

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 №4

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 поиска с ограничением глубины.

Цель: приобретение навыков по работе с поиском с ограничением глубины с помощью языка программирования Python версии 3.x

Ссылка на репозиторий:

https://github.com/Artorias1469/Artificial_Intelligence_laboratory_work_4.git

Ход работы:

Задание 1. Проработка примеров лабораторной работы

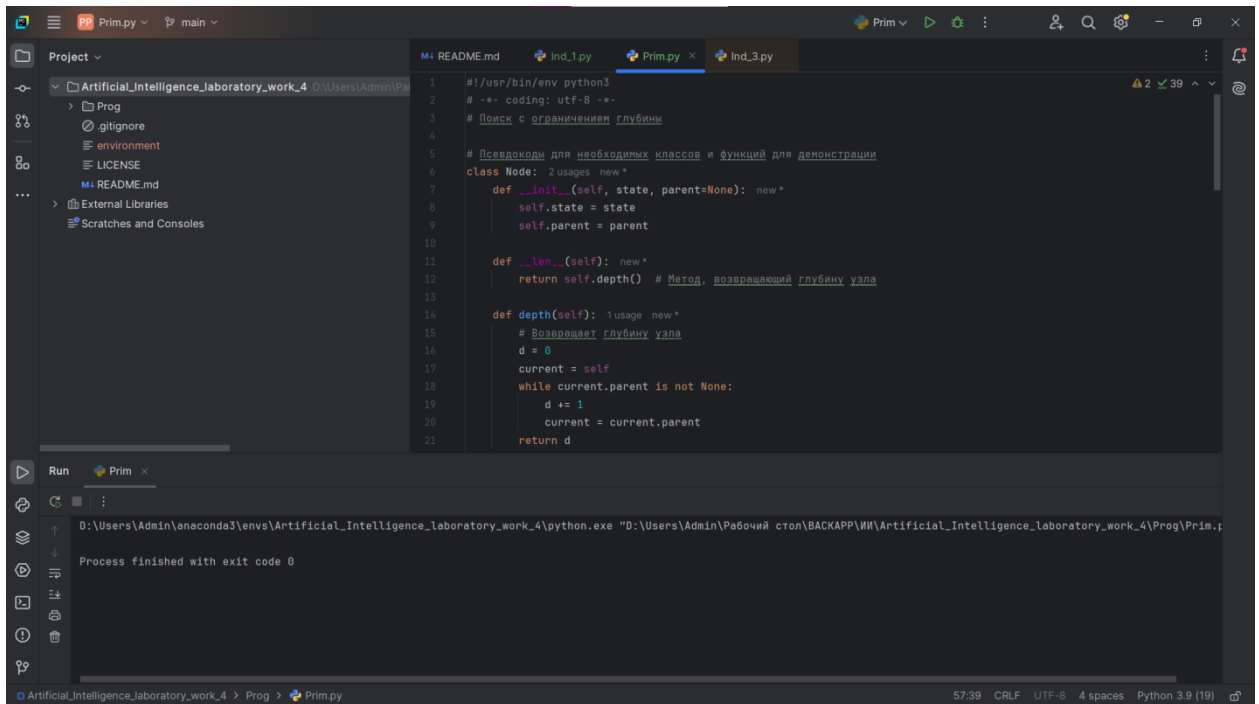


Рисунок 1. Пример и его результат выполнения

Задание 2. Система навигации робота-пылесоса

Вы работаете над разработкой системы навигации для робота-пылесоса. Робот способен передвигаться по различным комнатам в доме, но из-за ограниченности ресурсов (например, заряда батареи) и времени на уборку, важно эффективно выбирать путь. Ваша задача - реализовать алгоритм, который поможет роботу определить, существует ли путь к целевой комнате, не превышая заданное ограничение по глубине поиска.

Дано дерево, где каждый узел представляет собой комнату в доме. Узлы связаны в соответствии с возможностью перемещения робота из одной комнаты в другую.

Необходимо определить, существует ли путь от начальной комнаты (корень дерева) к целевой комнате (узел с заданным значением), так, чтобы робот не превысил лимит по глубине перемещения.

