ISSN 2225-9309 (Print)
2024. № 2 (31). С. 20–27
Молодежный Вестник УГАТУ
http://journal.ugatu.su

УДК 004.65

ПО ДЛЯ ПОИСКА МИНИМАЛЬНОГО ПУТИ НА ПЕРЕСЕЧЕННОЙ МЕСТНОСТИ

Ч. Д. Булатова[1]

[1] bulatovach@gmail.com

ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ)

Аннотация. В стремлении к постоянной автоматизации ручного труда и оптимизации различных процессов предлагается программное обеспечение для поиска минимального пути на пересечённой местности. Проанализирована актуальность данной задачи и предложен способ реализации с использованием открытых источников данных и библиотек с открытым исходным кодом. Описываются такие этапы работы, как обоснование выбора алгоритма, проектирование и разработка ПО. Дается схема работы основной части программы. Объясняются способы решения возникших в процессе разработки проблем.

Ключевые слова: алгоритм A*; алгоритм Дейкстры; эвристический алгоритм; жадный поиск; поиск минимального пути на пересеченной местности; ГИС; OSM.

ВВЕДЕНИЕ

Задача поиска пути на пересеченной местности известна и исследована намного слабее специалистами в области геоинформационных систем (ГИС) по сравнению с задачей поиска кратчайшего пути в графе, т.к. в случае с пересеченной местностью граф дорог становится очень сложным.

Несмотря на это, автоматизированный поиск минимального пути на пересеченной местности мог бы существенно облегчить работу специалистов в разных областях, связанных с изучением и перемещением по сложной местности.

Для экстренных служб и спасательных организаций определение наиболее эффективных маршрутов на пересечённой местности имеет жизненно важное значение. Спасатели должны оперативно и безопасно достигать мест происшествий, чтобы своевременно оказывать помощь и предотвращать развитие чрезвычайных ситуаций.

Оптимизация маршрутов доставки в условиях сложного рельефа позволяет сократить время и затраты на транспортировку, уменьшить износ оборудования, что повышает общую производительность и устойчивость логистических систем.

Инфраструктурные проекты, такие как строительство автомобильных и железных дорог и прочих объектов, требуют детального планирования маршрутов в условиях сложного рельефа. Это необходимо для обеспечения эффективного использования ресурсов и минимизации экологического воздействия.

Таким образом, актуальность проблемы поиска пути на пересеченной местности проникает в различные сферы деятельности, требуя инновационных подходов и решений для оптимизации процессов. Поэтому было принято решение попытаться разработать программное обеспечение (ПО), которое решало бы данную задачу.

РАЗРАБОТКА

Достижение сформулированной цели работы осуществляется путем решения следующих задач:

1) выбор эффективного алгоритма для поиска маршрута с учетом особенностей задачи;

2) подбор подходящих инструментов для разработки ПО с учетом их доступности;
3) проектирование и реализация ПО.

АЛГОРИТМ A*

Алгоритм A* (A-star) [1–4] является одним из наиболее эффективных и широко используемых алгоритмов для поиска кратчайшего пути в графах, особенно в задачах, где требуется учитывать различные стоимости переходов между узлами. A* сочетает в себе достоинства алгоритма Дейкстры и жадного поиска, используя функцию стоимости:

$$f(x) = g(x) + h(x),$$

где $g(x)$ представляет собой фактическую стоимость пути от начального узла до текущего узла (наследовано от алгоритма Дейкстры), а $h(x)$ — эвристическую оценку минимальной стоимости пути от текущего узла до целевого (наследовано от жадного алгоритма). Такая комбинация позволяет алгоритму эффективно балансировать между исследованием наиболее перспективных узлов и точным определением минимальной стоимости, что значительно ускоряет процесс поиска по сравнению с другими методами.

Применение алгоритма A* для решения задачи поиска минимального пути на пересечённой местности представляет собой оптимальный выбор благодаря его способности эффективно комбинировать точные вычисления стоимости пути и эвристические оценки [5,6]. В условиях сложного рельефа, где необходимо учитывать перепады высот и разнообразные препятствия, A* обеспечивает высокую точность и скорость нахождения оптимального маршрута. В качестве эвристики используем диагональное расстояние, учитывая, что задача будет решаться на сетке, представляющем собой карту местности.

