

# Лабораторная работа 3

Тема: Исследование поиска в глубину.

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

Ссылка на GitHub: [https://github.com/RomanGorchakov/AI\\_3](https://github.com/RomanGorchakov/AI_3)

## Порядок выполнения работы


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

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).



Owner \*      Repository name \*

 RomanGorchakov / Test

✔ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)


Choose a license



License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

 **Test** Public

 Pin  Unwatch 1

 main  1 branch  0 tags

Go to file

Add file 

 Code 

 **RomanGorchakov** Initial commit 87a4818 now  1 commit

 .gitignore

Initial commit

now

 LICENSE

Initial commit


















now

Help people interested in this repository understand your project by adding a README.

Add a README

 © 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#)

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».

	Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
	PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
	Plone.gitignore	Covered by global vim template	10 years ago
	Prestashop.gitignore	Update for Prestashop 1.7 ( <a href="#">#3261</a> )	3 years ago
	Processing.gitignore	Ignore transpiled .java and .class files ( <a href="#">#3016</a> )	4 years ago
	PureScript.gitignore	Update PureScript adding .spago ( <a href="#">#3278</a> )	3 years ago
	<a href="#">Python.gitignore</a>	Update Python.gitignore	last year
	Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
	Qt.gitignore	Remove trailing whitespace	2 years ago
	R.gitignore	Merge pull request <a href="#">#3792</a> from jl5000/patch-1	2 years ago
	README.md	Merge pull request <a href="#">#3854</a> from AnilSeervi/patch-1	2 years ago
	ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
	Racket.gitignore	Update Racket.gitignore	2 years ago
	Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
	Raku.gitignore	Changes the name of Perl 6 to Raku ( <a href="#">#3312</a> )	3 years ago
	RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
	Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files ( <a href="#">#3197</a> )	4 years ago

```

1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28

```

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

The screenshot shows the GitHub web interface for the repository 'RomanGorchakov / AI\_3'. The 'README.md' file is open in the 'main' branch. The file content is as follows:

```

1  Горчаков Роман Владимирович. Вариант 2
2  # Лабораторная работа 3. Исследование поиска в глубину
3
4  Поиск в глубину, подобно поиску в ширину, не предполагает использования информации, специфичной для задачи, за исключением данных, предоставленных в определении задачи. Это определение обычно включает в себя: пространство состояний,
5  функцию преемников, стоимость действий, начальное и целевое состояния. Любое предоставление дополнительной информации переводит поиск в категорию информированных.
6
7  Отличие поиска в глубину от поиска в ширину заключается в порядке расширения узлов, хотя оба метода представляют собой вариации поиска по дереву, этот процесс реализуется с использованием очереди типа "последний пришел - первым ушел"
8  (LIFO).
9
10 Мы начинаем с корневого узла A, который является единственным доступным для расширения. После расширения узла A он становится серым, а узлы B и C – его дочерними узлами, определенными функцией преемников.
11
12 Следующим этапом является выбор узла из списка на рассмотрение и его расширение. Проверив, что узел B не соответствует целевому состоянию, мы расширяем его, аналогично поиску в ширину. Теперь узел B становится серым, а узлы D, E и C
13 добавляются в список на рассмотрение. Здесь проявляется ключевое различие между поиском в глубину и в ширину: в отличие от поиска в ширину, который вернется бы к узлу C, поиск в глубину расширит самый глубокий из нерасширенных узлов.
14
15 Следующим для расширения становится один из узлов D или E. По канону, мы двигаемся слева направо, как в случае с D и E, поэтому рядом с D помещается красный треугольник, указывающий на следующий узел для расширения. Проверив, не является
16 ли он целевым узлом, мы добавляем его дочерние узлы в дерево. Теперь на границе находятся четыре узла: H, I, E и C, и мы выбираем один из самых глубоких для следующего расширения.
17
18 Если при поиске в глубину мы обнаруживаем, что текущий узел не соответствует целевому состоянию и является конечным узлом дерева, без возможности дальнейшего расширения из-за отсутствия дочерних узлов, мы переходим к следующему узлу из
19 списка на рассмотрение. В данном случае это узел I, выбираемый в соответствии с принципом максимальной глубины. Таким образом, узлы E и C не будут следующими кандидатами для расширения, так как они расположены на меньшей глубине. Стоит
20 отметить, что узел H теперь окрашен в черный цвет, что символизирует возможность его удаления из памяти.

