Исследование поиска в глубину

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

```
(venv) PS C:\!lab3\lab3> python flood_fill.py
Исходная матрица:
['Y', 'Y', 'Y', 'G', 'G', 'G', 'G', 'G', 'G', 'G']
['Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'G', 'X', 'X', 'X']
['G', 'G', 'G', 'G', 'G', 'G', 'G', 'X', 'X', 'X']
['W', 'W', 'W', 'W', 'W', 'G', 'G', 'G', 'G', 'X']
['W', 'R', 'R', 'R', 'R', 'R', 'G', 'X', 'X', 'X']
['W', 'W', 'W', 'R', 'R', 'G', 'G', 'X', 'X', 'X']
['W', 'B', 'W', 'R', 'R', 'R', 'R', 'R', 'R', 'X']
['W', 'B', 'B', 'B', 'B', 'R', 'R', 'X', 'X', 'X']
['W', 'B', 'B', 'X', 'B', 'B', 'B', 'B', 'X', 'X']
['W', 'B', 'B', 'X', 'X', 'X', 'X', 'X', 'X', 'X']

Матрица после заливки:
['Y', 'Y', 'Y', 'G', 'G', 'G', 'G', 'G', 'G', 'G']
['Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'G', 'C', 'C', 'C']
['G', 'G', 'G', 'G', 'G', 'G', 'G', 'C', 'C', 'C']
['W', 'W', 'W', 'W', 'W', 'G', 'G', 'G', 'G', 'C']
['W', 'R', 'R', 'R', 'R', 'R', 'G', 'C', 'C', 'C']
['W', 'W', 'W', 'R', 'R', 'G', 'G', 'C', 'C', 'C']
['W', 'B', 'W', 'R', 'R', 'R', 'R', 'R', 'R', 'C']
['W', 'B', 'B', 'B', 'B', 'R', 'R', 'C', 'C', 'C']
['W', 'B', 'B', 'C', 'B', 'B', 'B', 'B', 'C', 'C']
['W', 'B', 'B', 'C', 'C', 'C', 'C', 'C', 'C', 'C']
Алгоритм начал работу с точки (3, 9)
Целевой цвет для замены: 'X'
Цвет замены: 'C'
```

Рисунок 1 - Результат задачи Алгоритм заливки

```
(venv) PS C:\!lab3\lab3> python longest_path_matrix.py
Матрица (5x5):
C D E F G
B C D E F
A B C D E
B C D E F
C D E F G

Длина самого длинного пути, начинающегося с 'C': 5
```

Рисунок 2 – Результат задачи поиска самого длинного пути в матрице

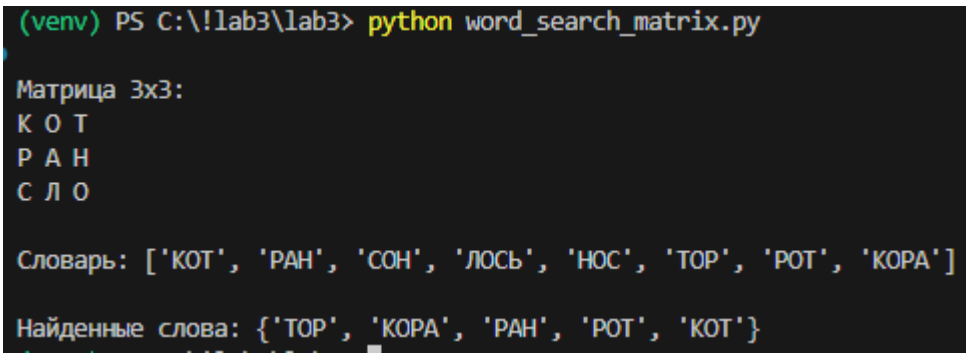

Рисунок 3 - Результат задачи генерирования списка возможных слов из матрицы символов

