Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Петрозаводский государственный университет (ПетрГУ)
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Промежуточный отчет по курсовая работа

Идентификация типичных маршрутов мобильного пользователя в Петрозаводском городском округе

Выполнил:
студент 4 курса группы 22403 М. А. Погорянская

____
подпись

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

____
подпись

Представлен на кафедру

« ____ » ______________________ 2017 г.

____
подпись принявшего работу

Петрозаводск

2017

Содержание

Введение **3**

1 Основные понятия и алгоритмы **4**

1.1 Терминология 6

1.2 Основные положения 7

2 Алгоритм поиска типичных маршрутов среди тестовых путей **8**

3 Пример работы алгоритма **11**

3.1 Входные данные 11

4 Дальнейшие планы **13**

4.1 Портирование кода алгоритма поиска типичных маршрутов на язык среды Android Studio JAVA 14

4.2 Адаптация кода программы под возможность сравнения маршрутов имеющих разную длинну пути. 14

4.3 Возможность работы с единым входным файлом. 15

4.4 Внедрение системы фильтрации данных, для отсеивания лишних параметров входного файла. 16

5 Заключение **17**

Библиографический список использованной литературы **18**

Введение

В настоящее время многим из нас сложно представить свою жизнь без мобильных телефонов, смартфонов, планшетов, ноутбуков. Техника становится все более мобильной и отправляется в ежедневное путешествие со своим пользователем, не готовым расстаться с ней ни на минуту. Многие постоянно держат свой телефон в руках, сверяясь с отметками в календаре, что-то записывая, решая множество задач, различных по времени и месту их выполнения. Поэтому системы, помогающие упорядочить текущие дела, координировать передвижения пользователя и отслеживать процесс выполнения поставленных задач, становятся все более востребованными. В связи с этим назрела необходимость реализации удобной системы, основанной на определении географического местоположения пользователя. По меткам, прикрепленным к определенным ориентирам на карте, посредством геолокацииможно предугадать дальнейший маршрут и вывести подсказку о потениальной цели движения в данном направлении.

Цель: Разработка прототипа системы подсказок на маршруте пользователя.

Задачи:

1. Выявление типовых маршрутов;
2. Упрощение маршрутов;
3. Разработка системы установки меток на карте;
4. Сопоставление карт городов с индивидуальными метками.

1 Основные понятия и алгоритмы

Для разработки прототипа системы подсказок на маршруте пользователя необходимо реализовать эффективную методику анализа передвижения человека и автоматического прогнозирования его движения. Также требуется введение вероятностного подхода при выборе потенциального маршрута пользователя из исходной точки на основе частоты использования определенных маршрутов. Необходимо организовать хранение всей требуемой индивидуальной информации: местоположения пользователя, его личных маршрутов, частопосещаемых мест. Нужно разработать системы создания, обработки и отслеживания выполнения задач, а также отображения подсказок при постоянном изменении местонахождения человека.

Предполагаем, что встроенный в мобильное устройство GPS(ГЛОНАСС)-навигатор будет с определенной периодичностью производить замер изменения положения пользователя и, используя систему GPS(ГЛОНАСС)-координат, сохранять требуемую информацию в формате, удобном для дальнейшего сопоставления с картами городов и необходимой обработки. Полученные данные будут анализироваться для выявления важных или более востребованных для определенного пользователя маршрутов. По ходу измененния местоположения мобильного устройства пользователем будут оставляться пометки о целях, из-за которых и был выбран конкретный маршрут передвижения. Эти подсказки будут использоваться для дальнейшего отслеживания выполнения этой задачи, привязанной к определенному ориентиру на карте. С момента создания задачи она будет считаться невыполненной до тех пор, пока пользователь не окажется территориально близко к ней и не поставит метку о выполнении задуманного. Также при следующем приближении к точке на карте, к которой была прикреплена подобная подсказка, будет выдано предположение о возобновлении необходимости выполнить изначальную задачу с повторным ожиданием ее завершения.

На Рис 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.

