Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Петрозаводский государственный университет»
Институт математики и информационных технологий
кафедра информатики и информационного обеспечения

(подпись соискателя)

Марина Алексеевна Погорянская

Магистерская дисертация

Идентификация типичных маршрутов мобильного пользователя в Петрозаводском городском округе

Направление 01.04.02 — Прикладная математика и информатика

Научный руководитель:
к.т.н., доцент О. Ю. Богоявленская

_____

(подпись руководителя)

Петрозаводск — 2020

Оглавление

Введение

В наше время, когда, любое перемещение человека оставляет за собой информационный след, все более актуальными становятся системы, предлагающие различные услуги ~~в зависимости от~~ текущего положения пользователя. ~~Будь то~~ заказ еды из ближайшего кафе или вызов такси на адрес рядом стоящего дома. ~~Таким образом появилась идея разработать более индивидуальную систему, работающую по аналогичному принципу:~~ систему, которая на основе личных предпочтений пользователя, исходя из его текущего местоположения будет напоминать о важных местах, которые пользователь часто посещал.

Установленное на мобильное устройство приложение, использующее GPS(ГЛОНАСС) - навигатор будет с определенной периодичностью производить измерения изменения положения пользователя и, используя систему GPS(ГЛОНАСС) - координат, сохранять требуемую информацию в формате, удобном для дальнейшего сопоставления с картами городов и необходимой обработки. Тогда полученные данные будут анализироваться для выявления типичных для определенного пользователя маршрутов, часто посещаемых мест.

По ходу посещения пользователем приложения мест, которые были им запланированы, будет необходимо оставлять пометки о целях, из-за которых и был выбран конкретный маршрут передвижения. Эти подсказки будут использоваться для дальнейшего прогнозирования перемещений в данном направлении, а так же для определения добрался ли пользователь, до отмеченного на карте места в рамках текущей сессии приложения. Добавленную подсказку о цели передвижения пользователя, будем называть задачей, а ее выполнение будет

отслеживаться при помощи оценки расстояния от пользователя до места на карте, к которому задача была прикреплена.

На Рисунке 1 изображен фрагмент карты с нанесенными маршрутами и задачами. Разными цветами отмечены различные траектории движения пользователя, а к важным ориентирам прикреплены подсказки с описанием задач.

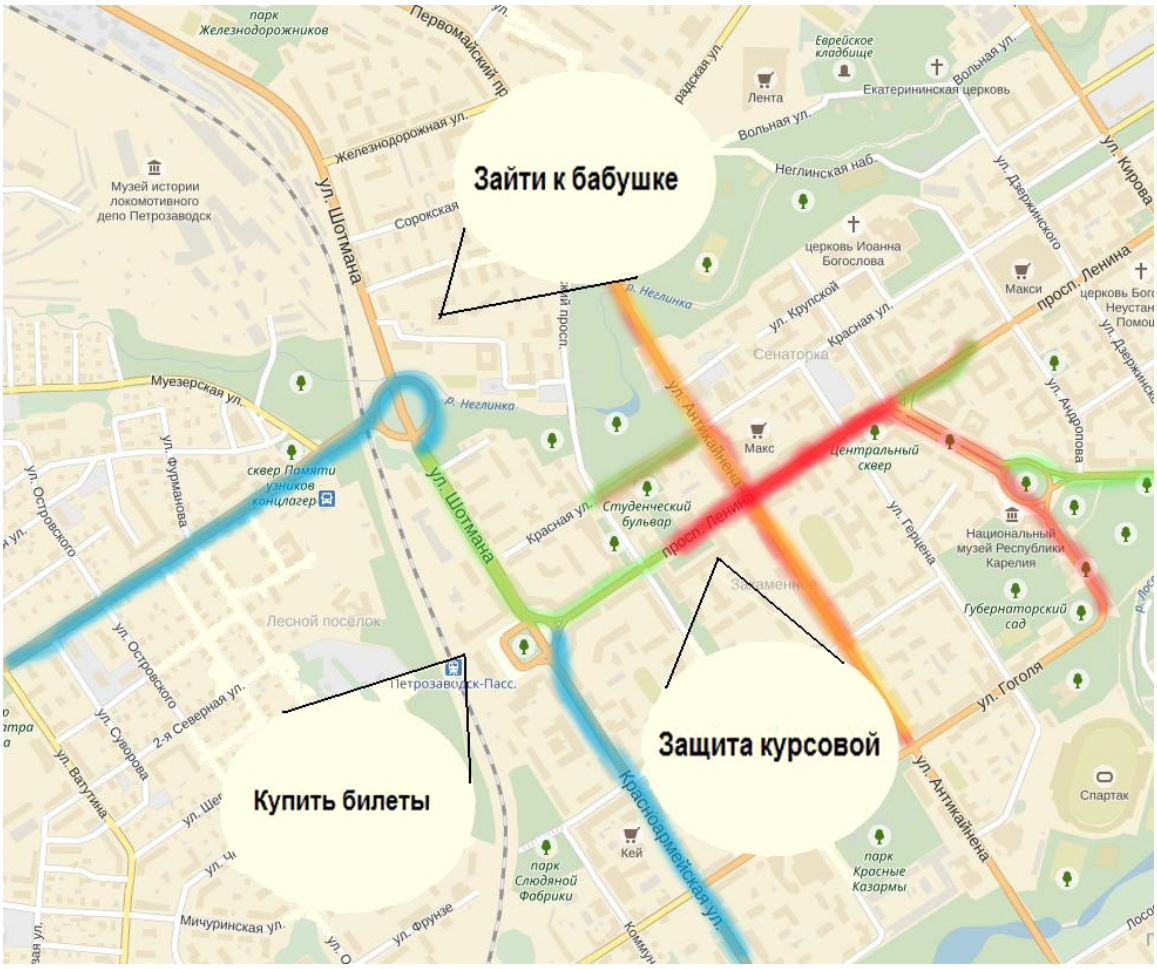

Рисунок 1 Используемые пути и подсказки на маршруте. [4] [5]

Цель:

Разработка прототипа системы подсказок на маршруте пользователя.

Задачи:

1. Выявление типовых маршрутов;
2. Анализ маршрутов и их составляющих;
3. Анализ типов связи между маршрутами;
4. Разработка системы предсказания направления пользователя;
5. Тестироние и анализ результата работы системы на экспериментальных данных.

Глава 1

Требования к системе

1.1 Терминология

Введем основные термины, используемые в работе:

Анализирование передвижения пользователя по GPS-координатам - это ~~бол~~ая задача, требующая понимания методов работы приложений, которые могут нам предоставить эти данные, понимания особнностей местности на которой проводятся замеры, изучения методов обощения и структуризации данных GPS-координат. Далее вводятся основные понятия, которые ~~помогут нам~~ структуризовать массивы данных с координатами, разработать методы разделения их на отдельные логические фрагменты, идентифицировать и удалить избыточные данные, ~~получаемые из-за порой излишне частых замеров GPS~~.

Рассмотрим множество S соеденненных некоторым образом точек такое что, $\forall$ точек A,B $\in$ S A $\neq$ B, и назовем его Множеством измерений, где под точкой понимается набор значений (x,y,z), где x, y $\in$ R и являются GPS(ГЛОНАСС)-координатами, а z - время в которое был осуществлен замер.

Введем функцию растояния между точками на метрическом пространстве (S, ρ), где S - множество соеденненных точек, а ρ - числовая

функция, принимающая вещественные значения, такая что:

1. $\rho(A, B) = 0$ тогда и только тогда когда A = B
2. $\rho(A, B) = \rho(B, A)$
3. $\rho(A, C) \leqq \rho(A, B) + \rho(B, C)$

в силу того что $\forall$ A и B $\in$ S состоят из координат, расстояние между ними задается следующим образом:

$\forall A, B, C \in S$, где $A = (x_a, y_a, z_a)$, $B = (x_b, y_b, z_b)$

$$rho(A, B) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \quad (1.1)$$

Следует отметить, что в расчетах функции растояния (1.1), не участвует координата Z, так как она имеет ~~др~~ ~~не~~ свойства, и используется для определения порядка соеденения точек множества S путем упорядочивания по времени.

Упорядоченное по времени множество - Множество M называется упорядоченным по времени, если $\forall$ A,B $\in$ M установленно отношение z1<z2, где A = (x1,y1,z1),B = (x2,y2,z2).

Такое упорядоченное по времени подмножество M $\in$ S называется Маршрутом, и имеет смысл набора координат, через которые последовательно двигался пользователь.

В силу того, что мы определяем порядок точек в маршруте путем упорядочиваниях их по времени, ~~мы должны понимать, что~~ разные маршруты, например принадлежащие разным датам, будут так же считаться продолжением друг друга, что не является верным. Множество S = (M1$\bigcup$M2$\bigcup$...$\bigcup$Mn), где n - количество маршрутов в множестве, которое определяется как количество пар из последовательных точек в упорядоченном по времени множестве S таких что A

$\in$ M1, B $\in$ M2,$|z_a - z_b| > Exp_1$.

Exp_1 - экспериментально полученное значение, характеризующее разрыв по времени между началом и концом разных маршрутов, в рамках текущей задачи выбран $Exp_1 = 30$ минутам.
Таким образом, если во временно упорядоченном множестве содержащем все точки множества S подряд окажутся две точки, время осуществления замера которых окажется большим чем Exp_1, мы будем утверждать, что эти точки относятся к разным маршрутам.
Другим важным экперементальным значением, является Exp_2 - значение характеризующее меру растояния между точками, на котором точки будут считаться равными, объединяться в одну.
Физический смысл этой переменной - ширина дороги, т.е расстояние меньшее ширины дороги - не значительно в рамках задачи и будет обобщаться. Нужно понимать, что работая с данными приложений которые с заданой периодичностью замеряют данные GPS не избежать повторяющихся значений. Причинами повторов могут стать автомобильные пробки, где в течении пары минут данные GPS могут дублироваться, или же слишком маленький интервал времени, в котором происходят замеры. Таким образом в исходных данных окаываются излишние данные, которые можно отбросить. Во время исследований карты местности, оказалось что значения отличающиеся друг от друга на растояние меньше средней ширины дороги, не дают информацию о смене направления движения пользователя, а скорее описывают практически незначимое смещение пользователя от основного маршрута, связанного с обходом препятствий или не точностью замера GPS, что приводит нас к понятию упрощенного маршрута.
Упрощенный маршрут - временно упорядоченное множество M, такое что $\forall$ A,B $\in$ M, ρ(A,B)$\geq Exp_2$.
В свою очередь точки, малое расстояние между которыми позволило нам их обобщить, называются Близкими. Строго говоря, близкие точки - $\forall$ m1,m2 $\in$ M ρ(A,B) $\leqq Exp_2$.

