

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»

Вариант ____

Выполнила:
Маньшина Дарья Алексеевна
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: исследование поиска с ограничением глубины.

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

Ход работы:

Ссылка на репозиторий: <https://github.com/Dash-AI/3lr4.git>

Необходимо клонировать репозиторий

```
C:\Users\ACER>git clone https://github.com/Dash-AI/3lr4.git
Cloning into '3lr4'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), 4.08 KiB | 417.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 1 – Клонирование репозитория

```
C:\Users\ACER>cd 3lr4
C:\Users\ACER\3lr4>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/ACER/3lr4/.git/hooks]
C:\Users\ACER\3lr4>
```

Рисунок 2 – Модель ветвления git-flow

```
(base) C:\Users\ACER>cd C:\Users\ACER\3lr4
(base) C:\Users\ACER\3lr4>conda create -n 3lr4 python=3.11
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.5.2
  latest version: 24.11.2

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=24.11.2

## Package Plan ##

  environment location: C:\ProgramData\miniconda3\envs\3lr4
  added / updated specs:
    - python=3.11
```

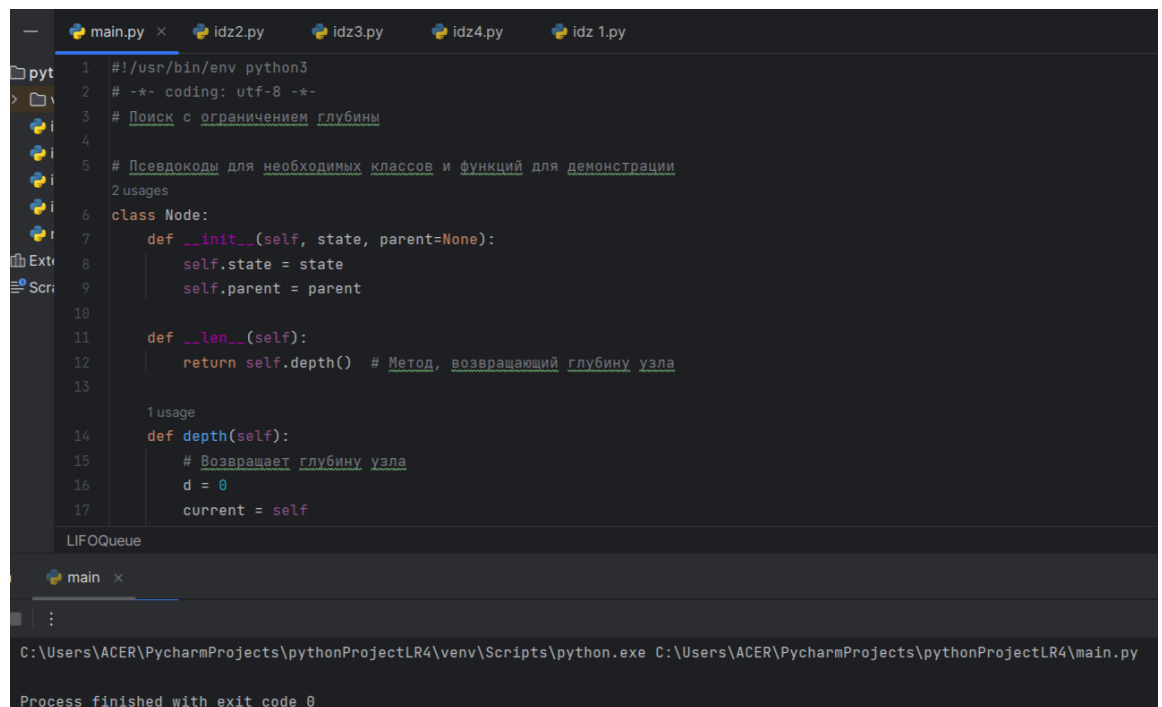
Рисунок 3 – Создание виртуального окружения