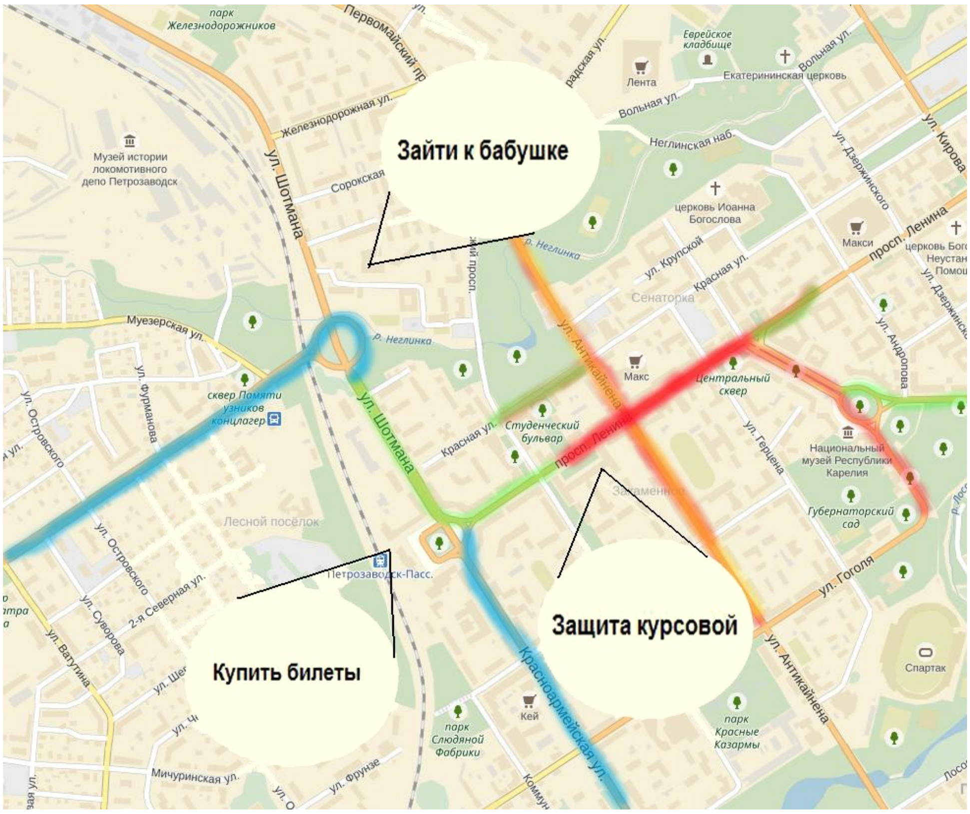

Рис. 1. Используемые пути и подсказки на маршруте

1.1 Терминология

Введем основные термины используемые в работе:

Определение 1: Маршрут(путь) - набор упорядоченных по значениям GPS(ГЛОНАСС)-координат, описывающих перемещение пользователя за ограниченный промежуток времени.

Определение 2: Точка - пара GPS(ГЛОНАСС)-координат, описывающих положение пользователя в определенный момент времени.

Определение 3: Типичный маршрут – часто используемый путь, который за период отслеживания пережвижения пользователя был обнаружен несколько раз с минимальными отклонениями в рамках заданной погрешности от первоначальных данных об этом маршруте.

Определение 4: Близкие точки - точки, находящиеся на настолько малом расстоянии друг от друга, что при сравнении их величин, являются приблизительно равными в рамках заданной погрешности.

Определение 5: Близкие пути – маршруты, целиком состоящие из близких точек.

Определение 6: Пересекающиеся пути – маршруты, в которых есть единственная общая точка пересечения.

Определение 7: Соприкасающиеся пути – маршруты, содержащие некоторое число близких точек расположенных по всему маршруту в любом порядке.

Определение 8: Различные пути – маршруты, не содержащие близких точек.

Определение 9: Важная точка - набор координат, обозначающий особое место на карте, возможно отмеченное задачей.

Определение 10: Ширина "дороги" - величина выбранная эксперементальным путем, значение которой позволяет говорить о близости точек, с погрешностью равной ее значению.

Определение 11: Отрезок пути(типичный отрезок) - общая часть двух пересекающихся путей.

2 Алгоритм поиска типичных маршрутов среди тестовых путей

1. Полный обход множества файлов тестовых маршрутов всех видов путем попарного считывания маршрутов из двух различных файлов ~~за раз~~ в соответствующие массивы;

2. Параллельное считывание координат путей, занесение в структуру данных, представляющую собой двухмерный массив ~~на~~ пять строк, в каждом столбце которого хранится:

   (a) Координата x;

   (b) Координата y;

   (c) Порядковый номер в файле, из которой были считаны данные;

   (d) Номер файла;

   (e) Поле, хранящее отладочную информацию о состоянии обработки маршрута.

   Данная структура ~~поможет~~ осуществить контроль растояния между путями в любых его точках, при этом благодаря сохранению данных об изначальных маршрутах нет потери необходимой информации.

```
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++;
}
```

Рис. 2. 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.

