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»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Кожуховский Виктор Андреевич  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) | |
|  | | Проверил:  Воронкин Роман Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

Тема: Исследование поиска в ширину

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

Порядок выполнения работы:

1. Изучил теоретический материал работы.

2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал примеры лабораторной работы.

8. Решите задания лабораторной работы с помощью языка программирования Python и элементов программного кода лабораторной работы 1 (имя файла начинается с PR.AI.001.). Проверьте правильность решения каждой задачи на приведенных тестовых примерах.

Для задачи "Расширенный подсчет количества островов в бинарной матрице" подготовить собственную матрицу, для которой с помощью разработанной в предыдущем пункте программы, подсчитать количество островов.

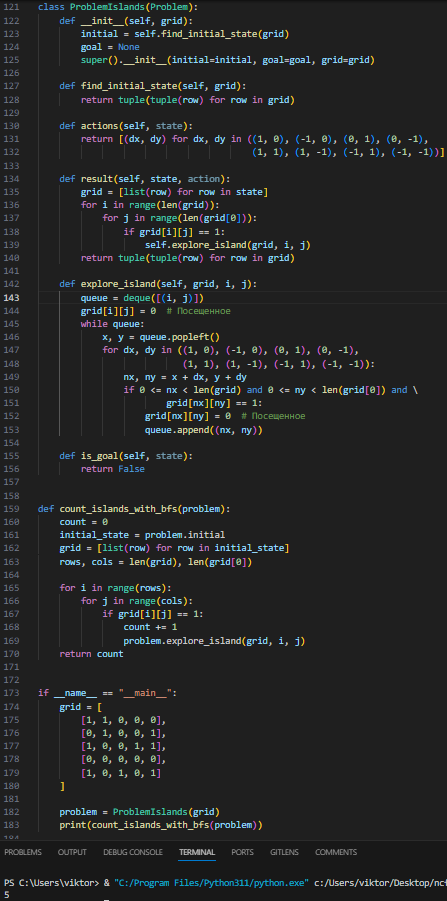


Рисунок 1. Решение задачи по поиску островов

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.

class ProblemLabyrinth(Problem):

def \_\_init\_\_(self, grid, initial, goal):

super().\_\_init\_\_(initial=initial, goal=goal, grid=grid)

def actions(self, state):

i, j = state

possible\_actions = []

directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]

for di, dj in directions:

ni, nj = i + di, j + dj

if (

0 <= ni < len(self.grid)

and 0 <= nj < len(self.grid[0])

and self.grid[ni][nj] == 1

):

possible\_actions.append((ni, nj))

return possible\_actions

def result(self, state, action):

return action

def is\_goal(self, state):

return state == self.goal

def breadth\_first\_search(problem):

node = Node(problem.initial)

if problem.is\_goal(problem.initial):

return node

frontier = FIFOQueue([node])

reached = {problem.initial}

while frontier:

node = frontier.pop()

for child in node.expand(problem):

s = child.state

if problem.is\_goal(s):

return child

if s not in reached:

reached.add(s)

frontier.appendleft(child)

return Node.failure

if \_\_name\_\_ == "\_\_main\_\_":

grid = [

[1, 1, 1, 1, 1, 0, 0, 1, 1, 1],

[0, 1, 1, 1, 1, 1, 0, 1, 0, 1],

[0, 0, 1, 0, 1, 1, 1, 0, 0, 1],

[1, 0, 1, 1, 1, 0, 1, 1, 0, 1],

[0, 0, 0, 1, 0, 0, 0, 1, 0, 1],

[1, 0, 1, 1, 1, 0, 0, 1, 1, 0],

[0, 0, 0, 0, 1, 0, 0, 1, 0, 1],

[0, 1, 1, 1, 1, 1, 1, 0, 0, 0],

[1, 1, 1, 1, 1, 0, 0, 1, 1, 1],

[0, 0, 1, 0, 0, 1, 1, 0, 0, 1],

]

initial = (0, 0)

goal = (5, 8)

problem = ProblemLabyrinth(grid, initial, goal)

solution = breadth\_first\_search(problem)

if solution is Node.failure:

print("Решение не найдено.")

else:

print("Решение найдено:")

print("Путь:", solution.path\_states())

print("Длина пути:", len(solution.path\_states()))



Рисунок 2. Решение задачи по поиску кратчайшего пути в лабиринте

Для задачи о льющихся кувшинах получить заданный объем воды в одном из кувшинов.