```
Администратор: Anaconda Prompt (miniconda3)

(base) C:\Users\ACER\3lr4>conda list
# packages in environment at C:\ProgramData\miniconda3:
#
# Name                        Version      Build      Channel
black                        24.8.0       py311haa95532_0
boltons                     23.0.0       py311haa95532_0
brotlipy                    0.7.0        py311h2bbff1b_1002
bzip2                       1.0.8        he774522_0
ca-certificates             2024.11.26   haa95532_0
certifi                     2024.12.14   py311haa95532_0
cffi                        1.15.1       py311h2bbff1b_3
charset-normalizer          2.0.4        pyhd3eb1b0_0
click                       8.1.7        win_pyh7428d3b_0
colorama                    0.4.6        py311haa95532_0
conda                       23.5.2       py311haa95532_0
conda-content-trust         0.1.3        py311haa95532_0
conda-libmamba-solver       23.5.0       py311haa95532_0
conda-package-handling      2.1.0        py311haa95532_0
conda-package-streaming     0.8.0        py311haa95532_0
console_shortcut_miniconda 0.1.1        haa95532_1
cryptography                39.0.1       py311h21b164f_2
flake8                      7.1.1        py311haa95532_0
fmt                         9.1.0        h6d14046_0
idna                        3.4          py311haa95532_0
isort                       5.13.2       py311haa95532_0
jsonpatch                   1.32         pyhd3eb1b0_0
jsonpointer                 2.1          pyhd3eb1b0_0
libarchive                  3.6.2        hb62f4d4_2
libcurl                     8.1.1        h86230a5_0
```

Рисунок 4 – Установка пакетов black, flake8, isort

Рассмотрим пример, предоставленный в методическом указании, а также выполним индивидуальные задания.



The screenshot shows a code editor with a file named `main.py` open. The code is a Python script that defines a `Node` class and a `LIFOQueue` class. The `Node` class has an `__init__` method and a `depth` method. The `LIFOQueue` class is partially visible. The script is executed, and the output shows the path to the Python executable and the exit code 0.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 # Поиск с ограничением глубины
4
5 # Псевдокоды для необходимых классов и функций для демонстрации
6 2 usages
7 class Node:
8     def __init__(self, state, parent=None):
9         self.state = state
10        self.parent = parent
11
12    def __len__(self):
13        return self.depth() # Метод, возвращающий глубину узла
14
15    1 usage
16    def depth(self):
17        # Возвращает глубину узла
18        d = 0
19        current = self
20
21 class LIFOQueue:
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

main x

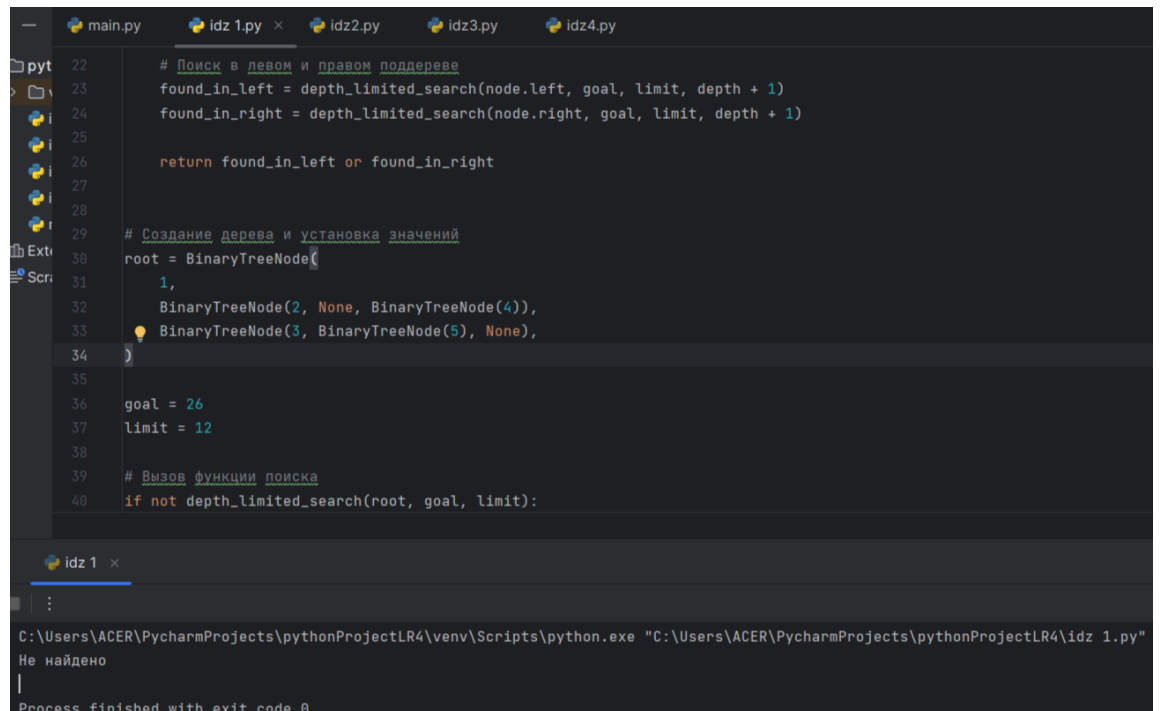
C:\Users\ACER\PycharmProjects\pythonProjectLR4\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProjectLR4\main.py

Process finished with exit code 0

Рисунок 5 – Выполнение примера №1

Индивидуальное задание №1: Решение задания «Система навигации робота-пылесоса». Необходимо реализовать алгоритм, который поможет

роботу определить, существует ли путь к целевой комнате, не превышая заданное ограничение по глубине поиска.