В рамках анализа передвижения пользователя стало важной задачей научиться различать маршруты по типам их взаимосвязи друг с другом:

1. Близкие маршруты – маршруты, целиком состоящие из близких точек, т.е $\forall$ m1 $\in$ M1 $\exists$ m2 $\in$ M2 такая что ρ(m1,m2) $\leqq Exp_2$.

2. Пересекающиеся маршруты – маршруты, в которых есть единственная общая близкая точка. $\forall$ m1 $\in$ M1 $\exists!$ m2 $\in$ M2 такая что ρ(m1,m2) $\leqq Exp_2$

3. Соприкасающиеся маршруты – маршруты, содержащие некоторое число близких точек $\sqsupset$ $\exists$ множество $M_1 \subset$ M1 и множество $M_2 = M1\$M_1, \forall$ m1 $\in M_1 \exists$ m2 $\in$ M2 такая что ρ(m1,m2) $\leqq Exp_2$, но $\forall$ m1 $\in M_2 !\exists$ m2 $\in$ M2 такая что ρ(m1,m2) $\leqq Exp_2$.

4. Различные маршруты – маршруты, не содержащие близких точек, $\forall$ m1 $\in$ M1 $\exists$ m2 $\in$ M2 такая что $\rho(m1, m2) > Exp_2$.

На Рисунке 2 в соответсвующем порядке представлены геометрические интерпретации типов взаимосвязи маршрутов (разными цветами выделены различные маршруты пользователя):

Особое внимание здесь следует уделить близким маршрутам, так как если мы на основе исходных данных обнаружим частое включение определенного маршрута, мы можем предположить что этот маршрут типичен для пользователя. ипичный маршрут – часто используемый маршрут, который за период отслеживания передвижения пользователя был обнаружен несколько раз с минимальными отклонениями от изначального маршрута. Разница между типичным и близким маршрутом заключается в частоте использования этого

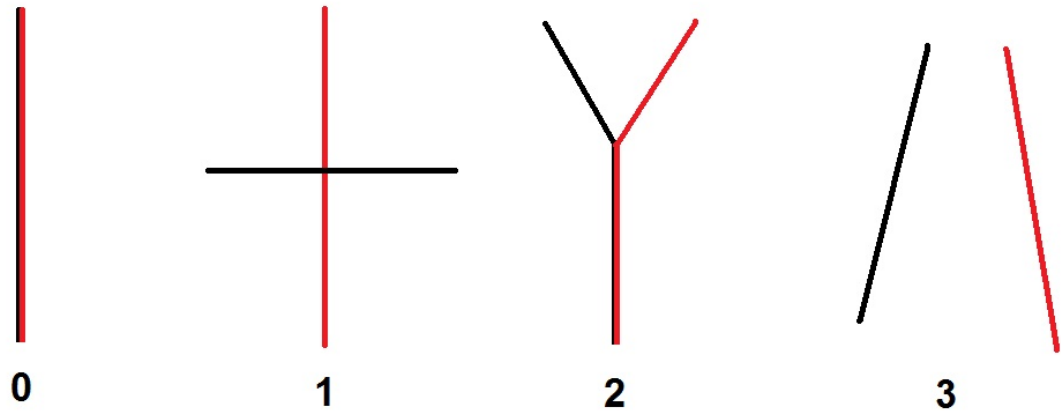

Рисунок 2. Типы взаимосвязи маршрутов: Близкие маршруты; Пересекающиеся маршруты; Соприкасающиеся маршруты; Различные маршруты.

маршрута, например из сотни маршрутов может оказаться что лишь десять являются различными между собой, каждый из них являлся близким маршрутом к другому, но типичными окажутся лишь те маршруты, которые использовались значительно большее количество раз, чем другие, какие-то маршруты могли встретится лишь дважды, когда другие типичные повторялись десятки раз.

В рамках поставленной задачи так же потребуются такие понятия как:

Точка, доступная за один шаг - следующая или предыдущая точка маршрута, относительно текущей. ∃ единственная пара точек m1, m3 ∈ M1 такая что, $z_m1 < z_m2 < z_m3$,где $m2$-текущая точка, а m1,m3 - точки доступные за один шаг. Так же точки доступные за один шаг можно определить через расстояние между ними: ∀ m1 ∈ M1 ∃ m2 ∈ M1, такая что ρ(m1,m2) $\leqq Exp_2 \cdot 2$, Exp_2 мы захватим соседнюю к нам точку, сразу за гранизей близости точек.

Точка прогноза - точка, доступная за один шаг, которая посещалась чаще других доступных точек.

1.2 Требования к прототипу системы подсказок на маршруте пользователя

1.2.1 Требования к алгоритмической части

Для разработки прототипа системы подсказок на маршруте пользователя необходимо:

1. Организовать хранение всей требуемой индивидуальной информации:
   1.1. Местоположение пользователя;
   1.2. Личные типичные маршруты;
   1.3. Частопосещаемые места (потенциальные задачи).
2. Реализовать методику анализа передвижения пользователя способную:
   2.1. Определить тип связи между маршрутами (Близкие, соприкасающиеся, пересекающиеся, различные);
   2.2. Сократить избыточную(повторяющуюся) информацию о маршрутах;
   2.3. Уточнить типичные маршруты пользователя по средствам усреднений значений при множественных замерах.
3. Реализовать методику прогнозирования движения пользователя на основе имеющихся индивидуальных замеров, способную:
   3.1. На основании текущего местоположения пользователя рассчитать точки доступные за один шаг.
   3.2. При помощи данных о частоте посещения ближайших точек выбрать наиболее вероятное направление движения.

4. Разработать системы создания, обработки и отслеживания выполнения задач и отображения подсказок при постоянном изменении местонахождения пользователя.

1.2.2 Требования к информационной части

Алгоритмы анализа работают с замерами GPS-координат и датами поэтому для корректной работы алгоритма необходимо унификация способов записи данных, а так же критерии осуществления замеров.

Требования к данным GPS:

1. Соответствие значений измерений вспомогательного приложения, реальным значениям GPS-координат.
2. Осуществление замеров должно происходит в заявленном территориальном округе, т.е в городе Петрозаводске.
3. Использование единого формата записи GPS-координат в алгоритме, и входных данных. Для данной работы был выбран формат состоящий из только из градусов (без минут), но с десятичной дробной частью.
   На Рисунке 3 изображены различные форматы записи GPS-координат.
4. Замеры должны осуществляться с постоянной частотой
5. Записываться в файл согласно хронологии осуществления замеров.

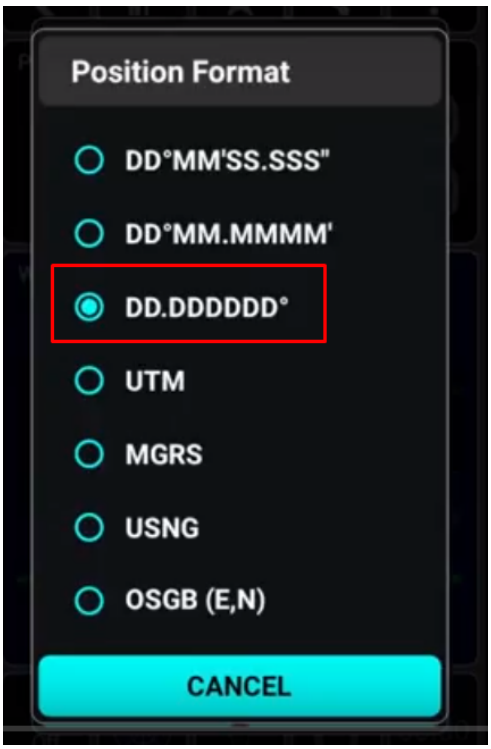

Рисунок 3. Форматы записи GPS-координат. [3]

Требования к данным с датами замеров:

1. Дата проведения замеров должна быть записана в формате ДД.ММ.ГГ, день-месяц-год, где в качестве разделителей между значениями ставятся точки.
2. Дата должна принадлежать Григорианскому календарю.
3. При подсчете даты должно учитываться территориальное расположение пользователя т.е часовой пояс, для заявленного округа это UTC+3.

Требования к оформлению и содержанию файлов:

1. Файл должен содержать в себе данные о передвижении пользователя за ограниченный промежуток времени. (Например один маршрут, ограниченный точками с задачами);
2. Названия файлов должны иметь единообразные названия и различаются на числовой коэффицент, увеличивающийся с добавлением новых файлов.(Например input1.txt input2.txt);

3. Файл должен иметь текстовое расширение ".txt";
4. Все файлы с маршрутами должны хранится в отдельной папке;
5. В самом файле не должно присутствовать данных или символов не описанных в предыдущих пунктах;
6. Замеры по различным точкам маршрута в файле должны быть отделены знаком перевода строки;
7. Значения координат в одной точке в файле должны быть разделены пробелом;

1.3 Перечень и краткое описание разработанных алгоритмов:

1. **Алгоритмы Анализа:** (обозначение - А.А.№ Номер алгоритма)
   1.1. Алгоритм (А.А.№1) поиска типичных маршрутов.
   Алгоритм основанный на интерпритации маршрутов, как ломаных линий, хранящихся в виде списков из взаимосвязанных точек, где для каждой точки хранится информация о соседних точках доступных за один шаг и маршрутах, к которым они принадлежат. Критерии оценивания маршрутов основаны, на типах связи описанных ранее.
   1.2. Алгоритм (А.А.№2) уточнения и сокращения количества типичных маршрутов.
   Алгоритм основан на результатах Алгоритма (А.А.№1), и проводит усреднение значений координат точек из множества близких маршрутов, для получения более точных координат типичных маршрутов. Результат алгоритма - список типичных уточненных маршрутов.
   1.3. Алгоритм (А.А.№3) анализа точек маршрутов среди тестовых маршрутов.