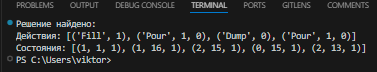


Рисунок 3. Решение задачи о льющихся кувшинах

10. Для построенного графа лабораторной работы 1 (имя файла начинается с PR.AI.001.) напишите программу на языке программирования Python, которая с использованием алгоритма поиска в ширину находит минимальное расстояние между начальным и конечным пунктами. Сравните найденное решение с решением, полученным вручную.

Рисунок 3. Код программы решения задания и его выполнение

11. Зафиксировал сделанные изменения в репозитории.

12. Выполнил слияние ветки для разработки с веткой master/main.

13. Отправил сделанные изменения на сервер GitHub.

Ссылка: https://github.com/Viktorkozh/AI\_2

Контрольные вопросы:

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

В стратегии поиска в ширину используется очередь типа "первым пришел - первым ушел" (FIFO).

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

Новые узлы в стратегии поиска в ширину добавляются в конец очереди, чтобы обеспечить порядок обработки узлов, начиная с наименее глубоких.

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

Узлы, которые дольше всего находятся в очереди, находятся впереди и обрабатываются в первую очередь.

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

Следующим для расширения будет узел, находящийся на том же уровне глубины, что и корневой узел, в данном случае узел C.

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

Важно расширять узлы с наименьшей глубиной в поиске в ширину, чтобы гарантировать нахождение решения на минимальной глубине.

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

Временная сложность алгоритма поиска в ширину зависит от коэффициента разветвления (b) и глубины наименее дорогого решения (d) по формуле O(b^(d+1)).

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

Основной фактор, определяющий пространственную сложность алгоритма поиска в ширину, — это максимальное количество узлов, которые необходимо хранить в памяти одновременно.

8. В каких случаях поиск в ширину считается полным?

Поиск в ширину считается полным, если коэффициент ветвления (b) конечен.

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

Поиск в ширину может быть неэффективен с точки зрения памяти из-за необходимости хранения всех узлов в памяти одновременно, что может привести к огромному количеству узлов.

10. В чем заключается оптимальность поиска в ширину?

Оптимальность поиска в ширину заключается в способности находить решение с наименьшей стоимостью среди всех возможных.

11. Какую задачу решает функция breadth\_first\_search?

Функция breadth\_first\_search решает задачу поиска решения для заданной задачи.

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

Объект problem, который передается в функцию, описывает задачу поиска.

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

Узел Node(problem.initial) в начале функции используется для создания начального узла поиска на основе начального состояния задачи.

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

Если начальное состояние задачи уже является целевым, то функция возвращает начальный узел как решение.

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

Frontier использует очередь FIFO, чтобы хранить узлы, которые нужно расширить, обеспечивая порядок обработки.

16. Какую роль выполняет множество reached ?

Множество reached выполняет роль отслеживания посещенных состояний, чтобы избежать повторного посещения одного и того же состояния.

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

Важно проверять, находится ли состояние в множестве reached, чтобы избежать повторного посещения одного и того же состояния и тем самым предотвратить зацикливание.

18. Какую функцию выполняет цикл while frontier ?

Цикл while frontier выполняет обработку узлов, пока в очереди есть узлы для обработки.

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

Узел, который извлекается из очереди в строке node = frontier.pop(), становится текущим узлом для расширения.

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

Цель функции expand(problem, node) — генерировать дочерние узлы для текущего узла.

21. Как определяется, что состояние узла является целевым?

Состояние узла является целевым, если оно удовлетворяет условию, определенному в методе is\_goal объекта problem.

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

Если состояние узла не является целевым, но также не было ранее достигнуто, оно добавляется в множество reached и дочерний узел добавляется в очередь frontier.

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

Дочерний узел добавляется в начало очереди с помощью appendleft(child) для обеспечения порядка обработки узлов.

24. Что возвращает функция breadth\_first\_search, если решение не найдено?

Если решение не найдено, функция breadth\_first\_search возвращает специальный узел failure.

25. Каково значение узла failure и когда он возвращается?

Значение узла failure — это индикатор того, что решение не было найдено, и он возвращается, когда все узлы были обработаны, но целевое состояние не было достигнуто.

Вывод: приобрел навыки по работе с поиском в ширину с помощью языка программирования Python версии 3.x