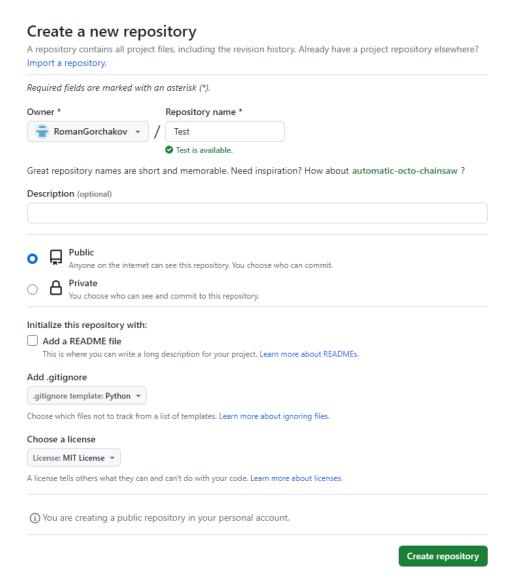
Лабораторная работа 2

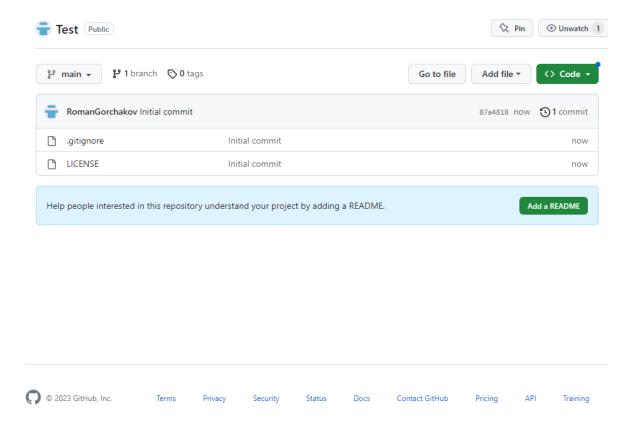
Тема: Исследование поиска в ширину.

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

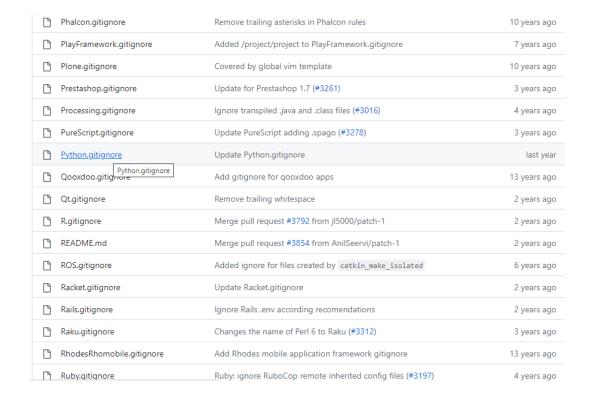
Порядок выполнения работы

1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия МІТ и язык программирования Python.



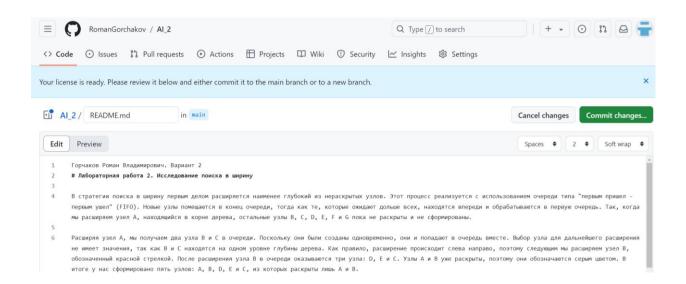


2. Теперь необходимо дополнить файл .gitignore с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «https://github.com/github/gitignore» и скачиваем оттуда файл «Python.gitignore».



```
# Byte-compiled / optimized / DLL files
       __pycache__/
 3
      *.py[cod]
      *$py.class
      # C extensions
 7
      *.50
      # Distribution / packaging
10
      .Python
      build/
11
     develop-eggs/
      dist/
13
      downloads/
15
      eggs/
16
      .eggs/
17
      lib/
18
      lib64/
19
      parts/
20
      sdist/
21
      var/
      wheels/
23
     share/python-wheels/
24
      *.egg-info/
25
       .installed.cfg
26
      *.egg
      MANIFEST
27
28
```

3. Теперь создаём файл «README.md», где вносим ФИО и теоретический конспект лекции. Сохраняем набранный текст через кнопку «Commit changes».



