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»

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

_____
(подпись)

Руководитель практики:
Воронкин Р. А., доцент департамента цифровых и робототехнических систем и электроники института перспективной инженерии

_____
(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.

Репозиторий:https://github.com/Artorias1469/Artificial_Intelligence_laboratory_work_1.git

Ход работы:

1. Выполню пример реализации краёв и узлов в коде, которые нам понадобятся при создании алгоритма поиска.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# разберем как край (fringe) и узел (node) могут быть реализованы в
# коде, они нам понадобятся когда мы будем реализовывать алгоритмы поиска

import heapq
import math
from collections import deque


class Problem:
    """Абстрактный класс для формальной задачи. Новый домен
    специализирует этот класс,
    переопределяя `actions` и `results`, и, возможно, другие методы.
    Эвристика по умолчанию равна 0, а стоимость действия по умолчанию
    равна 1 для всех состояний.
    Когда вы создаете экземпляр подкласса, укажите `начальное` и
    `целевое` состояния
    (или задайте метод `is_goal`) и, возможно, другие ключевые слова для
    подкласса."""


def __init__(self, initial=None, goal=None, **kwds):
    self.__dict__.update(initial=initial, goal=goal, **kwds)


def actions(self, state):  1 usage (1 dynamic)
    raise NotImplementedError


def result(self, state, action):  1 usage (1 dynamic)
```

Рис 1. Выполнение примера

2. Воспользовавшись сервисом Google карты, выбрал на карте более 20 населённых пунктов, связанных между собой дорогами. Построим граф, где узлы будут представлять населённые пункты, а рёбра — дороги, соединяющие их. Вес каждого ребра соответствует расстоянию между этими пунктами. Выберем начальный и конечный пункты на графе. Определим минимальный маршрут между ними, который должен проходить через три промежуточных населённых пункта. Пусть начальный пункт будет Париж, а конечный Дрезден.

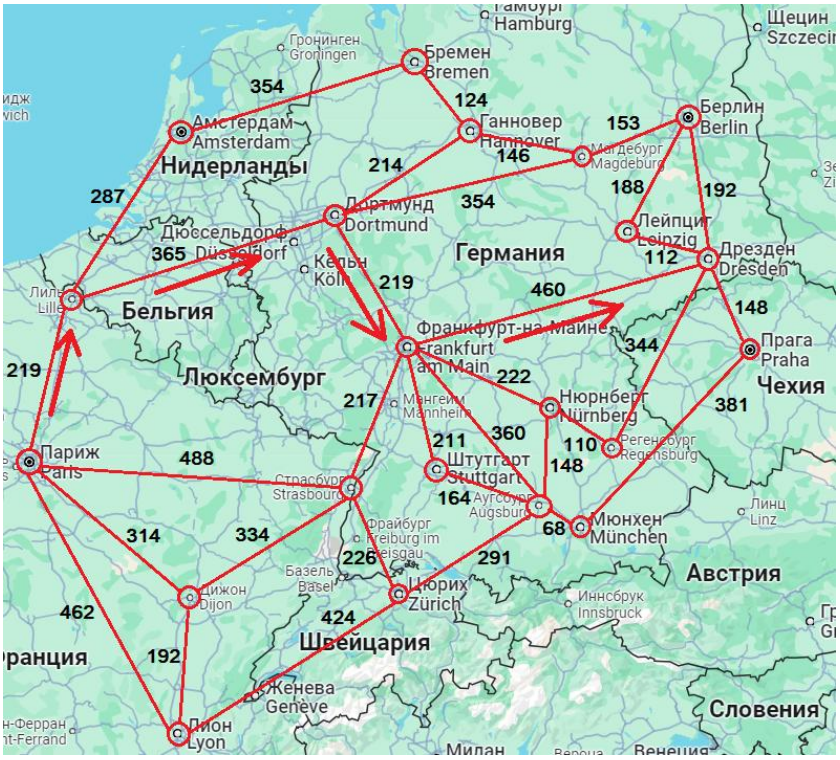

Рисунок 2. Граф и поиск кратчайшего маршрута

Всего на данном графе получилось определить два подходящих маршрута, но самым коротким оказался маршрут длинной 1263 км.

Для решения задачи коммивояжёра был применен метода полного перебора. Поскольку при достаточно большом количестве населенных пунктов время решения задачи увеличивается до непозволительного, количество узлов и ребер было решено уменьшить до 11.

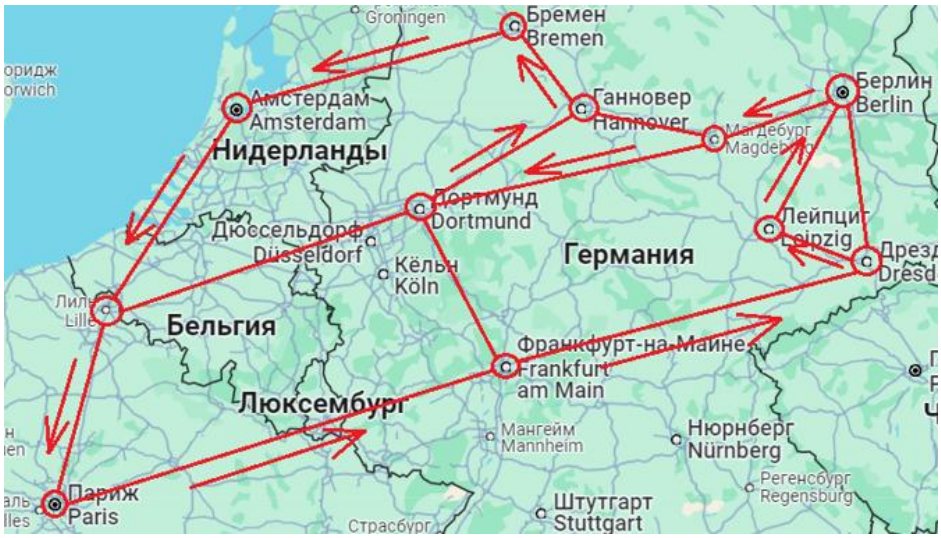

Рис 3. Урезанный граф

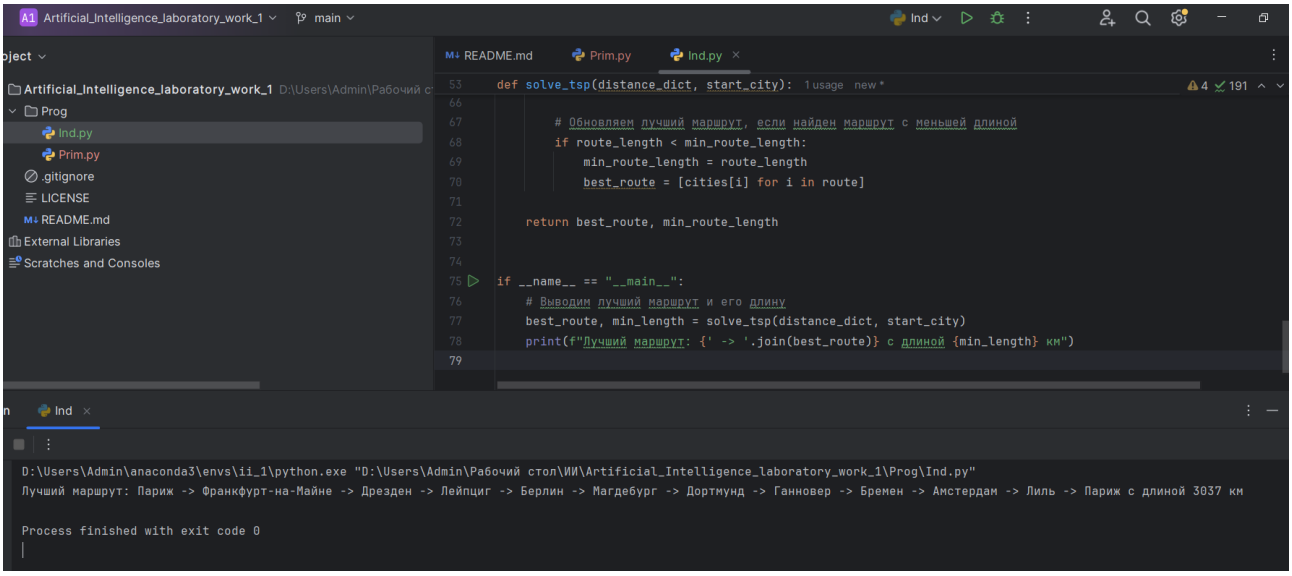

Рис 4. Результат работы индивидуальной программы