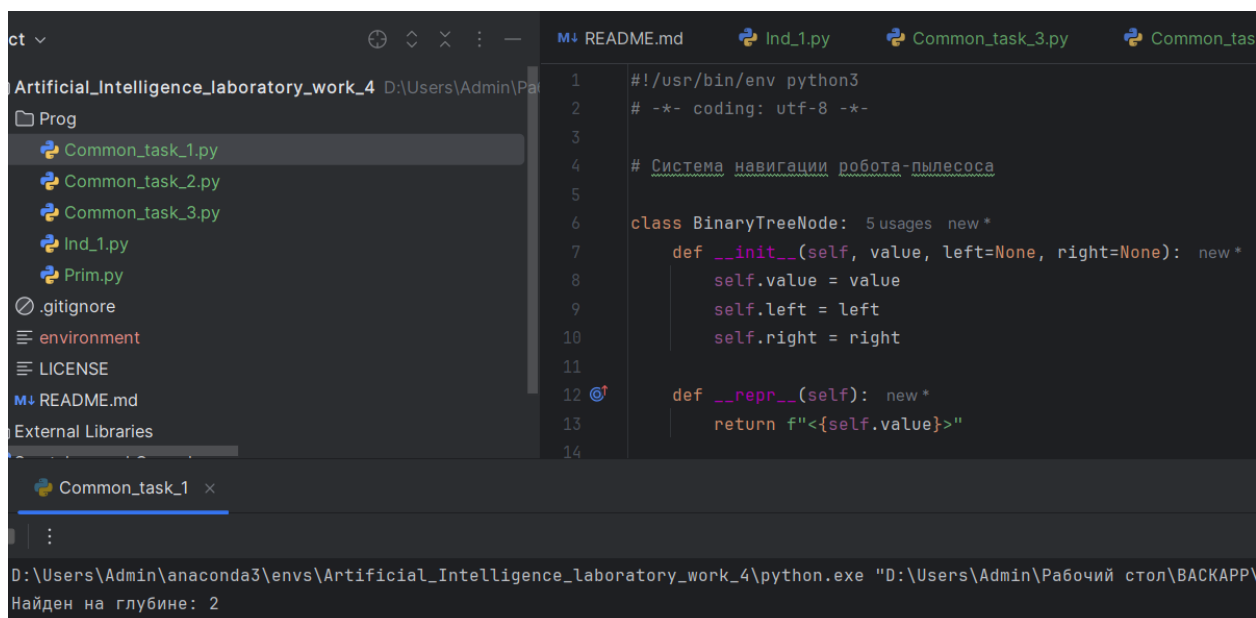


Рисунок 2. Результат выполнения первого общего задания

Задание 3. Система управления складом

Представьте, что вы разрабатываете систему для управления складом, где товары упорядочены в структуре, похожей на двоичное дерево. Каждый узел дерева представляет место хранения, которое может вести к другим местам хранения (левому и правому подразделу). Ваша задача — найти наименее затратный путь к товару, ограничив поиск заданной глубиной, чтобы гарантировать, что поиск займет приемлемое время.