```
22 # Поиск в левом и правом поддереве
23 found_in_left = depth_limited_search(node.left, goal, limit, depth + 1)
24 found_in_right = depth_limited_search(node.right, goal, limit, depth + 1)
25
26 return found_in_left or found_in_right
27
28
29 # Создание дерева и установка значений
30 root = BinaryTreeNode(
31     1,
32     BinaryTreeNode(2, None, BinaryTreeNode(4)),
33     BinaryTreeNode(3, BinaryTreeNode(5), None),
34 )
35
36 goal = 26
37 limit = 12
38
39 # Вызов функции поиска
40 if not depth_limited_search(root, goal, limit):
```

idz 1 x

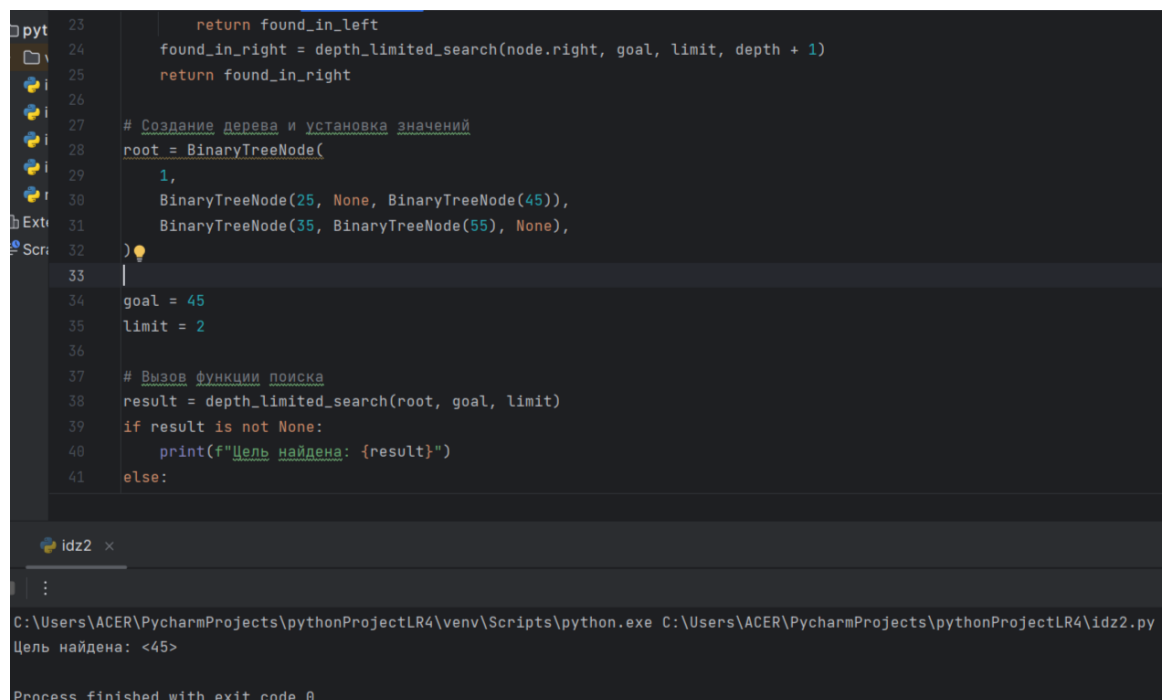
C:\Users\ACER\PycharmProjects\pythonProjectLR4\venv\Scripts\python.exe "C:\Users\ACER\PycharmProjects\pythonProjectLR4\idz 1.py"

Не найдено

Process finished with exit code 0

Рисунок 6 – Выполнение индивидуального задания №1

Индивидуальное задание №2: Система управления складом. Требуется найти наименее затратный путь к товару, ограничив поиск заданной глубиной, чтобы гарантировать, что поиск займет приемлемое время.