Ответы на контрольные вопросы:

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

Метод "слепого поиска" в ИИ — это метод, который не использует дополнительную информацию о целевом состоянии для оптимизации поиска. Примеры включают поиск в ширину и в глубину, где каждый узел проверяется без учета его близости к цели.

2. Как отличается эвристический поиск от слепого поиска?

Эвристический поиск отличается тем, что использует эвристику — оценочную функцию, которая позволяет алгоритму более эффективно выбирать, какие узлы проверять, основываясь на предполагаемом расстоянии до цели. Эвристический поиск может находить путь быстрее, чем слепой поиск.

3. Какую роль играет эвристика в процессе поиска?

Эвристика играет роль оценки "расстояния" или "стоимости" от текущего состояния до целевого, помогая алгоритму сосредоточиться на более перспективных путях и, таким образом, ускоряя процесс поиска.

4. Приведите пример применения эвристического поиска в реальной задаче.

Пример применения эвристического поиска — навигация GPS, где алгоритм находит кратчайший или оптимальный маршрут на основе данных о расстояниях и текущей дорожной обстановке.

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, что делает перебор всех ходов непрактичным даже для мощных компьютеров.

6. Какие факторы ограничивают создание идеального шахматного ИИ?

Факторы, ограничивающие создание идеального шахматного ИИ, включают ограничения вычислительных мощностей и времени, а также сложность оценки позиций, где требуется глубина понимания, приближенная к человеческой интуиции.

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

Основная задача ИИ при выборе ходов в шахматах — это определение оптимальных ходов, которые приводят к преимуществу в позиции, с учетом ограничений времени и вычислительных ресурсов.

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

Балансировка между скоростью вычислений и нахождением оптимальных решений достигается за счет эвристик и ограничений глубины поиска, которые позволяют ИИ быстро принимать приемлемые, но не всегда идеальные решения.

9. Каковы основные элементы задачи поиска маршрута по карте?

Основные элементы задачи поиска маршрута по карте — это начальная точка, целевая точка, граф дорог (или маршрутов), а также возможные ограничения, такие как расстояние и доступность дорог.

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

Оптимальность решения задачи маршрутизации на карте Румынии оценивается по следующим критериям:

Длина маршрута — кратчайшее расстояние между начальным и конечным пунктами.

Стоимость маршрута — учет дополнительных факторов, таких как время и расход топлива.