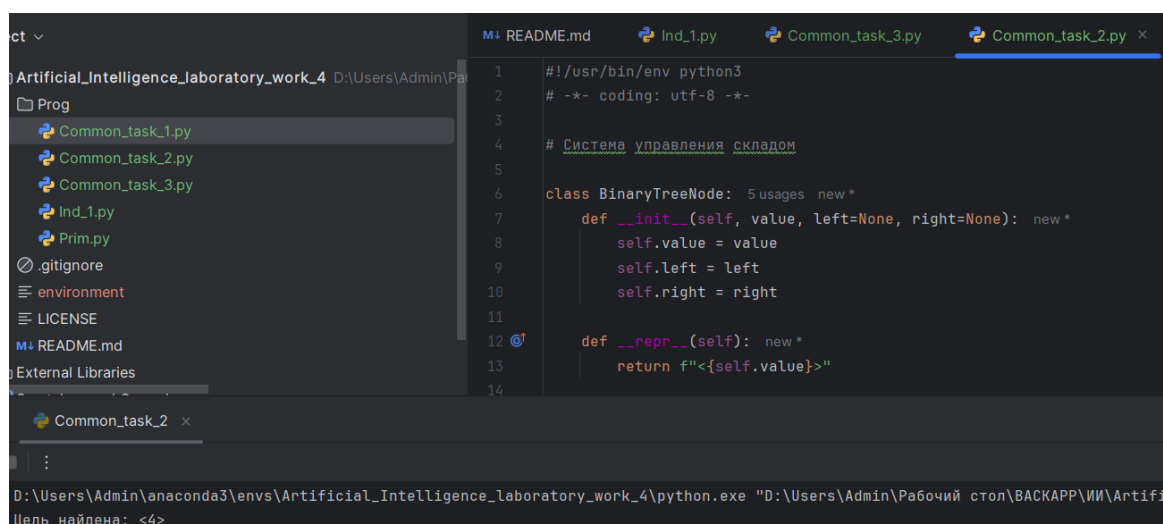
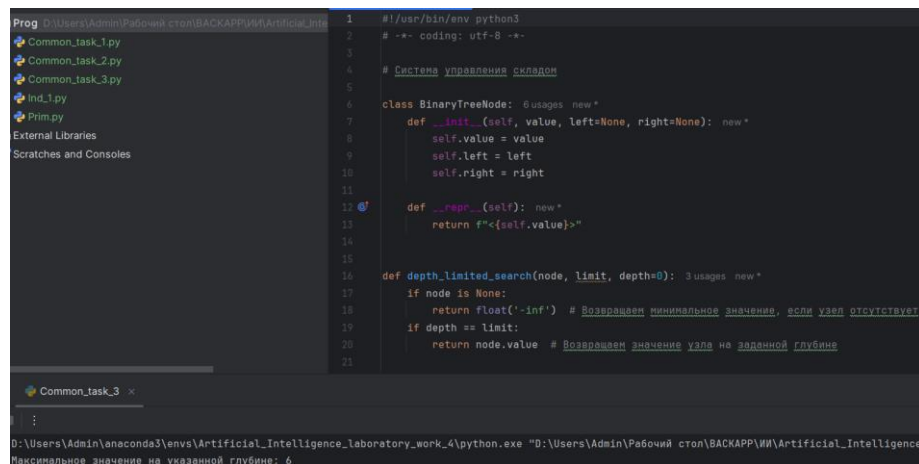


Рисунок 3. Результат выполнения второго общего задания

Задание 4. Система автоматического управления инвестициями

Представьте, что вы разрабатываете систему для автоматического управления инвестициями, где дерево решений используется для представления последовательности инвестиционных решений и их потенциальных исходов. Цель состоит в том, чтобы найти наилучший исход (максимальную прибыль) на

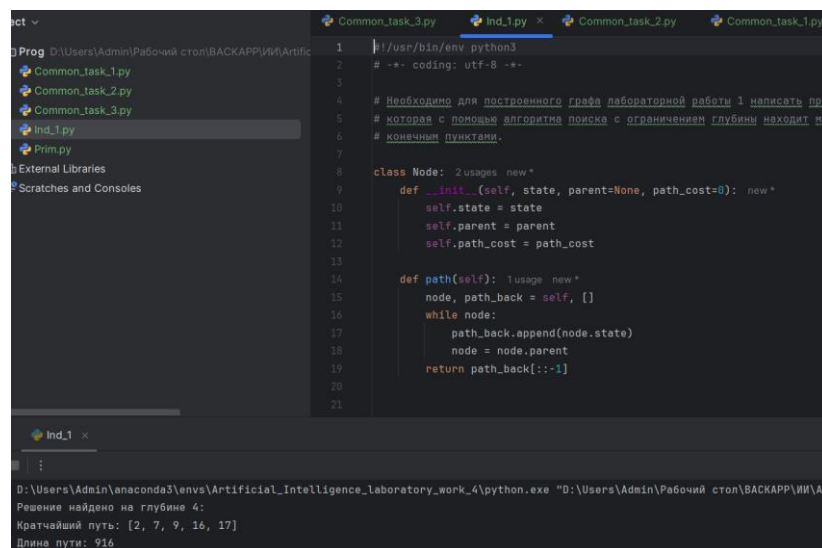
определённой глубине принятия решений, учитывая ограниченные ресурсы и время на анализ.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # Система управления складом
5
6 class BinaryTreeNode: 6 usages new *
7     def __init__(self, value, left=None, right=None): new *
8         self.value = value
9         self.left = left
10        self.right = right
11
12    def __repr__(self): new *
13        return f"<{self.value}>"
14
15
16 def depth_limited_search(node, limit, depth=0): 3 usages new *
17     if node is None:
18         return float('-inf') # Возвращаем минимальное значение, если узел отсутствует
19     if depth == limit:
20         return node.value # Возвращаем значение узла на заданной глубине
21
```

Рисунок 4. Результат выполнения третьего общего задания

Задание 5. напишите программу на языке программирования Python, которая с помощью алгоритма поиска с ограничением глубины находит минимальное расстояние между начальным и конечным пунктами. Определите глубину дерева поиска, на которой будет найдено решение. Сравните найденное решение с решением, полученным вручную.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # Необходимо для построения графа лабораторной работы 1 написать про
5 # которая с помощью алгоритма поиска с ограничением глубины находит ма
6 # конечным пунктами.
7
8 class Node: 2 usages new *
9     def __init__(self, state, parent=None, path_cost=0): new *
10         self.state = state
11         self.parent = parent
12         self.path_cost = path_cost
13
14     def path(self): 1 usage new *
15         node, path_back = self, []
16         while node:
17             path_back.append(node.state)
18             node = node.parent
19         return path_back[::-1]
20
21
```