```
23 return found_in_left
24 found_in_right = depth_limited_search(node.right, goal, limit, depth + 1)
25 return found_in_right
26
27 # Создание дерева и установка значений
28 root = BinaryTreeNode(
29     1,
30     BinaryTreeNode(25, None, BinaryTreeNode(45)),
31     BinaryTreeNode(35, BinaryTreeNode(55), None),
32 )
33
34 goal = 45
35 limit = 2
36
37 # Вызов функции поиска
38 result = depth_limited_search(root, goal, limit)
39 if result is not None:
40     print(f"Цель найдена: {result}")
41 else:
```

idz2 x

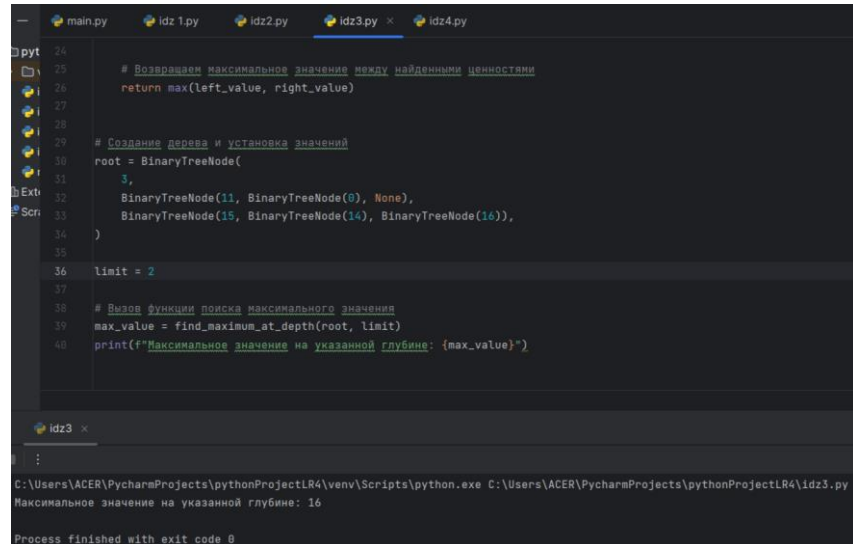
C:\Users\ACER\PycharmProjects\pythonProjectLR4\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProjectLR4\idz2.py

Цель найдена: <45>

Process finished with exit code 0

Рисунок 7 – Выполнение индивидуального задания №2

Индивидуальное задание №3: Задание «Система 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.