Идея алгоритма основана, на восприятии маршрутов, как отдельных точек, и их связи с другими точками доступными за один шаг. Благодаря информации о частоте повторения точек, можно прогнозировать дальнейший маршрут пользователя.

2. **Алгоритмы Генерации:** (обозначение - А.Г.№ Номер алгоритма) Для тестирования работы Алгоритвом Анализа требуется большое количество разнообразных данных, в которых будут присутствовать замеры передвижений пользователя по городу Петрозаводску. Среди таких данных желательно наличие повторяющихся маршрутов, или наличие часто посещаемых пользоваталем мест, чтобы мы могли проанализировать эти данные по описанным ранее методикам. Но к сожаленью, в силу того что мы не обладаем большой реальной базой передвижения множества пользователей, на длительном промежутке времени, что дало бы нам возможность всесторонне протестировать систему было принято решение разработать ряд алгоритмов, которые смогли бы сгенерировать нам подобные данные, для тестирования данных на начальных этапах разработки системы. Для упрощения тестирования системы анализа, были разработаны 4 алгоритма:

   2.1. Алгоритм (А.Г.№1) создания тестовых близких маршрутов. На вход алгоритму поступает маршрут, заданный списком координат взаимосвязанных точек. Для каждой точки заданного маршрута генерируется отклонение от изначального значения, для близких маршрутов величина отклонения должна быть меньше ширины дороги. Генерация множества маршрутов близких маршрутов осуществляется путем повторения запуска алгоритма необходимого количества раз.

   2.2. Алгоритм (А.Г.№2) создания тестовых непересекающихся маршрутов. Идея алгоритма аналогична (А.Г.№1), главное отличие - отклонения от заданного маршрута обязательно должно превышать ширину дороги.

2.3. Алгоритм (А.Г.№3) создания тестовых соприкасающихся маршрутов. Идея алгоритма аналогична (А.Г.№1), главное отличие, что генерируется отклонение от заданного маршрута(на величину больше ширины дороги) только для части точек маршрута, остальные остаются нетронутыми.

2.4. Алгоритм (А.Г.№4) создания тестовой точки маршрута. На вход алгоритму поступает пара координат с типичными для заданного округа значениями (GPS-координаты города Петрозаводска меняются в пределе 61.78 - 61.82 34.32 - 34.26) после чего генерируется отклонения от типичного значения, в рамках погрешности заданной территорией округа. Полученный результат интерпретируется, как новая точка маршрута пользователя. На рисунке 4 можно увидеть пример типичных координат города Петрозаводска.

Более подробно реализованные алгоритмы описаны в глава Алгоритмы Анализа и Алгоритмы Генерации, там будет рассмотрена работа алгоритмов на примерах, способы реализации алгоритмов на языке программирования C++, а так же пошаговый разбор работы проводимой алгоритмами.

1.4 Архитектура

На рисунке 5 ~~вы можете~~ увидеть архитектуру системы, которая делится на 4 основных блока:

1. Основные функции;
2. Генерация марщрутов;
3. Анализ типов маршрутов;
4. Прогнозирование.

Каждая из этих систем определенным образом связана друг с другом, и в свою очередь отвечает за ряд возложенных на нее функция. Например подсистема основных функций содержит в себе методы ввода

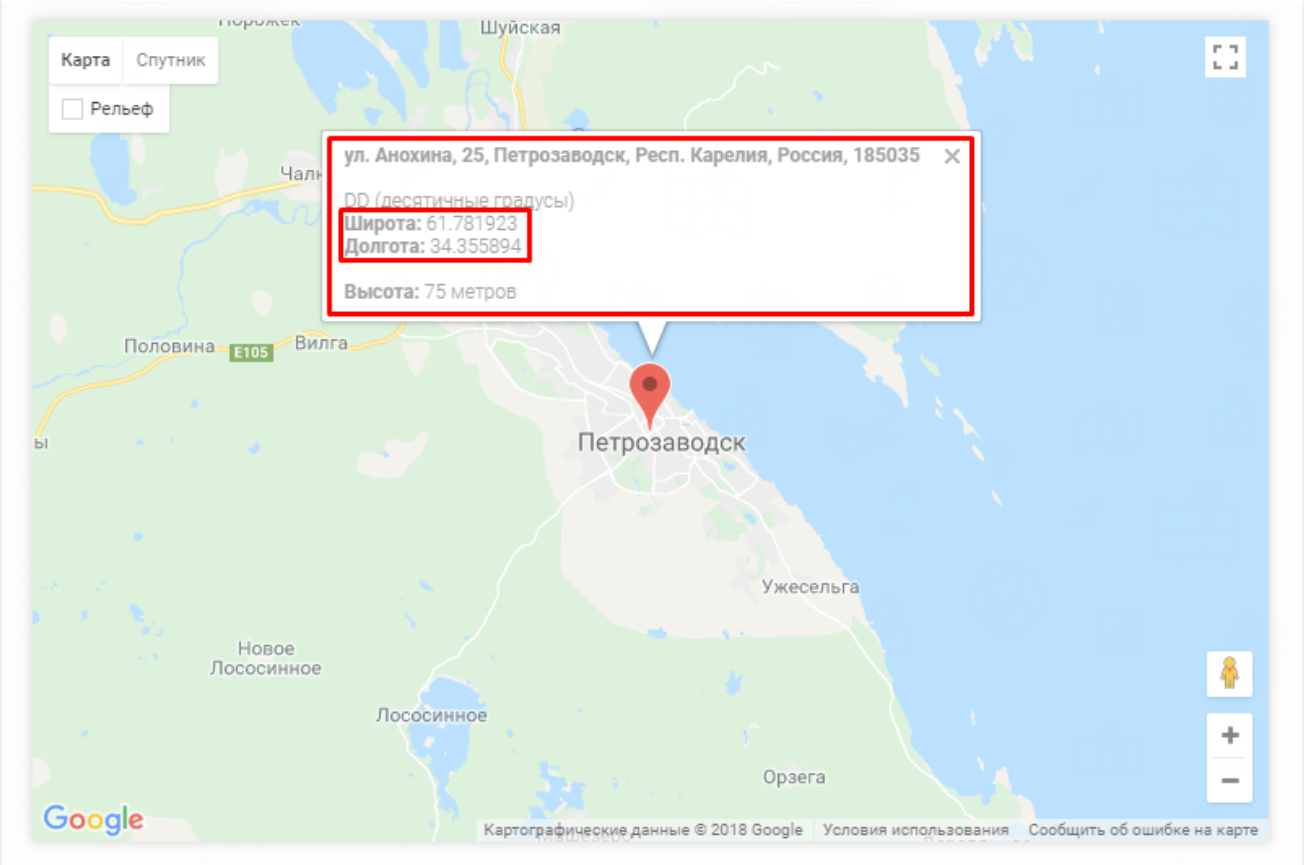

Рисунок 4.Типичные координаты города Петрозаводска. [2]

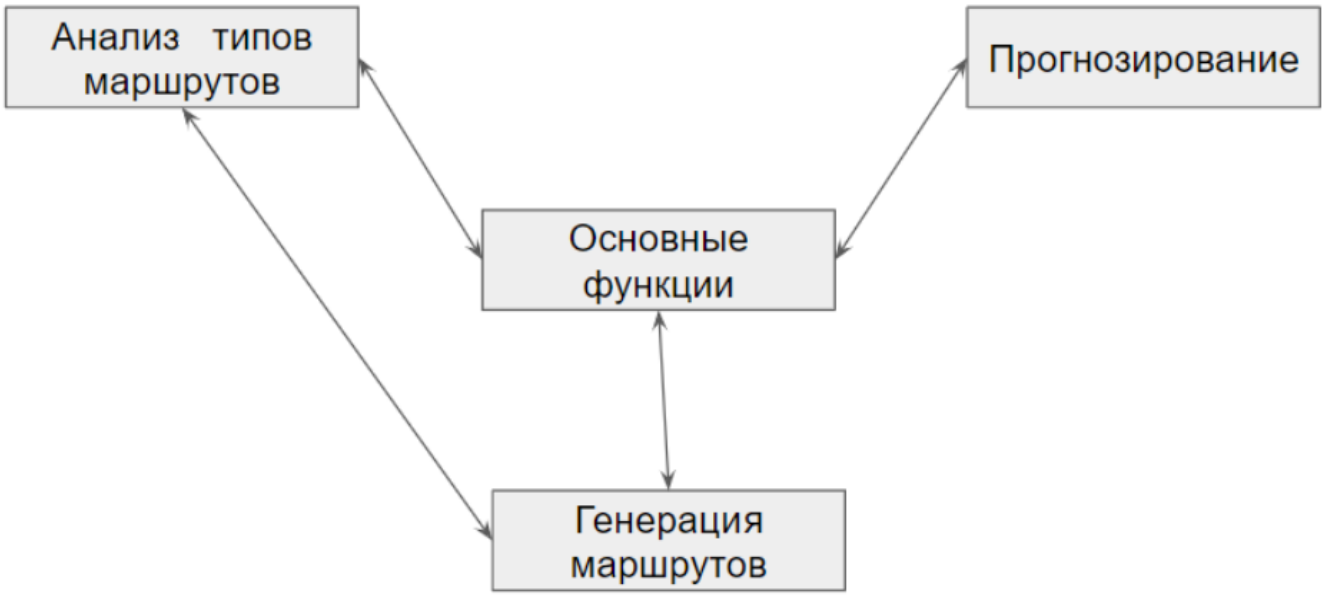

Рисунок 5.Взаимосвязи подсистем.

и вывода маршрутов в файл, сортировки координат маршрута и упрощения маршрута. Подсистема анализа типов маршрута осуществляет проверку маршрутов с целью по описанным критериям выбрать тип связи маршрутов: близкие, различные, соприкасающиеся, пересека-

ющиеся. Подсистема прогнозирования осуществляет расчет посещаемости координат и создает прогноз о более вероятном направлении передвижения пользователя.

