

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:

Евдаков Евгений Владимирович

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

### **Ход работы:**

**Задание 1.** Создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер. Организовал свой репозиторий в соответствие с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\evdak>git clone https://github.com/EvgenyEvdakov/Laba_ii_4.git
Cloning into 'Laba_ii_4'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 1. Клонирование репозитория

**Задание 2.** Создал виртуальное окружение conda и активировал его, также установил необходимые пакеты isort, black, flake8.

```
(base) PS C:\Users\evdak> cd C:\Users\evdak\Laba_ii_4
(base) PS C:\Users\evdak\Laba_ii_4> conda create -n ii_4 python=3.10
Retrieving notices: ...working... done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\evdak\.conda\envs\ii_4

  added / updated specs:
    - python=3.10

The following NEW packages will be INSTALLED:

  bzip2                pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6
  ca-certificates      pkgs/main/win-64::ca-certificates-2024.9.24-haa95532_0
  libffi               pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
  openssl              pkgs/main/win-64::openssl-3.0.15-h827c3e9_0
  pip                  pkgs/main/win-64::pip-24.2-py310haa95532_0
  python               pkgs/main/win-64::python-3.10.15-h4607a30_1
  setuptools           pkgs/main/win-64::setuptools-75.1.0-py310haa95532_0
  sqlite               pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  tk                   pkgs/main/win-64::tk-8.6.14-h0416ee5_0
  tzdata               pkgs/main/noarch::tzdata-2024b-h04d1e81_0
```

Рисунок 2. Создание виртуального окружения

**Задание 3.** Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл `primer.py`. Рассмотрим реализацию алгоритма поиска с ограничением глубины на практике, в программном коде:

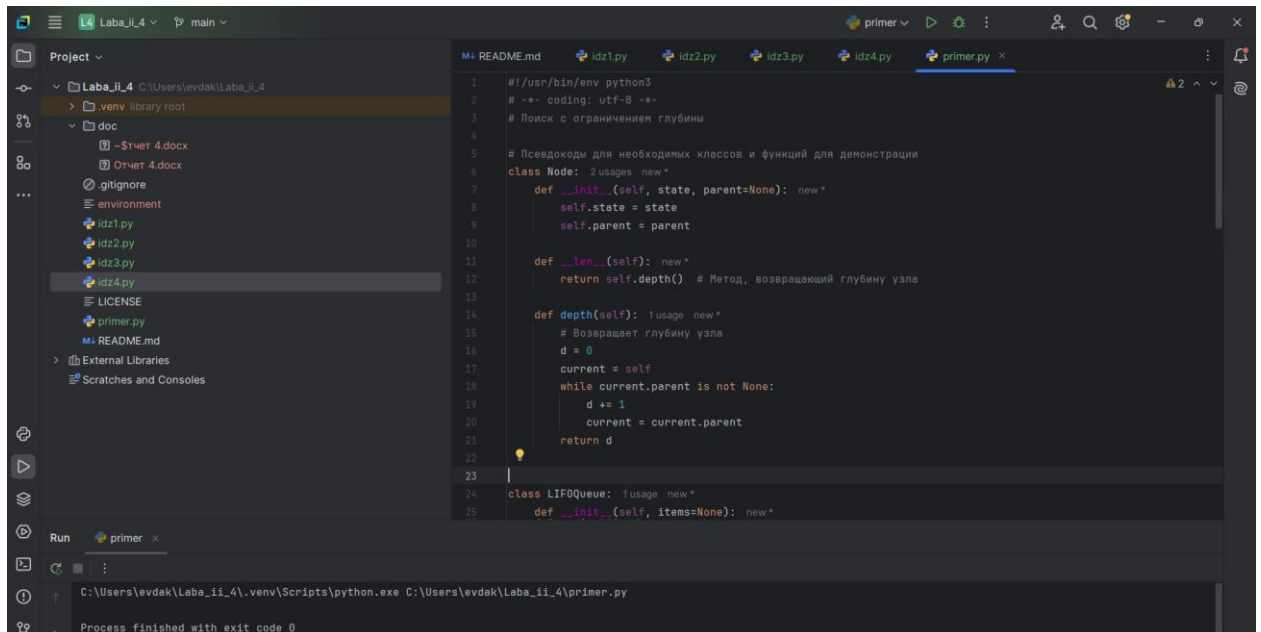


Рисунок 3. Выполнение примера

**Задание 4.** Необходимо реализовать систему навигации 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. Необходимо 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.

