

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамента цифровых, роботехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
дисциплины «Искусственный интеллект в профессиональной сфере»

Выполнила:
Гайчук Дарья Дмитриевна
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика
и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А.-доцент департамента
цифровых, роботехнических систем и
электроники института перспективной
инженерии

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: «Исследование поиска в глубину»

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

Ссылка на репозиторий: https://github.com/Ichizuchi/PR_AI_003

Порядок выполнения работы:

1. Создала новый репозиторий и клонировала его на свой компьютер.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * / Repository name *
Ichizuchi / PR_AI_003
PR_AI_003 is available.

Great repository names are short and memorable. Need inspiration? How about [improved-waffle](#) ?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template: Python
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: MIT License
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Рисунок 1. Создание репозитория

2. Клонировала репозиторий на свой компьютер.

```
@Ichizuchi →/workspaces/PR_AI_003 (main) $ git branch develop
@Ichizuchi →/workspaces/PR_AI_003 (main) $ git checkout develop
Switched to branch 'develop'
@Ichizuchi →/workspaces/PR_AI_003 (develop) $
```

Рисунок 2. Модель ветвления git-flow

3. Создала виртуальное окружение Anaconda с именем репозитория.

```

@Ichizuchi →/workspaces/PR_AI_003 (develop) $ conda create -n myenv python=3.10
Retrieving notices: ...working... done
Channels:
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): | 

```

Рисунок 3. Создание виртуального окружения

Вариант 3

Задание №1. **Flood fill** алгоритм для определения и изменения области, связанной с начальным узлом в многомерной матрице.

```

Tasks > Task_1.py > ...
1  def flood_fill(matrix, start_node, target_color, replacement_color):
2      rows, cols = len(matrix), len(matrix[0])
3      x, y = start_node
4      if matrix[x][y] != target_color:
5          return matrix
6
7      def dfs(row, col):
8          if not (0 <= row < rows and 0 <= col < cols):
9              return
10             if matrix[row][col] != target_color:
11                 return
12             matrix[row][col] = replacement_color
13             for dr, dc in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
14                 dfs(row + dr, col + dc)
15
16         dfs(x, y)
17         return matrix
18

```

Рисунок 4. Код программы

```

def main():
    # Задание 1
    matrix = [
        ["Y", "Y", "Y", "G", "G", "G", "G", "G", "G", "G"],
        ["Y", "Y", "Y", "G", "G", "G", "X", "X", "X", "X"],
        ["W", "W", "Y", "W", "G", "G", "G", "X", "X", "X"],
        ["W", "W", "W", "W", "G", "G", "G", "G", "G", "X"],
        ["W", "W", "W", "R", "R", "R", "R", "X", "X", "X"],
        ["W", "B", "R", "R", "R", "R", "R", "R", "R", "X"],
        ["W", "B", "B", "B", "B", "B", "B", "R", "X", "X"],
    ]
    start_node = (3, 9)
    target_color = "X"
    replacement_color = "C"

    print("Flood Fill Result:")
    result = flood_fill(matrix, start_node, target_color, replacement_color)
    for row in result:

```

Рисунок 5. Входные данные задания №1

```
Flood Fill Result:
['Y', 'Y', 'Y', 'G', 'G', 'G', 'G', 'G', 'G', 'G']
['Y', 'Y', 'Y', 'G', 'G', 'G', 'C', 'C', 'C', 'C']
['W', 'W', 'Y', 'W', 'G', 'G', 'G', 'C', 'C', 'C']
['W', 'W', 'W', 'W', 'G', 'G', 'G', 'G', 'G', 'C']
['W', 'W', 'W', 'R', 'R', 'R', 'R', 'C', 'C', 'C']
['W', 'B', 'R', 'R', 'R', 'R', 'R', 'R', 'R', 'C']
['W', 'B', 'B', 'B', 'B', 'B', 'B', 'R', 'C', 'C']
```

Рисунок 6. Вывод задания №1

Задание №2. Поиск самого длинного пути в матрице символов, начиная с заданного символа, где символы должны следовать в алфавитном порядке.