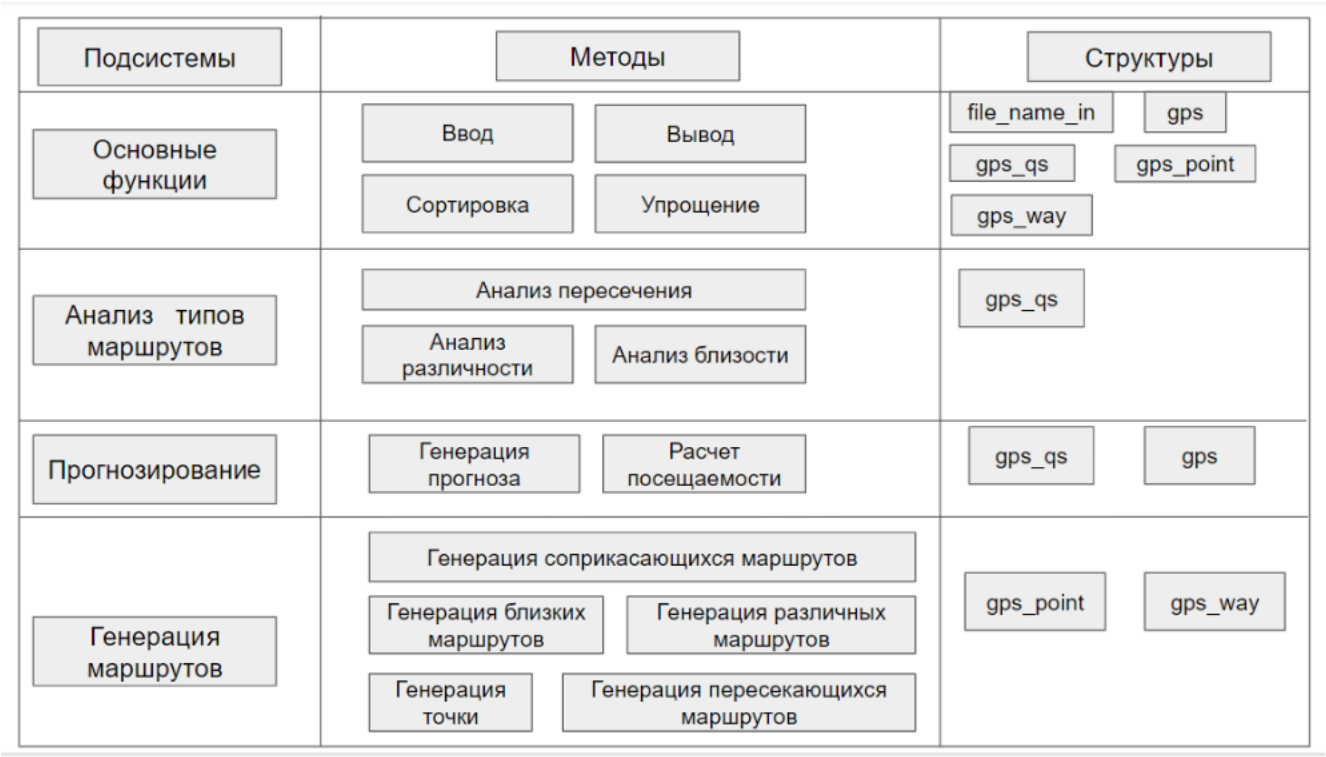

Рисунок 6.Методы и структуры подсистем.

На рисунке 6 можно более наглядно ознакомится с методами используемыми в рамках подсистем, а так же отметить с какими типами стуктур работает каждая подсистема. В качестве дополнения к рисунку ниже представлено детальное описание работы каждого метода и структур данных с которыми ведется работа.

1. Структуры

   1.1. Структура file_name_in

   Структура представляет собой строку формата: название директории_порядковый номер.txt

   1.2. Структура gps

   Представляет собой массив gps[4][n], где первый столбец - координата x, второй столбец координата y, третий столбец - порядковый номер в маршруте, четвертый - номер маршрута(файла из которого взят маршрут).

1.3. Структура gps_qs

Представляет собой массив gps[6][n], где первый столбец - координата x, второй столбец координата y, третий столбец - порядковый номер в маршруте, четвертый - номер маршрута(файла из которого взят маршрут) пятый - код, характеризующий тип связи маршрутов, шестой - частота помещения координат.

1.4. Структура gps_point

Структура представляет собой пару переменных типа float содержащие по отдельности координату x и y.

1.5. Структура gps_way

Представляет собой массив gps_way[2][n], где первый столбец - координата x, второй столбец координата y.

2. Методы

2.1. Метод Ввод

Данный метод последовательно открывает все файлы маршрутов в указанной директории структуры file_name_in и переносит данные в структуру gps.

2.2. Метод Вывод

Данный метод содержит два варианта вывода данных:

i. Генерирует последовательные названия для файлов и загружает в них данные маршрутов, для структур типа gps_way gps_point.

ii. Вывод в консоль для структур типа gps, gps_qs.

2.3. Метод Сортировки

Метод содержит в себе способы сортировки координат маршрутов для структур типа gps и выводит результат в структуру типа gps_qs

2.4. Метод Упрощения

Метод содержит в себе способы обобщения координат маршрута для структур типа gps и gps_qs.

2.5. Метод Анализа пересечения

Метод содержит в себе 2 методики анализа пересечения для структур типа gps_qs:

i. Анализа пересечения.

ii. Анализа соприкосновения.

2.6. Метод Анализа близости

Метод содержит в себе методику анализа близости для структур типа gps_qs.

2.7. Метод Анализа различности

Метод содержит в себе методику анализа различности для структур типа gps_qs.

2.8. Метод Расчета посещаемости

Метод позволяет на основе структуры gps_qs рассчитать частоту посещаемости для координат маршрутов.

2.9. Метод Генерации прогноза

Метод на основе структуры gps_qs и gps_point строит прогноз направления движения пользователя.

2.10. Метод Генерации соприкасающихся

Метод позволяет сгенерировать маршруты типа соприкасающиеся в структуре типа gps_way.

2.11. Метод Генерации близких

Метод позволяет сгенерировать маршруты типа близкие в структуре типа gps_way.

2.12. Метод Генерации различных

Метод позволяет сгенерировать маршруты типа различные в структуре типа gps_way.

2.13. Метод Генерации пересекающихся

Метод позволяет сгенерировать маршруты типа пересекающиеся в структуре типа gps_way.

2.14. Метод Генерации точки

Метод позволяет сгенерировать точку структуры типа

gps_point

Глава 2

Алгоритмы Анализа

2.1 Алгоритм (А.А.№1) поиска типичных маршрутов

Данный алгоритм создает структуры для хранения значений координат маршрутов, анализирует схожесть отдельных маршрутов пользователя, выводит данные о близости и, на основе этого, типичности маршрутов.

2.1. Попарно открыть все файлы с замерами передвижения пользователя;

Предполагается, что алгоритм начинает свою работу при наличии двух или более файлов описывающих передвижение пользователя, ранее начать работу считаем не возможным из-за малого количества информации для анализа.

2.2. Параллельно считать значения GPS-координат из файлов, и занести в структуру данных, представляющую собой двухмерный массив на пять столбцов и количеством строк соответствующим количеству замеров в файле, в которых хранится:

i. Координата x;

ii. Координата y;

iii. Порядковый номер в файле, из которой были считаны данные;

iv. Название файла;

v. Поле, хранящее отладочную информацию о состоянии обработки маршрута.

Данная структура помогает осуществлять контроль расстояний между точками маршрутов, при этом сохраняя всю необходимую информацию об изначальных маршрутах, На рисунке 5 представлена реализация метода ввода данных в описанную структуру на языке программирования C. описанную в предыдущих пунктах.

```
for(j; j < m; j++)
{
    scanf("%f", &mass_cif[j][0]);
    scanf("%f", &mass_cif[j][1]);

    double_mass[j][0]=mass_cif[j][0];
    double_mass[j][1]=mass_cif[j][1];
    double_mass[j][2]= i;
    double_mass[j][3] = jj;
    double_mass[j][4] = 0;

    mass_cif[j][2]= i;
    mass_cif[j][3] = jj;
    mass_cif[j][4] = 0;

    jj++;
}
```

Риснок 5. Реализация: ввод массивов, где mass_cif - массив ввода первого файла, double_mass - общий массив в который будут добавлены все файлы, jj - номер файла, j - номер в файле.

2.3. Отсортировать полученный массив по координате x, для формирования структуры состоящей из близко расположенных (из-за сортировки) значений координат.

2.4. Поиск близких точек:

i. Взять наименьшую по x координате точку;

ii. Сравнить ее со следующим элементом упорядоченного массива;

iii. Считать точки близкими, если разность величин будет меньше или равна заданной погрешности ширины доро-

ги по обеим координатам, т.е $\forall$ m1 $\in$ S $\exists$ m2 $\in$ S такая что ρ(m1,m2) $\leqq Exp_2$, где под S подразумеваем множество всех маршрутов;

iv. Искать близкие точки среди оставшихся элементов массива, если не выполнено предыдущее условие.

2.5. Если точки являются близкими по обеим координатам, запустить пункт 4 для точек которые имеют:

i. Такой же номер файла;

ii. Следующий порядковый номер в файле.

На рисунке 6 вы можете увидеть реализацию метода анализа близости точек.

```
int obxod(int start, int kolvo)
{

    int  j = kolvo;
    if( mass_cif[start][4]==PUSTO)
    {
        for( int k=0; k<j; k++)
        {
           if(mass_cif[start][2] != upmass_cif[k][2])
            {
                if(((mass_cif[start][0] - upmass_cif[start][0])<=ROAD)&&((mass_cif[start][0] - upmass_cif[start][0])>=-ROAD))
                {
                    mass_cif[start][4]=1; //где 1 - одинаковые значения
                    upmass_cif[start][4]=1;
                    obxod(start+1,j);
                }
                else
                {
                    mass_cif[start][4]=0;
                    upmass_cif[start][4]=0;
                    obxod(start+1,j);
                 }


            }
        }
    }
    return 0;
}
```

Рисунок 6. Реализация поиска и отметки близких точек.

