

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 №5
дисциплины
«Искусственный интеллект в профессиональной сфере»

Выполнил:
Гуляницкий Александр
3 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи», очная
форма обучения

(подпись)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Исследование поиска с итеративным углублением.

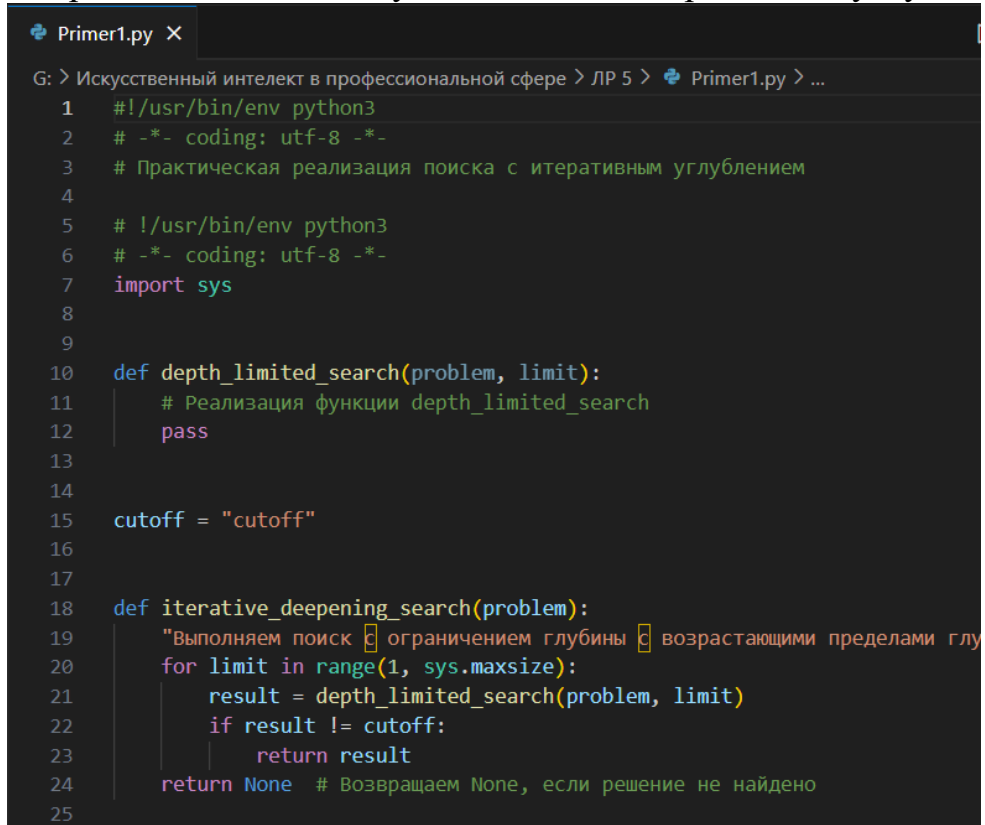
Цель работы: приобретение навыков по работе с поиском с итеративным углублением с помощью языка программирования Python версии 3.x.

Ссылка на репозиторий: <https://github.com/Alexander-its/LR5.git>

Вариант 6.

Ход работы:

Пример 1. Реализация на Python поиска с итеративным углублением.

A screenshot of a code editor window titled 'Primer1.py'. The editor shows Python code for an iterative deepening search algorithm. The code includes comments in Russian and defines two functions: 'depth_limited_search' and 'iterative_deepening_search'. The 'iterative_deepening_search' function loops through increasing depth limits, calling 'depth_limited_search' until a solution is found or a cutoff is reached.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Практическая реализация поиска с итеративным углублением
4
5  # !/usr/bin/env python3
6  # -*- coding: utf-8 -*-
7  import sys
8
9
10 def depth_limited_search(problem, limit):
11     # Реализация функции depth_limited_search
12     pass
13
14
15 cutoff = "cutoff"
16
17
18 def iterative_deepening_search(problem):
19     "Выполняем поиск с ограничением глубины с возрастающими пределами гл
20     for limit in range(1, sys.maxsize):
21         result = depth_limited_search(problem, limit)
22         if result != cutoff:
23             return result
24     return None # Возвращаем None, если решение не найдено
25
```

Рисунок 1 – Код для выполнения примера 1

Задание 1. Поиск элемента в дереве с использованием алгоритма итеративного углубления.

```
IDZ1.py  X
G: > Искусственный интеллект в профессиональной сфере > ЛР 5 > IDZ1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Поиск элемента в дереве с использованием алгоритма итеративного углублен
4
5  class BinaryTreeNode:
6      def __init__(self, value, left=None, right=None):
7          self.value = value
8          self.left = left
9          self.right = right
10
11      def add_children(self, left, right):
12          self.left = left
13          self.right = right
14
15      def __repr__(self):
16          return f"<{self.value}>"
17
18  def depth_limited_search(node, goal, limit):
19      if limit < 0:
20          return "cutoff"
21      if node is None:
22          return None
23      if node.value == goal:
24          return True
25
26      left_result = depth_limited_search(node.left, goal, limit - 1)
27      if left_result is True:
28          return True
29      elif left_result == "cutoff":
30          cutoff = "cutoff"
31      else:
```

Рисунок 2 – Код для выполнения задания 1

Задание 2. Поиск в файловой системе.

```
IDZ2.py  X
G: > Искусственный интеллект в профессиональной сфере > ЛР 5 > IDZ2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Поиск в файловой системе
4
5  class TreeNode:
6      def __init__(self, value):
7          self.value = value
8          self.children = []
9
10     def add_child(self, child):
11         self.children.append(child)
12
13     def add_children(self, *args):
14         for child in args:
15             self.add_child(child)
16
17     def __repr__(self):
18         return f"<{self.value}>"
19
20
21     def depth_limited_search(node, goal, limit, path):
22         if limit < 0:
23             return "cutoff"
24         if node is None:
25             return None
26         # Добавим текущий узел в путь
27         path.append(node.value)
28         if node.value == goal:
29             return path[:] # Возвращаем путь до текущего узла
30
31         for child in node.children:
```

Рисунок 3 - Код для выполнения задания 2

Задание 3.

```

6 class TreeNode:
7     def __init__(self, value, permissions):
8         self.value = value # Имя файла или каталога
9         self.permissions = permissions # Права доступа
10        self.children = [] # Дочерние узлы
11
12        def add_child(self, child):
13            self.children.append(child)
14
15        def add_children(self, *args):
16            for child in args:
17                self.add_child(child)
18
19        def __repr__(self):
20            return f"<{self.value}: {self.permissions}>"
21
22
23 def depth_limited_search(node, goal_perms, limit, found_files):
24     if limit < 0:
25         return found_files
26     if node is None:
27         return found_files
28
29     # Если права доступа совпадают с заданными, добавляем файл в список
30     if node.permissions == goal_perms:
31         found_files.append(node.value)
32         if len(found_files) >= 10: # Останавливаемся, если найдено 10 файлов
33             return found_files
34
35     for child in node.children:
36         result = depth_limited_search(child, goal_perms, limit - 1, found_files)
37         if len(found_files) >= 10:
38             return found_files # Возвращаем найденные файлы, если достигли 10
39

```

Рисунок 4 - Код для выполнения задания 3

Вывод: в ходе выполнения лабораторной работы приобрели навыки по работе с поиском с итеративным углублением с помощью языка программирования Python версии 3.x.