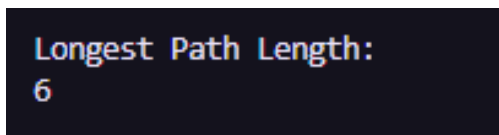
```
Tasks > Task_2.py > ...
1 def longest_path_in_matrix(matrix, start_char):
2     rows, cols = len(matrix), len(matrix[0])
3     directions = [(-1, 0), (1, 0), (0, -1), (0, 1), (-1, -1), (-1, 1), (1, -1), (1, 1)]
4     memo = {}
5
6     def is_valid_move(x, y, prev_char):
7         return 0 <= x < rows and 0 <= y < cols and ord(matrix[x][y]) == ord(prev_char) + 1
8
9     def dfs(x, y):
10         if (x, y) in memo:
11             return memo[(x, y)]
12         max_path = 1
13         for dx, dy in directions:
14             nx, ny = x + dx, y + dy
15             if is_valid_move(nx, ny, matrix[x][y]):
16                 max_path = max(max_path, 1 + dfs(nx, ny))
17         memo[(x, y)] = max_path
18         return max_path
19
20     max_length = 0
21     for i in range(rows):
22         for j in range(cols):
23             if matrix[i][j] == start_char:
24                 max_length = max(max_length, dfs(i, j))
25     return max_length
26
```

Рисунок 7. Код программы

```
# Задание 2
matrix = [
    ["D", "E", "H", "X", "B"],
    ["A", "O", "G", "P", "E"],
    ["D", "D", "C", "F", "D"],
    ["E", "B", "E", "A", "S"],
    ["C", "D", "Y", "E", "N"],
]
start_char = "C"

print("\nLongest Path Length:")
print(longest_path_in_matrix(matrix, start_char))
```

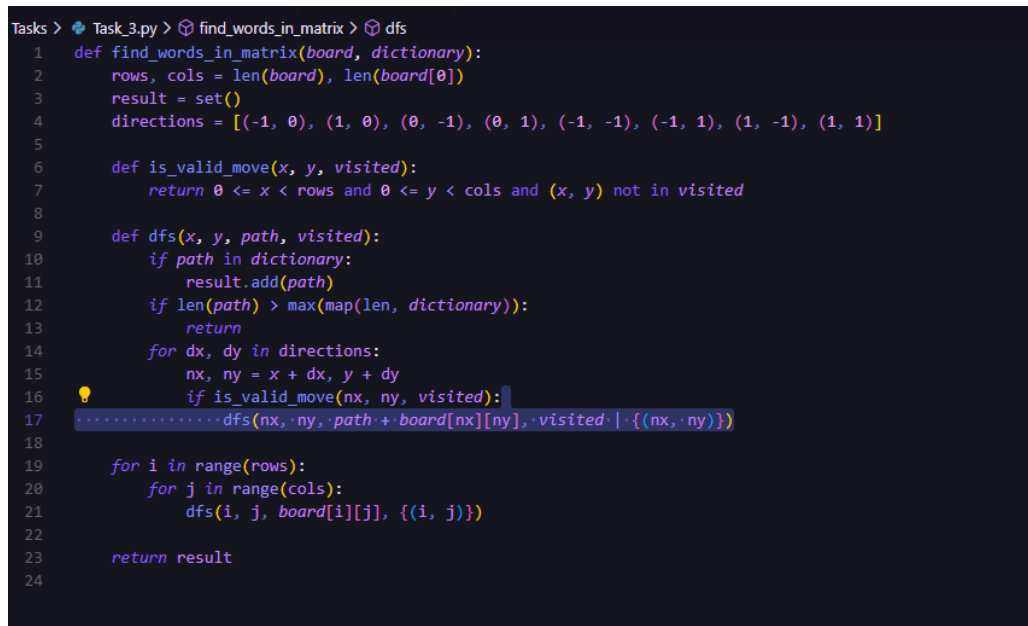
Рисунок 8. Входные данные задания №2



```
Longest Path Length:
6
```