2.6. Аналогично 5 выполнить сравнение по описанным критериям предыдущих значений массива предположительно типичных между собой путей;

2.7. Если все точки маршрута оказались близкими друг для друга, считать этот путь типичным, провести уточнение маршрута, средствами усреднения данных по общим точкам с занесением полученного нового маршрута, в дополнительный массив хранящий текущие типичные маршруты (пути, что были усреднены хотя бы 1 раз); На рисунке 7 можно увидеть реализацию метода упрощения, целью которого является усреднить значения координат маршрутов, значение которых оказались близкими.

```
if (flag == 1)// где flag означает, что маршруты близкие
{

    for(l=0; l<j; l++)
    {
    end_mass[l][0]= ( upmass_cif[l][0] + mass_cif[l][0] ) / 2;
    end_mass[l][1]= ( upmass_cif[l][1] + mass_cif[l][1] ) / 2;
    end_mass[l][2]= upmass_cif[l][2];
    end_mass[l][3]= upmass_cif[l][3];
    end_mass[l][4]= 20;//где 20 означает, что поле пересчитано

    }
```

Рисунок 7. Данная часть программы обобщает маршруты в единый путь если они были близки по большинству точек.

2.8. Если близкая точка лишь одна, считать эту точку пересечением путей;

2.9. Если близкие точки отсутствуют, закончить поиск и закрыть все файлы.

2.2 Пример работы алгоритма А.А.№1

2.2.1 Входные данные

Для тестирования и оценки точности работы алгоритма А.А.№1 необходимы правдоподобные или реальные входные данные. При содействии аналогичных дипломных проектов (Прохоров Илья, Анализ проблем адаптации алгоритмов упрощения ломаных на мобильных устройствах) был получен GPS(ГЛОНАСС) - маршрут замеренный с интервалами в 5 минут во время передвижения пользователя по территории города Петрозаводска. Формат входных данных включает в себя время и дату совершенного замера, а так же заголовки Широта и Долгота (x и y) необходимые для поиска типичных маршрутов величины GPS(ГЛОНАСС)-координат.

```
2:42 PM, дек. 15, '16 Широта: 61.7820635  Долгота: 34.3090038
2:47 PM, дек. 15, '16 Широта: 61.7820635  Долгота: 34.3090038
2:52 PM, дек. 15, '16 Широта: 61.78149  Долгота: 34.3163508
2:57 PM, дек. 15, '16 Широта: 61.78149  Долгота: 34.3163508
3:02 PM, дек. 15, '16 Широта: 61.78149  Долгота: 34.3163508
3:07 PM, дек. 15, '16 Широта: 61.78149  Долгота: 34.3163508
3:12 PM, дек. 15, '16 Широта: 61.78149  Долгота: 34.3163508
```

Рисунок 8. Часть полученного первичного GPS(ГЛОНАСС)-маршрута.

Входной файл содержал 58 строк-измерений, часть которых имела подряд идущие повторяющиеся значения, что являлось избыточной информацией для описания пути. Необходимо было обработать имеющийся маршрут алгоритмом упрощения, для удаления излишних элементов маршрута. За основу алгоритма упрощения планировалось взять алгоритм упрощения кривой Алгоритм Рамера — Дугласа — Пекера[7][8] , но точность этого ал-

горитма оказалось избыточной для наших данных(Алгоритм сохранял те данные которые, в рамках нашей точности не требовались, считались лишними) и в силу чего, требовалась доработка алгоритма. Было решено использовать алгоритм построчной обработки и сравнения входных данных на повторяющиеся строки. Повторяющимися строками назывались такие пары значений x (Широта) и y (Долгота) которые совпадали с аналогичными значениями предыдущей или следующей строк с погрешностью равной ширине дороги. После обработки адаптированным алгоритмом упрощения и удаления не востребованных описательных характеристик замеров (согласно пункту Требования к файлам), исходный файл имел лишь 19 значимых замеров и хранил их в формате x(Широта) y(Долгота).

```
1  61.7825857 34.30827
2  61.7820635 34.3090038
3  61.78149 34.3163508
4  61.7828846 34.3088731
5  61.78149 34.3163508
6  61.7864136 34.3536091
7  61.78149 34.3163508
8  61.7864 34.3534732
9  61.78149 34.3163508
10 61.7864407 34.3538809
11 61.78149 34.3163508
12 61.7864407 34.3538809
13 61.7863695 34.353764
14 61.7864136 34.3536091
15 61.7865763 34.3528537
16 61.7853949 34.3481889
17 61.7825417 34.3084249
18 61.7825993 34.3084059
19 61.7825417 34.3084249
```

Рисунок 9. Упрощенный исходный маршрут.

За отсутствием большого количества экспериментальных GPS(ГЛОНАСС) - измерений было решено искусственно сгенерировать аналогичные дополнительные маршруты различных типов: близкие и пересекающиеся пути. Отсутствие в тестировании различных путей обуславливалось тем, что они не интересны

для тестирования поиска типичных маршрутов и будут отброшены на первом же этапе, а так же, изучением экспериментальных замеров, которые утверждали наличие хотя бы одной пары близких точек, практически в каждом замеренном пути в связи с небольшой площадью местности. Для генерации маршрутов использовались алгоритмы тестовой генерации типичных маршрутов и тестовой генерации пересекающихся маршрутов, разработанные для тестирования методов упрощения кривых.

Для простоты рассмотрения работы алгоритма на примере для каждого входного файла дано название поясняющее тип взаимосвязи с другими файлами.

- **Маршрут №1** - исходный, полученный после упрощения и приведения к виду описанному в Требованиях к файлам виду, маршрут.
- **Типичный Маршрут №Т-1, Типичный Маршрут №Т-2** - маршруты полученные при использовании алгоритма генерации типичных маршрутов А.Г.№1, на основе Маршрута №1. Ожидается, что в ходе работы алгоритм сможет обнаружить, что маршруты: Маршрут №1, Типичный Маршрут №Т-1 и Типичный Маршрут №Т-2, являются близкими между собой, благодаря чему, можно будет назвать маршруты типичными для пользователя.
- **Пересекающийся Маршрут №2** - маршрут сгенерированный при помощи алгоритма генерации пересекающихся маршрутов А.Г.№2, на основе Маршрута №1. Ожидается, что в ходе работы алгоритм сможет обнаружить, что маршруты: Маршрут №1 и Пересекающийся Маршрут №2, являются пересекающимися в одной точке.
- **Типичный Маршрут №П-1, Типичный Маршрут №П-2** - аналогично Типичный Маршрут №Т-1, сгенерированный маршрут на основе Пересекающийся Маршрут №2. Ожидается, что алгоритм сможет обнаружить взаимосвязь маршру-

тов: Типичный Маршрут №П-1, Типичный Маршрут №П-2 и Пересекающийся Маршрут №2, посчитав их близкими.

Точные значения координат всех описанных маршрутов указаны на Рисунке 10.

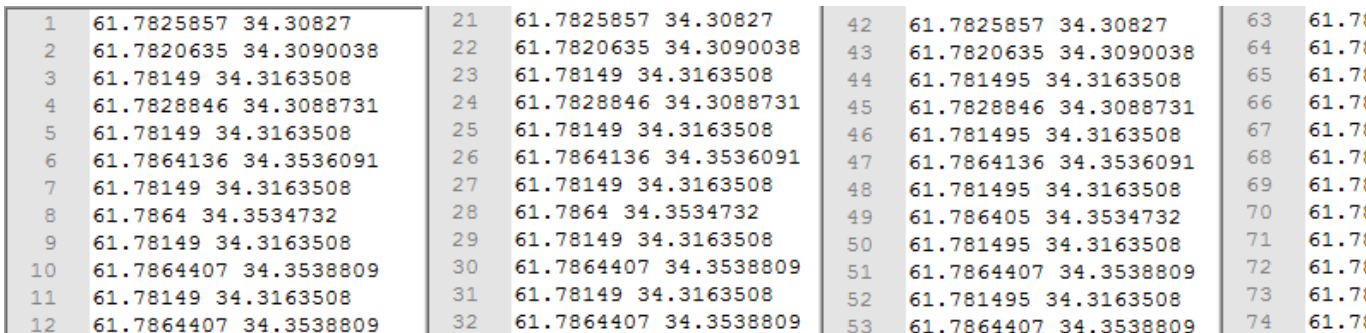

Рисунок 10. (Слева на право) Исходный упрощенный Маршрут №1, Типичный Маршрут №Т-1, Типичный Маршрут №Т-2, Пересекающийся Маршрут №2, Типичный Маршрут №П-1, Типичный Маршрут №П-2.

2.2.2 Работа Алгоритма А.А.№1 на примере

В данном случае для наглядной демонстрации работы алгоритма, было решено использовать следующую систему хранения входных фалов, где в файлах ik1.txt-ik5.txt хранятся маршруты Маршрут №1, Типичный Маршрут №Т-1, Типичный Маршрут №Т-2, Пересекающийся Маршрут №2, Типичный Маршрут №П-1, Типичный Маршрут №П-2 соответственно.

После обработки алгоритмом первичных входных данных, первыми двумя шагами первые два столбца каждого файла - GPS(ГЛОНАСС)-координаты, далее номер входного файла (соответствующий цифре указанной в названии), далее номер строки в которой были указаны пара координат. После поиска в маршрутах близких точек и соответственно маршрутов, можно увидеть следующий результат, где к ранее описанным структурам полей, было добавлено еще одно, описывающее близость

точки, с соответствующей точкой сравниваемого маршрута.(1 - близкие точки, 0 - отсутствие близости координат),(Рисунок 12) после чего на основе подсчета количества близких точек в маршрутах, дается характеристика маршрутов: Близкие, Пересекающиеся, Различные (Рисунок 13).

