

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:  
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 ограничением глубины.

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

Ссылка на репозиторий: <https://github.com/Alexander-its/LR4.git>

Ход работы:

Пример 1. Реализация на Python поиска с ограничением глубины.



```
Primer1.py X
G: > Искусственный интеллект в профессиональной сфере > ЛР 4 > Primer1.py > depth_
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  class Node:
7      def __init__(self, state, parent=None):
8          self.state = state
9          self.parent = parent
10
11      def __len__(self):
12          return self.depth() # Метод, возвращающий глубину узла
13
14      def depth(self):
15          # Возвращает глубину узла
16          d = 0
17          current = self
18          while current.parent is not None:
19              d += 1
20              current = current.parent
21          return d
22
23
24  class LIFOQueue:
25      def __init__(self, items=None):
26          if items is None:
27              items = []
28          self.items = items
29
30      def pop(self):
31          return self.items.pop() if self.items else None
32
```

Рисунок 1 – Код для выполнения примера 1

Задание 1. Система навигации робота-пылесоса.

```
IDZ1.py ×
G: > Искусственный интеллект в профессиональной сфере > ЛР 4 > IDZ1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Система навигации робота-пылесоса
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 __repr__(self):
12         return f"<{self.value}>"
13
14
15     def depth_limited_search(node, goal, limit, depth=0):
16         if node is None:
17             return False
18         if node.value == goal:
19             print(f"Найден на глубине: {depth}")
20             return True
21         if depth >= limit:
22             return False
23
24         # Поиск в левом и правом поддереве
25         found_in_left = depth_limited_search(node.left, goal, limit, depth + 1)
26         found_in_right = depth_limited_search(node.right, goal, limit, depth + 1)
27
28         return found_in_left or found_in_right
29
30
31     # Создание дерева и установка значений
32     root = BinaryTreeNode(
```

Рисунок 2 – Код для выполнения задания 1

Задание 2. Система управления складом.

```
IDZ2.py x
G: > Искусственный интеллект в профессиональной сфере > ЛР 4 > IDZ2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Система управления складом
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 __repr__(self):
12         return f"<{self.value}>"
13
14
15     def depth_limited_search(node, goal, limit, depth=0):
16         if node is None:
17             return None
18         if node.value == goal:
19             return node
20         if depth >= limit:
21             return None
22
23         # Поиск в левом и правом поддереве
24         found_in_left = depth_limited_search(node.left, goal, limit, depth + 1)
25         if found_in_left is not None:
26             return found_in_left
27
28         found_in_right = depth_limited_search(node.right, goal, limit, depth + 1)
29         return found_in_right
30
31
32     # Создание дерева и установка значений
```

Рисунок 3 - Код для выполнения задания 2

Задание 3. Система автоматического управления инвестициями.

```
IDZ3.py X
G: > Искусственный интеллект в профессиональной сфере > ЛР 4 > IDZ3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Система управления складом
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 __repr__(self):
12         return f"<{self.value}>"
13
14
15     def find_maximum_at_depth(node, limit, depth=0):
16         if node is None:
17             return float('-inf') # Возвращаем минимальное значение, если узел
18         if depth == limit:
19             return node.value # Возвращаем значение узла на заданной глубине
20
21         # Рекурсивный поиск в левом и правом поддереве
22         left_value = find_maximum_at_depth(node.left, limit, depth + 1)
23         right_value = find_maximum_at_depth(node.right, limit, depth + 1)
24
25         # Возвращаем максимальное значение между найденными ценностями
26         return max(left_value, right_value)
27
28
29     # Создание дерева и установка значений
30     root = BinaryTreeNode(
31         3,
32         BinaryTreeNode(1, BinaryTreeNode(0), None),
```

Рисунок 4 - Код для выполнения задания 3

Задание 4. Алгоритм поиска с ограничением глубины (минимальное расстояние между начальным и конечным пунктами).

```

IDZ4.py
G: > Искусственный интеллект в профессиональной сфере > ЛР 4 > IDZ4.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # С помощью алгоритма поиска с ограничением глубины найдем минимальное рас
4  import itertools
5
6  def calculate_total_distance(route, distance_matrix):
7      """Вычисление общей дистанции для данного маршрута."""
8      total_distance = 0
9      for i in range(len(route) - 1):
10         total_distance += distance_matrix[route[i]][route[i + 1]]
11         # Добавляем расстояние обратно к начальной точке
12         total_distance += distance_matrix[route[-1]][route[0]]
13     return total_distance
14
15 def traveling_salesman(distance_matrix):
16     """Решение задачи коммивояжера методом полного перебора."""
17     num_cities = len(distance_matrix)
18     cities = list(range(num_cities))
19     min_distance = float('inf')
20     best_route = None
21     # Генерация всех возможных маршрутов
22     for route in itertools.permutations(cities):
23         current_distance = calculate_total_distance(route, distance_matrix)
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

```

Рисунок 5 - Код для выполнения задания 4

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