

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 №2
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 в ширину.

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

Ход работы:

Ссылка на репозиторий: <https://github.com/Dash-AI/3lr2.git>

Создали репозиторий и клонировали его

```
C:\Users\ACER>git clone https://github.com/Dash-AI/3lr2.git
Cloning into '3lr2'...
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 | 464.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.

C:\Users\ACER>cd 3lr2

C:\Users\ACER\3lr2>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/3lr2/.git/hooks]
```

Рисунок 1 – Клонирование репозитория

Рассмотрим приведённый пример в методическом указании

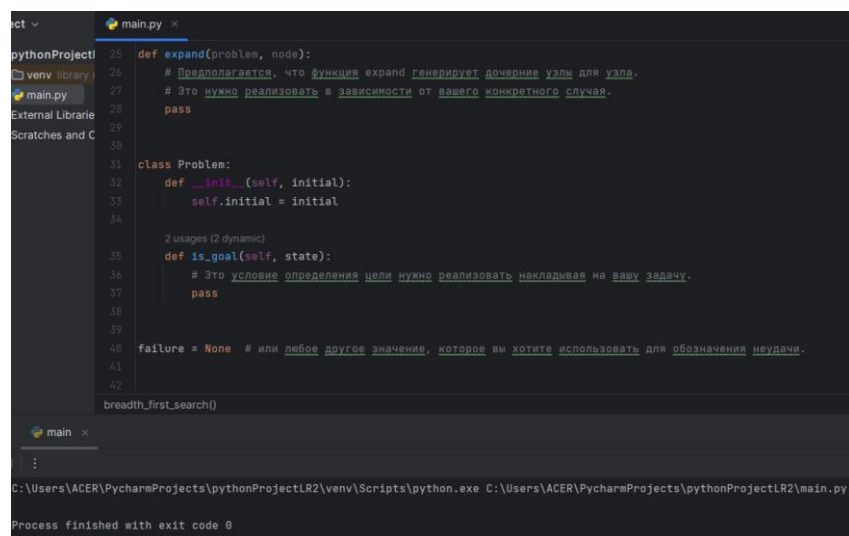


Рисунок 2 – Пример №1

Индивидуальное задание №1: для задачи «Расширенный подсчет количества островов в бинарной матрице» необходимо создать собственную матрицу и подсчитать количество островов.

```

pythonProjectLR2  Version control  ID31
pythonProjectLR2
venv library
main.py
ID31.py
External Libraries
Scratches and Console

return island
66
67
68
69 if __name__ == '__main__':
70     mat = [
71         [1, 0, 0, 0, 0, 1, 0],
72         [0, 1, 1, 0, 1, 1, 1],
73         [1, 0, 0, 1, 0, 0, 1],
74         [0, 0, 0, 0, 0, 1, 1],
75         [0, 0, 0, 0, 0, 1, 0],
76         [0, 1, 1, 1, 0, 0, 1]
77     ]
78
79     print('Общее количество островов:', countIslands(mat))

if __name__ == '__main__':

C:\Users\ACER\PycharmProjects\pythonProjectLR2\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProjectLR2\ID31.py
Общее количество островов: 2
Process finished with exit code 0
  