ПРОЕКТИРОВАНИЕ ПО

Для решения задачи ПО должно иметь следующие функциональные возможности:
- добавления начального и конечного пунктов маршрута;
- добавление промежуточных точек маршрута;
- нахождение алгоритмом кратчайшего пути;
- отрисовка найденного алгоритмом пути;
- возможность укрупнения/измельчения *сетки* (для поиска пути на разных расстояниях с разной точностью);
- выделение прямоугольной области, определяющей *окно поиска* – часть карты местности, на которой работает алгоритм. Возможность изменения размера окна поиска.
- возможность выбора дополнительных условий построения маршрута;
- возможность экспорта пути в достаточно популярном формате.

Предполагаемое решение для основной части программы может быть описано блок-схемой (рис. 1).

Учитывая вышеперечисленные критерии, было принято решение разрабатывать ГИС в виде картографического веб-сервиса. Сервер будет построен на Node.js без использования какого-либо фреймворка. Для работы с геоданными и отображения карты на веб-сервисе была выбрана библиотека OpenLayers, т.к. она обладает всем необходимым функционалом и является открытой и доступной. Данные о местности будут извлекаться из открытых источников: сервиса Terrain Tiles от Mapzen, где хранятся (на Amazon S3 хранилище) плитки данных высотной модели поверхности, и источников, которые используют геоданные из OpenStreetMap (OSM) – открытого проекта, направленного на создание и предоставление детализированных картографических данных, собранных сообществом добровольцев.