3. Поиск близких точек:

    (a) Взятие наименьшей по x координате точки;

    (b) Сравнение ее со следующим элементом упорядоченного массива;

    (c) Если разность величин будет меньше или равна заданной погрешности ширины “дороги”по обеим координатам, а номера файлов, из которых эти точки были взяты, различными, то считать точки близкими;

    (d) Иначе перейти к поиску близких точек среди оставшихся элементов массива.

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

    (a) Такой же номер файла;

    (b) Следующий порядковый номер в файле.

```
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;
}
```

Рис. 3. Реализация поиска и отметки близких точек.

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

6. Если все точки маршрута оказались близкими друг для друга, считать этот путь типичным, провести уточнение маршрута, по средствам усреднения данных по общим точкам с занесением полученного нового маршрута, в дополнительный массив хранящий текущие типичные маршруты(пути что были усреднены хотябы 1 раз);

```
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

    }
```

Рис. 4. Данная функция обобщает маршруты в единый путь если они были близки по большенству точек.

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

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

9. Если все файлы уже обработаны, перейти к обработке массива типичный маршрутов, для поиска повторяющихся в нем путей, дальнейшего их уточнения и вывода информации о найденных типичных маршрутах. (Такая ситуация может вознинуть при наличие более чем двух одинаковых маршрутов.)

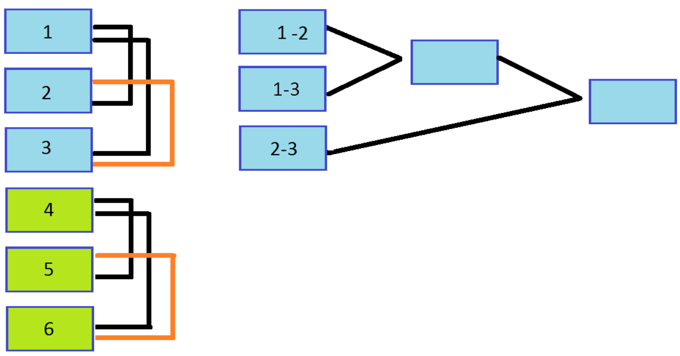

Рис. 5. Описание алгоритма уточнения маршрута.

3 Пример работы алгоритма

3.1 Входные данные

Для ~~обеспечения~~ ранее описанного алгоритма необходимыми входными данными, и его тестировании на реальных данных, были ~~приняты следующие меры~~. При содействии аналогичных курсовых проектов был получен первичный пробный GPS(ГЛОНАСС) - маршрут пользователя 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 под заголовками Широта и Долгота, скрывались 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
```

Рис. 6. Часть полученного первичного GPS(ГЛОНАСС)-маршрута.

Входной файл содержал 58 строк-замеров, 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, но его точность оказалось избыточной и требовала доработки. 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, исходный файл имел лишь 19 значимых замеров и хранил их в формате x(Широта) y(Долгота).

За отсутствием большого количества настоящих GPS(ГЛОНАСС)-измерений было решено искуственно 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 реальных замеров, которые