4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой

репозиторий. После этого организуем репозиторий в соответствие с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout —b develop» для создания ветки разработки; «git branch feature_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix. Устанавливаем библиотеки isort, black и flake8 и создаём файлы .pre-commit-config.yaml и environment.yml.

```
    @RomanGorchakov →/workspaces/AI_2 (main) $ git checkout -b develop Switched to a new branch 'develop'
    @RomanGorchakov →/workspaces/AI_2 (develop) $ git branch feature_branch
    @RomanGorchakov →/workspaces/AI_2 (develop) $ git branch release/1.0.0
    @RomanGorchakov →/workspaces/AI_2 (develop) $ git checkout main Switched to branch 'main'
    Your branch is up to date with 'origin/main'.
    @RomanGorchakov →/workspaces/AI_2 (main) $ git branch hotfix
    @RomanGorchakov →/workspaces/AI_2 (main) $ git checkout develop Switched to branch 'develop'
    @RomanGorchakov →/workspaces/AI_2 (develop) $ [
```

```
Collecting black
   Downloading black-24.10.0-cp312-cp312-manylinux 2 17 x86 64.manylinux2014 x86 64.manylinux 2 28 x86 64.whl.metadata (79 kB)
 Collecting click>=8.0.0 (from black)
   Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
 Collecting mypy-extensions>=0.4.3 (from black)
   Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
 Requirement already satisfied: packaging>=22.0 in /home/codespace/.local/lib/python3.12/site-packages (from black) (24.1)
 Collecting pathspec>=0.9.0 (from black)
   Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
 Requirement already satisfied: platformdirs>=2 in /home/codespace/.local/lib/python3.12/site-packages (from black) (4.3.6)
 Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (1.8 MB)
                                              1.8/1.8 MB 33.9 MB/s eta 0:00:00
 Downloading click-8.1.7-py3-none-any.whl (97 kB)
 Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
 Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
 Installing collected packages: pathspec, mypy-extensions, click, black
 Successfully installed black-24.10.0 click-8.1.7 mypy-extensions-1.0.0 pathspec-0.12.1
• @RomanGorchakov →/workspaces/AI_2 (develop) $ pip install flake8
 Collecting flake8
   Downloading flake8-7.1.1-py2.py3-none-any.whl.metadata (3.8 kB)
 Collecting mccabe<0.8.0,>=0.7.0 (from flake8)
   Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
 Collecting pycodestyle<2.13.0,>=2.12.0 (from flake8)
   Downloading pycodestyle-2.12.1-py2.py3-none-any.whl.metadata (4.5 kB)
 Collecting pyflakes<3.3.0,>=3.2.0 (from flake8)
   Downloading pyflakes-3.2.0-py2.py3-none-any.whl.metadata (3.5 kB)
 Downloading flake8-7.1.1-py2.py3-none-any.whl (57 kB)
 Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
 Downloading pycodestyle-2.12.1-py2.py3-none-any.whl (31 kB)
 Downloading pyflakes-3.2.0-py2.py3-none-any.whl (62 kB)
 Installing collected packages: pyflakes, pycodestyle, mccabe, flake8
 Successfully installed flake8-7.1.1 mccabe-0.7.0 pycodestyle-2.12.1 pyflakes-3.2.0
• @RomanGorchakov →/workspaces/AI_2 (develop) $ pre-commit sample-config > .pre-commit-config.yaml

    @RomanGorchakov →/workspaces/AI_2 (develop) $ conda env export > environment.yml
    @RomanGorchakov →/workspaces/AI_2 (develop) $ [
```

5. Создаём файл «PR.AI.001_1.py», в котором нужно подсчитать общее количество единиц в бинарной матрице.

```
© Command Prompt × + ∨ − □ ×

Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\cd C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\2\Задания\Задание 1

C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\2\Задания\Задание 1>руthon PR.AI.001_1.ру
Общее количество островов: 5

C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\2\Задания\Задание 1>
```