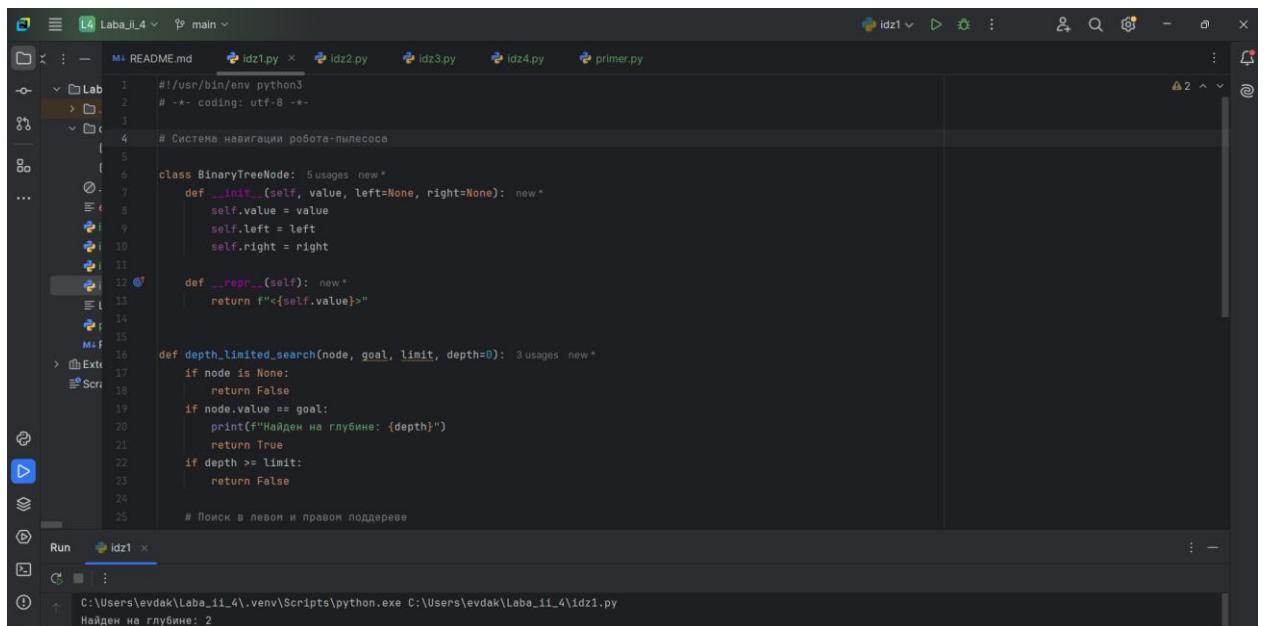


Рисунок 4. Реализация программы для системы навигации робота-пылесоса

**Задание 5.** Необходимо реализовать систему управления складом. Каждый узел дерева представляет место хранения, которое может вести к другим местам хранения (левому и правому подразделу). Необходимо найти наименее затратный путь к товару, ограничив поиск заданной глубиной, чтобы гарантировать, что поиск займет приемлемое время.

