

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:
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.

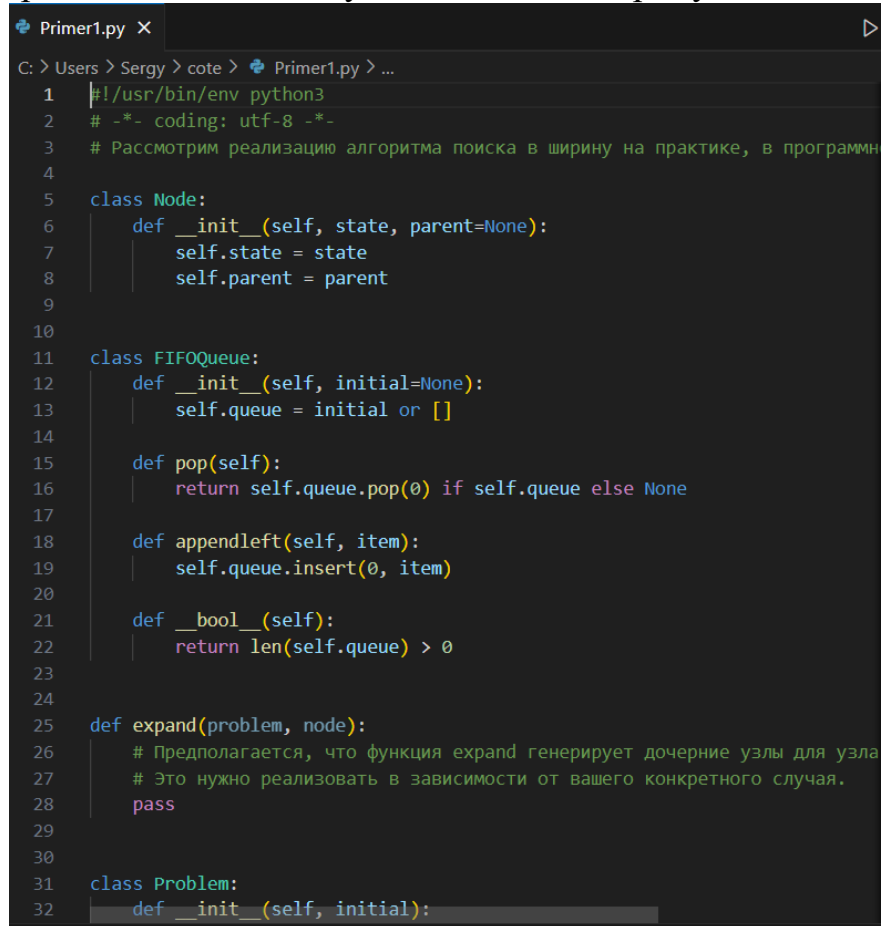
Исследование поиска в ширину.

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

Ссылка на репозиторий: <https://github.com/Alexander-its/LR2.git>

Ход работы:

Пример 1. Реализация на Python поиска в ширину.



```
Primer1.py X
C: > Users > Sergy > cote > Primer1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Рассмотрим реализацию алгоритма поиска в ширину на практике, в программн
4
5  class Node:
6      def __init__(self, state, parent=None):
7          self.state = state
8          self.parent = parent
9
10
11 class FIFOQueue:
12     def __init__(self, initial=None):
13         self.queue = initial or []
14
15     def pop(self):
16         return self.queue.pop(0) if self.queue else None
17
18     def appendleft(self, item):
19         self.queue.insert(0, item)
20
21     def __bool__(self):
22         return len(self.queue) > 0
23
24
25 def expand(problem, node):
26     # Предполагается, что функция expand генерирует дочерние узлы для узла
27     # Это нужно реализовать в зависимости от вашего конкретного случая.
28     pass
29
30
31 class Problem:
32     def __init__(self, initial):
```

Рисунок 1 – Код для выполнения примера 1

Задание 1. Расширенный подсчет количества островов в бинарной матрице.

```
IDZ1.py X
C: > Users > Sergy > cote > IDZ1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Расширенный подсчет количества островов в бинарной матрице.
4
5  from collections import deque
6
7  # Ниже перечислены все восемь возможных перемещений из ячейки.
8  # (верхний, правый, нижний, левый и четыре диагональных хода)
9  row = [-1, -1, -1, 0, 1, 0, 1, 1]
10 col = [-1, 1, 0, -1, -1, 1, 0, 1]
11
12
13 # Функция проверки безопасного перехода в позицию (x, y)
14 # с текущей позиции. Функция возвращает false, если (x, y)
15 # недействительные матричные координаты или (x, y) представляет воду или
16 # Позиция (x, y) уже обработана.
17
18 def isSafe(mat, x, y, processed):
19     return (x >= 0 and x < len(processed)) and (y >= 0 and y < len(processed)) and \
20         mat[x][y] == 1 and not processed[x][y]
21
22
23 def BFS(mat, processed, i, j):
24     # создает пустую очередь и ставит в очередь исходный узел
25     q = deque()
26     q.append((i, j))
27
```

Рисунок 2 – Код для выполнения задания 1

Задание 2. Поиск кратчайшего пути в лабиринте.

```
IDZ2.py X
C: > Users > Sergy > cote > IDZ2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # Поиск кратчайшего пути в лабиринте.
4
5  from collections import deque
6
7  # Функция для поиска кратчайшего пути в лабиринте
8  def bfs(maze, start, goal):
9      # Дирекции для перемещения (верх, низ, влево, вправо)
10     directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
11     queue = deque([start])
12     visited = {start: None} # Словарь для отслеживания предков
13
14     while queue:
15         current = queue.popleft()
16
17         # Если достигли цели, строим путь
18         if current == goal:
19             path = []
20             while current is not None:
21                 path.append(current)
22                 current = visited[current]
23             return path[::-1] # Возвращаем путь в обратном порядке
24
25         # Проверка соседних клеток
26         for d in directions:
27             neighbor = (current[0] + d[0], current[1] + d[1])
28             if (0 <= neighbor[0] < len(maze) and
29                 0 <= neighbor[1] < len(maze[0]) and
```

Рисунок 3 - Код для выполнения задания 2

Задание 3. Алгоритм поиска в ширину (минимальное расстояние между начальным и конечным пунктами).

```

IDZ3.py
C: > Users > Sergy > cote > IDZ3.py > ...
1  from collections import deque
2
3
4  def bfs_min_distance(distance_matrix, start, end):
5      """Поиск минимального расстояния с помощью алгоритма BFS."""
6
7      # Количество городов
8      num_cities = len(distance_matrix)
9
10     # Очередь для BFS
11     queue = deque([start])
12
13     # Список для хранения расстояний
14     distances = [float('inf')] * num_cities
15     distances[start] = 0
16
17     # Массив для отслеживания посещенных узлов
18     visited = [False] * num_cities
19     visited[start] = True
20
21     while queue:
22         current = queue.popleft()
23         for neighbor in range(num_cities):
24             if distance_matrix[current][neighbor] > 0 and not visited[neighbor]:
25                 visited[neighbor] = True
26                 distances[neighbor] = distances[current] + distance_matrix[current][neighbor]
27                 queue.append(neighbor)
28
29     return distances[end] if distances[end] != float('inf') else -1 # -1
30
31

```

Рисунок 4 - Код для выполнения задания 5

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