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 и машинное обучение»

Выполнил:
Быковская Стефания Станиславовна
3 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»

_____
(подпись)

Проверил: доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Исследование методов поиска в пространстве состояний

Цель: Приобретение навыков по работе с методами поиска в пространстве состояний с помощью языка программирования Python версии 3.x

Порядок выполнения работы:

Задание: Воспользуйтесь сервисом Google Maps или аналогичными картографическими приложениями, чтобы выбрать на карте более 20 населённых пунктов, связанных между собой дорогами. Постройте граф, где узлы будут представлять населённые пункты, а рёбра — дороги, соединяющие их. Вес каждого ребра соответствует расстоянию между этими пунктами. Выберите начальный и конечный пункты на графе. Определите минимальный маршрут между ними, который должен проходить через три промежуточных населённых пункта. Покажите данный путь на построенном графе.

Была выбрана страна – Тайланд. Населенные пункты узлы, а ребра – дороги между ними, каждое ребро имеет свой вес – расстояние между населенными пунктами.

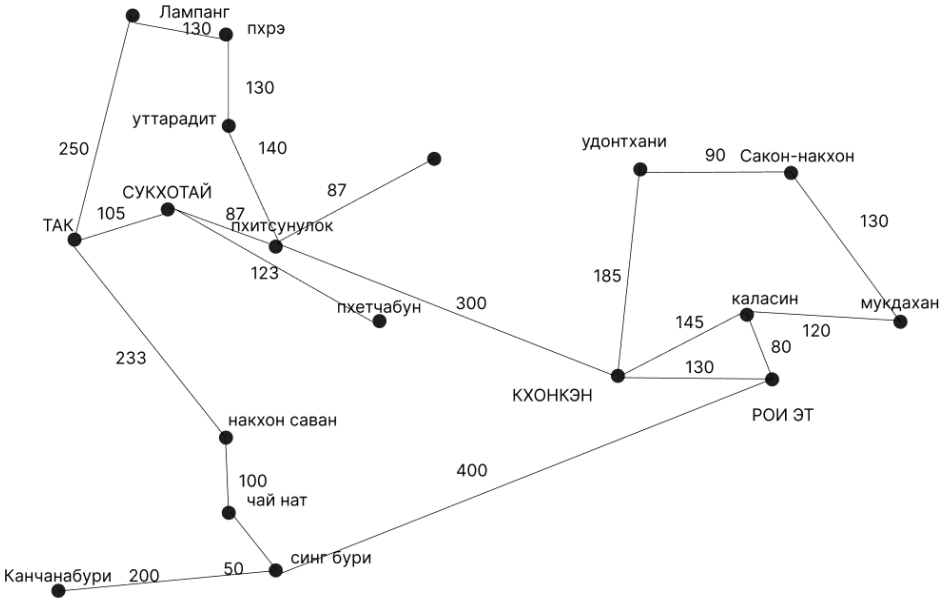

Рисунок 1. Граф

Необходимо определить начальные точки и конечную, в моем случае начальная Мукдахан, а конечная –Пхитсунулок (рис.2).