```

Рисунок 3 – Выполнение индивидуального задания №1

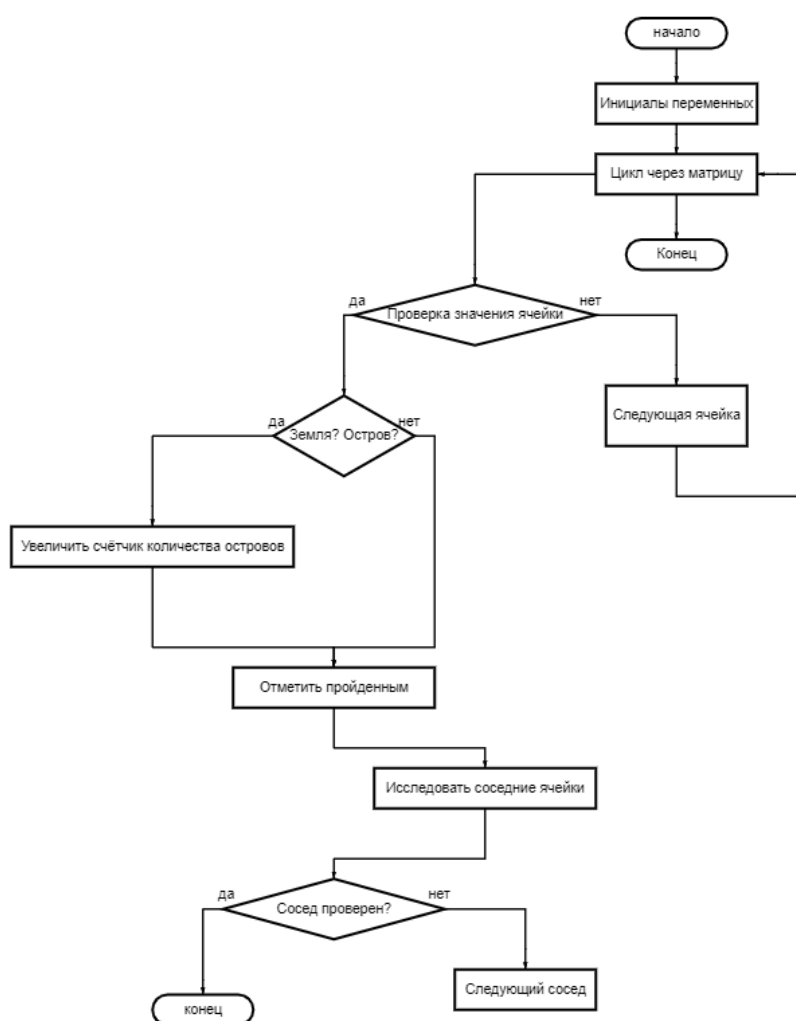
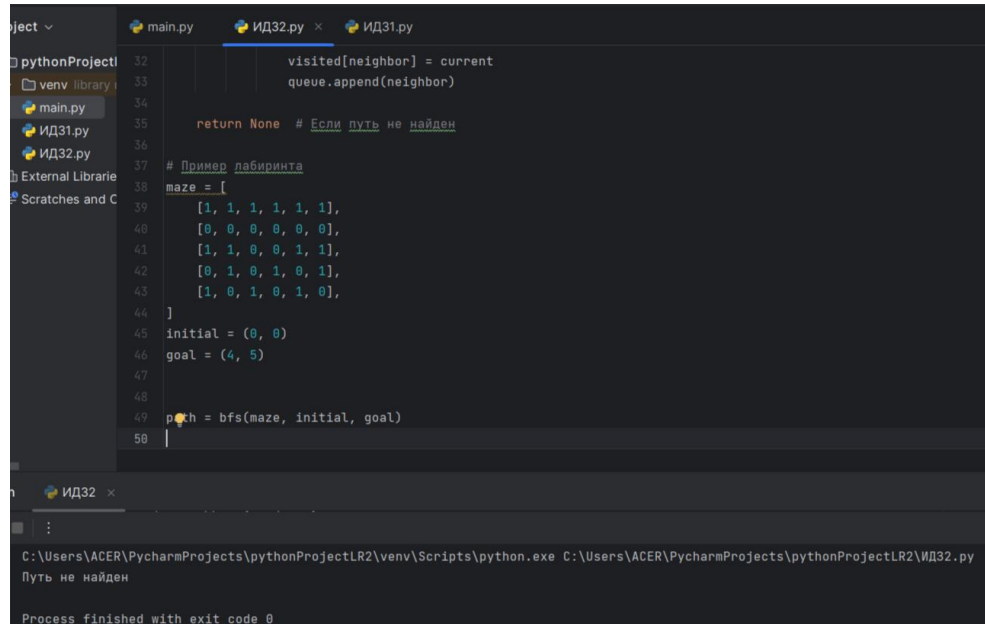


Рисунок 4 – Блок-схема для индивидуального задания №1

Индивидуальное задание №2: для задачи «Поиск кратчайшего пути в лабиринте» подготовить собственную схему лабиринта, а также определить начальную и конечную позиции в лабиринте. Для данных найти минимальный путь в лабиринте от начальной к конечной позиции.



```
32     visited[neighbor] = current
33     queue.append(neighbor)
34
35     return None # Если путь не найден
36
37 # Пример лабиринта
38 maze = [
39     [1, 1, 1, 1, 1, 1],
40     [0, 0, 0, 0, 0, 0],
41     [1, 1, 0, 0, 1, 1],
42     [0, 1, 0, 1, 0, 1],
43     [1, 0, 1, 0, 1, 0],
44 ]
45 initial = (0, 0)
46 goal = (4, 5)
47
48
49 path = bfs(maze, initial, goal)
50
```

ИД32

C:\Users\ACER\PycharmProjects\pythonProjectLR2\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProjectLR2\ИД32.py