```
61.7825857 34.30827
61.7820635 34.3090038
61.78149 34.3163508
61.7828846 34.3088731
61.78149 34.3163508
61.7864136 34.3536091
61.78149 34.3163508
61.7864 34.3534732
61.78149 34.3163508
61.7864407 34.3538809
61.78149 34.3163508
61.7864407 34.3538809
61.7863695 34.353764
61.7864136 34.3536091
61.7865763 34.3528537
61.7853949 34.3481889
61.7825417 34.3084249
61.7825993 34.3084059
61.7825417 34.3084249
```

Рис. 7. Упрощенный исходный маршрут.

утверждали наличие хотябы одной пары близких точек, практически в каждом замеренном пути в связи с не большой площадью местности с проводимыми замерами. Для генерации маршрутов использовались алгоритмы тестовой генерации типичных маршрутов, и тестовой генерации пересекающихся маршрутов, разработанные ранее для тестирования методов упрощения кривых. На вход алгоритму генерации типичных маршрутов подавался изначальный реальный замер(Маршрут №1), тот в рамках ширины "дороги" увеличивал или уменьшал значения переменных x y в каждом замере, на выходе получали запрошенное количество типичных к заданному маршрутов (Маршруты №Т-1...Т-n, где n - количество необходимых типичных маршрутов). Аналогично происходила разработка пересекающихся маршрутов, при помощи алгоритма генерации пересекающихся маршрутов был построен путь отличающиеся от оригинильного в ряде точек более чем на шрину "дороги"(Маршрут №2), что не позволяло считать данные маршрут близким к исходному(Маршруту №1), далее по аналогии с предыдущим шагом, были сгенерированы типичные маршруты к Маршруту №2 (Маршруты №П-1...П-n, где n - количество необходимых типичных маршрутов).

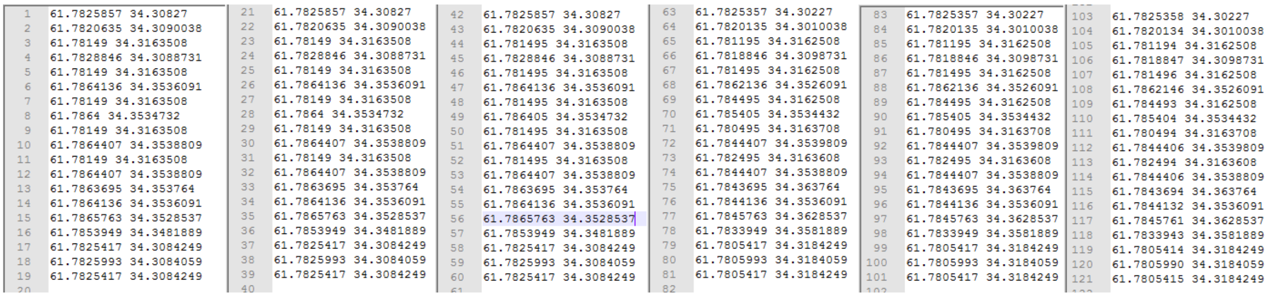

Рис. 8. (Слева на право) Исходный упрощенный Маршрут №1, Типичный Маршрут №Т-1, Типичный Маршрут №Т-2, Пересекающийся Маршрут №2, Т ипичный Маршрут №П-1, Типичный Маршрут №П-2.

4 Дальнейшие планы

После реализации алгоритма поиска типичных маршрутов, было принято решения внедрить алгоритм в уже частично разработанную систему алгоритмов упрощения ломаных на мобильных устройствах (на территории города Петрозаводска). Которая представляет собой приложение на мобильное устройство Android, работающая при помощи осуществления замеров GPS координат с некоторым промежутком времени. Система имеет графический интерфейс, и отмечает на карте ломаными передвижение пользователя. В процессе изчения этой системы был выявлен ряд факторов которые должны быть учтены для успешного слияния систем.Часть этих проблем описывалась в пункте “Пример работы алгоритма”, где для пробных данных полученной ситемой алгоритмов упрощения ломаных на мобильных устройствах, были получены необходимые данные, но для слияния систем необходимы более точные и автоматизированные подходы объединения форматов данных и решения сопутсвующих проблем.

Основные проблемы слияния систем:

- Система алгоритмов реализована на языке JAVA, в среде разработки мобильных приложений Android Stidio.
- Количество замеров в сравниваемых путях может быть различным.
- На выходе работы системы алгоритмов существует один файл, пути дозаписываются друг за другом.
- Выходой файл содержит избыточную информацию: даты замеров, подряд повторяющиеся элементы, текстовые константы.

Пути решения проблем:

- Необходимо портировать исходный код алгоритма поиска типичных маршрутов на язык среды разрботки Android Studio JAVA.
- Адаптировать код программы под возможность сравнения маршрутов имеющих разную длинну пути.
- Добавить возможность работы с единым входным файлом.
- Внедрить систему фильтрации данных, для отсеивания лишних параметров.

4.1 Портирование кода алгоритма поиска типичных маршрутов на язык среды Android Studio JAVA

Для успешного перенесения кода программы на другой язык стало необходимым дополнение знаний по темам:

- Перегрузка и переопределение методов.
- Абстрактные методы и классы.
- Конструкторы класса.
- Вложенные классы.
- Пакеты и интерфейсы.
- Синтаксический разбор строки Класс StringTokenizer
- Файловый ввод/вывод,Прямой доступ к файлу

Так же было необходмо изучить особнности работы в среде Android Studio.

4.2 Адаптация кода программы под возможность сравнения маршрутов имеющих разную длинну пути.

Для поточечного сравнения двух маршрутов имеющих одинаковую длинну алгоритм поиска типичных маршрутов работает без ошибочно. Но наличие хотябы одной точки не имеющей достаточно близко к ней расположенной пары во втором пути заставит алгоритм считать эти маршруты лишь пересекающимися, а не типичными друг к другу, хотя разное количество точек в нем может быть обусловлено лишь разной скоростью прохождения маршрута, или разными упрощениями путей. Для решения этой проблемы необходимо обобщить алгоритм: любые пары подряд идущих точек в замерах будут считаться некими маршрутами. Таким образом можно обеспечить попарное сравнение точек, и два маршрута будут считаться близкими друг к другу если большая часть пар идущих подряд точек 1-го пути, будет соответствовать парам 2-го пути. Таким образом, алгоритм на вход имея два маршрута, будет искать типичные отрезки пути пользователя, даже в небольших пересечениях разных маршрутов, что поможет повысить точность рассчетов, об использовании пользователем тех или иных маршрутов.