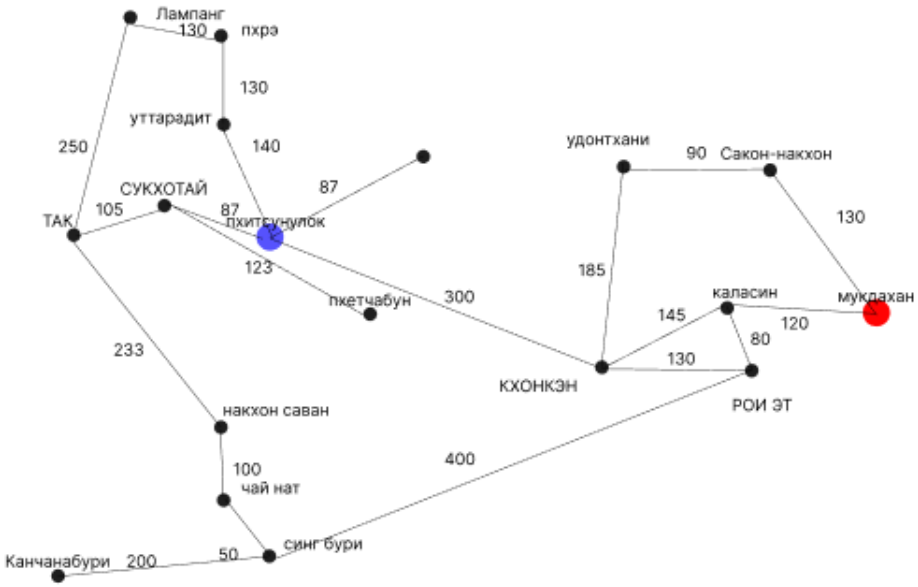

Рисунок 2. Начало и конец путей

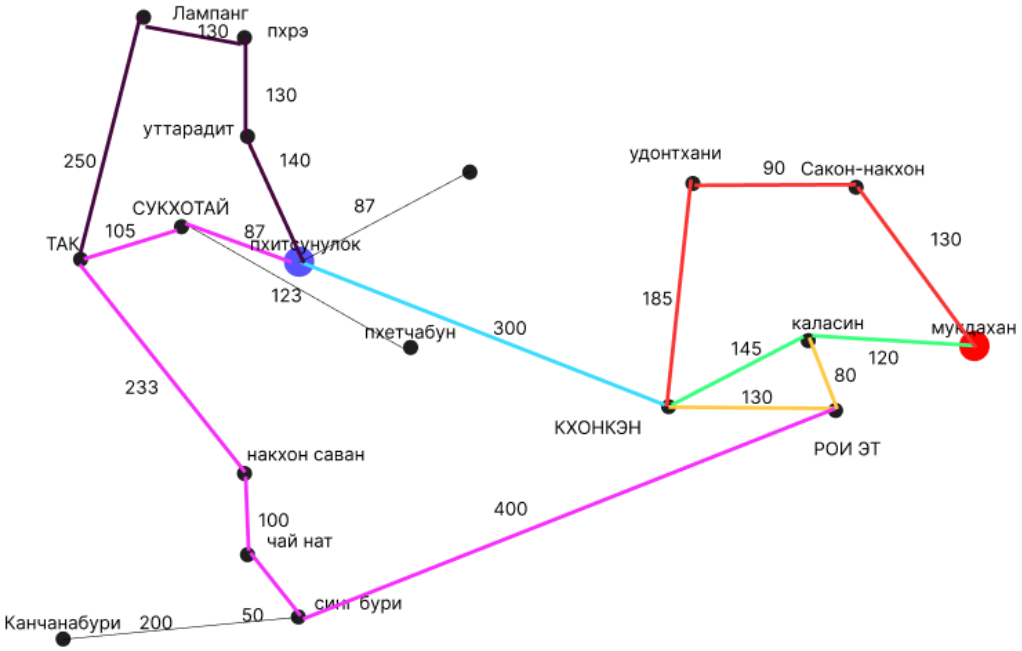

Рисунок 3. Возможные пути

На рисунке 3 отображены возможные пути до пхитсунулока, синим цветом, изображена общая дорога для жёлтого, зеленого и красного путей, по итогу имеем 5 маршрутов

Расчёт длинны маршрутов:

Красный:

130 + 90 + 185 + 300 =705 км

Зеленый:

120 + 145 + 300 = 565 км

Жёлтый:

120 + 80 + 130 + 300 = 630 км

Фиолетовый:

120 + 80 + 400 + 50 + 100 +233+ 105 + 87 =1175 км

Темно фиолетовый:

120 + 80 + 400 + 50 + 100 +233+ 250 +130 +130 + 140 =1633 км

По итогу кратчайший маршрут Зеленый(565км), он отображен на рис. 5

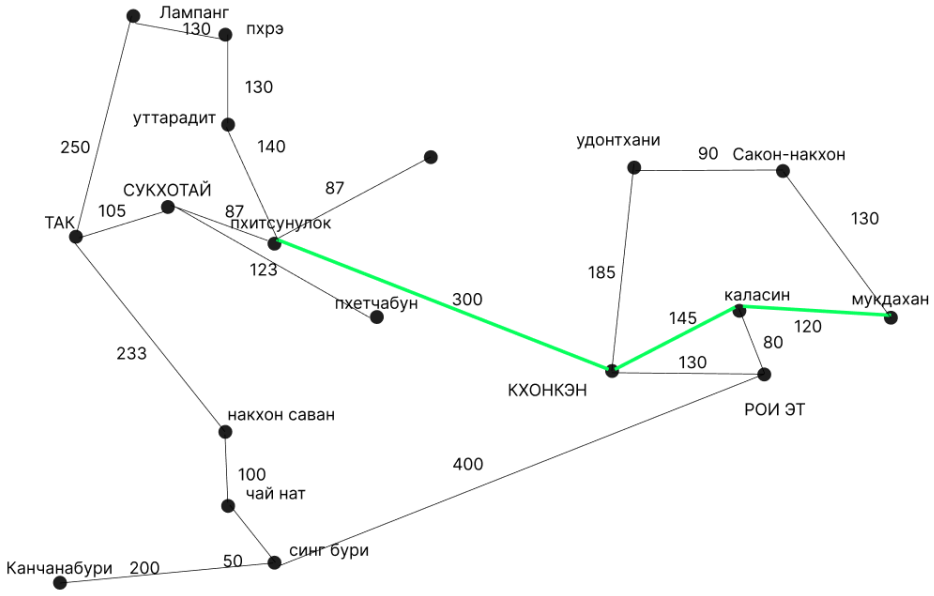

Рисунок 4. Кратчайший путь до Пхитсунулока от Мукхадан

Задача коммивояжёра (TSP) заключается в нахождении кратчайшего пути, который проходит через заданные узлы (города) и возвращается в исходный. Метод полного перебора включает в себя генерацию всех возможных маршрутов и выбор самого короткого.

Дерево поиска (рис. 5) может реализовать данную задачу, корень дерева город Мукдахан, а цель – город Мукхадан. Основная задача древа – проложить каждый маршрут.

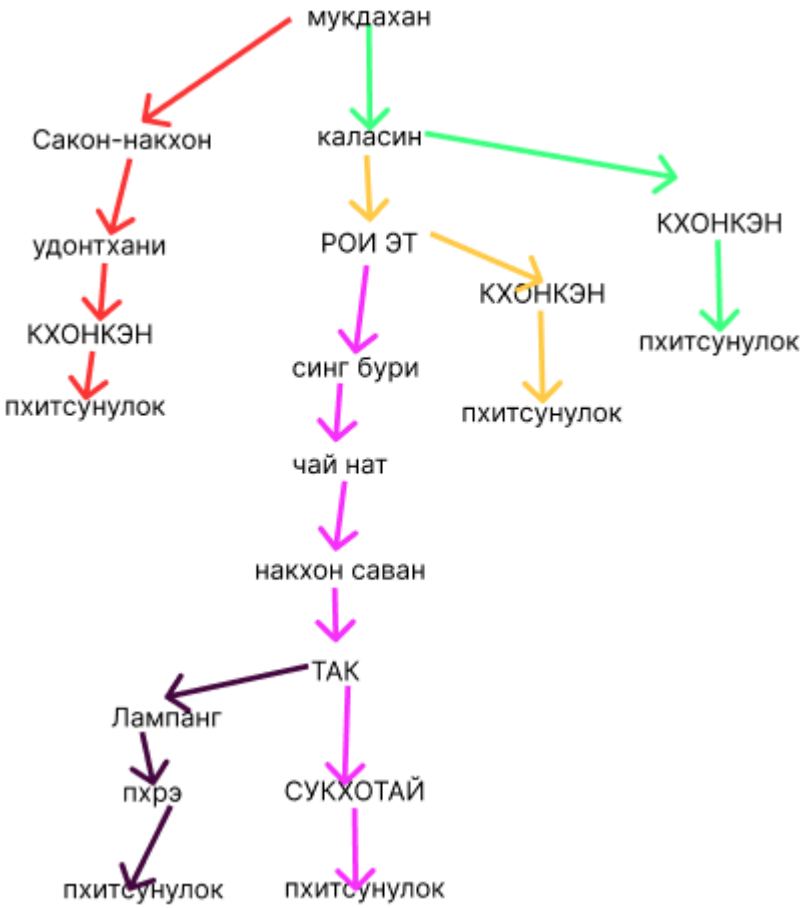

Рисунок 5. Древо поиска с узлами

Дерево поиска (рис. 5) может реализовать данную задачу, корень дерева город Мукдахан, а цель – город Пхитсунулок. Основная задача древа – проложить каждый маршрут.

Время решения задачи коммивояжёра методом полного перебора составляет O(n!), где n — количество узлов. Это связано с тем, что необходимо проверить все возможные перестановки узлов.

Контрольные вопросы:

1. Что представляет собой метод "слепого поиска" в искусственном интеллекте? Метод "слепого поиска" (или неэвристического поиска) — это подход к поиску решений, который не использует никакой дополнительной информации о проблеме, кроме самой структуры задачи. Он исследует все возможные варианты, не делая предположений о том, какой из них может быть более перспективным.