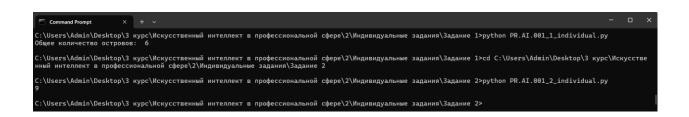
6. Создаём файл «PR.AI.001_2.py», в котором нужно найти кратчайший путь через лабиринт, используя алгоритм поиска в ширину.

```
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\2\Задания\Задание 2>python PR.AI.001_2.py
12
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\2\Задания\Задание 2>
```

7. Создаём файл «PR.AI.001_3.py», в котором нужно реализовать алгоритм поиска в ширину (BFS) для решения задачи о льющихся кувшинах, где цель состоит в том, чтобы получить заданный объем воды в одном из кувшинов.

```
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\2\Задания\Задание 3>python PR.AI.001_3.py [('Fill', 1), ('Pour', 1, 0), ('Dump', 0), ('Pour', 1, 0)] [(1, 1, 1), (1, 16, 1), (2, 15, 1), (0, 15, 1), (2, 13, 1)]
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\2\Задания\Задание 3>
```

8. Создаём файлы «PR.AI.001_1_individual.py» И «PR.AI.001_2_individual.py», файлам «PR.AI.001_1.py» идентичные И «PR.AI.001_2.py» использующие соответственно, ДЛЯ расчётов НО пользовательские данные.



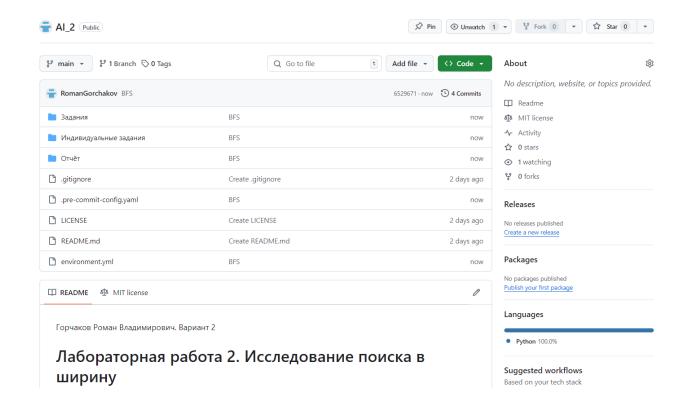
9. Создаём файл «PR.AI.001_3_individual.py», в котором нужно найти минимальное расстояние между начальным и конечным пунктами для построенного графа лабораторной работы 1 с использованием алгоритма поиска в ширину.



10. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.

```
.pre-commit-config.yaml
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      40 ++++++++++++++++
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      environment.yml
            .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/PR.AI.001_1.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/PR.AI.001 2.py"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265\27P\R.AI.001_2.py
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265\3/PR.AI.001_3.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265\3/PR.AI.001_1 individual.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265\3/PR.AI.001_2 individual.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265\3/PR.AI.001_3 individual.py"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      3\32\2402_\32\23\32\276\32\\200\321\200\321\207\32\207\32\200\320\276\32\202\32\202\32\200\32\200\32\
    ...3\\\220\\220\\223\\230\\\276\\321\\200\\321\\207\\320\\260\\320\\272\\320\\276\\320\\262\\320\\240\\320\\222.docx" | Bin 0 -> 1486858 bytes

9 files changed, 555 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 environment.yml
create mode 100644 "\\320\\227\\320\\260\\320\\260\\320\\260\\320\\275\\320\\270\\321\\217\\320\\260\\320\\260\\320\\260\\320\\275\\320\\270\\320\\260\\320\\260\\320\\275\\320\\270\\320\\260\\320\\260\\320\\275\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\270\\320\\27
            create \ mode \ 100644 \ \ "\ 320\ 236\ 321\ 202\ 321\ 202\ 321\ 202\ 321\ 202\ 320\ 233\ 320\ 240\ 233\ 320\ 276\ 320\ 223\ 320\ 276\ 320\ 272\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276\ 320\ 276
          .docx"
  ● @RomanGorchakov →/workspaces/AI_2 (main) $ git push -u
       Enumerating objects: 21, done
         Counting objects: 100% (21/21), done.
      Delta compression using up to 2 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (20/20), 1.24 MiB | 4.50 MiB/s, done.
Total 20 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
       remote: Resolving deltas: 100% (2/2), done. To https://github.com/RomanGorchakov/AI_2
                    de94110..6529671 main -> main
       branch 'main' set up to track 'origin/main'
@RomanGorchakov →/workspaces/AI_2 (main) $ []
```