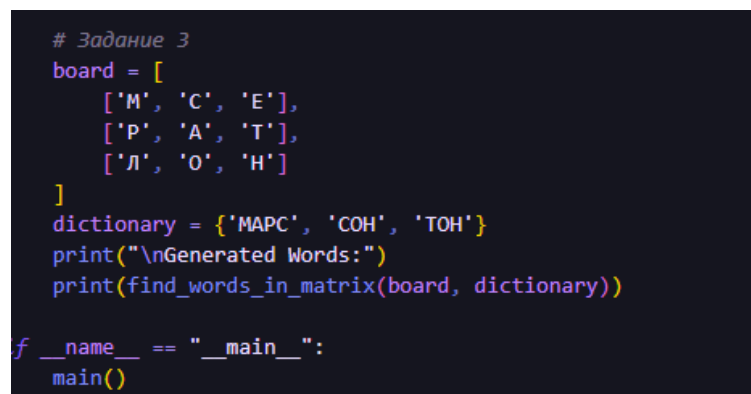
Рисунок 9. Вывод задания №2

Задание №3. Генерация списка возможных слов из матрицы СИМВОЛОВ по заданным условиям.



```
Tasks > Task_3.py > find_words_in_matrix > dfs
1 def find_words_in_matrix(board, dictionary):
2     rows, cols = len(board), len(board[0])
3     result = set()
4     directions = [(-1, 0), (1, 0), (0, -1), (0, 1), (-1, -1), (-1, 1), (1, -1), (1, 1)]
5
6     def is_valid_move(x, y, visited):
7         return 0 <= x < rows and 0 <= y < cols and (x, y) not in visited
8
9     def dfs(x, y, path, visited):
10         if path in dictionary:
11             result.add(path)
12         if len(path) > max(map(len, dictionary)):
13             return
14         for dx, dy in directions:
15             nx, ny = x + dx, y + dy
16             if is_valid_move(nx, ny, visited):
17                 dfs(nx, ny, path + board[nx][ny], visited | {(nx, ny)})
18
19     for i in range(rows):
20         for j in range(cols):
21             dfs(i, j, board[i][j], {(i, j)})
22
23     return result
24
```

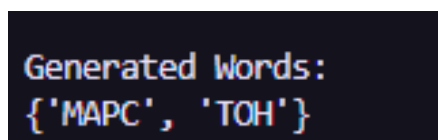
Рисунок 10. Код программы



```
# Задание 3
board = [
    ['M', 'C', 'E'],
    ['P', 'A', 'T'],
    ['L', 'O', 'H']
]
dictionary = {'МАРС', 'СОН', 'ТОН'}
print("\nGenerated Words:")
print(find_words_in_matrix(board, dictionary))

if __name__ == "__main__":
    main()
```

Рисунок 11. Входные данные задания №3



```
Generated Words:
{'МАРС', 'ТОН'}
```

Рисунок 12. Вывод задания №3

Задание №4. Поиск в глубину для графов: Использовать алгоритм из лабораторной работы 1 для поиска минимального расстояния.

```

Tasks > Task 4.py > ...
1 # Поиск в глубину по личному графу
2 from collections import deque
3
4 # Граф с расстояниями между городами
5 graph = {
6     "Стамбул": [("Бурса", 155), ("Эскишехир", 480), ("Анкара", 450)],
7     "Бурса": [("Стамбул", 155), ("Анкара", 480), ("Маниса", 70)],
8     "Эскишехир": [("Стамбул", 480), ("Анкара", 230)],
9     "Анкара": [("Эскишехир", 230), ("Стамбул", 450), ("Бурса", 480), ("Сивас", 320), ("Конья", 260)],
10    "Маниса": [("Бурса", 70), ("Измир", 330)],
11    "Измир": [("Маниса", 330), ("Анталья", 460)],
12    "Конья": [("Анкара", 260), ("Анталья", 290), ("Мерсин", 460)],
13    "Анталья": [("Измир", 460), ("Конья", 290)],
14    "Мерсин": [("Конья", 460), ("Адана", 70)],
15    "Адана": [("Мерсин", 70), ("Газантеп", 220)],
16    "Газантеп": [("Адана", 220), ("Шанлыурфа", 150)],
17    "Шанлыурфа": [("Газантеп", 150), ("Диярбакыр", 180)],
18    "Диярбакыр": [("Шанлыурфа", 180), ("Ван", 320)],
19    "Ван": [("Диярбакыр", 320), ("Эрзурум", 370)],
20    "Эрзурум": [("Ван", 370), ("Трабзон", 250)],
21    "Трабзон": [("Эрзурум", 250), ("Самсун", 330)],
22    "Самсун": [("Трабзон", 330), ("Сивас", 410)],
23    "Сивас": [("Самсун", 410), ("Анкара", 320), ("Кайсери", 240)],
24    "Кайсери": [("Сивас", 240), ("Малатья", 240)],
25    "Малатья": [("Кайсери", 240), ("Шанлыурфа", 90)]
26 }
27
28 def dfs_min_path(graph, start, goal, visited=None, path_cost=0):
29     if visited is None:
30         visited = set()
31     if start == goal:
32         return path_cost
33     visited.add(start)
34     min_cost = float('inf')
35     for neighbor, cost in graph[start]:
36         if neighbor not in visited:
37             current_cost = dfs_min_path(graph, neighbor, goal, visited.copy(), path_cost + cost)
38             min_cost = min(min_cost, current_cost)
39     return min_cost
40
41 # Пример использования
42 start_city = "Стамбул"
43 goal_city = "Анкара"
44 min_distance = dfs_min_path(graph, start_city, goal_city)
45 print(f"Минимальное расстояние между {start_city} и {goal_city}: {min_distance}")
46