```
24 # Возвращаем максимальное значение между найденными значениями
25 return max(left_value, right_value)
26
27
28
29 # Создание дерева и установка значений
30 root = BinaryTreeNode(
31     3,
32     BinaryTreeNode(11, BinaryTreeNode(0), None),
33     BinaryTreeNode(15, BinaryTreeNode(14), BinaryTreeNode(16)),
34 )
35
36 limit = 2
37
38 # Вызов функции поиска максимального значения
39 max_value = find_maximum_at_depth(root, limit)
40 print(f"Максимальное значение на указанной глубине: {max_value}")
```

idz3

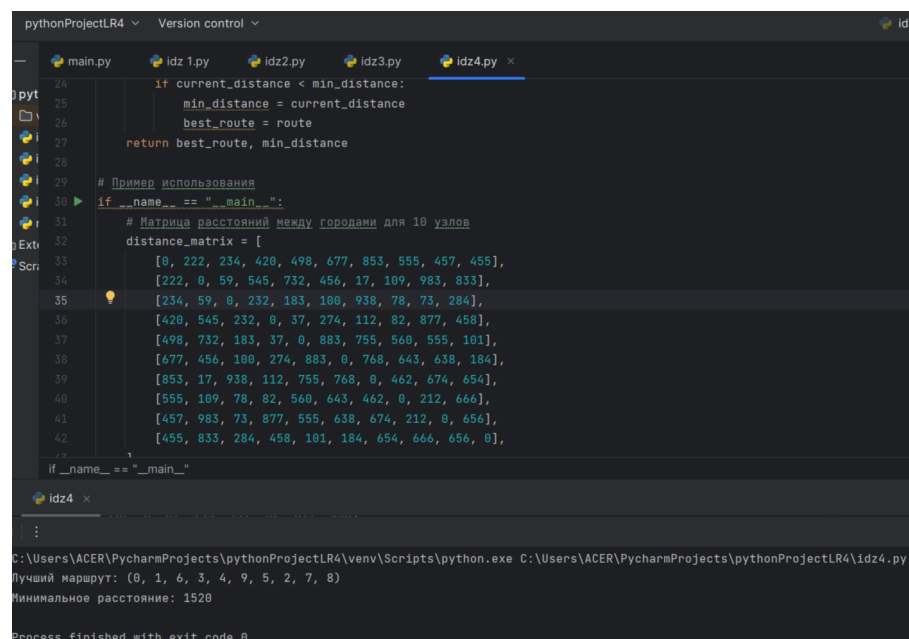
C:\Users\ACER\PycharmProjects\pythonProjectLR4\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProjectLR4\idz3.py

Максимальное значение на указанной глубине: 16

Process finished with exit code 0

Рисунок 8 – Выполнение индивидуального задания №3

Индивидуальное задание №4: для построенного графа лабораторной работы 1 напишите программу на языке программирования Python, которая с помощью алгоритма поиска с ограничением глубины находит минимальное расстояние между начальным и конечным пунктами.