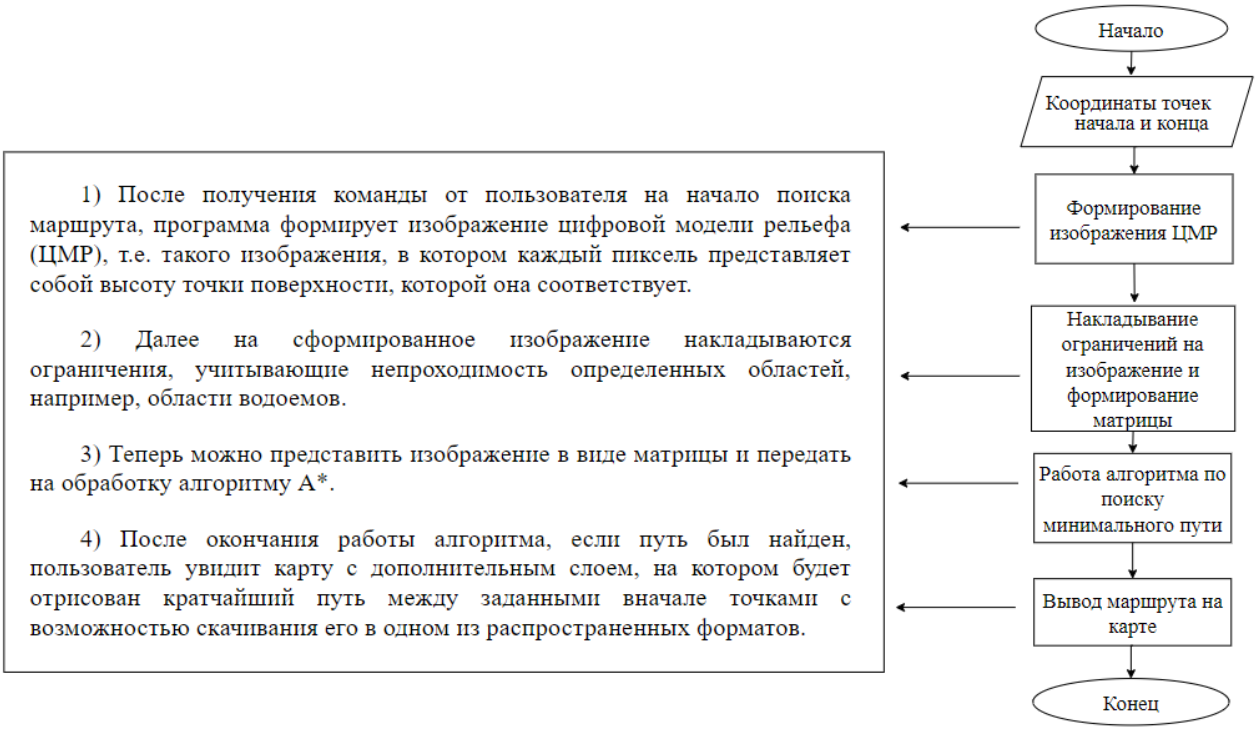

Рис. 1. Блок-схема предполагаемой работы основной части программы

РАЗРАБОТКА ПО

В соответствии с условиями и предполагаемой схемой было реализовано веб-приложение. В качестве дополнительных условий были выбраны проходимость в лесу и переход по сложному рельефу (подъём/спуск по горной местности) – проходимость в горах. В качестве формата скачивания маршрута был выбран GPX. Интерфейс приложения представлен (рис. 2).

Рис. 2. Демонстрация работы программы. Поиск кратчайшего пути между заданными точками с обходом препятствий в виде водных объектов и зданий

В левой части окна сайта мы видим боковую панель, на которой расположены элементы управления. Первый блок – слайдеры для настройки маршрута. *Zoom сетки* позволяет пользователю самому задать нужный уровень масштаба сетки: уровень zoom=15 соответствует масштабу 1:15000, уровень zoom=10 – масштабу 1:500000. С помощью 2го и 3го слайдера (4 деления) пользователь может слабо/очень сильно учитывать проходимость по горной местности и по лесной зоне. Ползунок, отодвинутый полностью вправо, делает лес непроходимым препятствием. Далее блок, состоящий из 3х кнопок, позволяющих увеличить/уменьшить, а также скрыть/показать окно поиска. В самом нижнем блоке располагаются 3 слайдера, предназначенные для удобного пользователям отображения рельефа на карте: можно изменить интенсивность тени от высотных объектов, а также азимут и высоту солнца. Под блоком окна поиска распложены 3 кнопки: зеленая(верхняя) кнопка для запуска алгоритма, голубая(средняя) – для скачивания построенного пути, красная(нижняя) – для очистки всех обозначенных точек, маршрута и окна поиска. Интерфейс приложения простой и чистый, что соответствует стилю картографических сервисов.

В ходе работы были выявлены и устранены слабые стороны программы, подробный алгоритм основной части программы может быть представлен схемой (рис. 3).

В качестве данных о свойствах местности извлекались данные обо всех водных объектах, включая болота и мелкие ручьи, данные о зданиях, заборах или преградах, о мостах, а также данные о расположении лесов. Для извлечения данных было решено использовать инструмент Overpass API, который позволяет получить выборочные данные из базы OSM по пользовательскому запросу с помощью языка запросов OverpassQL. Зная координаты начальной и конечной точки, а также уровень zoom, мы можем определить индексы тайлов – это изображения размером 256x256 пикселей, на которые разбивается карта. Тайлы отображаются рядом друг с другом, создавая одно большое изображение карты. Координаты верхнего левого угла верхнего левого тайла определяют начальную координату окна поиска. Мы формируем общее изображение рельефа местности, скачивая и объединяя нужные тайлы. Далее переводим изображение в матрицу (матрицу рельефа) для удобства дальнейшей работы с ней. Параллельно мы формируем матрицы данных о свойствах местности, которые объединяются в две конечные: матрицу препятствий (водные объекты и пути с учетом мостов, здания, заборы) и матрицу лесов. Полученные 3 матрицы подаются на вход функции, строящей сетку карты местности, которая затем передается на обработку алгоритму.

Основным недостатком программы являлась скорость обработки данных, полученных от Overpass API. Полученные геоданные представляют собой пути (way) и мультиполигоны (relation), часто с избыточной (выходящей за пределы окна поиска) информацией о водных объектах и лесах. Проверка всех ячеек матрицы на нахождение на нем какого-либо объекта занимала много времени. Для решения этой проблемы был реализован следующий подход: сначала обрезаем мультиполигоны и пути по рамке окна поиска, затем создаем холст (canvas), на котором рисуем наши объекты, и далее переводим изображения в матрицы. Были дополнительно подключены библиотеки osmtogeojson для преобразования данных OpenStreetMap (OSM) в формат GeoJSON и turf.js для работы с геообъектами (функция пересечения и другие).