```

Рисунок 13. Код программы

```

@Ichizuchi →/workspaces/PR_AI_003 (develop) $ /home/codespace/.python/current/bin/python3 /workspaces/PR_AI_003/Tasks/Task_4.py
Минимальное расстояние между Стамбул и Анкара: 450
@Ichizuchi →/workspaces/PR_AI_003 (develop) $

```

Рисунок 14. Вывод задания №4

```

==> For changes to take effect, close and re-open your current shell. <==
@Ichizuchi →/workspaces/PR_AI_003 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@Ichizuchi →/workspaces/PR_AI_003 (main) $ git merge develop
Updating f7f6cee..e44c66f
Fast-forward
 Tasks/Task_1.py | 17 ++++++
 Tasks/Task_2.py | 25 ++++++
 Tasks/Task_3.py | 23 ++++++
 Tasks/Task_4.py | 45 ++++++
 Tasks/tasks.py | 116 ++++++
 5 files changed, 226 insertions(+)
 create mode 100644 Tasks/Task_1.py
 create mode 100644 Tasks/Task_2.py
 create mode 100644 Tasks/Task_3.py
 create mode 100644 Tasks/Task_4.py
 create mode 100644 Tasks/tasks.py
@Ichizuchi →/workspaces/PR_AI_003 (main) $

```

Рисунок 15. Слияние веток

Ответы на контрольные вопросы

1. Ключевое отличие поиска в глубину от поиска в ширину

Поиск в глубину углубляется по одной ветви до конца, пока возможно, а затем возвращается назад. Поиск в ширину, напротив, проходит все узлы текущего уровня перед переходом на следующий.

2. Четыре критерия качества поиска

- 1) Полнота (гарантия нахождения решения, если оно существует),
- 2) Оптимальность (находится ли лучшее решение),
- 3) Временная сложность,
- 4) Пространственная сложность.

3. Расширение узла в поиске в глубину

При расширении узла алгоритм добавляет его дочерние узлы в стек (или вызывает функцию рекурсивно для дочерних узлов).

4. Почему поиск в глубину использует LIFO?

Очередь LIFO позволяет алгоритму возвращаться к последнему исследованному узлу, обеспечивая углубление в одной ветви перед переходом к другой.

5. Как поиск в глубину удаляет узлы из памяти?

Узлы удаляются из стека, как только обработка их потомков завершается. Это экономит память по сравнению с поиском в ширину, который хранит все узлы текущего уровня.

6. Какие узлы остаются в памяти при максимальной глубине?

Только узлы текущего пути и дочерние узлы последнего узла.

7. Когда поиск в глубину может "застыть"?

Если дерево содержит бесконечные ветви или если алгоритм возвращается к уже посещённым узлам.

8. Временная сложность поиска в глубину

Зависит от максимальной глубины дерева $O(b^d)$, где b — фактор ветвления, d — максимальная глубина.

9. Почему поиск в глубину не оптимален?

Он не гарантирует нахождение кратчайшего пути, так как не исследует все узлы текущего уровня перед углублением.

10. Когда предпочтительно использовать поиск в глубину?

При ограниченной памяти, когда дерево поиска большое, а решение находится на глубине.

11. Функция `depth_first_recursive_search`

Реализует рекурсивный обход дерева поиска. Она принимает текущий узел, проблему (объект, описывающий задачу), стек уже посещённых узлов.

12. Задача проверки `if node is None`

Защита от попытки обработки несуществующего узла.

13. Когда функция возвращает узел как решение?

14. Механизм обхода дерева

Используется рекурсивный вызов для перехода между уровнями дерева.

15. Если решение не найдено в ходе рекурсии

Функция возвращает "failure" или пустое значение.

16. Почему функция вызывает саму себя?

Для углубления по ветвям дерева и обработки всех уровней.

17. Функция `expand(problem, node)`

Создаёт список дочерних узлов для текущего узла на основе структуры задачи.

18. Роль функции `is_cycle(node)`

Проверяет, был ли узел уже посещён, чтобы избежать повторной обработки.

19. Проверка `if result`

Позволяет прервать цикл и вернуть результат, если решение найдено.

20. Когда алгоритм может вернуть failure?

Если ни один из узлов не удовлетворяет критериям задачи.

21. Разница между рекурсивной и итеративной реализациями

Итеративная реализация использует явный стек для хранения узлов, а рекурсивная использует стек вызовов.

22. Проблемы при бесконечных деревьях

Алгоритм может застрять в бесконечной ветви или использовать слишком много памяти.

Вывод: в ходе работы были приобретены навыки по работе с поиском в глубину с помощью языка программирования Python версии 3.x