Рисунок 5. Результат выполнения индивидуального задания

Ответы на контрольные вопросы:

1. Что такое поиск с ограничением глубины, и как он решает проблему бесконечных ветвей?

Поиск с ограничением глубины (Depth-Limited Search, DLS) — это модификация поиска в глубину, которая ограничивает глубину рекурсии до заданного предела (limit). Узлы, которые находятся на уровне глубже заданного предела, не исследуются.

Как решает проблему бесконечных ветвей:

В случае бесконечных графов поиск в глубину может уйти в бесконечную рекурсию, так как он продолжает углубляться. Поиск с ограничением глубины останавливается, как только достигнута заданная глубина, предотвращая заикливание.

2. Какова основная цель ограничения глубины в данном методе поиска?

Основная цель ограничения глубины — ограничить объем работы поиска, избегая бесконечных рекурсий или исследования ненужных частей дерева. Это особенно важно в графах с бесконечными или очень глубокими ветвями.

3. В чем разница между поиском в глубину и поиском с ограничением глубины?

Поиск в глубину углубляется до самого нижнего уровня дерева или графа, что может привести к заикливанию в бесконечных графах.

Поиск с ограничением глубины ограничивает глубину поиска заданным значением limit, предотвращая исследование узлов, которые находятся глубже этого уровня.

4. Какую роль играет проверка глубины узла в псевдокоде поиска с ограничением глубины?

Проверка глубины узла определяет, следует ли продолжать исследование текущей ветви. Если глубина узла достигает значения limit, дальнейшее исследование прекращается, чтобы не нарушить ограничение.

5. Почему в случае достижения лимита глубины функция возвращает «обрезание»?

Когда достигается лимит глубины, алгоритм возвращает результат

"обрезание" (cutoff), чтобы сигнализировать, что узел на этой ветви находится на пределе глубины и не может быть исследован дальше. Это помогает алгоритму сообщить, что в текущей ветви может находиться решение, но его невозможно проверить на данном уровне.

6. В каких случаях поиск с ограничением глубины может не найти решение, даже если оно существует?

Поиск с ограничением глубины не найдет решение, если:

- Решение находится на глубине, превышающей заданный лимит.
- Путь к решению слишком длинный, и алгоритм прекращает углубление до его нахождения.

7. Как поиск в ширину и в глубину отличаются при реализации с использованием очереди?

Поиск в ширину (BFS): использует очередь FIFO (first-in, first-out), добавляя узлы в конец очереди и извлекая из начала. Это обеспечивает уровень за уровнем исследование.

Поиск в глубину (DFS): использует стек LIFO (last-in, first-out), добавляя узлы в начало структуры данных, что позволяет углубляться по одной ветви.

8. Почему поиск с ограничением глубины не является оптимальным?

Поиск с ограничением глубины не гарантирует нахождение кратчайшего пути до цели, так как он прекращает исследование ветвей, которые превышают установленный лимит. Если решение находится на большой глубине, оно не будет найдено.

9. Как итеративное углубление улучшает стандартный поиск с ограничением глубины?

