пользователям, а также объяснения причин возникновения этих потребностей.

Этот документ описывает видение предлагаемой системы RuteMaker. Система RuteMaker позволяет своим пользователям прокладывать оптимальный маршрут линейных объектов, в частности трубопроводов, используя Web-интерфейс (существующий браузер).

Цель (Purpose)

Цель написания документа-видения – окончательно сформировать требования к разрабатываемому продукту.

Область применения (Scope)

Этот документ фиксирует видение системы RuteMaker. Он включён в проект, который предоставляет возможность прокладывать оптимальный маршрут линейных объектов через Web-интерфейс.

Определения, акронимы и сокращения (Definitions, Acronyms and Abbreviations)

Определения терминов и допустимых операций являются стандартным для ГИС. Более подробная информация приведена в Глоссарии.

Ссылки (References)

Отсутствуют.

Обзор (Overview)

Этот документ содержит положения о позиционировании продукта, анализ заинтересованных лиц системы, анализ ожидаемых пользователей системы и список функциональных возможностей, которые необходимо обеспечить системе. Эти возможности выявлены благодаря исходной информации, полученной от заинтересованных сторон.

Позиционирование (Positioning)

Цели проекта (Business Opportunity)

Этот продукт в дальнейшем станет модулем для автоматизированного расчетного комплекса - «Модель расчета стоимости обустройства месторождений». Ожидается, что комплекс будет внедрен в группе компаний ООО «Роснефть» и ДО.

Постановка задачи (Problem Statement)

Проблема: высокая стоимость создания ТЭО проектной документации

Оказывает влияние на: общую стоимость проектной документации

Проявляется в виде: высокой стоимости работ по созданию проектной документации

Успешное решение: сократит стоимость полного пакета проектной документации на 5-20%

Позиционирование продукта (Product Position Statement)

Для: специалистов.

Которые: занимаются расчетом стоимости обустройства месторождений.

RuteMaker: web-приложение для автоматического построения оптимального маршрута трубопроводов

Которое: автоматизируют создание оптимального маршрута трубопровода и сократит трудозатраты на создание ТЭО

Заинтересованные стороны и пользователи (Stakeholder and User Descriptions)

Описание рынка (Market Demographics)

На рынке не существует продуктов предоставляющих подобный функционал через веб-интерфейс.

Обзор заинтересованных сторон (Stakeholder Summary)

В этом продукте принимает участие небольшое количество заинтересованных сторон. Следующая таблица перечисляет тех из них, которые несут основную ответственность и оказывают основное влияние на успех проекта.

Название	Представляет	Роль
Менеджер продукта (Product Manager)	Пользователей системы	Определяет продукт, оценивает возможности и области применения разных версий системы.
Конечный пользователь (User)	Специалистов, занимающихся расчетом стоимости обустройства месторождений	Использует фунционал системы для автоматизации своей работы

Обзор пользователей (User Summary)

Название	Описание	Заинтересованное лицо
Конечный пользователь (End User)	Грамотный пользователь, который часто находится в системе и использует ее расширенные возможности.	Конечный пользователь RuteMaker
Разработчик RuteMaker	Часть команды, которая разрабатывает систему (разработчик программного обеспечения, тестировщик и т.д.)	Руководитель разработки

Пользовательское окружение (User Environment)

Пользователи будут иметь доступ к системе через стандартный Интернет-браузер; им нет необходимости изучать новый пользовательский интерфейс. Они будут обращаться к системе с большого количества разнообразных клиентских платформ, таких, как Windows, Macintosh, Linux, OS/2 и т.д.