Для учета условий были использованы коэффициенты, определяемые положениями ползунка на слайдерах. К $g(x)$ добавляются дополнительные стоимости, определенные переходом через гору или лес. Чем больше коэффициенты, тем выше стоимость перехода к ячейке с препятствием.

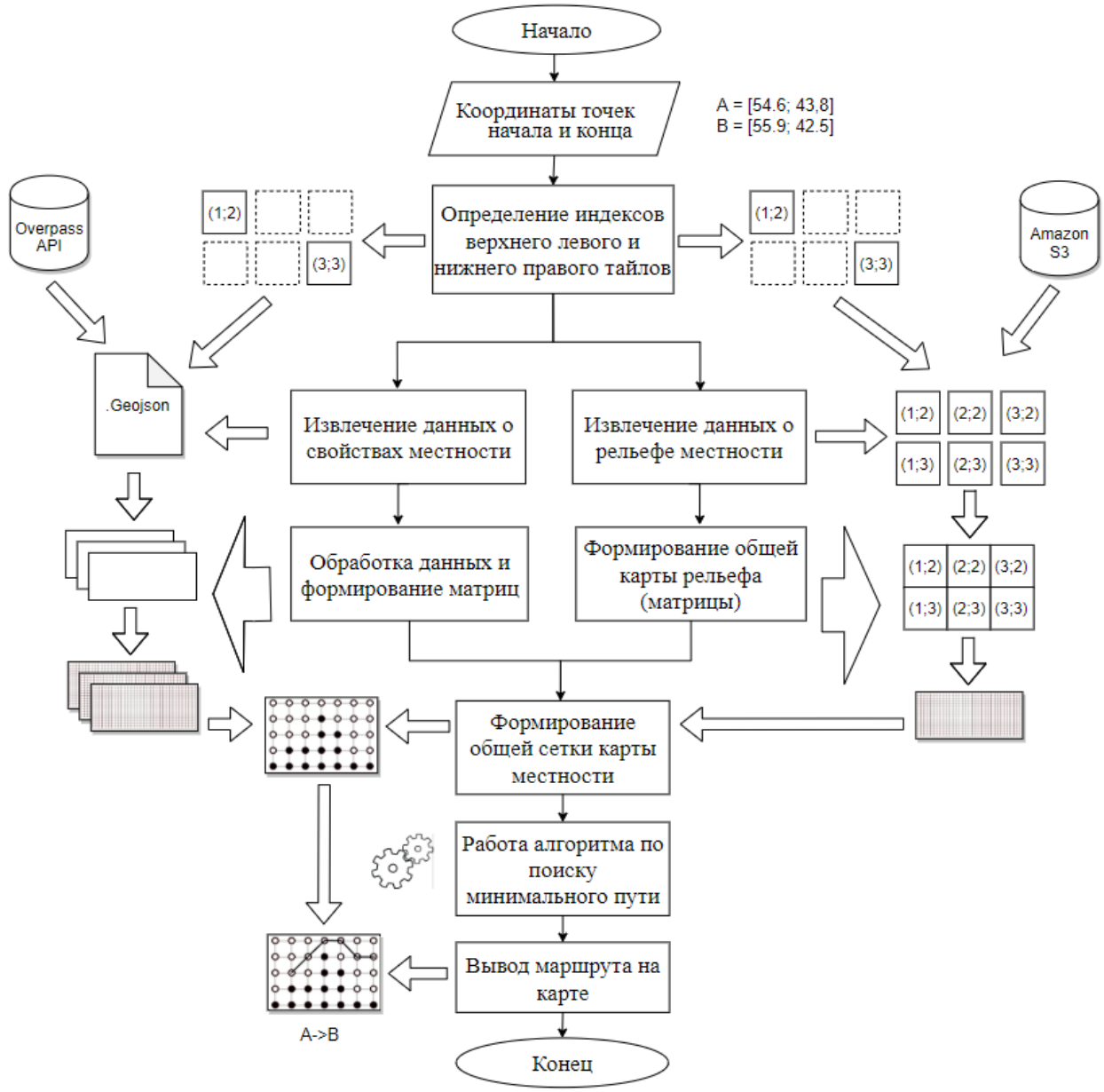

Рис. 3. Схема работы основной части программы

На примере работы приложения (см. рис. 2) мы видим, как программа находит минимальный путь между 4 точками с учетом всех отмеченных на карте препятствий.