2. Как отличается эвристический поиск от слепого поиска? Эвристический поиск использует дополнительные знания или правила (эвристики), чтобы направить процесс поиска и сократить количество рассматриваемых вариантов. В отличие от слепого поиска, который исследует все возможные пути, эвристический поиск пытается выбрать более вероятные пути к решению.

3. Какую роль играет эвристика в процессе поиска? Эвристика помогает оценить, насколько близко текущее состояние к целевому, что позволяет алгоритму выбирать более перспективные пути и избегать менее продуктивных. Это значительно ускоряет процесс поиска и повышает его эффективность.

4. Приведите пример применения эвристического поиска в реальной задаче. Примером применения эвристического поиска является алгоритм A*, который используется в навигационных системах для нахождения кратчайшего пути между двумя точками на карте, учитывая расстояние и другие факторы, такие как пробки или дорожные условия.

5. Почему полное исследование всех возможных ходов в шахматах затруднительно для ИИ? Полное исследование всех возможных ходов в шахматах затруднительно из-за огромного количества возможных комбинаций ходов. Даже в начальной позиции количество возможных ходов может достигать тысяч, а общее количество возможных партий превышает количество атомов во Вселенной.

6. Какие факторы ограничивают создание идеального шахматного ИИ? Ограничивающими факторами являются вычислительные ресурсы, время на анализ каждого хода, необходимость учитывать множество стратегий и тактик, а также сложность оценки позиций на доске.

7. В чем заключается основная задача искусственного интеллекта при выборе ходов в шахматах? Основная задача ИИ при выборе ходов в шахматах заключается в нахождении оптимального хода, который максимизирует шансы на победу, минимизируя при этом риски проигрыша.

8. Как алгоритмы ИИ балансируют между скоростью вычислений и нахождением оптимальных решений? Алгоритмы ИИ используют различные техники, такие как отсечение менее перспективных вариантов (например, с помощью альфа-бета отсечения) и применение эвристик для оценки позиций, что позволяет им находить хорошие решения за разумное время.

9. Каковы основные элементы задачи поиска маршрута по карте? Основные элементы задачи поиска маршрута включают узлы (пункты на карте), рёбра (дороги между узлами), начальное состояние, целевое состояние и стоимость (расстояние или время) перемещения между узлами.

10. Как можно оценить оптимальность решения задачи маршрутизации на карте Румынии? Оптимальность решения можно оценить по критериям, таким как минимальное расстояние, минимальное время в пути, а также с учетом дополнительных факторов, таких как пробки или дорожные условия.

11. Что представляет собой исходное состояние дерева поиска в задаче маршрутизации по карте Румынии? Исходное состояние дерева поиска представляет собой начальную точку маршрута, от которой начинается поиск, и может включать информацию о текущем местоположении и доступных путях. Арад.

12. Какие узлы называются листовыми в контексте алгоритма поиска по дереву? Листовые узлы — это узлы, которые не имеют дочерних узлов, то есть они представляют собой конечные состояния в процессе поиска, где больше нет доступных путей для дальнейшего исследования.

13. Что происходит на этапе расширения узла в дереве поиска? На этапе расширения узла происходит генерация всех возможных дочерних узлов, которые могут быть достигнуты из текущего узла, что позволяет продолжить процесс поиска.

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

15. Как определяется целевое состояние в алгоритме поиска по дереву? Целевое состояние определяется как состояние, которое соответствует условиям задачи и является конечной целью поиска, например, достижение определённого пункта назначения на карте.

16. Какие основные шаги выполняет алгоритм поиска по дереву? Основные шаги алгоритма поиска по дереву включают:

1) Выбор исходного состояния
2) Формирование листовых узлов
3) Определение целевого состояния

17. Чем различаются состояния и узлы в дереве поиска? Состояния представляют собой конкретные конфигурации проблемы, которые могут быть достигнуты в процессе поиска. Узлы, в свою очередь, являются элементами дерева, которые содержат информацию о состоянии, а также о пути, по которому было достигнуто это состояние. Узлы могут также содержать дополнительные данные, такие как стоимость или глубина.

18. Что такое функция преемника и как она используется в алгоритме поиска? Функция преемника — это функция, которая генерирует все возможные состояния (дочерние узлы) из текущего состояния. Она используется в алгоритме поиска для расширения узлов, позволяя исследовать все возможные пути, которые могут быть предприняты из данного состояния.