Ключевые потребности заинтересованных сторон и пользователей (Key Stakeholder / User Needs)

Потребность	Приоритет	Текущее решение	Предлагаемое решение
Построение оптимального маршрута прокладки трубопровода на карте	Высокий	Построение производится вручную, на усмотрение специалиста	Построение производится автоматически, весовые коэфициенты типов местности задаются по усмотрению специалиста
Построение оптимального маршрута прокладки трубопровода по геоданным из корпоративной базы данных	Низкий	Построение производится только по картам	Построение производится автоматически, весовые коэфициенты типов местности заложены в базе

Построение оптимального маршрута прокладки трубопровода по геоданным из корпоративной базы данных получило низкий приоритет после совещания с руководителем группы проектирования линейных объектов. На совещании было отмечено, что маршруты строятся по данным карт (по причине того, что данные на картах, как правило, более актуальны и достоверны, в отличие от данных, полученных из корпоративной базы данных)

Альтернативные варианты (Alternatives and Competition)

На данный момент существуют следующие альтернативы:

- Продолжить создавать оптимальный маршрут вручную, опираясь только на субъективное мнение специалиста.
- Покупка большого количества лицензий программного продукта ArcGIS.
- Разработка и внедрение дополнительных модулей для уже существующей внутри компании системы РН-КИН.

Обзор продукта (Product Overview)

Все конечные пользователи будут обращаться к возможностям системы с помощью стандартных Интернет-браузеров. Система предоставляет специалистам функционал для построения оптимального маршрута по имеющимся данным.

Перспектива продукта (Product Perspective)

RuteMaker является полноценным модулем системы для расчета стоимости обустройства месторождений. В дальнейшем планируется расширить функционал, например, добавить возможность для построения оптимального маршрута на основе геоданных из корпоративного хранилища.

Обзор возможностей (Summary of Capabilities)

Выгода покупателя (Customer Benefit)	Опорные функциональные возможности (Supporting Features)
Автоматизированное создание маршрута трубопровода	– Выгрузка координат точек перегибов; – Внесение правок в маршрут, построенный автоматически; – Работа с актуальными геоданными, взятыми из корпоративной БД; – Работа с картами растрового формата.

Допущения и зависимости (Assumptions and Dependencies)

Базовые возможности системы достижимы при помощи современных технологий. Рискованных зависимостей не предвидится.

Затраты и ценообразование (Cost and Pricing)

Необходимо определить.

Лицензирование и установка (Licensing and Installation)

Необходимо определить.

Функциональные возможности продукта (Product Features)

1. Импорт карты в растровом формате;
2. Импорт геоданных из базы для создания карты местности (месторождения), на которой предполагается строить маршрут;
3. Определение стоимости прохода по конкретному типу участка;
4. Создание оптимального маршрута прокладки трубопровода по двум точка;
5. Изменение автоматически построенного маршрута в ручном режиме;
6. Экспорт карты с отрисованным маршрутом;
7. Экспорт координат точек перегиба.

Ограничения (Constraints)

Все возможности режима онлайн должны (must) функционировать в выбранном Интернет-браузере (Internet Explorer 11), при условии наличия модуля Microsoft Silverlight.

Диапазон качества (Quality Ranges)

Требуется дальнейшее исследование.

Очередность и приоритет (Precedence and Priority)

Приоритетным направлением является построение оптимального маршрута прокладки трубопровода на основе импортированной карты.

Другие требования к продукту (Other Product Requirements)

Системные требования (System Requirements)

Пользователю необходим компьютер (клиент) с возможностью доступа к Интернету. Скорость доступа и использование жесткого диска зависят от конфигурации пользователя и практики применения.

Технические требования (Performance Requirements)

Отображение карты и маршрута должно (must) выполняться со скоростью, удобной для пользователей Интернета (менее трех секунд, не считая времени на построение оптимального маршрута).

Требования к условиям эксплуатации (Environmental Requirements)

Дополнительных требований кроме тех, которые являются обычными для компьютера с возможностью работы в Интернете, нет.

Требования к документации (Documentation Requirements)

Вся документация для системы RuteMaker должна быть доступна и в печатном виде, и в режиме онлайн.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 ГОСТ Р 21.1001-2009. Основные требования к проектной и рабочей документации. – М. : Национальные стандарты, 2009. – 51 с.

2 ГОСТ 28397-89. Языки программирования. Термины и определения. – М. : Издательство стандартов, 1989. – 8 с.

3 СНиП 2.05.06-85. Магистральные трубопроводы. – М. : ФГУП ЦПП, 2000. – 7 с.

