Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
**дисциплины**
**«Искусственный интеллект в профессиональной сфере»**
**Вариант 11**

Выполнил:
Репкин Александр Павлович
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Ассистент департамента цифровых, робототехнических систем и электроники Богданов С.С.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

**Тема:** Исследование методов поиска в пространстве состояний

**Цель:** приобрести навыки работы с методами поиска в пространстве состояний с помощью языка программирования Python версии 3.x.

**Порядок выполнения работы:**

1. При помощи сервиса Yandex Maps были выбраны 22 связанных между собой населённых пункта. В задании требовалось построить граф, где узлы будут представлять выбранные населённые пункты, а рёбра – дороги, соединяющие их. Вес ребра означает расстояние между населёнными пунктами. Требовалось выбрать начальный и конечный пункты на графе и определить минимальный маршрут между ними, проходящий минимум через 3 населённых пункта (Не считая начальный и конечный пункты). Так, если взять за начальную и конечную точки населённые пункты Актогай и Нура соответственно, то минимальным маршрутом будет: Актогай -> Аксу-Аюлы -> Караганда -> Кертинды -> Нура. Общее расстояние составит: 124+131+152+38 = 445 км. Также, возможны иные 4 маршрута, составляющие соответственно: 450, 447, 539, 597 км.

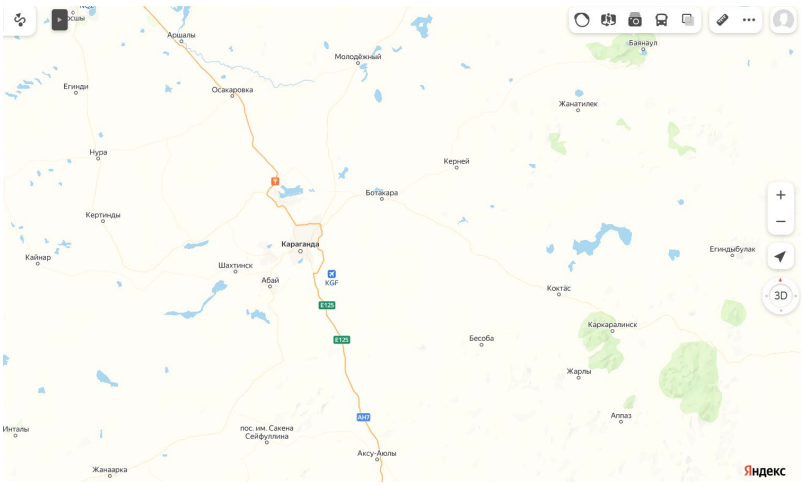


Рисунок 1 – Открытый сервис Yandex Maps

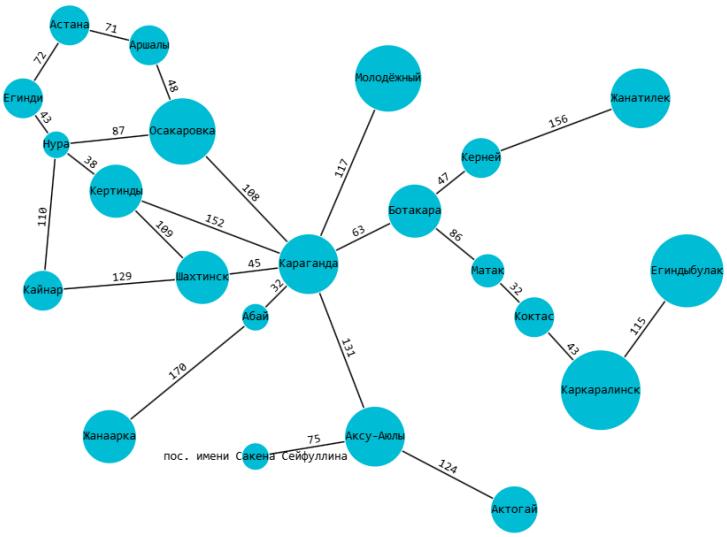


Рисунок 2 – Полученный взвешенный граф с участием Астаны и областей: Карагандинской, Акмолинской, Павлодарской и Улытау