4.3 Возможность работы с единым входным файлом.

Наличие единого выходного файла упрощает систему обращения к входным/выходным потокам данных, но усложняет обращение к находящейся в нем информации. Для корректной работы алгоритма необходимо хранение последовательностей переменных описывающих маршрут пользователя, здесь критичным является количесто переменных в маршруте, порядок переменных(необходимо хранение информации о следующей и предыдущей переменной) Частично проблема решается по средствам метода описанного в Адаптация кода программы под возможность сравнения маршрутов имеющих разную длинну пути. Так как при сортировке всех точек маршрута мы будем сравнивать лишь две близкие по значениям точки, а далее переходить к следующей паре, не обращая внимание на порядок записи этих точек в маршрут, но хранение информации о том к какому маршруту принадлежит точка - все еще нобходим. Здесь возникает проблема отделения в едином файле точек из разных маршрутов. В ходе работы было предложено несколько вариантов ее решения:

- Разделение маршрутов по времени замеров. (Если между соседними точками в пути большой зазор по времени, то считать путь новым)
- Разработка системы разделителей. (Выключение приложение, отключение GPS как конец пути)
- Разработка системы точек на карте, по которым будет разделяться маршруты.(Попросить пользователя при первом запуске приложения отметить на карте места, которые он часто посещает, и обрезать маршруты, по достижении этих точек)

Так же необходимо дополнить входной файл программы данными о том к какому маршруту он принадлежит, т.е выработать систему отметок или номеров, отвечающих за нумерацию маршрутов пользователя.

4.4 Внедрение системы фильтрации данных, для отсеивания лишних параметров входного файла.

Для обработки входных данных было решено использовать класс StringTokenizer(String str, String delim), который разбивает входную строку на части, при помощи заданного разделителя delim. Так же возможно использование методов класса:

- int countTokens() Определяющий количество элементов между разделителями
- boolean hasMoreElements() Определяющий наличия разделителей
- boolean hasMoreTokens() Функция определения наличия лексем в формате true or false
- Object nextElement() Возвращает объект, на который указывает токенайзер
- String nextToken() Возвращает подстроку, на которую указывает токенайзер
- String nextToken(String delim) Возвращает подстроку, на которую указывает разделитель delim

5 Заключение

В дальнейшем планируется найти и реализовать решение по обнаруженным проблемам, провести слияние систем алгоритмов поиска типичных маршрутов и системы алгоритмов упрощения ломанных на мобильных устройствах. Усовершенствовать методы анализа пересечений близких и пересекающихся маршрутов, уточнить понятие типичный маршрут, разработать систему особых точек на карте. Важно не просто хранить все типичные маршруты, но уметь их обрабатывать и использовать. Необходимо высчитывать средние значения между одинаковыми путями, чтобы иметь единственный предельно точный экземпляр каждого типичного маршрута. Также стоит учитывать и частоту появления типичных путей. В перспективе нужно применить вероятностный подход для выстраивания предположений о дальнейших передвижениях пользователя мобильного устройства по точке, из которой началось движение, и дальнейшее уточнение вероятного маршрута по мере добавления новых замеров. При приближении к важным точкам должно выдаваться предположение о возможной цели выбора данного маршрута. Например, после анализа направления пути можно будет высчитать, куда вероятно движется пользователь, и напомнить ему, что в прошлый раз он был здесь, поставив себе задачу, содержание которой будет указано, стараясь предугадать цели пользователя, даже если они не указаны в рамках текущего маршрута. После накопления некоторой базы реальных измерений можно будет преступить к реализации более сложный алгоритмов, включающих в себя необходимость анализировать данные что ранее были занесены в програму, а также улучшить алгоритмы для возможности динамического появления данных о местоположении пользователя.

Список литературы

1. maps.google.ru [Электронный ресурс]: Карты Google - Электрон. дан. - [Москва], cop.2005 - URL: https://www.google.ru/maps/@61.78637,34.3413988,11z

2. maps.yandex.ru - [Электронный ресурс]: Яндекс.Карты: город Петрозаводск - Электрон. дан. - [Москва], cop.2004 -

3. 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&ll=34.347996%2C61.788058&z=15

4. 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

5. 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

6. 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

7. www.km.ru - [Электронный ресурс]: Java онлайн для разработчиков - Электрон. дан. - [Москва], cop.2005

   - http://java-online.ru/blog-tokenizer.xhtml