```
24 if current_distance < min_distance:
25     min_distance = current_distance
26     best_route = route
27 return best_route, min_distance
28
29 # Пример использования
30 if __name__ == "__main__":
31     # Матрица расстояний между городами для 10 узлов
32     distance_matrix = [
33         [0, 222, 234, 420, 498, 677, 853, 955, 457, 455],
34         [222, 0, 59, 545, 732, 456, 17, 109, 983, 833],
35         [234, 59, 0, 232, 183, 100, 938, 78, 73, 284],
36         [420, 545, 232, 0, 37, 274, 112, 82, 877, 458],
37         [498, 732, 183, 37, 0, 883, 755, 560, 555, 101],
38         [677, 456, 100, 274, 883, 0, 768, 643, 638, 184],
39         [853, 17, 938, 112, 755, 768, 0, 462, 674, 654],
40         [955, 109, 78, 82, 560, 643, 462, 0, 212, 666],
41         [457, 983, 73, 877, 555, 638, 674, 212, 0, 656],
42         [455, 833, 284, 458, 101, 184, 654, 666, 656, 0],
43     ]
44
45 if __name__ == "__main__":
46     # ... (code for finding the minimum distance) ...
```

idz4

C:\Users\ACER\PycharmProjects\pythonProjectLR4\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProjectLR4\idz4.py

Лучший маршрут: (0, 1, 6, 3, 4, 9, 5, 2, 7, 8)

Минимальное расстояние: 1520

Process finished with exit code 0

Рисунок 9 – Выполнение индивидуального задания №4

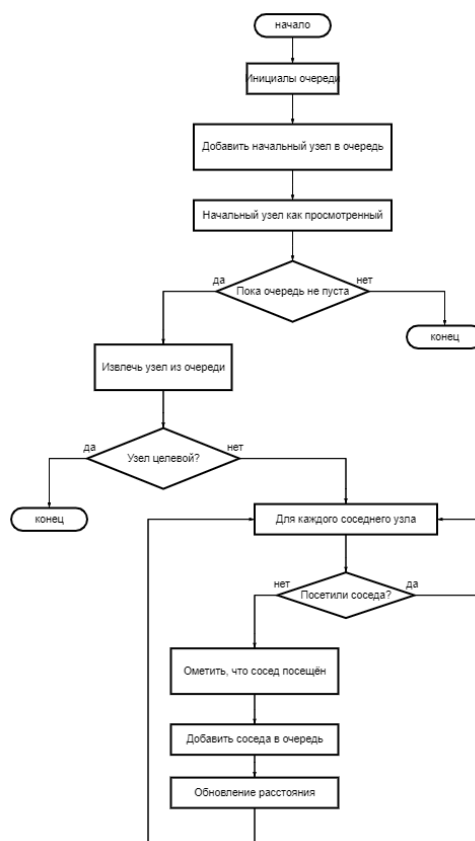


Рисунок 10 – Блок-схема для индивидуального задания №3