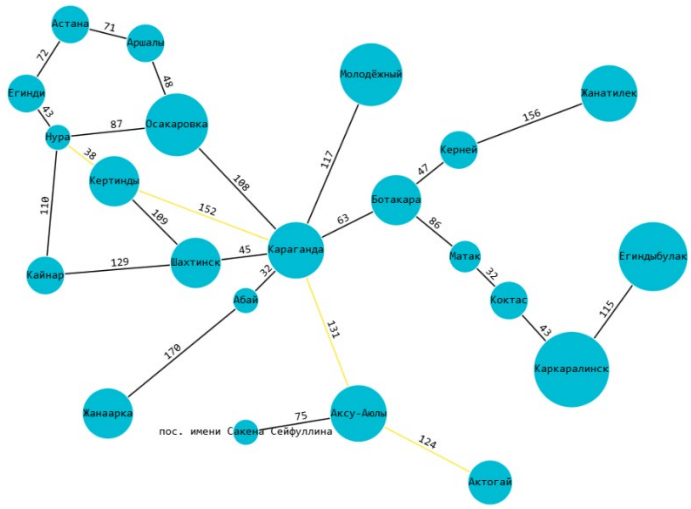


Рисунок 3 – Минимальный маршрут от Актогай до Нура, 445 км

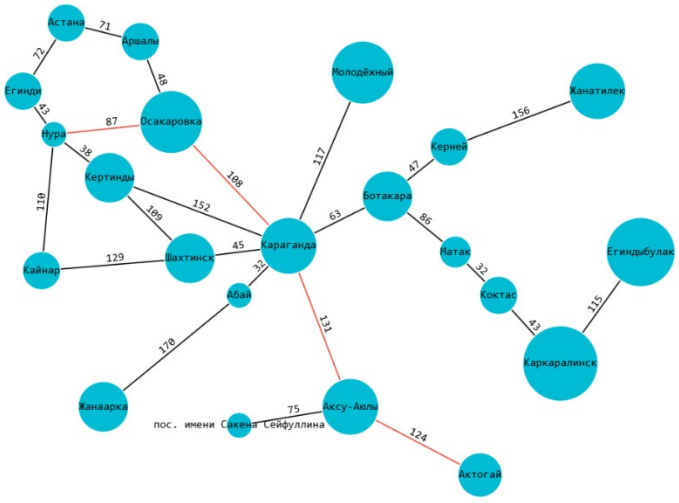


Рисунок 4 – Второй возможный маршрут от Актогай до Нура, 450 км

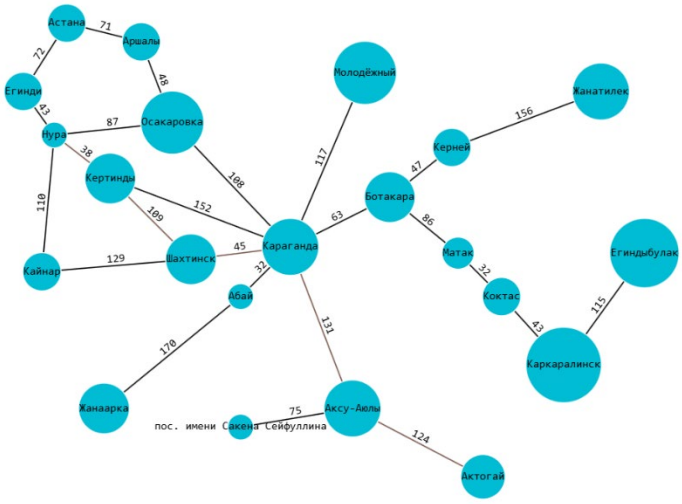


Рисунок 5 – Третий возможный маршрут от Актогай до Нура, 447 км

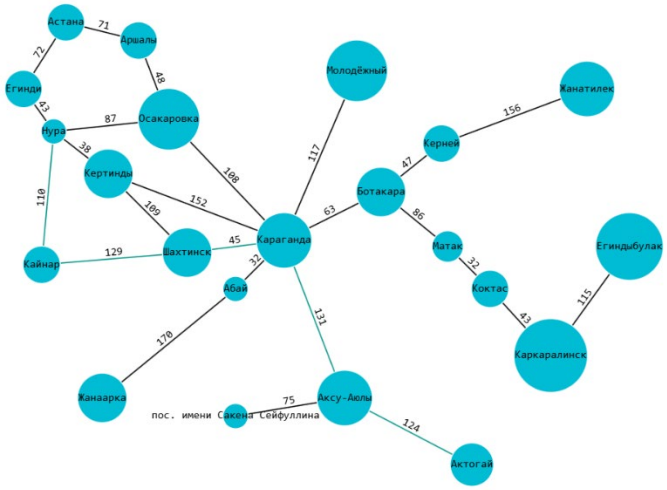


Рисунок 6 – Четвёртый возможный маршрут от Актогай до Нура, 539 км

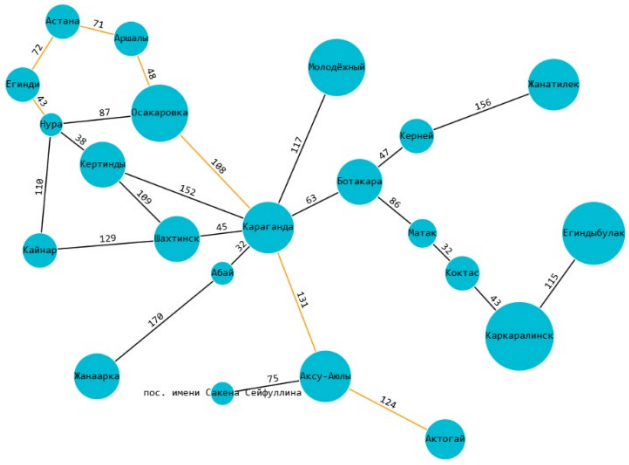


Рисунок 7 – Пятый возможный маршрут от Актогай до Нура, 597 км

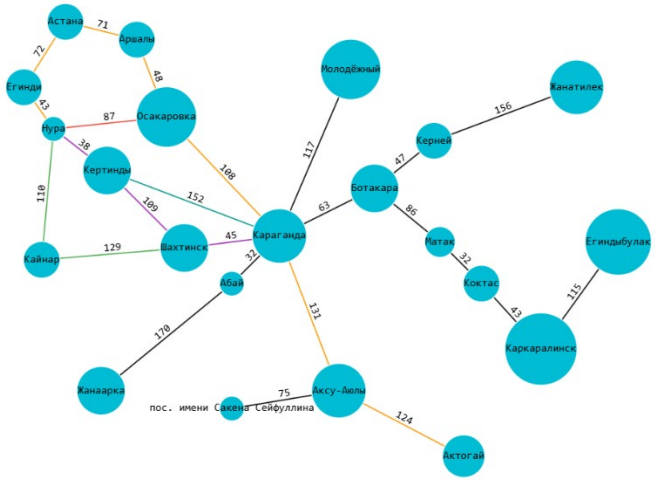


Рисунок 8 – Все возможные маршруты от Актогай до Нура

2. Для поиска оптимального маршрута можно также использовать дерево поиска, которое включает в себя листовые узлы, преемники, дочерние узлы. Основная задача дерева – это проложить каждый представленный маршрут, даже если он заканчивается не на цели. Так и получается полное дерево путей. На рисунке №9 представлено дерево поиска оптимального пути от Актогай до Нура, содержащее корневой узел – Актогай. Поиск по дереву продолжается до тех пор, пока не будет достигнута цель – Нура.