Итеративное углубление (Iterative Deepening Depth-First Search, IDDFS):

- Выполняет поиск с ограничением глубины для увеличивающегося лимита.
- На каждом шаге лимит увеличивается на единицу.
- Это позволяет находить решения на минимальной глубине, подобно поиску в ширину, при этом используя меньшую память (как в поиске в глубину).

10. В каких случаях итеративное углубление становится эффективнее простого поиска в ширину?

Итеративное углубление становится эффективнее, когда:

- Пространство состояний очень большое или бесконечное.
- Глубина целевого узла мала по сравнению с размером пространства состояний.
- Ограничение памяти является критическим фактором, так как итеративное углубление использует память, эквивалентную поиску в глубину (не хранит все узлы).

11. Какова основная цель использования алгоритма поиска с ограничением глубины?

Алгоритм поиска с ограничением глубины предотвращает заикливание в бесконечных пространствах состояний, ограничивая глубину поиска заданным параметром `limit`. Это помогает эффективно исследовать пространство состояний до фиксированной глубины.

12. Какие параметры принимает функция `depth_limited_search`, и каково их назначение?

Функция обычно принимает следующие параметры:

- `problem`: описание задачи, содержащей начальное состояние, операторы, функции проверки цели и т.д.
- `limit`: максимальная глубина поиска, которая предотвращает заикливание и неконтролируемое углубление.

13. Какое значение по умолчанию имеет параметр `limit` в функции `depth_limited_search`?

Значение по умолчанию зависит от реализации. Часто значение не задается явно, и пользователь обязан указать его, или же используется большой фиксированный предел.

14. Что представляет собой переменная `frontier`, и как она используется в алгоритме?

`frontier` представляет собой структуру данных, хранящую узлы, которые нужно исследовать. В поиске с ограничением глубины это LIFO-очередь (стек), которая обеспечивает поведение поиска в глубину.

15. Какую структуру данных представляет LIFOQueue, и почему она используется в этом алгоритме?

LIFOQueue — это стек, реализованный на базе принципа "последним вошел — первым вышел". Он используется, чтобы исследовать узлы в порядке обратного хода (углубляться в дочерние узлы перед возвратом к родительским).

16. Каково значение переменной result при инициализации, и что оно означает?

Обычно result инициализируется как None, failure или другое значение, указывающее, что целевой узел пока не найден.

17. Какое условие завершает цикл while в алгоритме поиска?

Цикл завершается, когда:

- frontier становится пустым (все возможные узлы исследованы).
- Целевой узел найден.

18. Какой узел извлекается с помощью frontier.pop() и почему?

С помощью frontier.pop() извлекается последний добавленный узел (верхний элемент стека). Это обеспечивает углубление поиска, следуя стратегии "глубина сначала".

19. Что происходит, если найден узел, удовлетворяющий условию цели (условие problem.is_goal(node.state))?

Алгоритм немедленно завершает работу и возвращает найденный узел как решение.

20. Какую проверку выполняет условие elif len(node) >= limit, и что означает его выполнение?

Условие проверяет, достиг ли текущий узел максимальной глубины, определенной параметром limit. Если достиг, узел больше не расширяется, чтобы предотвратить заикливание или избыточное углубление.

21. Что произойдет, если текущий узел достигнет ограничения по глубине поиска?

Алгоритм прекращает расширение этого узла. Обычно возвращается

результат `cutoff`, чтобы показать, что поиск был прерван из-за ограничения глубины.

22. Какую роль выполняет проверка на циклы `elif not is_cycle(node)` в алгоритме?

Она предотвращает повторное исследование уже пройденных узлов в текущем пути, исключая заикливание.

23. Что происходит с дочерними узлами, полученными с помощью функции `expand(problem, node)`?

Дочерние узлы добавляются в `frontier` для дальнейшего исследования, если они соответствуют условиям (например, не достигли предела глубины и не образуют цикл).

24. Какое значение возвращается функцией, если целевой узел не был найден?

Если целевой узел не найден, возвращается `failure`, что означает отсутствие решения в рамках заданного ограничения глубины.

25. В чем разница между результатами `failure` и `cutoff` в контексте данного алгоритма?

- `failure`: целевой узел не найден, и поиск завершен.
- `cutoff`: поиск был прерван из-за достижения ограничения глубины, возможно, целевой узел находится на большей глубине.

Вывод: приобрел навыки по работе с поиском с ограничением глубины с помощью языка программирования Python версии 3.x