Теперь покажем на примере, как будет меняться работа алгоритма при различных значениях проходимости через горы (рис. 4).

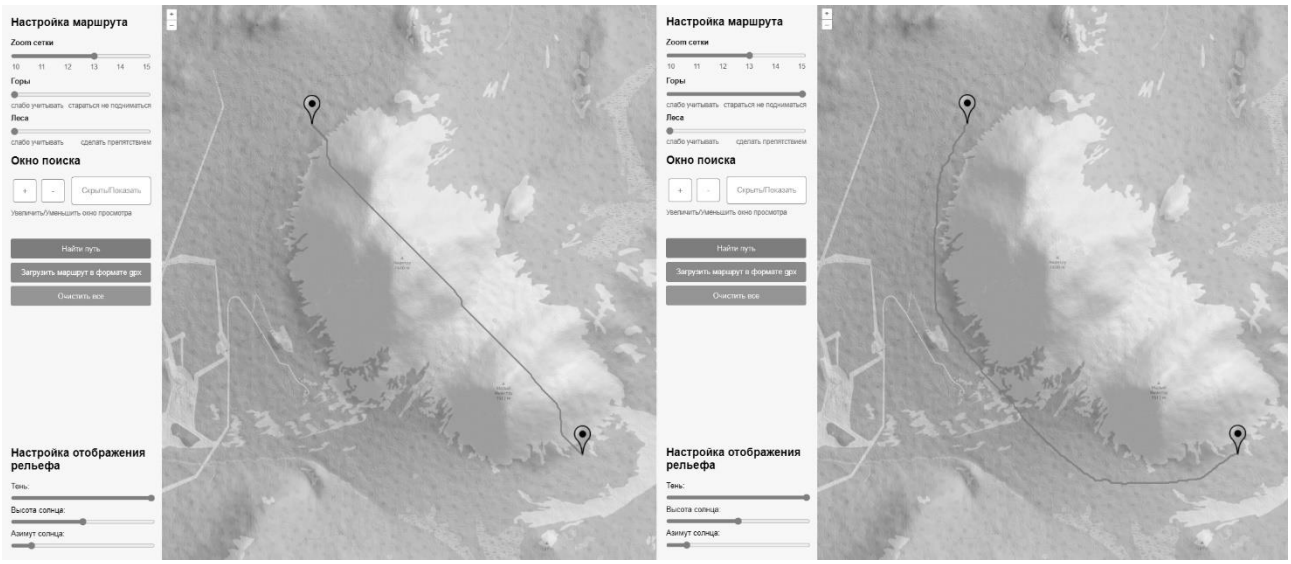

Рис. 4. Демонстрация работы программы. Слева, проходимость через горы высокая, справа проходимость через горы низкая

Мы видим, что алгоритм «проходит» через гору, когда ползунок слайдера «Горы» сдвинут влево, и огибает гору, когда ползунок слайдера сдвинут вправо – коэффициент, учитывающий сложный рельеф, достигает максимума в этом примере.

Представлен еще один пример, демонстрирующий возможности программы (рис. 5).