4 СНиП Ш-42-80. Магистральные трубопроводы. – М. : ВНИИСТ, 1997. – 33 с.

5 СП 103-34-96. Свод правил сооружения магистральных трубопроводов. Подготовка строительной полосы. - М. : ИРЦ Газпром, 1996. – 13 с.

6 СП 104-34-96. Свод правил сооружения магистральных трубопроводов. Производство земляных работ. – М. : ИРЦ Газпром, 1996. – 33 с.

7 Алиев, Р.А. Трубопроводный транспорт нефти и газа / Р.А. Алиев, В.Д. Белоусов, А.Г. Немудров, В.А. Юфин, Е.И. Яковлев. – М. : Недра, 1988. – 235 с.

8 Басов, Е.Д. Технология и организация строительства переходов магистральных трубопроводов через малые преграды / Е.Д. Басов. - М. : МИНГП им. И.М. Губкина, 2004. – 109 с.

9 Бородавкин, П.П. Выбор оптимальных трасс трубопроводов большой протяженности / П.П. Бородавкин, Е.М. Сощенко, Б.И. Ким. – М. : ВНИИОЭНГ, 1977. – 57 с.

10 Бородавкин, П.П. Строительство магистральных трубопроводов в сложных условиях / П.П. Бородавкин, А.Х. Сунарчин. - М. : Недра, 1965. – 207 с.

11Гольштейн, Е.Г., Юдин Д.Б. Задачи линейного программирования транспортного типа / Е.Г. Гольштейн, Д.Б. Юдин. - М. : Наука, 1969. – 384 с.

12Иванина, Д.С. Совершенствование проектирования линейной части магистральных трубопроводов с использованием математических методов и

средств вычислительной техники / Д.С. Иванина. - К. : ИК им. В.М. Глушкова, 1984. – 234 с.

13Кузнецов, И.С. Поиск маршрута прокладки инженерных сетей с наименьшей стоимостью / И.С. Кузнецов, Р.Н. Кузнецов, А.А. Горских // Строительство и архитектура. – 2009. - № 4. – С. 12-16.

14Совершенствование технологии и организации строительства линейной части трубопроводов, сборник статей. - М. : Недра, 1982. - 289 с.

15СПБЦ. Справочник базовых цен на проектные работы для строительства. – М. : ГУП ЦПП, 1999. – 22 с.

16СПБЦ. Справочник базовых цен на инженерно-геологические изыскания для строительства. - М. : ФГУП ПНИИС , 2004. - 134 с.

17Строительство магистральных трубопроводов, справочник. - М. : Недра, 1991. - 474 с.

18Трубопроводный транспорт, справочник. - М. : Нац. Ком.СССР по нефти, 1971. – 91 с.

19Рыльский, И.А. Оптимизация трасс трубопроводов с использованием ГИС-технологий / И.А. Рыльский // Вестник Московского Университета, серия География. – 2004. - № 4. - С. 34-41.

20Решетников, А.Д. Разработка методов сооружения трубопроводов в условиях Западной Сибири, обеспечивающих рациональную загрузку линейного потока по сезонам производства работ / А.Д. Решетников. - М. : ВНИСТ, 1984. – 120 с.

21Адигеев, М.Г. Анализ сложности параллельных алгоритмов. Методические указания для студентов факультета математики, механики и компьютерных наук [Электронный ресурс] / М.Г. Адигеев. – Ростов-на-Дону: ЮФУ, 2013. – Режим доступа: https://open-edu.sfedu.ru/pub/2416. Дата обращения: 08.03.2016 г.

22Алгоритм A* для новичков [Электронный ресурс]. – Режим доступа: http://www.policyalmanac.org/games/aStarTutorial_rus.htm. Дата обращения: 25.11.2015 г.

23Алгоритм поиска A* [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0_A*. Дата обращения: 21.12.2015 г.

24Алгоритм поиска пути Jump Point Search [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/162915/. Дата обращения: 05.10.2015 г.