```

The interface includes a search bar, navigation tabs (Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings), a license notice, and a commit button.

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 соответствии 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_3 (main) $ git checkout -b develop
Switched to a new branch 'develop'
• @RomanGorchakov → /workspaces/AI_3 (develop) $ git branch feature_branch
• @RomanGorchakov → /workspaces/AI_3 (develop) $ git branch release/1.0.0
• @RomanGorchakov → /workspaces/AI_3 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @RomanGorchakov → /workspaces/AI_3 (main) $ git branch hotfix
• @RomanGorchakov → /workspaces/AI_3 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov → /workspaces/AI_3 (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 56.4 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_3 (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_3 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
• @RomanGorchakov → /workspaces/AI_3 (develop) $ conda env export > environment.yml
○ @RomanGorchakov → /workspaces/AI_3 (develop) $
```

5. Создаём файл «PR.AI.001\_1.py», в котором нужно заменить один цвет ячеек матрицы на другой.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 1>python PR.AI.001_1.py
['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', 'C', 'C', 'C']
['W', 'B', 'B', 'C', 'C', 'C', 'C', 'C', 'C', 'C']
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 1>
```

6. Создаём файл «PR.AI.001\_2.py», в котором нужно найти самый длинный путь от одной ячейки матрицы до другой.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 1>cd C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 2
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 2>python PR.AI.001_2.py
6
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 2>
```

7. Создаём файл «PR.AI.001\_3.py», в котором нужно сгенерировать список ВОЗМОЖНЫХ СЛОВ ИЗ МАТРИЦЫ СИМВОЛОВ.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 2>cd C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 3
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 3>python PR.AI.001_3.py
{'МАРС', 'ТОН'}
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Задания\Задание 3>
```

8. Создаём файлы «PR.AI.001\_1\_individual.py» и «PR.AI.001\_2\_individual.py», идентичные файлам «PR.AI.001\_1.py» и «PR.AI.001\_2.py» соответственно, но использующие для расчётов пользовательские данные.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 1>python PR.AI.001_1_individual.py
['Y', 'Y', 'Y', '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', 'P', 'W', 'R', 'R', 'R', 'R', 'R', 'X']
['W', 'P', 'P', 'P', 'P', 'R', 'R', 'X', 'X', 'X']
['W', 'P', 'P', 'X', 'P', 'P', 'P', 'X', 'X', 'X']
['W', 'P', 'P', 'X', 'X', 'X', 'X', 'X', 'X', 'X']

C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 1>cd C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 2
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 2>python PR.AI.001_2_individual.py
4
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 2>
```

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 глубину.

```
Command Prompt
C:\Users\Admin>cd C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 3
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 3>python PR.AI.001_3_individual.py
Кратчайший путь от Новоалександровск до Озёрный : Новоалександровск -> Присадовый -> Ударный -> Краснодарский -> Красночервоный -> Южный -> Расшеватская -> Славенский -> Озёрный
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\3\Индивидуальные задания\Задание 3>
```

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