```

(base) C:\Users\ACER\3lr4>conda env export > environment.yml
(base) C:\Users\ACER\3lr4>conda deactivate
  
```

Рисунок 11 – Деактивация виртуального окружения

Вывод по лабораторной работе: в ходе выполнения лабораторной работы были приобретены навыки по работе с поиском с ограничением глубины с помощью языка программирования Python версии 3.x–

Контрольные вопросы

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

Поиск с ограничением глубины – это метод поиска, в котором алгоритм ограничивает глубину обхода дерева или графа. Он решает

проблему бесконечных ветвей, устанавливая максимальное значение глубины, на которое алгоритм будет исследовать узлы. Если это ограничение достигнуто, дальнейшее исследование прекращается, предотвращая заикливание.

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

Основная цель ограничения глубины – избежать излишнего расходования ресурсов при поиске, обеспечивая при этом возможность исследования более глубоких уровней дерева поиска, если они потенциально ведут к решению.

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

Поиск в глубину продолжает обходить узлы, пока не найдет решение или не достигнет конца дерева, тогда как поиск с ограничением глубины прекращает поиск, если достигнута заранее заданная глубина, что ограничивает количество исследуемых узлов.

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

Проверка глубины узла позволяет алгоритму определить, достигнуто ли установленное ограничение, и соответственно решить, продолжать поиск дальше или обрезать его.

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

Возврат «обрезания» указывает на то, что поиск достиг установленного лимита глубины и больше не будет продолжаться. Это позволяет алгоритму понимать, что данная ветвь не приведёт к решению.

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

Поиск с ограничением глубины может не найти решение, если целевое состояние находится на глубине, превышающей установленный предел, что может произойти, если глубина решения не предсказуема заранее.

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

Поиск в ширину использует очередь (FIFO), помещая узлы в один конец и извлекая из другого, что обеспечивает исследование всех узлов на текущем уровне перед переходом к следующему. Поиск в глубину использует стек (LIFO), помещая новые узлы на вершину и извлекая с неё, что значит углубление до конца в одной ветке перед исследованием других.

8. Почему поиск с ограничением глубины не является оптимальным?

Поиск с ограничением глубины не является оптимальным, потому что он может не рассматривать более глубокие уровни, которые могут привести к более коротким и эффективным решениям из-за ограничения глубины.

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

Итеративное углубление сочетает преимущества поиска в глубину и поиска с ограничением глубины. Он выполняет поиск с увеличивающимся пределом глубины, начиная с нуля и увеличивая его, что позволяет находить решение, если оно существует, даже если оно находится глубже, чем изначально предполагалось.

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

Итеративное углубление становится более эффективным, когда память ограничена, поскольку оно не требует хранения всех уровней узлов, как это делает поиск в ширину. Это позволяет избежать потребления ресурсов за счёт сохранения только текущего уровня узлов.

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

Основная цель использования алгоритма поиска с ограничением глубины заключается в контроле глубины поиска для избежания чрезмерного расхода ресурсов и управления бесконечными ветвями.

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

Функция `depth_limited_search` принимает параметры, такие как `problem` (задача) и `limit` (максимальная глубина). `problem` определяет структуру задачи и её состояние, а `limit` ограничивает глубину обхода.

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

Значение по умолчанию для параметра `limit` чаще всего устанавливается в ноль или в некое небольшое значение, что позволяет избежать излишних ресурсов в случае отсутствия явных указаний на глубину.

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

Переменная `frontier` представляет собой структуру данных, в которой хранятся узлы, ожидающие обработки. Она управляет узлами, которые будут исследованы на следующих этапах алгоритма.

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

`LIFOQueue` – это структура данных, работающая по принципу «последний пришёл – первый вышел» (`LastIn, FirstOut`). Она используется в алгоритме для реализации поиска в глубину.

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

Значение переменной `result` обычно инициализируется как `None`, указывая на то, что решение ещё не найдено.

17. Какое условие завершает цикл `while` в алгоритме поиска?

Цикл `while` завершается, когда `frontier` пуст, что означает, что больше нет узлов для исследования.

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

С помощью `frontier.pop()` извлекается последний добавленный узел, так как алгоритм работает по принципу LIFO, что позволяет исследовать узлы в порядке их добавления.

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

Если найден узел, удовлетворяющий условию цели, алгоритм возвращает его как решение.

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

Условие ``eliflen(node) >= limit`` проверяет, достиг ли узел предельной глубины. Если это так, то узел не будет дальше обрабатываться и алгоритм вернётся к предыдущему узлу.

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

Если текущий узел достигает ограничения по глубине, алгоритм не исследует его дочерние узлы и возвращает «обрезание», указывая, что дальше не имеет смысла продолжать поиск на этой ветви.

22. Какую роль выполняет проверка на циклы `elifnotiscycle(node)` в алгоритме?

Проверка `elifnotis_cycle(node)` определяет, является ли узел циклом, и, при необходимости, предотвращает повторное исследование узлов, чтобы избежать заикливания.

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

Дочерние узлы, полученные с помощью `expand`, добавляются в `frontier` для дальнейшего исследования.

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

Если целевой узел не был найден, функция обычно возвращает значение, указывающее на неудачу, например `failure` или `cutoff`.

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

Результат failure указывает на то, что решение не существует, в то время как cutoff означает, что достигнута предельная глубина, и дальнейший поиск не возможен, но решение могло бы быть найдено на больших глубинах.