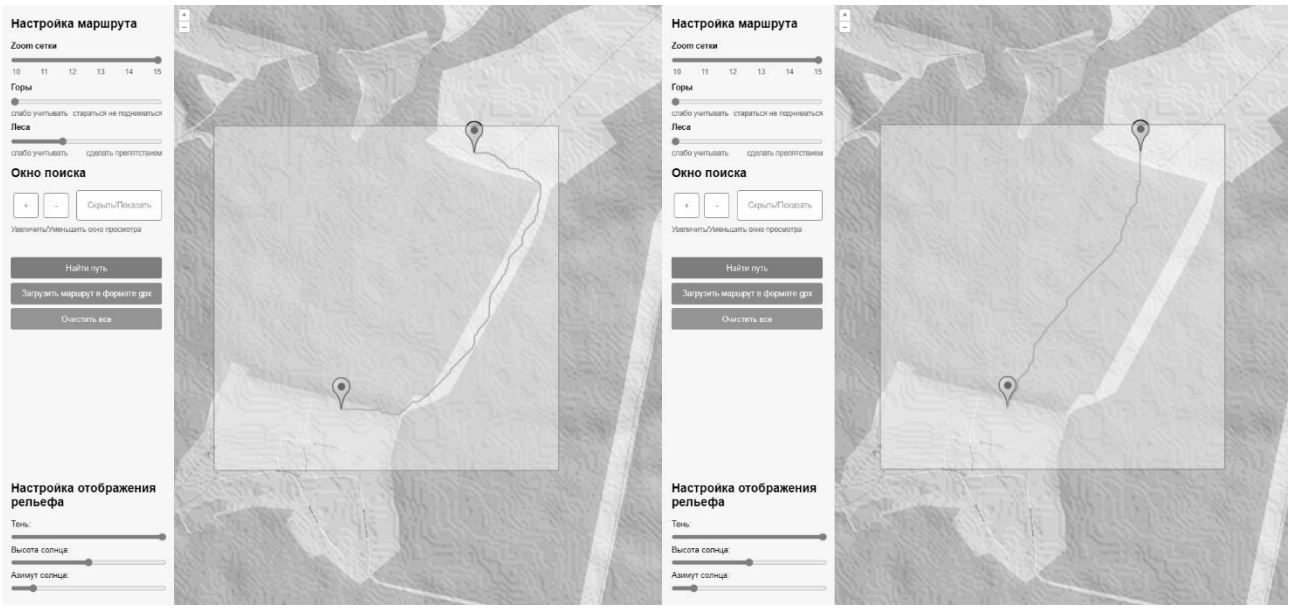

Рис. 5. Демонстрация работы программы. Слева проходимость через лес низкая, справа проходимость через лес высокая

В данном случае мы видим, как на построенный алгоритмом путь, влияет коэффициент, учитывающий проходимость через лесную зону.

ЭФФЕКТИВНОСТЬ ПО

Время работы основной функции программы складывается из времени, необходимого для формирования сетки, и времени работы алгоритма. Время, необходимое для формирования сетки – время предобработки данных – зависит в основном от скорости Интернет-соединения и загруженности серверов, от которых мы получаем наши геоданные, а также от выбранных нами способов обработки этих данных в конечную сетку.

Время работы алгоритма, т.е. время построения пути на готовой сетке, в свою очередь зависит от следующих факторов:

- расстояния между заданными точками;
- уровня zoom сетки – чем крупнее сетка, т.е. меньше zoom, тем быстрее работает алгоритм, но менее точным становится полученный маршрут;
- наличия препятствий между заданными точками – алгоритму приходится проверять большее количество узлов сетки, пока он не найдет путь прохода;
- дополнительных условий – если пользователь хочет дополнительно учитывать проходимость по лесной и/или горной местности, то алгоритму потребуется выполнять дополнительные вычисления и проверять большее количество узлов сетки.

Покажем примерную зависимость времени нахождения маршрута от размера окна поиска, вычисленную на тестовых данных, в табл. 1.

Таблица 1

Эффективность работы приложения

Размер окна поиска	Количество точек маршрута	Общее время выполнения (сек)	Время предобработки данных (сек)	Время работы алгоритма a_star (сек)
256x256	159	2.429	2.389	0.040
512x512	277	3.335	3.146	0.187
768x768	528	4.899	4.367	0.531
1024x1024	1123	154.669	3.600	151.069
1280x1280	1361	51.966	5.069	46.896
1536x1536	1271	35.697	10.263	25.432
1792x1792	2009	281.123	5.552	275.568
2048x2048	2263	1297.278	15.817	1281.459

Видим, что при увеличении размера окна поиска основной нагрузкой для производительности становится работа алгоритма по поиску пути. Время предобработки данных было оптимизировано изменением в способах формирования матриц в ходе разработки, поэтому является вполне приемлемым и не нуждается в дополнительных ускорениях. Для оптимизации скорости работы алгоритма A* можно воспользоваться методом укрупнение сетки [1]. Таким образом, на данном этапе пользователь может увеличить скорость изменением уровня zoom на меньший – уменьшится размер окна поиска, при этом построенный маршрут несколько потеряет точность.