```
Your branch is up to date with 'origin/main'.
@RomanGorchakov →/workspaces/AI_3 (main) $ git merge develop
Updating ca2d58c..cbaba2b
Fast-forward
.../320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/PR.AI.001_1.py" | 54 +++++
.../320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/PR.AI.001_2.py" | 54 +++++
.../320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/PR.AI.001_3.py" | 55 +++++
.../320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/PR.AI.001_1_individual.py" | 54 +++++
.../320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/PR.AI.001_2_individual.py" | 57 +++++
.../320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/PR.AI.001_3_individual.py" | 108 +++++
...33\320\2402_\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf" | Bin 0 -> 837557 bytes
7 files changed, 382 insertions(+)
create mode 100644 ".../320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/PR.AI.001_1.py"
create mode 100644 ".../320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/PR.AI.001_2.py"
create mode 100644 ".../320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/PR.AI.001_3.py"
create mode 100644 ".../320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 \320\267\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/PR.AI.001_1_individual.py"
create mode 100644 ".../320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 \320\267\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/PR.AI.001_2_individual.py"
create mode 100644 ".../320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 \320\267\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/PR.AI.001_3_individual.py"
create mode 100644 ".../320\236\321\202\321\207\321\221\321\202\320\233\320\2402_\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
@RomanGorchakov →/workspaces/AI_3 (main) $ git push -u
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (18/18), 753.65 KiB | 22.17 MiB/s, done.
Total 18 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/RomanGorchakov/AI_3
ca2d58c..cbaba2b main -> main
branch 'main' set up to track 'origin/main'.
@RomanGorchakov →/workspaces/AI_3 (main) $
```

AI\_3 Public

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

RomanGorchakov Taks cbaba2b · now 5 Commits

File	Operation	Time
Задания	Taks	now
Индивидуальные задания	Taks	now
Отчёт	Taks	now
.gitignore	Create .gitignore	15 minutes ago
.pre-commit-config.yaml	Deep Search	now
.python-version	Deep Search	now
LICENSE	Create LICENSE	15 minutes ago
README.md	Create README.md	8 minutes ago
edu.pyoop.code-workspace	Deep Search	now
environment.yml	Deep Search	now
pyproject.toml	Deep Search	now
uv.lock	Deep Search	now

README MIT license

Горчаков Роман Владимирович. Вариант 2

## Лабораторная работа 3. Исследование поиска в глубину

About: No description, website, or topics provided.

Releases: No releases published. [Create a new release](#)

Packages: No packages published. [Publish your first package](#)

Languages: Python 100.0%

Suggested workflows: Based on your tech stack

- Django: Build and Test a Django Project. [Configure](#)
- Python Package using Anaconda: [Configure](#)

## Контрольные вопросы

1. В чем ключевое отличие поиска в глубину от поиска в ширину?

Отличие поиска в глубину от поиска в ширину заключается в порядке расширения узлов, хотя оба метода представляют собой вариации поиска по дереву, этот процесс реализуется с использованием очереди типа "последним пришел - первым ушел" (LIFO).

2. Какие четыре критерия качества поиска обсуждаются в тексте для оценки алгоритмов?

Временная сложность, пространственная сложность, оптимальность и полнота.

3. Что происходит при расширении узла в поиске в глубину?

При расширении узла в поиске в глубину происходит обход всей ветки дерева и всех прилегающих узлов. Для этого используется подход "последним пришёл – первым ушёл", который реализуется через стек.

4. Почему поиск в глубину использует очередь типа "последним пришел – первым ушел" (LIFO)?

Поиск в глубину использует структуру данных стек, основанную на принципе "последним пришёл – первым ушёл", потому что для поиска данных в графе используется алгоритм DFS.

5. Как поиск в глубину справляется с удалением узлов из памяти, и почему это преимущество перед поиском в ширину?

После достижения самого глубокого узла все остальные узлы извлекаются из стека, который используется для отслеживания текущего узла. Затем происходит обход прилегающих узлов, которые ещё не посещались.

6. Какие узлы остаются в памяти после того, как достигнута максимальная глубина дерева?

Максимальное количество узлов, которые мы должны хранить в памяти во время поиска в глубину, равно произведению коэффициента ветвления на  $m$ .

7. В каких случаях поиск в глубину может "застрять" и не найти решение?