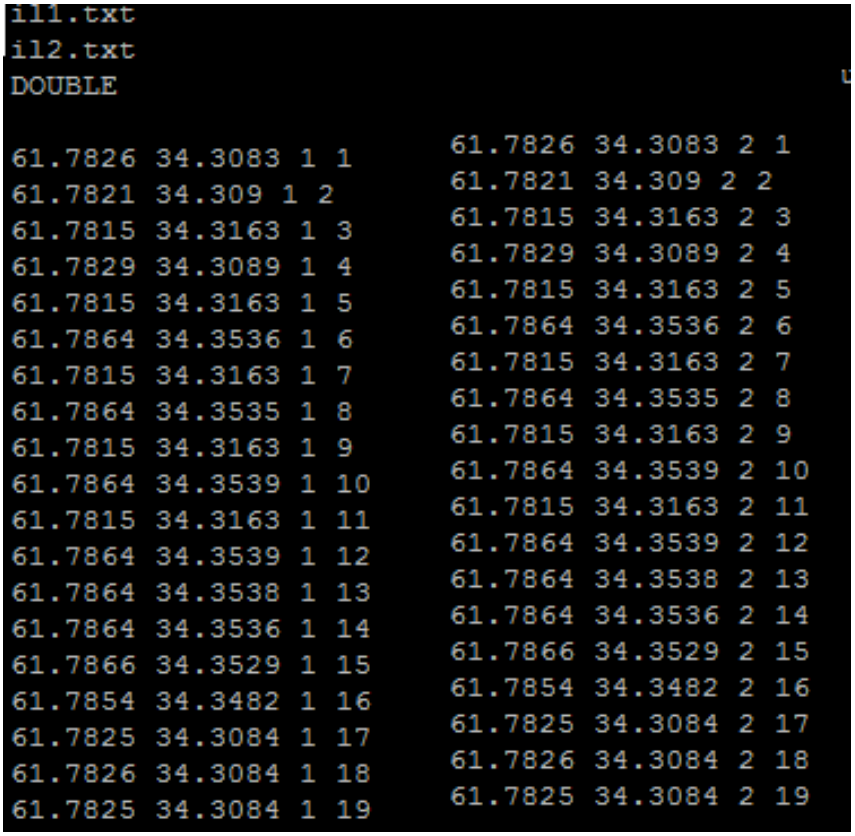

Рисунок 12. Координаты маршрутов после добавления требуемых полей.

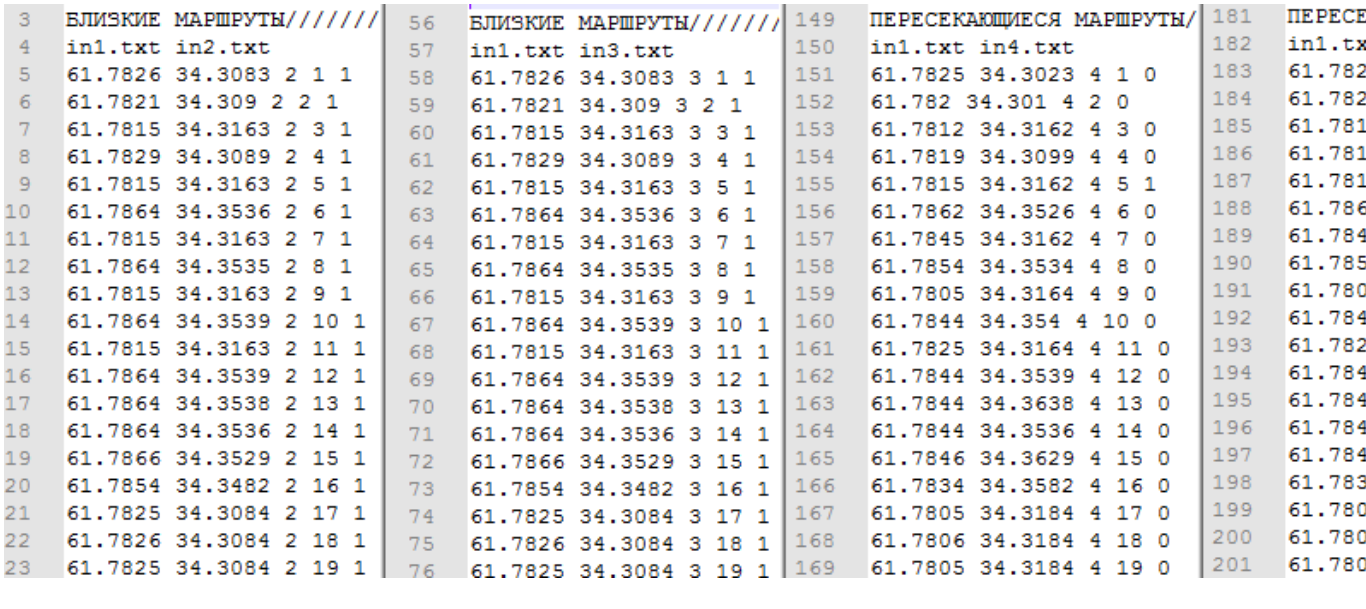

Рисунок 13. Результат работы программы на этапе сравнивания.

2.3 Алгоритм (А.А.№2) уточнения и сокращения количества типичных маршрутов.

Алгоритм работает с входным массивом данных состоящим из координат близких маршрутов, с отметками о типе взаимосвязи между каждым из них. При этом массив отсортирован следующим образом: если ранее были найдены взаимосвязанные маршруты (близкие или пересекающиеся), то в массив они будут внесены последовательно друг за другом, что ускорит работу алгоритма. Из-за наличия возможности появления погрешности и минимальных изменений в измерениях GPS-координат, которыми пользуется алгоритм, на основе множества близких маршрутов формируется единственный типичный маршрут, значения которого, являются усреднением по каждой из координат всех близких между собой маршрутов. Таким образом сформированный маршрут, считается типичным для пользователя, так как периодически повторялся несколько раз.

2.1. Ввод данных о передвижении пользователя в структуру аналогичную описанной в "Алгоритме поиска типичных маршрутов среди тестовых путей"пункте 2, с добавлением дополнительного столбца, в котором будут хранится отладочные значения по усреднению данных.

2.2. Если первые два маршрута в массиве близкие то:

   i. Из полученного массива попарно берутся значения принадлежащие двум различным маршрутам, их значения суммируются и делятся пополам по обеим координатам, осуществляя усреднение значений

   ii. По окончании усреднения значений двух маршрутов, ме-

сто в массиве, которое занимал первый маршрут заменяется усредненным значением, а все оставшиеся не усредненные значения других маршрутов смещаются, удаляя второй из маршрутов участвующий в получении усредненных данных.

iii. Далее пункт №2 повторяется для следующих первых двух указанных в массиве маршрутов, один из которых уже усреднен, а второй еще не был задействован.

2.3. Если маршруты не близкие, то перейти к сравнению следующей пары.

2.4 Пример работы алгоритма А.А.№2

Пример работы алгоритма А.А.№2 показан, на данных полученных в результате работа алгоритма А.А.№1 подробно рассмотренных в примере работы алгоритма А.А.№1

На Рисунке 15 можно отметить, что после удачного усреднения значений, система выводит все повторившиеся более 1-го раза(и соответственно уточненные) значение теперь уже считающихся Типичными маршрутов.

Как и ожидалось, работа алгоритма завершилась выводом двух усредненных маршрутов, которые были получены из первоначального набора различных маршрутов по средствам приведенного алгоритма. Промежуточные данные о близости ряда точек маршрутов не участвуют в итоговом выводе программы, так как не являются важными для отбора типичных маршрутов.

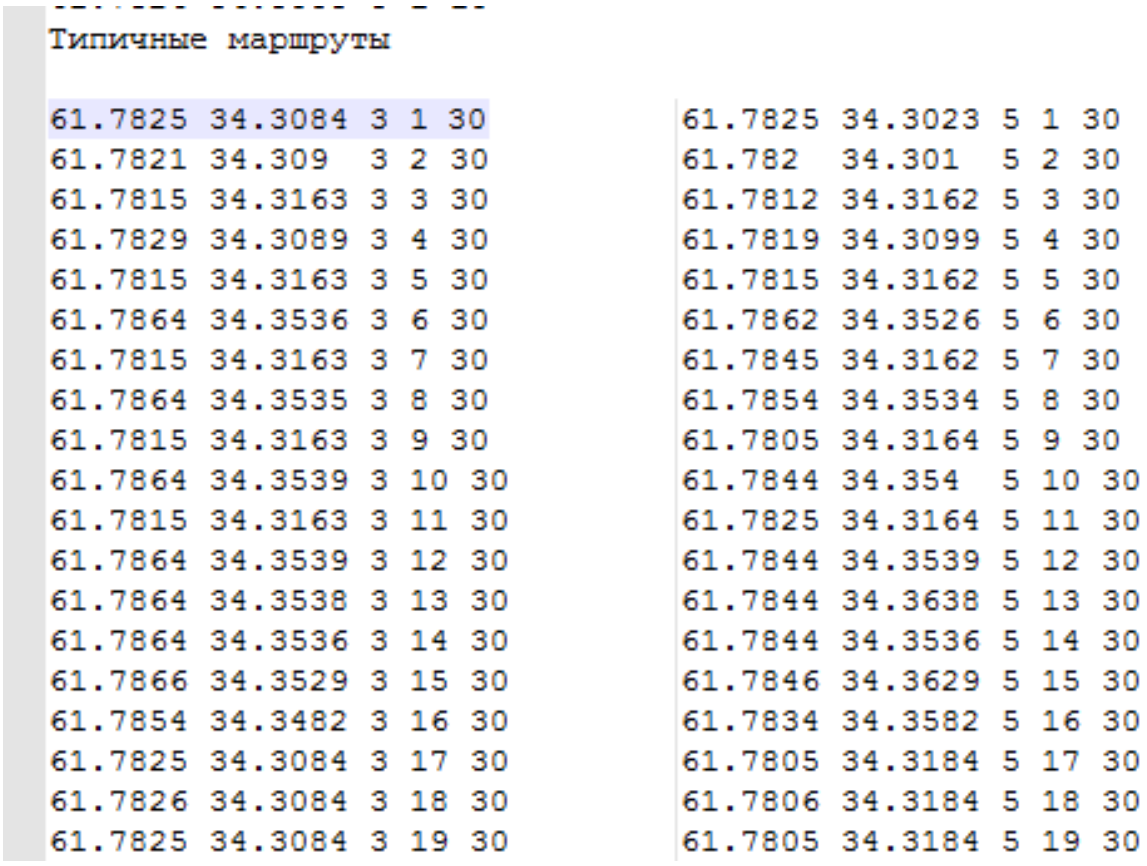

Рисунок 15. Итоговый вывод программы, нашедшей уточненные маршруты.