25Алгоритмы поиска путей на JavaScript [Электронный ресурс]. - Режим доступа: http://habrahabr.ru/post/238001/ http://qiao.github.com/PathFinding.js/visual. Дата обращения: 18.06.2015 г.

26Аничкин, А. Минэкономразвития атакует «Газпром» [Электронный ресурс] / А. Аничкин. – М. : Km.ru, 2012. – Режим доступа: http://www.km.ru/economics/2012/08/29/gazprom/minekonomrazvitiya-atakuet-gazprom. Дата обращения: 22.02.2015 г.

27Грачева, Е. "Роснефть" открыла крупнейшее месторождение нефти в Арктике [Электронный ресурс] / Е. Грачева. – М. : Вести.Ru, 2014 г. – Режим доступа http://www.vesti.ru/doc.html?id=2001908. Дата обращения: 20.09.2015 г.

28Дискретные структуры: матан для айтишников [Электронный ресурс]. - Режим доступа: http://habrahabr.ru/post/252077/ https://stepic.org/course/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D1%8B%D0%B5-%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D1%8B-83. Дата обращения: 12.02.2016 г.

29Индексация (программирование) [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Дата обращения: 08.01.2016 г.

30Климентьева, Л. ЕС добивается от США отмены запрета на экспорт нефти / Л. Климентьева. – М. : ВЕДОМОСТИ - деловая газета, 2014. - Режим доступа: http://www.vedomosti.ru/politics/news/33231791/es-dobivaetsya-ot-ssha-otmeny-zapreta-na-eksport-nefti-v. Дата обращения: 11.02.2015 г.

31Корсей, С.Г. Применение ГИС в трубопроводном транспорте [Электронный ресурс] / С.Г. Корсей, Р.Г. Прохожаев, И.А. Рыльский. – М. : ЗАО «АРКОН» , 2008. – Режим доступа: http://neftegaz.ru/science/view/265. Дата обращения: 12.12.2015 г.

32Корсей, С.Г. ГИС-технологии в трубопроводном транспорте [Электронный ресурс] / С.Г. Корсей, Н.Б. Дьякова. – М. : ЗАО «АРКОН», 2007. - Режим доступа: http://neftegaz.ru/science/view/208. Дата обращения: 17.11.2015 г.

33Котов, А.Н. Метод минимизации суммарных затрат по развертыванию распределительных сетей на основе модифицированного метода Дейкстры [Электронный ресурс] / А.Н. Котов. – СПб. : СПбГУТ им. проф. М.А. Бонч-Бруевича, 2007. – Режим доступа: http://jurnal.org/articles/2007/radio4.html. Дата обращения: 26.04.2015 г.

34Кратчайшие пути в графах [Электронный ресурс]. - Режим доступа: http://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%9A%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D0%B5_%D0%BF%D1%83%D1%82%D0%B8_%D0%B2_%D0%B3%D1%80%D0%B0%D1%84%D0%B0%D1%85. Дата обращения: 05.04.2015 г.

35Кузнецов, Р. Н. Определение оптимального маршрута прокладки газопровода [Электронный ресурс] / Р.Н. Кузнецов. – Воронеж : ВГАСУ, 2009. – Режим доступа: http://www.dissercat.com/content/opredelenie-optimalnogo-marshruta-prokladki-gazoprovoda. Дата обращения: 16.07.2015 г.

36Курносов, М.Г. Амортизационный анализ (amortized analysis) [Электронный ресурс] / М.Г. Курносов. – Режим доступа:

http://www.mkurnosov.net/teaching/index.php/DSA/Fall2015. Дата обращения: 28.01.2016 г.

37Лекции Технопарка. 1 семестр. Алгоритмы и структуры данных [Электронный ресурс]. - Режим доступа: http://habrahabr.ru/company/abbyy/blog/251561/. Дата обращения: 01.12.2015 г.

38Нахождение пути на карте [Электронный ресурс]. - Режим доступа: http://www.firststeps.ru/theory/karta.html#marshr. Дата обращения: 07.03.2016 г.