Эвристическая оценка — использование функции, которая помогает находить путь быстрее и ближе к оптимальному.

Эти критерии позволяют выбрать наилучший маршрут в зависимости от задачи.

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

Исходное состояние дерева поиска в задаче маршрутизации — это город, из которого начинается поиск, например, Арад, если цель — найти путь в Бухарест.

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

Листовые узлы — это узлы, не имеющие потомков в дереве, что означает, что для них больше нет доступных действий для расширения.

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

Этап расширения узла включает создание дочерних узлов, представляющих возможные состояния, достижимые из текущего узла с помощью доступных действий.

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

Города, которые можно посетить из Арада за одно действие: Зеринд, Сибиу и Тимишоара.

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

Целевое состояние в алгоритме поиска определяется как состояние, в котором выполнено условие задачи, например, достижение города Бухарест.

16. Какие основные шаги выполняет алгоритм поиска по дереву?

Основные шаги алгоритма поиска по дереву включают выбор узла для расширения, проверку, является ли он целевым состоянием, и, если нет, создание дочерних узлов.

17. Чем различаются состояния и узлы в дереве поиска?

Состояния и узлы различаются тем, что узел — это представление состояния в дереве поиска, которое включает дополнительную информацию, такую как путь и затраты.

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

Функция преемника генерирует возможные состояния, достижимые из текущего состояния, и используется для расширения узлов в дереве.

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

Параметры b (разветвление), d (глубина решения) и m (максимальная глубина) влияют на время и ресурсы, необходимые для поиска. Увеличение b и d значительно увеличивает количество узлов для проверки.

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

Алгоритмы поиска оцениваются по критериям полноты (способности найти решение, если оно существует), временной и пространственной сложности, а также оптимальности (нахождения наилучшего решения).

21. Какую роль выполняет класс Problem в приведенном коде?

Класс Problem в приведенном коде задает структуру задачи поиска, включая начальное состояние, целевое состояние и методы для выполнения действий и проверки цели.

22. Какие методы необходимо переопределить при наследовании класса Problem?