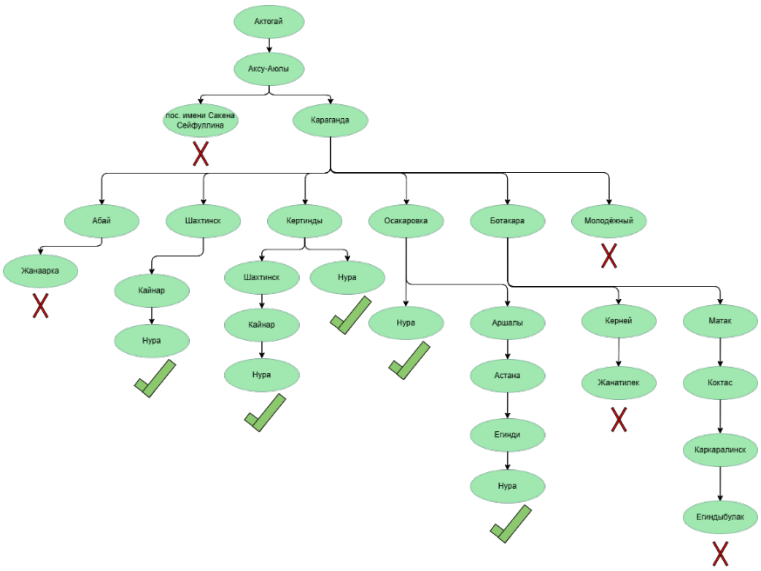


Рисунок 9 – Дерево путей, получаемые 5 маршрутов

**Ответы на контрольные вопросы:**

**1.** Что представляет собой метод "слепого поиска" в искусственном интеллекте?

**Ответ:** метод “слепого поиска” можно представить как попытку найти выход из затемнённого лабиринта, ощупывая каждую стену, каждый угол, без заранее известного плана или карты. Он исследует пространство возможных решений методом проб и ошибок, не обладая информацией о том, насколько близко каждое принятое решение к финальной цели. Такой подход, хотя и элементарен, является основой для разработки более сложных алгоритмов, включая эвристический поиск.

**2.** Чем отличается эвристический поиск от слепого поиска?

**Ответ:** эвристический поиск, в отличие от слепого, использует дополнительные знания или "эвристики" для направления процесса поиска, подобно тому, как путешественник использует компас в неизведанных землях. Один из наиболее известных примеров такого поиска - алгоритм A*, который находит наиболее оптимальный путь к цели, основываясь на заранее заданных критериях и предположениях.

**3.** Какую роль играет эвристика в процессе поиска?

**Ответ:** благодаря эвристическому алгоритму поиска может быть найден наиболее оптимальный путь к цели, основываясь на заранее заданных критериях и предположениях.

**4.** Приведите пример применения эвристического поиска в реальной задаче.

**Ответ:** шахматы представляют собой классическую арену, где можно применить стратегию эвристического поиска. На первый взгляд, задача простая: программа должна уметь генерировать возможные ходы и следовать базовым правилам игры. Однако, чем глубже мы погружаемся в задачу, тем сложнее становится её выполнение. Разработка ИИ для игры в шахматы требует глубокого понимания не только правил, но и бесчисленных стратегий и тактик.

**5.** Почему полное исследование всех возможных ходов в шахматах затруднительно для ИИ?

**Ответ:** существует большой вариативный разброс ходов в шахматах и их влияний на противника, из-за чего даже ИИ не может предвидеть всего.

**6.** Какие факторы ограничивают создание идеального шахматного ИИ?

**Ответ:** даже если технически возможно построить структуру данных для представления всех возможных ходов в шахматах, встает вопрос о ресурсах - времени и памяти. Для многих задач эти ограничения могут быть преодолены, но в контексте шахмат они становятся критическими факторами. Необходимость быстро обрабатывать информацию и принимать решения в условиях, когда время может быть ограниченно, делает задачу создания эффективного шахматного ИИ особенно сложной.