39Обзорные онлайн-курсы по математике [Электронный ресурс]. - Режим доступа: http://habrahabr.ru/company/stepic/blog/249955/. Дата обращения: 03.04.2016 г.

40Обход препятствий: волновой алгоритм (Алгоритм Ли) 8 клеточный [Электронный ресурс]. - Режим доступа: http://suvitruf.ru/2012/05/16/1211/. Дата обращения: 30.08.2015 г.

41Оверченко, М. Россия может потерять до 38% газовых доходов из-за экспорта СПГ из США [Электронный ресурс] / М. Оверченко, А. Агамалова. - М. : ВЕДОМОСТИ - деловая газета, 2014. – Режим доступа: http://www.vedomosti.ru/companies/news/33694971/gazprom-mozhet-poteryat-18-vyruchki-iz-za-konkurencii-s. Дата обращения: 27.03.2015 г.

42Поиск кратчайших путей: алгоритм Дейкстры [Электронный ресурс]. - Режим доступа: http://traditio-ru.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%BA%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D1%85_%D0%BF%D1%83%D1%82%D0%B5%D0%B9:%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B. Дата обращения: 18.04.2015 г.

43Поиск пути в гексагональной сетке [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/115689/. Дата доступа: 14.11.2015 г.

44Принцип работы алгоритма поиска пути Астар (A*) [Электронный ресурс]. - Режим доступа: http://www.gamedev.ru/code/articles/?id=4246. Дата обращения: 11.08.2015 г.

45Расстояние Чебышёва [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5_%D0%A7%D0%B5%D0%B1%D1%8B%D1%88%D1%91%D0%B2%D0%B0. Дата обращения: 14.10.2015 г.

46Рыльский, И.А. Геоинформационное обеспечение автоматизированного трассирования трубопроводов [Электронный ресурс] / И.А. Рыльский. – М. : МГУ им. М.В. Ломоносова, 2006. - Режим доступа: http://www.dissercat.com/content/geoinformatsionnoe-obespechenie-avtomatizirovannogo-trassirovaniya-truboprovodov-0. Дата обращения: 03.02.2016 г.

47Эвристики для поиска кратчайших путей [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru/wiki/index.php?title=%D0%AD%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8_%D0%B4%D0%BB%D1%8F_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0_%D0%BA%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D1%85_%D0%BF%D1%83%D1%82%D0%B5%D0%B9. Дата обращения: 01.09.2015 г.

48Эвристический поиск [Электронный ресурс]. – Режим доступа: http://chernykh.net/content/view/293/493/. Дата обращения: 13.06.2015 г.

49Хеш-таблица [Электронный ресурс]. – Режим доступа: http://kvodo.ru/hash-table.html. Дата обращения: 19.11.2015 г.

50Хеш-таблица (Hash-Table). Реализация C++ [Электронный ресурс]. – Режим доступа: http://evilcoderr.blogspot.ru/2013/01/hash-table-c.html. Дата обращения: 11.12.2015 г.

51Cormen, Thomas H. Introduction to Algorithms Third Edition Thomas / Thomas H Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. – Ldn. : The MIT Press, 2009. – 1313 с.

52Deo, N. Shortest-path algorithms: Taxonomy and Annotation / N. Deo, C. Pang // Networks. – 1984. - № 14. - С. 275–323.

53Hart, P. A Formal Basis for the Heuristic Determination of Minimum Cost Paths / P. Hart, N. Nilsson, B. Raphael // IEEE Trans. Syst. Science and Cybernetics. – 1968. - № SSC-4(2). – С. 100-107.

54Koenig, S. Incremental Heuristic Search in Artificial Intelligence / S. Koenig, M. Likhachev, Y. Liu, D. Furcy // Artificial Intelligence Magazine. – 2004. - № 25(2). - С. 99-112.

55Koenig, S. Fast Replanning for Navigation in Unknown Terrain / S. Koenig, M. Likhachev // Transactions on Robotics. – 2005. - 21, (3). - С. 354-363.

56Koenig, S. Lifelong Planning A* / S. Koenig, M. Likhachev, D. Furcy // Artificial Intelligence Journal 155. – 2004. – С. 93-146.