Путь не найден

Process finished with exit code 0

Рисунок 5 – Выполнение индивидуального задания №2



Рисунок 6 – Блок-схема для индивидуального задания №2

Индивидуальное задание №3: с помощью алгоритма поиска в ширину находит минимальное расстояние между начальным и конечным пунктами.

```

29
30     return distances[end] if distances[end] != float('inf') else -1 # -1 если путь не найден
31
32
33 # Пример использования
34 if __name__ == "__main__":
35     # Матрица расстояний между городами
36     distance_matrix = [
37         [0, 222, 234, 420, 498, 677, 853, 555, 457, 455],
38         [222, 0, 59, 545, 732, 456, 17, 109, 983, 833],
39         [234, 59, 0, 232, 183, 100, 938, 78, 73, 284],
40         [420, 545, 232, 0, 37, 274, 112, 82, 877, 458],
41         [498, 732, 183, 37, 0, 883, 755, 560, 555, 101],
42         [677, 456, 100, 274, 883, 0, 768, 643, 638, 184],
43         [853, 17, 938, 112, 755, 768, 0, 462, 674, 654],
44         [555, 109, 78, 82, 560, 643, 462, 0, 212, 666],
45         [457, 983, 73, 877, 555, 638, 674, 212, 0, 656],
46         [455, 833, 284, 458, 101, 184, 654, 666, 656, 0],
47     ]
48
49 if __name__ == "__main__":

```

ИД33 ×

C:\Users\ACER\PycharmProjects\pythonProjectLR2\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProjectLR2\ИД33.py

Минимальное расстояние между городами 0 и 3: 420

Process finished with exit code 0

Рисунок 7 – Выполнение индивидуального задания №3

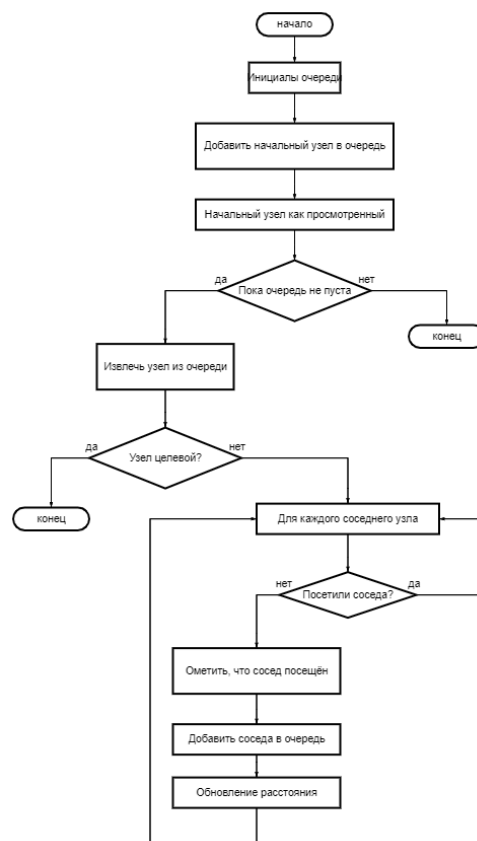


Рисунок 8 – Блок-схема для индивидуального задания №3

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

Контрольные вопросы

1. Какой тип очереди используется в стратегии поиска в ширину?

В стратегии поиска в ширину используется очередь (обычно реализованная как FIFO - FirstIn, FirstOut). Это означает, что узлы обрабатываются в том порядке, в котором они были добавлены в очередь.

2. Почему новые узлы в стратегии поиска в ширину добавляются в конец очереди?

Новые узлы добавляются в конец очереди, чтобы гарантировать, что узлы, которые были добавлены в очередь раньше, будут обработаны первыми. Это соответствует принципу FIFO и позволяет исследовать узлы на каждом уровне глубины перед тем, как перейти к следующему уровню.

3. Что происходит с узлами, которые дольше всего находятся в очереди в стратегии поиска в ширину?

Узлы, которые дольше всего находятся в очереди, будут обработаны первыми, когда очередь будет извлечена. Это происходит потому, что BFS обрабатывает узлы в порядке их добавления, и по мере извлечения узлов из очереди части графа, ближайшие к корневому узлу, будут исследованы первыми.

4. Какой узел будет расширен следующим после корневого узла, если используются правила поиска в ширину?

Следующий узел, который будет расширен, будет первым узлом, добавленным в очередь после корневого. Это зависит от порядка, в котором дочерние узлы были добавлены в очередь.

5. Почему важно расширять узлы с наименьшей глубиной в поиске в ширину?

Важно расширять узлы с наименьшей глубиной, чтобы гарантировать, что когда будет найдено решение, оно будет оптимальным, и вы получите кратчайший путь к целевому узлу.

6. Как временная сложность алгоритма поиска в ширину зависит от коэффициента разветвления и глубины?

Временная сложность BFS составляет $O(b^d)$, где b — коэффициент разветвления (максимальное количество дочерних узлов для одного узла), а d — глубина решения. Это означает, что время выполнения алгоритма растет экспоненциально с увеличением количества дочерних узлов и глубины решения.

7. Каков основной фактор, определяющий пространственную сложность алгоритма поиска в ширину?

Основной фактор, определяющий пространственную сложность алгоритма BFS, — это максимальное количество узлов, которые могут находиться в очереди одновременно. Это количество связано с коэффициентом разветвления и глубиной. Пространственная сложность также составляет $O(b^d)$.