2.5 Алгоритм (А.А.№3) анализа точек маршрутов среди тестовых путей

Данный алгоритм анализирует каждую точку маршрута, как отдельный элемент, на основе собранной статистики ведет учет частоты посещения отдельных точек на карте и близости их расположения. Алгоритм дает возможность протестировать гипотезу о предсказании маршрута пользователя, исходя из координат положения пользователя, и таблицы частот посещения точек в которые пользователь может переместится за один шаг из заданной точки.

2.1. Ввод данных о передвижении пользователя в структуру аналогичную описанной в "Алгоритме поиска типичных маршрутов среди тестовых путей"пункте 2, с добавлением дополнительного столбца, в котором будут хранится данные о ча-

стоте посещения данной точки.

2.2. Поиск близких точек:

i. Упорядочить массив данных по координате X.

ii. Взять наименьшую по x координате точку;

iii. Сравненить ее со следующим элементом упорядоченного массива;

iv. Считать точки близкими, если разность величин будет меньше или равна Exp_2 по обеим координатам. T.e $\forall$ m1 $\in$ M1 $\exists$ m2 $\in$ M2 такая что ρ(m1,m2) $\leqq$ Exp_2, где под M1,M2 подразумеваются сравнивыемые на близость маршруты.

v. Если точки близкие:

A. Увеличить значение в столбце частота для первой из близких точек на единицу.

B. Удалить вторую из близких точек.

vi. Если точки не являются близкими, то перейти к анализу следующей точки.

2.3. Ввод (генерация при помощи алгоритма А.Г.№4) координат X и Y "текущего места положения пользователя и добавление этой точки в структуру данных пункта 1.

2.4. Поиск следующей точки маршрута:

i. Провести Поиск близких точек для сгенерированных значений, аналогично пункту 2.

ii. Сравнить частоты посещения точек доступных за один шаг, таких что $\forall$ m1 $\in$ M1 $\exists$ m2 $\in$ M1, такая что ρ(m1,m2) $\leqq$ $Exp_2 \cdot 2, , , ,$ Exp$_2$ будут объеденяться в одну благодаря методам упрощения.

iii. Считать точку с большей частотой посещение точкой прогноза.

2.6 Пример работы алгоритма А.А.№3

2.6.1 Входные данные

Для корректной работы алгоритма необходимо соответствие входных данных требованиям указанным в Требованиях к GPS-координатам, и подробно рассмотренным в пункте Пример работы алгоритма А.А.№1, входные данные.

На Рисунке 16 изображен один из тестовых маршрутов занесенный в массив структура которого была описана ранее.

ill.txt
mass

61.782585	34.308270	1.000000	1.000000	10.000000	1.000000
61.782063	34.309002	1.000000	2.000000	10.000000	1.000000
61.781490	34.316349	1.000000	3.000000	10.000000	1.000000
61.782887	34.308872	1.000000	4.000000	10.000000	1.000000
61.781490	34.316349	1.000000	5.000000	10.000000	1.000000
61.786415	34.353611	1.000000	6.000000	10.000000	1.000000
61.781490	34.316349	1.000000	7.000000	10.000000	1.000000
61.786400	34.353474	1.000000	8.000000	10.000000	1.000000
61.781490	34.316349	1.000000	9.000000	10.000000	1.000000
61.786442	34.353882	1.000000	10.000000	10.000000	1.000000
61.781490	34.316349	1.000000	11.000000	10.000000	1.000000
61.786442	34.353882	1.000000	12.000000	10.000000	1.000000
61.786369	34.353764	1.000000	13.000000	10.000000	1.000000
61.786415	34.353611	1.000000	14.000000	10.000000	1.000000
61.786575	34.352852	1.000000	15.000000	10.000000	1.000000
61.785397	34.348190	1.000000	16.000000	10.000000	1.000000
61.782543	34.308426	1.000000	17.000000	10.000000	1.000000
61.782600	34.308407	1.000000	18.000000	10.000000	1.000000
61.782543	34.308426	1.000000	19.000000	10.000000	1.000000

Рисунок 16.

Для дальнейшей работы, с собранными из файлов данными по маршрутам, все массивы объединены в один общий массив, данные которого отсортированы по координате x. (Рисунок 17)

Легко заметить, что после сортировки подряд сгруппировались повторяющиеся (типичные) значения точек.(Рисунок 18)

Дублирующиеся значения координат точек удаляются из массива, но поле с частотой появления точки в массиве увеличивается на единицу.(Рисунок 19)

```
QSORT
size_q 76
61.786575 34.352852 1.000000 15.000000 10.000000 1.000000
61.786575 34.352852 2.000000 15.000000 10.000000 1.000000
61.786575 34.352852 3.000000 15.000000 10.000000 1.000000
61.786575 34.352852 4.000000 15.000000 10.000000 1.000000
61.786442 34.353882 1.000000 12.000000 10.000000 1.000000
61.786442 34.353882 2.000000 12.000000 10.000000 1.000000
61.786442 34.353882 3.000000 12.000000 10.000000 1.000000
61.786442 34.353882 4.000000 12.000000 10.000000 1.000000
61.786442 34.353882 1.000000 10.000000 10.000000 1.000000
61.786442 34.353882 2.000000 10.000000 10.000000 1.000000
61.786442 34.353882 3.000000 10.000000 10.000000 1.000000
61.786442 34.353882 4.000000 10.000000 10.000000 1.000000
61.786415 34.353611 1.000000 14.000000 10.000000 1.000000
61.786415 34.353611 2.000000 14.000000 10.000000 1.000000
61.786415 34.353611 3.000000 14.000000 10.000000 1.000000
61.786415 34.353611 4.000000 14.000000 10.000000 1.000000
61.786415 34.353611 1.000000 6.000000  10.000000 1.000000
61.786415 34.353611 2.000000 6.000000  10.000000 1.000000
61.786415 34.353611 3.000000 6.000000  10.000000 1.000000
61.786415 34.353611 4.000000 6.000000  10.000000 1.000000
61.786400 34.353474 1.000000 8.000000  10.000000 1.000000
```

Рисунок 17.

```
size_d 16
61.786575 34.352852 1.000000 15.000000 10.000000 4.000000
61.786442 34.353882 1.000000 12.000000 10.000000 8.000000
61.786415 34.353611 1.000000 14.000000 10.000000 8.000000
61.786400 34.353474 1.000000 8.000000  10.000000 4.000000
61.786369 34.353764 1.000000 13.000000 10.000000 4.000000
61.785397 34.348190 1.000000 16.000000 10.000000 4.000000
61.782887 34.308872 1.000000 4.000000  10.000000 4.000000
61.782600 34.308407 1.000000 18.000000 10.000000 4.000000
61.782585 34.308270 1.000000 1.000000  10.000000 4.000000
61.782543 34.308426 1.000000 19.000000 10.000000 8.000000
61.782063 34.309002 1.000000 2.000000  10.000000 4.000000
61.781490 34.316349 1.000000 9.000000  10.000000 20.000000
```

Рисунок. 18.

```
new_x = 61.784409
new_y = 34.302238
mass
 61.784409  34.302238  0.000000 0.000000  10.000000  1.000000
 QSORT
```

Рисунок 19.

Далее представлены значения новой сгенерированной точки, которая далее также будет внесена в общий массив.(Рисунок 20)

```
 QSORT
size_q 17
61.786575 34.352852 1.000000 15.000000 10.000000 4.000000
61.786442 34.353882 1.000000 12.000000 10.000000 8.000000
61.786415 34.353611 1.000000 14.000000 10.000000 8.000000
61.786400 34.353474 1.000000 8.000000  10.000000 4.000000
61.786369 34.353764 1.000000 13.000000 10.000000 4.000000
61.785397 34.348190 1.000000 16.000000 10.000000 4.000000
61.784409 34.302238 0.000000 0.000000  10.000000 1.000000
61.782887 34.308872 1.000000 4.000000  10.000000 4.000000
61.782600 34.308407 1.000000 18.000000 10.000000 4.000000
61.782585 34.308270 1.000000 1.000000  10.000000 4.000000
61.782543 34.308426 1.000000 19.000000 10.000000 8.000000
61.782063 34.309002 1.000000 2.000000  10.000000 4.000000
61.781490 34.316349 1.000000 9.000000  10.000000 20.000000
```

Рисунок 20.

После внесения новых значений, массив повторно сортируется, для наглядного определения доступных за один шаг точек, из заданной. На основе сравнения частот посещения пользователем точек, выбирается более посещаемая, на основе чего строится предположение о направлении движения пользователя.(Рисунок 21)

```
Для точки: 61.784409 34.302238
Точка прогноза: 61.785397 34.348190
```

Рисунок 21.

Глава 3

Алгоритмы Генерации

3.1 Алгоритм (№1) создания тестовых близких путей

2.1. В структуру данных описанную в "Алгоритме поиска типичных маршрутов среди тестовых путей"пункте 2 внесем экспериментальный упрощенный маршрут, полученный при помощи измерений GPS - координат при передвижении по городу Петрозаводску;

2.2. Для каждого и сгенерируем случайное отклонение от изначального маршрута, при этом отклонение не должно превышать заданную ширину "дороги";

2.3. Пункт №2 повторяется с исходным маршрутом до получения нужного количества тестовых близких маршрутов.

Создание нового множества типичных маршрутов подразумевает под собой запуск данного алгоритма с новыми значениями координат исходного упрощенного маршрута.

```
#define bound_rand_x() ( fabs(sin(rand())) * 0.01)// функция генерации случайного числа в формате
#define bound_rand_y() ( fabs(sin(rand())) * 0.01)
    for(int ii = 0; ii < MAX; ++ii)
    {
        new[ii][0] = old[ii][0] + bound_rand_x()
        new[ii][1] = old[ii][1] + bound_rand_y()
        new[ii][2] = old[ii][2]
        new[ii][3] = old[ii][3]
        new[ii][4] = old[ii][4]
        new[ii][5] = old[ii][5]
    }
```

Рисунок 22. Пример реализации алгоритма генерации псевдослучайного дробного числа.