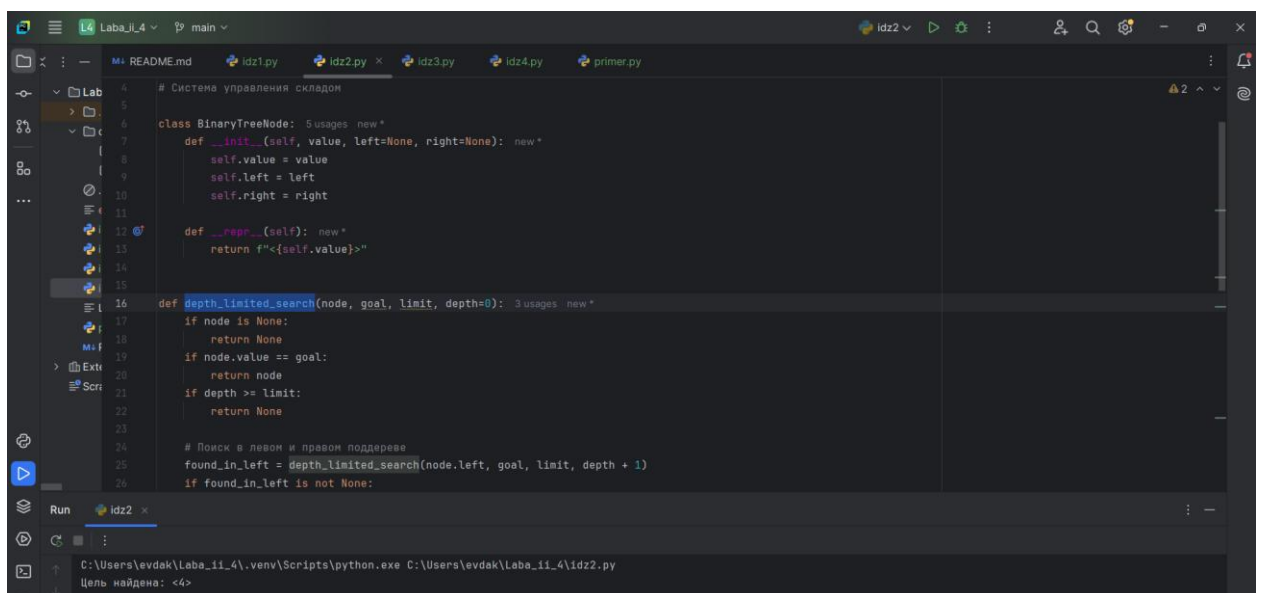


Рисунок 5. Реализация программы для системы управления складом

**Задание 6.** Необходимо реализовать систему автоматического управления инвестициями. 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: @staticmethod 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): @staticmethod 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
22     left_value = depth_limited_search(node.left, limit, depth + 1)
23     right_value = depth_limited_search(node.right, limit, depth + 1)

```

Run idz3

C:\Users\evdak\Lab11\_4\venv\Scripts\python.exe C:\Users\evdak\Lab11\_4\idz3.py  
Максимальное значение на указанной глубине: 6

Рисунок 6. Реализация программы для системы автоматического управления инвестициями

**Задание 7.** Необходимо для построенного графа лабораторной работы 1 написать программу на языке программирования Python, которая с помощью алгоритма поиска с ограничением глубины находит минимальное расстояние между начальным и конечным пунктами. И так сравним найденное решение с решением, полученным вручную.

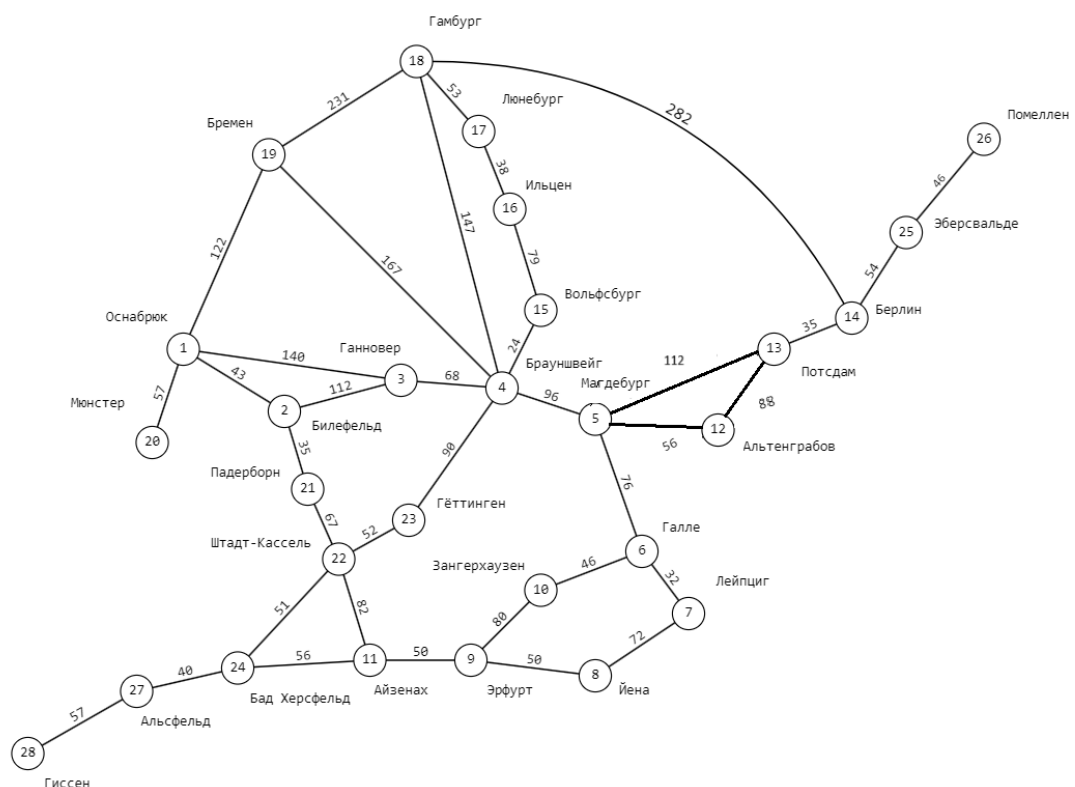
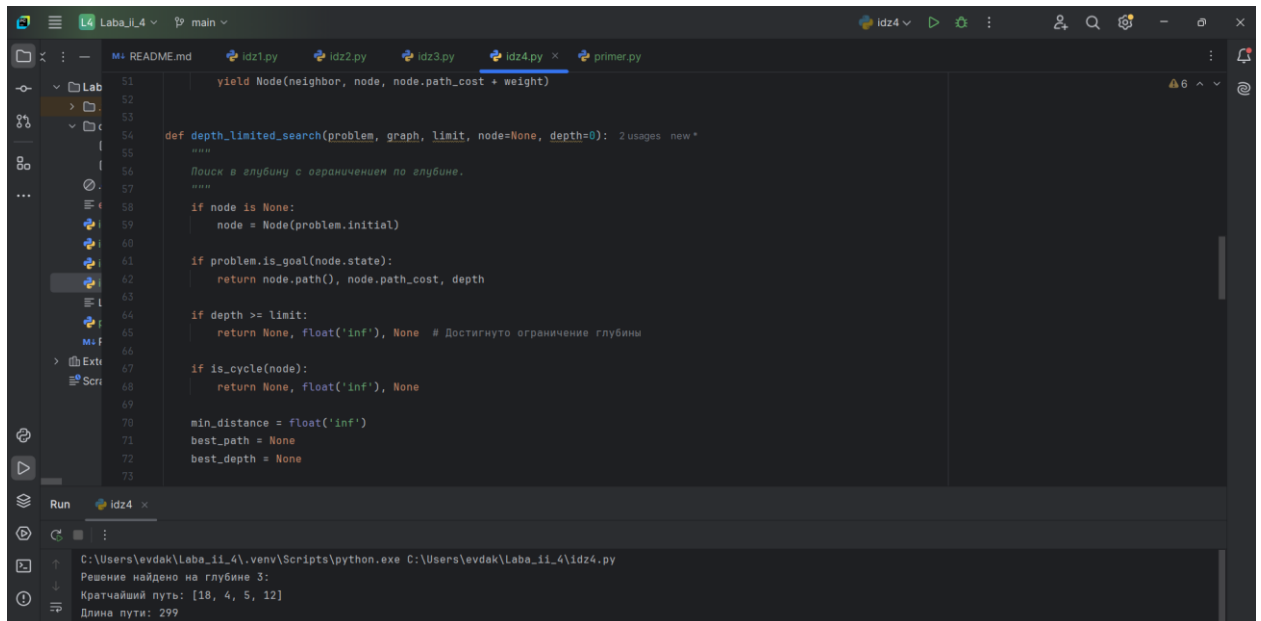


Рисунок 7. Построение графа

Найдем минимальное расстояние между городами Гамбург и Альтенграбов. Если считать вручную, то минимальное расстояние составляет 299 км. Далее составим программу и проверим:



```
51 yield Node(neighbor, node, node.path_cost + weight)
52
53
54 def depth_limited_search(problem, graph, limit, node=None, depth=0):
55     """
56     Поиск в глубину с ограничением по глубине.
57     """
58     if node is None:
59         node = Node(problem.initial)
60
61     if problem.is_goal(node.state):
62         return node.path(), node.path_cost, depth
63
64     if depth >= limit:
65         return None, float('inf'), None # Достигнуто ограничение глубины
66
67     if is_cycle(node):
68         return None, float('inf'), None
69
70     min_distance = float('inf')
71     best_path = None
72     best_depth = None
73
```

Run idz4 x

C:\Users\evdak\Laba\_ii\_4\venv\Scripts\python.exe C:\Users\evdak\Laba\_ii\_4\idz4.py

Решение найдено на глубине 3:  
Кратчайший путь: [18, 4, 5, 12]  
Длина пути: 299

Рисунок 8. Выполнение программы

Результат программы вывел так же 299 км.

### Задание 7.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер. Создал файл environment.yml и деактивировал виртуальное окружение.

```
(ii_4) PS C:\Users\evdak\Laba_ii_4> conda env export > environment
(ii_4) PS C:\Users\evdak\Laba_ii_4> conda deactivate
```

Рисунок 9. Деактивация ВО

Ссылка: [https://github.com/EvgenyEvdakov/Laba\\_ii\\_4](https://github.com/EvgenyEvdakov/Laba_ii_4)

### 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