Контрольные вопросы

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

В стратегии поиска в ширину первым делом расширяется наименее глубокий из нераскрытых узлов. Этот процесс реализуется с использованием очереди типа "первым пришел – первым ушел".

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

Новые узлы в стратегии поиска в ширину добавляются в конец очереди для эффективной планировки дальнейшего исследования этого узла, но не прежде, чем будут исследованы остальные вершины из списка смежности текущего узла.

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

Они находятся впереди и обрабатываются в первую очередь.

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

Расширяя узел A, мы получаем два узла B и C в очереди. Поскольку они были созданы одновременно, они и попадают в очередь вместе. Выбор узла для дальнейшего расширения не имеет значения, так как B и C находятся на одном уровне глубины дерева.

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

Это позволяет быстрее находить кратчайший путь. Такой поиск сначала изучает ближайшие узлы и затем переходит всё дальше в сторону от исходной точки.

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

Формула временной сложности: $b + b^2 + b^3 + \dots + b^d = (b^{d+1} - 1)/(b - 1) = O(b^{d+1})$ b - коэффициент разветвления, <math>d - глубина наименее дорогого решения.

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

Основной фактор, определяющий пространственную сложность алгоритма поиска в ширину, – количество развёрнутых узлов, которые хранятся в памяти.

8. В каких случаях поиск в ширину считается полным?

Поиск в ширину, выполняясь достаточно долго, исследует все дерево поиска. Следовательно, если решение находится где-то в этом дереве, алгоритм обязательно его найдет, что делает его полным.

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

Хранение всех узлов в памяти одновременно невозможно на практике из-за их огромного количества. Это делает алгоритм как времязатратным, так и пространственно неэффективным, хотя основная проблема заключается именно в необходимости большого объема памяти.

10. В чем заключается оптимальность поиска в ширину?

Оптимальность подразумевает способность алгоритма находить решение с наименьшей стоимостью среди всех возможных. Оптимальный алгоритм всегда

предоставляет решение с минимальными затратами. Это не означает, что алгоритм самый быстрый или экономичный по памяти, но он гарантирует нахождение решения с наименьшей стоимостью.

11. Какую задачу решает функция breadth first search?

Функция breadth_first_search является реализацией алгоритма поиска в ширину.

- 12. Что представляет собой объект problem, который передается в функцию? Объект problem описывает задачу поиска.
- 13. Для чего используется узел Node(problem.initial) в начале функции?

Создается начальный узел поиска, используя начальное состояние задачи problem.initial.

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

Если начальное состояние является целевым, то возвращается начальный узел как решение.

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

Очередь frontier имеет структуру данных типа FIFO (First-In-First-Out). Эта очередь будет использоваться для хранения узлов, которые нужно расширить.

16. Какую роль выполняет множество reached?

Множество reached нужно для отслеживания посещенных состояний.

- 17. Почему важно проверять, находится ли состояние в множестве reached? Чтобы избежать повторного посещения одного и того же состояния.
- 18. Какую функцию выполняет цикл while frontier?

Цикл while frontier продолжается, пока в границе есть узлы для обработки. Она извлекает узел из очереди для расширения и ищет целевой узел среди дочерних.

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

Для каждого дочернего узла текущего узла node (расширение узла) получаем состояние дочернего узла.

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

Она ищет целевой узел среди дочерних. Если состояние дочернего узла является целевым, возвращается этот узел как решение. Если состояние еще не было достигнуто (не в множестве reached), добавляем состояние в множество достигнутых и дочерний узел в начало очереди frontier. Если решение не найдено, возвращается специальный узел failure.

21. Как определяется, что состояние узла является целевым? if problem.is_goal(s): return child