Методы, которые нужно переопределить в классе Problem при наследовании: actions, result, возможно, is_goal и action_cost.

23. Что делает метод is_goal в классе Problem?

Метод is_goal в классе Problem проверяет, достигнуто ли целевое состояние для текущего состояния.

24. Для чего используется метод action_cost в классе Problem?

Метод action_cost возвращает стоимость выполнения действия между

состояниями и используется для расчета общей стоимости пути.

25. Какую задачу выполняет класс Node в алгоритмах поиска?

Класс Node представляет узел в дереве поиска и хранит информацию о состоянии, родителе, действии и стоимости пути.

26. Какие параметры принимает конструктор класса Node?

Конструктор класса Node принимает параметры: state (состояние узла), parent (родительский узел), action (действие, которое привело к узлу), path_cost (стоимость пути до узла).

27. Что представляет собой специальный узел failure?

Специальный узел failure указывает, что алгоритм не смог найти решение для задачи.

28. Для чего используется функция expand в коде?

Функция expand создает дочерние узлы для текущего узла, основываясь на возможных действиях и результатах.

29. Какая последовательность действий генерируется с помощью функции path_actions?

Функция path_actions генерирует последовательность действий для достижения данного узла от начального состояния.

30. Чем отличается функция path_states от функции path_actions?

Функция path_states возвращает последовательность состояний, в то время как path_actions — последовательность действий.

31. Какой тип данных используется для реализации FIFOQueue?

Тип данных для реализации FIFOQueue — это deque из библиотеки collections, обеспечивающая быструю вставку и удаление с обоих концов.

32. Чем отличается очередь FIFOQueue от LIFOQueue?

Очередь FIFOQueue (первый пришел, первый ушел) работает по принципу очереди, в то время как LIFOQueue (последний пришел, первый ушел) — по принципу стека.

33. Как работает метод add в классе PriorityQueue?

Метод add в классе PriorityQueue добавляет элемент с приоритетом, чтобы обеспечить доступ к элементам с наименьшим значением приоритетной функции.

34. В каких ситуациях применяются очереди с приоритетом?

Очереди с приоритетом используются, когда элементы должны обрабатываться в порядке важности, например, в алгоритмах поиска по наименьшей стоимости.

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

Функция heappop извлекает элемент с наименьшим значением приоритетной функции и помогает в реализации очереди с приоритетом с минимальным значением на вершине.

Вывод: приобрел навыки по работе с методами поиска в пространстве состояний с помощью языка программирования Python.