**7.** В чем заключается основная задача искусственного интеллекта при выборе ходов в шахматах?

**Ответ:** основная задача ИИ при выборе ходов в шахматах – поиск наиболее оптимального хода.

**8.** Как алгоритмы ИИ балансируют между скоростью вычислений и нахождением оптимальных решений?

**Ответ:** некоторые алгоритмы могут быстро находить решения, но они могут быть далеки от оптимальных. Другие, напротив, способны находить наилучшие решения, но требуют значительных временных и ресурсных затрат. В конечном итоге, многие задачи в искусственном интеллекте, будь то шахматы, планирование маршрута или доказательство математических теорем, сводятся к поиску эффективного пути в огромном пространстве возможных решений. ИИ должен найти способ эффективно исследовать это пространство, балансируя между доступными ресурсами и желаемым качеством решения, что делает его уникальным и незаменимым инструментом в современном мире.

**9.** Каковы основные элементы задачи поиска маршрута по карте?

**Ответ:** основные элементы задачи поиска маршрута по карте: изначальный населённый пункт и конечный населённый пункт, между

которыми располагаются другие точки городов; длины путей между населёнными пунктами.

**10.** Как можно оценить оптимальность решения задачи маршрутизации на карте Румынии?

**Ответ:** оптимальность решения в данном случае предполагает нахождение маршрута с минимальной стоимостью. Среди менее эффективных вариантов могут быть маршруты через Тимишоару и Лугож с нелогичными блужданиями по кругу. Важно, чтобы итоговый маршрут завершался в Бухаресте, даже если он включает неоптимальные повторения. Оптимальное решение – это такое, которое достигается с минимальными затратами времени и расстояния.

**11.** Что представляет собой исходное состояние дерева поиска в задаче маршрутизации по карте Румынии?

**Ответ:** исходное состояние дерева поиска в задаче маршрутизации – отправная точка – город Арад.

**12.** Какие узлы называются листовыми в контексте алгоритма поиска по дереву?

**Ответ:** листовой узел – возможное решение при построении и расширении дерева.

**13.** Что происходит на этапе расширения узла в дереве поиска?

**Ответ:** расширение узла в дереве поиска подразумевает применение функции преемника, определяющей все возможные действия, применимые к текущему состоянию.

**14.** Какие города можно посетить, совершив одно действие из Арада в примере задачи поиска по карте?

**Ответ:** совершив одно действие из Арада в примере задачи поиска по карте можно посетить: Сибиру, Тимишоара, Зеринд.

**15.** Как определяется целевое состояние в алгоритме поиска по дереву?

**Ответ:** целевое состояние в алгоритме поиска по дереву – некая конечная точка, то есть точка, до которой будут разрастаться листовые узлы.

**16.** Какие основные шаги выполняет алгоритм поиска по дереву?

**Ответ:** основные шаги алгоритма поиска по дереву: выбор исходного состояния; формирование листовых узлов; определение целевого состояния.

**17.** Чем различаются состояния и узлы в дереве поиска?

**Ответ:** состояние – это, например, конфигурация плиток в головоломке, а узел – это структура данных, включающая это состояние и дополнительную информацию.

**18.** Что такое функция преемника и как она используется в алгоритме поиска?

**Ответ:** функция расширения играет ключевую роль в алгоритме поиска по дереву. Она создаёт новые узлы, применяя действия к родительскому узлу и заполняя различные поля с использованием функции преемника, чтобы создать соответствующие состояния. Так образуется основная структура поиска: дерево инициализируется начальным состоянием, затем функция преемника многократно применяется к листьям дерева, проверяя, достигли ли мы цели.

**19.** Какое влияние на поиск оказывают такие параметры, как b (разветвление), d (глубина решения) и m (максимальная глубина)?

**Ответ:** b (максимальный коэффициент разветвления) – параметр, показывающий, сколько дочерних узлов может иметь один узел; d (глубина наименее дорогого решения) – параметр, определяющий, насколько далеко нужно спуститься по дереву для нахождения оптимального решения; m (максимальная глубина дерева) – параметр, показывающий, насколько глубоко можно в принципе спуститься по дереву (В некоторых случаях это значение может быть бесконечным).

**20.** Как алгоритмы поиска по дереву оцениваются по критериям полноты, временной и пространственной сложности, а также оптимальности?