57Stentz, A. The Focussed D* Algorithm for Real-Time Replanning / A. Stentz // Proceedings of the International Joint Conference on Artificial Intelligence. – 1995. - С. 1652–1659.

58Sun, X. Generalized Adaptive A*S / X. Sun, S. Koenig, W. Yeoh // In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). – 2008. – С. 469-476.

59Sun, X. The Fringe-Saving A* Search Algorithm - A Feasibility Study / X. Sun, S. Koenig // In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). – 2007. – С. 2391-2397.

60A* Variants [Электронный ресурс]. – Режим доступа: http://movingai.com/astar-var.html. Дата обращения: 17.09.2015 г.

61Algorithm A* [Электронный ресурс]. Режим доступа: http://qiao.github.io/PathFinding.js/visual/. Дата обращения: 03.03.2015 г.

62Amit’s Game Programming Information [Электронный ресурс]. – Режим доступа: www-cs-students.stanford.edu/~amitp/gameprog.html#hex. Дата обращения: 24.02.2016 г.

63ArcGIS Resources [Электронный ресурс]. – Режим доступа: http://resources.arcgis.com/ru/help/main/10.1/index.html#//004700000053000000. Дата обращения: 29.03.2016 г.

64D* [Электронный ресурс]. - Режим доступа: http://en.wikipedia.org/wiki/D*. Дата обращения: 02.04.2015 г.

65Chowdhury, Rezaul A. Analysis of Algorithms [Электронный ресурс] / Rezaul A. Chowdhury. – NY. : Stony Brook University, 2015. - Режим доступа: http://www3.cs.stonybrook.edu/~rezaul/CSE548-S15.html. Дата обращения: 09.09.2015 г.

66Harabor, D. Clearance-based Pathfinding [Электронный ресурс] / D. Harabor. – Cbr. : The Australian National University, 2009. – Режим доступа: https://harablog.wordpress.com/2009/01/29/clearance-based-pathfinding/. Дата обращения: 13.10.2015 г.

67Harabor, D. Fast Pathfinding via Symmetry Breaking [Электронный ресурс] / D. Harabor. – Cbr. : The Australian National University, 2011. – Режим доступа: https://harablog.wordpress.com/2011/08/26/fast-pathfinding-via-symmetry-breaking/. Дата обращения: 02.10.2015 г.

68Harabor, D. Jump Point Search [Электронный ресурс] / D. Harabor. – Cbr. : The Australian National University, 2011. – Режим доступа: https://harablog.wordpress.com/2011/09/07/jump-point-search/. Дата обращения: 02.10.2015 г.

69Harabor, D. Rectangular Symmetry Reduction [Электронный ресурс] / D. Harabor. – Cbr. : The Australian National University, 2011. – Режим доступа: https://harablog.wordpress.com/2011/09/01/rectangular-symmetry-reduction/. Дата обращения: 02.10.2015 г.

70Heuristics and A* Pathfinding [Электронный ресурс]. – Режим доступа: http://www.policyalmanac.org/games/heuristics.htm. Дата обращения: 16.10.2015 г.

71Incremental heuristic search [Электронный ресурс]. - Режим доступа: http://en.wikipedia.org/wiki/Incremental_heuristic_search. Дата обращения: 12.05.2015 г.

72Real-Time Replanning in Dynamic and Unknown Environments [Электронный ресурс]. - Режим доступа:

http://www.frc.ri.cmu.edu/~axs/dynamic_plan.html. Дата обращения: 06.07.2015 г.

73Pathfinding [Электронный ресурс]. - Режим доступа: http://en.wikipedia.org/wiki/Pathfinding. Дата обращения: 27.09.2015 г.

74Step-by-Step Examples- A Star Search [Электронный ресурс]. – Режим доступа: www.youtube.com/watch?feature=player_embedded&v=g0MJRpquEOk. Дата обращения: 29.01.2016 г.

75Taxicab geometry [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Taxicab_geometry. Дата обращения: 24.05.2015 г.