8. В каких случаях поиск в ширину считается полным?

Поиск в ширину считается полным, если он гарантирует нахождение решения, если оно существует. BFS полный, если граф конечен или если у нас есть возможность обнаруживать циклы и избегать их.

9. Объясните, почему поиск в ширину может быть неэффективен с точки зрения памяти.

Поиск в ширину неэффективен с точки зрения памяти, потому что он хранит все узлы на текущем уровне и все узлы на следующем уровне в памяти, что может быстро потреблять большое количество памяти, особенно при большом коэффициенте разветвления.

10. В чем заключается оптимальность поиска в ширину?

Поиск в ширину оптимален в том смысле, что если существует решение, то он найдет его в кратчайшем возможном пути, потому что он исследует все узлы на одном уровне перед переходом на следующий уровень.

11. Какую задачу решает функция ``breadth_first_search``?

Функция ``breadth_first_search`` решает задачу поиска решения в графе или дереве, начиная с корневого узла и исследуя узлы по уровням, чтобы найти целевой узел.

12. Что представляет собой объект ``problem``, который передается в функцию?

Объект ``problem`` представляет собой задачу, включающую в себя начальное состояние, цель и, возможно, функцию перехода, которая описывает, как переходить от одного состояния к другому.

13. Для чего используется узел ``Node(problem.initial)`` в начале функции?

Узел ``Node(problem.initial)`` инициализируется с начальным состоянием задачи и представляет собой стартовый узел для поиска. Он добавляется в очередь BFS и является базовой точкой для дальнейшего расширения поиска.

14. Что произойдет, если начальное состояние задачи уже является целевым?

Если начальное состояние уже является целевым, алгоритм немедленно найдет решение, и будет возвращен этот узел без дальнейших расширений.

15. Какую структуру данных использует ``frontier`` и почему выбрана именно очередь FIFO?

``frontier`` использует структуру данных очереди (FIFO), поскольку это позволяет расширять узлы в порядке их добавления, что важно для стратегии поиска в ширину, где необходимо исследовать узлы по уровням их глубины.

16. Какую роль выполняет множество ``reached``?

- Множество ``reached`` используется для отслеживания узлов, которые были уже обработаны или посещены, чтобы избежать повторных обработок и циклов.

17. Почему важно проверять, находится ли состояние в множестве ``reached``?

Важно проверять, находится ли узел в множестве ``reached``, чтобы избежать бесконечного цикла и повторной обработки узлов, что может привести к излишнему потреблению памяти и времени.

18. Какую функцию выполняет цикл ``whilefrontier``?

Цикл ``whilefrontier`` выполняет основную работу алгоритма BFS, периодически извлекая узел из очереди, расширяя его и добавляя его дочерние узлы в очередь, пока не будет найдено решение или очередь не станет пустой.

19. Что происходит с узлом, который извлекается из очереди в строке ``node = frontier.pop()``?

Когда узел извлекается из очереди, он становится текущим узлом, который будет расширен. Это означает, что будут проверены все его дочерние узлы и добавлены в очередь для дальнейшего исследования.

20. Какова цель функции ``expand(problem, node)``?

Функция ``expand(problem, node)`` возвращает дочерние узлы для данного узла, раскрывая возможности перехода к новым состояниям.

21. Как определяется, что состояние узла является целевым?

Состояние узла определяется как целевое, если оно совпадает с состоянием, описанным в целевой функции задачи, указанной в объекте ``problem``.

22. Что происходит, если состояние узла не является целевым, но также не было ранее достигнуто?

Если состояние узла не является целевым и ранее не было достигнуто, то его дочерние узлы будут добавлены в ``frontier``, и узел будет добавлен в множество ``reached`` для последующего отслеживания.

23. Почему дочерний узел добавляется в начало очереди с помощью ``appendleft(child)``?

Использование `'appendleft'` имеет смысл только в тех случаях, когда очередь реализована как двусторонняя. В типичном подходе BFS дочерние узлы добавляются в конец. Если вы хотите организовывать хуухаки (на данный момент) в другой порядок, следует использовать аналогичные возможности, предоставляемые данными структурами.

24. Что возвращает функция `'breadth_first_search'`, если решение не найдено?

Если решение не найдено, функция `'breadth_first_search'` обычно возвращает `'None'`, `'failure'`, или аналогичные значения, специфичные для реализации, указывающие на то, что целевой узел не был найден.

25. Каково значение узла `'failure'` и когда он возвращается?

Узел `'failure'` представляет собой специальный флаг или значение, указывающее на то, что алгоритм не нашёл целевой узел. Он возвращается в случае, если существуют все возможные узлы и все они были исследованы, но целевого узла не оказалось.