**Ответ:** полнота означает, что алгоритм находит решение, если оно существует; отсутствие решения возможно только в случае, когда задача невыполнима; временная сложность измеряется количеством сгенерированных узлов, а не временем в секундах или циклах ЦПУ, она пропорциональна общему количеству узлов; пространственная сложность относится к максимальному количеству узлов, которые нужно хранить в памяти в любой момент времени (В некоторых алгоритмах она совпадает с временной сложностью); оптимальность – способность алгоритма всегда находить наилучшее решение с минимальной стоимостью, если таковое существует (Она не связана напрямую с временной или пространственной сложностью, которые измеряют количество узлов и не гарантирует быстродействие или экономию памяти, а лишь обеспечивает нахождение решения с наименьшей стоимостью).

**21.** Какую роль выполняет класс Problem в приведенном коде?

**Ответ:** Problem – абстрактный класс для формальной задачи. Новый домен специализирует этот класс, переопределяя “actions” и “results”, и, возможно, другие методы.

**22.** Какие методы необходимо переопределить при наследовании класса Problem?

**Ответ:** при наследовании класса Problem необходимо переопределить методы: actions, result, is_goal, action_cost и h.

**23.** Что делает метод is_goal в классе Problem?

**Ответ:** метод is_goal проверяет, достигнуто ли целевое состояние.

**24.** Для чего используется метод action_cost в классе Problem?

**Ответ:** метод action_cost предоставляет стандартные реализации для стоимости действия.

**25.** Какую задачу выполняет класс Node в алгоритмах поиска?

**Ответ:** класс Node является узлом в дереве поиска.

**26.** Какие параметры принимает конструктор класса Node?

**Ответ:** конструктор класса Node принимает: текущее состояние (state), ссылку на родительский узел (parent), действие, которое привело к этому узлу (action), и стоимость пути (path_cost).

**27.** Что представляет собой специальный узел failure?

**Ответ:** узел failure обозначает неудачу в поиске.

**28.** Для чего используется функция expand в коде?

**Ответ:** функция expand расширяет узел, генерируя дочерние узлы.

**29.** Какая последовательность действий генерируется с помощью функции path_actions?

**Ответ:** функция path_actions возвращает последовательность действий, которые привели к данному узлу.

**30.** Чем отличается функция path_states от функции path_actions ?

**Ответ:** функция path_states возвращает последовательность состояний, ведущих к данному узлу, в то время как функция path_actions возвращает последовательность действий, которые привели к данному узлу.

**31.** Какой тип данных используется для реализации LIFOQueue?

**Ответ:** для реализации LIFOQueue используется обычный список Python (List).

**32.** Чем отличается очередь FIFOQueue от LIFOQueue ?

**Ответ:** FIFOQueue – это очередь, где первый добавленный элемент будет первым извлеченным. Она реализуется с помощью deque из модуля collections (deque – это обобщенная версия стека и очереди, которая поддерживает добавление и удаление элементов с обоих концов), в то время как LIFOQueue – это стек, где последний добавленный элемент будет первым извлеченным. В этом случае для реализации используется обычный список Python (List). Добавление и извлечение элементов происходит с одного конца списка.

**33.** Как работает метод add в классе PriorityQueue ?

**Ответ:** метод add добавляет элемент в очередь, с указанием его приоритета, определённым функцией key.

**34.** В каких ситуациях применяются очереди с приоритетом?

**Ответ:** очереди с приоритетом часто применяются в алгоритмах поиска кратчайшего пути, пример – A*.

**35.** Как функция heappop помогает в реализации очереди с приоритетом?

**Ответ:** функция heapop из модуля heapq позволяет извлекать элемент с наименьшим приоритетом.

**Вывод:** в ходе выполнения лабораторной работы приобретены практические навыки работы в среде визуального программирования Visual Programming Language с блоками данных, переменной и вычисления, выполнения вывода информации на дисплей с помощью сервиса простого диалога и сервиса симуляции консольного интерфейса, а также использования блока сравнения и блока простого слияния