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

1) b (разветвление): чем больше значение b, тем больше узлов необходимо исследовать на каждом уровне, что увеличивает время и ресурсы, необходимые для поиска;

2) d (глубина решения): глубина решения определяет, насколько далеко нужно углубляться в дерево, чтобы найти решение. Большая глубина может привести к увеличению времени поиска;

3) m (максимальная глубина): максимальная глубина ограничивает, насколько глубоко алгоритм может исследовать дерево, что может предотвратить исчерпание ресурсов, но также может привести к пропуску решения, если оно находится глубже, чем m.

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

Полнота: Алгоритм считается полным, если он гарантированно находит решение, если оно существует.

Временная сложность: оценивает, сколько времени потребуется алгоритму для нахождения решения, обычно выражается в зависимости от b и d.

Пространственная сложность: оценивает, сколько памяти потребуется для хранения узлов, что также зависит от b и d.

Оптимальность: Алгоритм считается оптимальным, если он находит лучшее (наименее затратное) решение среди всех возможных.

21. Какую роль выполняет класс в приведенном коде? Problem

Класс Problem служит базовым классом для определения конкретных задач поиска. Он задает структуру и интерфейс, которые должны быть реализованы в подклассах для решения различных задач.

22. Какие методы необходимо переопределить при наследовании класса Problem? При наследовании класса Problem необходимо переопределить методы actions, result, и, возможно, h, чтобы определить конкретные действия, результаты этих действий и эвристическую функцию для данной задачи.

23. Что делает метод is_goal в классе Problem? Метод is_goal проверяет, достигнуто ли целевое состояние, сравнивая текущее состояние с заданным целевым состоянием.

24. Для чего используется метод action_cost в классе Problem? Метод action_cost используется для определения стоимости выполнения действия, что позволяет учитывать затраты при поиске решения.

25. Какую задачу выполняет класс Node в алгоритмах поиска? Класс Node представляет состояние в пространстве поиска и хранит информацию о текущем состоянии, родительском узле, выполненном действии и стоимости пути до этого состояния.

26. Какие параметры принимает конструктор класса Node? Конструктор класса Node принимает параметры: state (текущее состояние), parent (родительский узел), action (действие, приведшее к этому состоянию) и path_cost (стоимость пути до этого состояния).

27. Что представляет собой специальный узел failure? Специальный узел failure представляет собой состояние, указывающее на неудачу в поиске решения, с бесконечной стоимостью пути.

28. Для чего используется функция expand в коде? Функция expand генерирует дочерние узлы для данного узла, основываясь на возможных действиях, которые можно выполнить из текущего состояния.

29. Какая последовательность действий генерируется с помощью функции path_actions? Функция path_actions генерирует последовательность действий, которые были выполнены для достижения данного узла, начиная с корневого узла.

30. Чем отличается функция path_states от функции path_actions? Функция path_states возвращает последовательность состояний, через которые прошел узел, в то время как path_actions возвращает последовательность действий, которые были выполнены для достижения этих состояний.

31. Какой тип данных используется для реализации FIFOQueue? Для реализации FIFOQueue используется тип данных deque, который обеспечивает эффективные операции добавления и удаления элементов с обоих концов.

32. Чем отличается очередь FIFOQueue от LIFOQueue? Очередь FIFOQueue (First In, First Out) обрабатывает элементы в порядке их поступления, тогда как LIFOQueue (Last In, First Out) обрабатывает элементы в обратном порядке, т.е. последний добавленный элемент будет первым, который будет удален.

33. Как работает метод add в классе PriorityQueue? Метод add в классе PriorityQueue добавляет элемент в очередь, создавая пару, состоящую из приоритета элемента и самого элемента, и помещает эту пару в кучу с помощью heapq.heappush.

34. В каких ситуациях применяются очереди с приоритетом? Очереди с приоритетом применяются в ситуациях, когда необходимо обрабатывать элементы не по порядку их поступления, а в зависимости от их важности или приоритета, например, в алгоритмах планирования задач или в графовых алгоритмах.

35. Как функция heappop помогает в реализации очереди с приоритетом? Функция heappop извлекает элемент с наивысшим приоритетом (наименьшим значением ключа) из кучи, что позволяет эффективно управлять очередью с приоритетом, обеспечивая быструю обработку элементов.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с методами поиска в пространстве состояний с помощью языка Python версии 3.x.