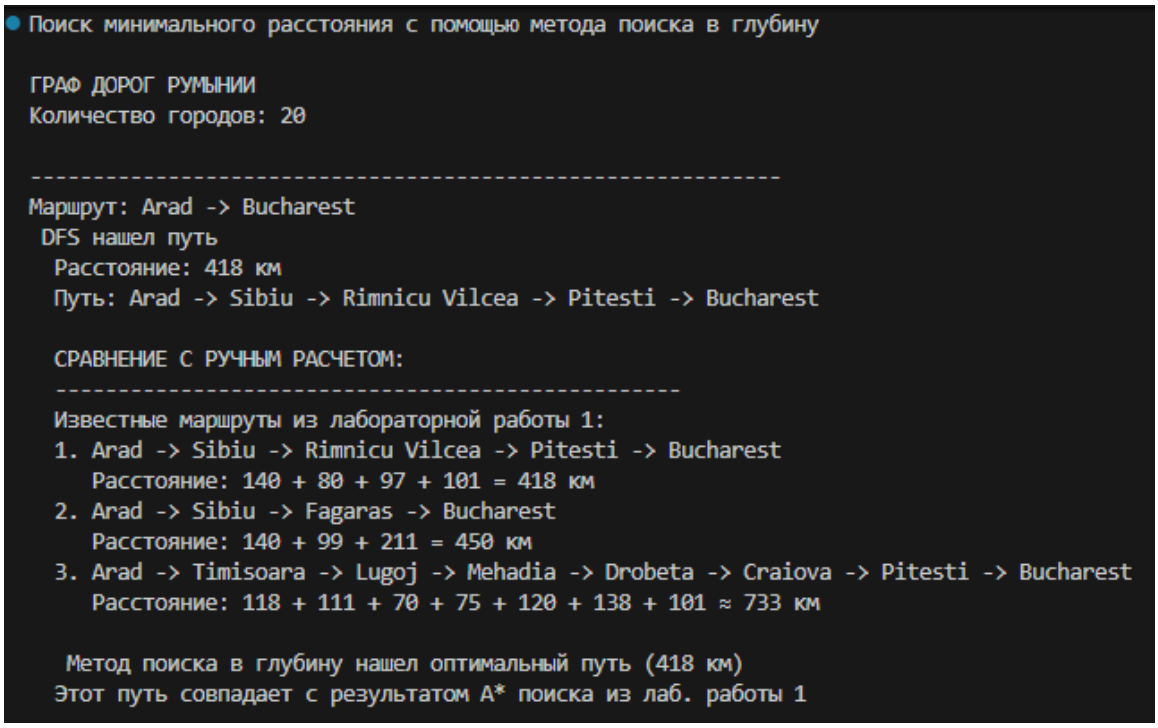

Рисунок 4 - Результат поиска минимального расстояния между пунктами в графе Румынии с помощью алгоритма поиска в глубину (DFS)

Ответы на вопросы:

1. Поиск в глубину исследует путь до конца перед переходом к соседним узлам, а поиск в ширину исследует все соседние узлы перед углублением.
2. Полнота, оптимальность, сложность по времени и сложность по памяти.
3. Генерируются все возможные дочерние узлы, которые добавляются в начало границы поиска.
4. Потому что это позволяет алгоритму сначала исследовать самые глубокие узлы, что соответствует стратегии углубления.

5. DFS хранит только текущий путь, освобождая память при возврате, что эффективнее BFS, который хранит все узлы уровня.
6. Только узлы на текущем пути от корня до листа.
7. При наличии бесконечных путей или при отсутствии проверки на циклы.
8. Временная сложность экспоненциально зависит от максимальной глубины дерева.
9. Потому что он находит первое попавшееся решение, не гарантируя его минимальную стоимость.
10. При ограниченной памяти или когда решение находится на большой глубине.
11. Функция выполняет рекурсивный поиск в глубину, принимая задачу и начальный узел.
12. Она инициализирует начальный узел, если он не передан.
13. Когда состояние узла соответствует целевой цели задачи.
14. Чтобы избежать зацикливания на уже посещенных узлах.
15. Возвращает значение failure, указывающее на неудачу.
16. Она рекурсивно вызывает себя для каждого дочернего узла в цикле.
17. Используется рекурсия с возвратом при неудаче.
18. Функция вернет значение failure.
19. Для последовательного исследования всех возможных путей в глубину.
20. Она генерирует дочерние узлы на основе возможных действий из текущего узла.
21. Она предотвращает бесконечную рекурсию при зацикливании.
22. Потому что она позволяет вернуть найденное решение без лишних продолжений поиска.
23. При отсутствии решения или при обнаружении цикла.
24. Рекурсивная реализация использует стек вызовов, а итеративная явный стек.
25. Может возникнуть бесконечная рекурсия или исчерпание стека вызовов.

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