В дальнейшем уменьшить время работы алгоритма можно, например, путем распараллеливания поиска точных путей на отдельных отрезках общего пути, грубо вычисленного перед этим [7].

ЗАКЛЮЧЕНИЕ

В данной статье представлены результаты разработки программного обеспечения для поиска кратчайшего пути на пересеченной местности. Нахождение маршрута возможно с учетом дополнительных условий (проходимость по горной и лесной местности), выбранных пользователем. Также пользователь может самостоятельно задать уровень масштаба сетки и величину окна поиска, т.е. части карты, на которой будет осуществляться поиск. Это позволяет ему регулировать время работы программы, выбирая подходящий ему вариант соотношения точности будущего пути и времени поиска.

Программное обеспечение представляет собой картографический веб-сервис, реализованный на языке JavaScript на платформе Node.js и *отличается* использованием бесплатных источников геоданных (от Mapzen и OSM), а также использованием библиотек с открытым исходным кодом OpenLayers, Turf.js, Osmtogeojson.js.

В работе приведено краткое описание основных этапов создания ПО и работы основной части программы, также отмечены возникшие проблемы и их решения. Разработанное приложение *отличается* уникальными решениями, которые были реализованы для достижения поставленной в начале работы цели: способами формирования матриц, модификацией алгоритма A* для установленных типов поступающих данных и др.

Дальнейшие исследования предполагается продолжить в направлении увеличения показателей производительности, в частности, уменьшении времени работы алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. **Hart P. E., Nilsson N. J., Raphael B.** A Formal Basis for the Heuristic Determination of Minimum Cost Paths. — IEEE Transactions on Systems Science and Cybernetics, 1968. Vol. 4 (2). — P. 100–107.

2. **Lester P.**. A* Pathfinding for Beginners [Электронный ресурс] URL: https://csis.pace.edu/~benjamin/teaching/cs627/webfiles/Astar.pdf (дата обращения: 05.04.2024).

3. **Stout B.** Smart move: Intelligent path-finding. [Электронный ресурс]. URL: https://www.gamedeveloper.com/programming/smart-move-intelligent-path-finding (дата обращения 10.04.2024).

4. **Изотова Т.Ю.** Обзор алгоритмов поиска кратчайшего пути в графе // Новые информационные технологии в автоматизированных системах. 2016. №19 С. 341–344.

5. **Russell S. J., Norvig P.** Artificial Intelligence: A Modern Approach. — Prentice Hall, 2009.

6. **Клочкова Е.Н.** Обоснование выбора алгоритма поиска пути решения задач построения маршрута к месту назначения // Вестник Московского университета МВД России. 2015. №5. С. 205–209.

7. **Lester P.** Two-Tiered A* Pathfinding [Электронный ресурс] URL: https://www.cnblogs.com/liusy1988/archive/2011/10/16/2214370.html (дата обращения: 15.05.2024).

ОБ АВТОРАХ

БУЛАТОВА Чулпан Динаровна, студент ПРО ИИМРТ УУНиТ.

METADATA

Title: Software for finding the shortest path on rough terrain.

Authors: C. D. Bulatova

Affiliation:

Ufa University of Science and Technology (UUST), Russia.

Email: bulatovach@gmail.com

Language: Russian.

Source: Molodezhnyj Vestnik UGATU (scientific journal of Ufa University of Science and Technology), no. 2 (31), pp. 20-27, 2024. ISSN 2225-9309 (Print).

Abstract: In an effort to permanently automate manual labor and optimize various processes, software for finding the shortest path on rough terrain is proposed. The relevance of this task is analyzed and a method of implementation using open data sources and open source libraries is proposed. Such stages of work as justification of algorithm selection, design and software development are described. The scheme of operation of the main part of the program is given. The ways of solving the problems arisen in the process of development are explained.

Key words: algorithm A*; Dijkstra's algorithm; heuristic algorithm; greedy search; finding the shortest path on rough terrain; GIS; OSM.

About authors:

BULATOVA, Chulpan Dinarovna, student PRO IIMRT UUST.