3.2 Алгоритм (№2) создания тестовых непересекающихся маршрутов

2.1. При помощи генератора случайных чисел, в рамках типичный координат города Петрозаводска генерируется случайная точка, далее относительноо этой точки выбирается следующая, так чтобы они не оказались друг другу близкими, таким образом и создается упрощенный маршрут пользователя.

2.2. Пункт №1 повторяется до получения требуемого количества непересекающихся маршрутов, при этом каждая новая сгенерированная точка добавляется в маршрут, только при условии, что она не является близкой для любых других точек из всех ранее сгенерированных маршрутов.

3.3 Алгоритм (№3) создания тестовых соприкасающихся маршрутов

2.1. Аналогично Алгоритму №1 пункт №1 на вход алгоритму требуется маршрут, соприкасающиеся маршруты к которому мы хотим построить.

2.2. В отличае от алгоритма генерации близких маршрутов здесь для и генерируется случайное отклонение от изначального маршрута, при этом отклонение превышает заданную ширину "дороги чем и иметируется расхождение от изначального маршрута.;

2.3. Пункт №2 повторяется для созданий нужного количества отклонений от типичного маршрута.

2.4. Пункт №2 повторяется с исходным маршрутом до получения нужного количества тестовых соприкасающихся маршрутов.

3.4 Алгоритм (№4) создания тестовой точки маршрута

2.1. При помощи генератора случайных чисел, создается единственная точка из маршрута пользователя, правдоподобность значений создается при помощи подбора коэффицентов.

2.2. Пункт №1 повторяется до получения требуемого количества тестовых точек.

```
#define bound_rand_x() ( fabs(sin(rand())) * 0.01)// функция генерации случайного числа в формате flo
#define bound_rand_y() ( fabs(sin(rand())) * 0.01)

    float new_x, new_y;

    new_x = 61.78 + bound_rand_x();// 61.78 типичное значение широты по городу Петрозаводску
    new_y = 34.3 + bound_rand_y();// 34.3 типичное значение долготы по городу Петрозаводску
    printf("new_x = %f\n",  new_x);
    printf("new_y = %f\n",  new_y);
```

Рисунок 23.

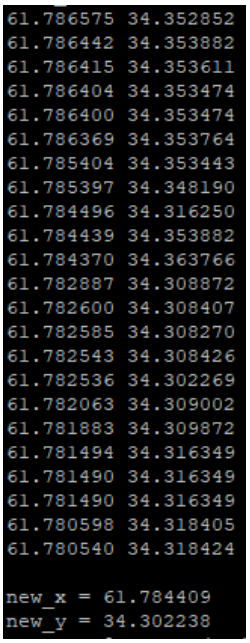

Рисунок 24. Пример работы алгоритма, где в последних двух строчках отображаются новые сгенерированные значения x и y.

Глава 4

Тестирование

4.1 План тестирования

Для проверки работоспособности системы было решено провести ручное тестирование. Выбор данного подхода к тестированию обусловлен спецификой исследуемой области и необходимостью получения реальных данных о работе приложения для его возможной последующей отладки. Далее приведен перечень основных тестов, которые проводились для проверки работоспособности системы.

4.2 Тестирование подсистемы основных функций

Тест 1 Метод ввода
Тип: Общий
Описание:
Задать название директории в которой находятся файлы с маршрутами. Запустить приложение. Полученный результат: Приложение открывает и считывает все находящиеся в указанной директории файлы.

Тест 2 Метод ввода
Тип: Общий

Описание:

Задать название директории в которой находятся файлы с маршрутами.

Запустить приложение.

Полученный результат: Приложение создает структуру типа gps, вносит в нее данные из файлов типа $\mathrm{gps}_{w}ay$.

Тест 3 Метод вывода

Тип: Общий

Описание:

1. Задать тип связи для генерации маршрутов 2. Запустить приложение.

Полученный результат: В результате получен не повторяющийся набор файлов с названиями типа $\mathrm{file}_{n}ame_{i}n$.

Тест 4 Метод вывода

Тип: Общий

Описание:

1. Задать тип связи для генерации маршрутов 2. Запустить приложение. Полученный результат: Для каждой сгенерированной структуры типа $\mathrm{gps}_{w}ay$.

Тест 5 Метод вывода

Тип: Общий

Описание:

1. Задать тип связи для генерации маршрутов

2. Запустить приложение.

Полученный результат: В файле каждая пара значений из структуры типа $\mathrm{gps}_{w}ay$.

4.3 Тестирование подсистемы Анализа типов маршрутов

Тест 1 Анализ пересечения
Тип: Общий
Описание:
Задать название директории в которой находятся файлы с пересекающимися и соприкасающимися маршрутами.
Запустить приложение.

Полученный результат: Система выявляет, что представленные маршруты являлись пересекающимися, выводит данные о том какие маршруты пересекались и в скольких точках. Аналогичный результат при смене входных данных на любой другой тип маршрутов.

Тест 2
Тип: Общий
Описание:
Задать название директории в которой находятся файлы с близкими, пересекающимися, соприкасающимися и различными маршрутами.
Запустить приложение.

Полученный результат: Система выявляет, что часть представленных маршрутов являлась близкими, часть различными, некоторые пересекались, выводит данные о том какие маршруты пересекались и в скольких точках.

4.4 Тестирование прогнозирования

Тест 1 Расчет посещаемости
Тип: Общий

Описание:
1. Запустить приложение с входными данными маршрутами различных типов.

Полученный результат: Таблица частот посещения строится на основе входных данных.

Тест 2 Генерация прогноза
Тип: Общий
Описание:
1. Запустить приложение со сгенерированной точкой.

Полученный результат: На основе таблицы частоты посещений алгоритм выбирает вариант с большим количеством посещений в зоне досягаемости от заданной точки.

4.5 Тестирование генерации маршрутов

Тест 1 Генерация маршрутов всех типов
Тип: Общий
Описание:
1. Запустить приложение с параметром любого типа.
Полученный результат: Программа генерирует маршруты заданного типа и выводит их в файлы по одному маршруту в файл.

4.6 Результаты тестирования

В итоговом варианте работы, все тесты проходятся успешно, система успешно генерирует маршруты требуемого типа, осуществляет корректный ввод и вывод полученных данных в файлы и структуры дан-

ных. Способна осуществлять работу с описанныси стуктурами и типами файлов, грамотно высчитывает расстояние между точками, упрощает маршруты с излишними данными, проводит усреднение данных, строит таблицу частот. Справляется с анализом данных, корректно определяет вид взаимосвязи между маршрутами, способна прогнозировать перемещение пользователя, на основе построенной таблицы частот. Также система соответствует всем представленным ранее требованиям к алгоритмической части и точности подсчетов.

Заключение

Полученные результаты:

2.1. Разработана и обоснована терминология задачи

2.2. Были разработаны и реализованы алгоритмы:

   i. Позволяющие установить тип взаимосвязи между тестовыми путями;

   ii. Позволяющие находить типичные маршруты среди тестовых путей;

   iii. Анализирующие точки маршрутов;

2.3. Все реализованные алгоритмы были успешно протетированы.

Таким образом, цель работы и поставленные задачи были реализованы, и хоть созданная система является лишь прототипом системы подсказок на маршруте пользователя, но уже имеет алгоритмическую основу, которую можно внедрить в класс разрабатываемых на кафедре приложений, работающих с GPS-координатами.

Список использованной литературы

1. edgetime.ru [Электронный ресурс]: Как прочитать координаты GPS - Электрон. дан. - [Москва], cop.2018 - URL: https://edgetime.ru/smartphone/kak-prochitat-koordinatyi-gps/
2. gps-coordinates.ru [Электронный ресурс]: ПОИСК ГЕОГРАФИЧЕСКИХ КООРДИНАТ - Электрон. дан. - [Москва], cop.2016 - URL: https://gps-coordinates.ru/
3. offroadrest.ru [Электронный ресурс]: GPS и связь - Электрон. дан. - [Санкт-Петербуг], cop.2015 - URL: https://offroadrest.ru/gps-format/
4. maps.google.ru [Электронный ресурс]: Карты Google - Электрон. дан. - [Москва], cop.2005 - URL: https://www.google.ru/maps/@61.78637,34.3413988,11z
5. maps.yandex.ru - [Электронный ресурс]: Яндекс.Карты: город Петрозаводск - Электрон. дан. - [Москва], cop.2004 -
6. www.u-karty.ru- [Электронный ресурс]: Карты городов России и мира - Электрон. дан. - [Санкт-Петербуг], cop. 2011 - URL: http://u-karty.ru/opredelenie-koordinat-na-karte-yandex

   URL: https://yandex.ru/maps/18/petrozavodsk/?source=wizgeo&l=map&C61.788058&z=15
7. ru.enc.tfode.com [Электронный ресурс]: The Free Online Dictionary and Encyclopedia: Douglas-Peucker Line-Simplification Algorithm - Электрон. дан. - [USA], cop. 2003 - URL: http://ru.enc.tfode.com/Алгоритм_Рамера-Дугласа-Пекера

8. forum.oszone.net [Электронный ресурс]: Компьютерный информационный портал: реализации алгоритма Дугласа-Пекера - Электрон. дан. - [Москва], cop. 2001 - URL: http://forum.oszone.net/post-1504226.html

9. www.km.ru - [Электронный ресурс]: Справочно-энциклопедический ресурс: Алгоритм фильтрации геолокационных данных - Электрон. дан. - [Москва], cop.1999 - URL: http://www.km.ru/referats/335854-blochno-vremennoi-algoritm-filtratsii-geolokatsionnykh-dannykh

10. www.java-online.ru - [Электронный ресурс]: Java онлайн для разработчиков - Электрон. дан. - [Москва], cop.2005 - http://java-online.ru/blog-tokenizer.xhtml

11. www.cyberforum.ru - [Электронный ресурс]: Кибер-портал для разработчиков - Электрон. дан. - [Москва], cop.2003 - http://www.cyberforum.ru/visual-cpp/thread169285.html