В случае конечного дерева поиск в глубину гарантированно найдёт решение, так как он исследует все ветви дерева. Однако в бесконечном дереве поиск в глубину может "застрять" в бесконечной ветви, таким образом, не рассмотрев другие потенциальные решения.

8. Как временная сложность поиска в глубину зависит от максимальной глубины дерева?

Если  $b$  – коэффициент ветвления, а  $m$  – максимальная глубина дерева, то общее количество узлов, сгенерированных поиском в глубину, составит  $b^m$ .

9. Почему поиск в глубину не гарантирует нахождение оптимального решения?

В то время как поиск в глубину эффективен в управлении памятью и может быть полезен в конечных деревьях, его неполнота в бесконечных деревьях и



потенциально высокая временная сложность делают его менее предпочтительным в некоторых случаях по сравнению с поиском в ширину.

10. В каких ситуациях предпочтительно использовать поиск в глубину, несмотря на его недостатки?

Использование поиска в глубину предпочтительно в следующих ситуациях, несмотря на его недостатки:

1) когда нет полных данных о графе. В таком случае поиск в глубину является практически оптимальным решением;

2) если известно, что путь решения будет длинным. В этом случае поиск в глубину не будет тратить время на поиск большого количества «поверхностных» состояний на графе;

3) для областей поиска с высокой степенью связности. Это связано с тем, что поиску в глубину не нужно помнить все узлы данного уровня.

11. Что делает функция `depth_first_recursive_search`, и какие параметры она принимает?

Функция принимает два параметра: `problem`, представляющий задачу, которую необходимо решить, и `node`, который является текущим узлом в процессе поиска. Если узел не указан (т.е., `None`), функция создает начальный узел с использованием начального состояния задачи.

12. Какую задачу решает проверка `if node is None`?

Если текущий узел не предоставлен (то есть, при первом вызове функции), создается новый узел `Node` с начальным состоянием `problem.initial`.

13. В каком случае функция возвращает узел как решение задачи?

Функция `problem_is_goal` проверяет, достигнуто ли целевое состояние. Если состояние текущего узла соответствует целевому состоянию задачи (`problem.is_goal(node.state)` возвращает `True`), поиск успешно завершается, и текущий узел возвращается как решение.

14. Почему важна проверка на циклы в алгоритме рекурсивного поиска в глубину?

Проверка на циклы в алгоритме рекурсивного поиска в глубину важна, потому что она позволяет избежать заикливания.

15. Что возвращает функция при обнаружении цикла?

Если текущий узел создает цикл, функция возвращает специальное значение `failure`, указывающее на неудачу в поиске пути.

16. Как функция обрабатывает дочерние узлы текущего узла?

Если текущий узел не является целевым и не создаёт цикл, функция генерирует всех дочерних узлов (`child`) путем расширения текущего узла (`expand(problem, node)`). Затем она рекурсивно вызывает себя для каждого дочернего узла. Если рекурсивный вызов возвращает не `failure`, то найдено решение, и оно возвращается.

17. Какой механизм используется для обхода дерева поиска в этой реализации?

Для обхода дерева поиска в рекурсивной реализации используется механизм стека. Стек позволяет запоминать маршрут к конечному узлу и обратно, реализуя подход "последним пришёл — первым ушёл". После достижения самого глубокого узла все остальные узлы извлекаются из стека, затем происходит обход прилегающих узлов, которые ещё не посещались.

18. Что произойдет, если не будет найдено решение в ходе рекурсии?

Функция возвращает `failure`.

19. Почему функция рекурсивно вызывает саму себя внутри цикла?

Рекурсивный алгоритм обхода графа принимает на вход некоторую вершину графа и рекурсивно запускает себя для всех ещё не посещённых соседей данной вершины.

20. Как функция `expand(problem, node)` взаимодействует с текущим узлом?

Если текущий узел не является целевым и не создает цикл, функция генерирует всех дочерних узлов (`child`) путем расширения текущего узла (`expand(problem, node)`). Затем она рекурсивно вызывает себя для каждого дочернего узла. Если рекурсивный вызов возвращает не `failure`, то найдено решение, и оно возвращается.

21. Какова роль функции `is_cycle(node)` в этом алгоритме?

Для предотвращения заикливания (когда алгоритм постоянно возвращается к уже посещенным узлам) используется проверка `is_cycle(node)`.

22. Почему проверка `if result` в рекурсивном вызове важна для корректной работы алгоритма?

Проверка `if result` в рекурсивном вызове важна для корректной работы алгоритма, потому что она позволяет при определённых условиях завершить функцию, то есть вернуть значение, а не выполнить очередной рекурсивный вызов.

23. В каких ситуациях алгоритм может вернуть `failure`?

Если функция достигает этой точки, это означает, что ни один из дочерних узлов не привел к решению. В этом случае функция возвращает `failure`, указывая на то, что решение не найдено.

24. Как рекурсивная реализация отличается от итеративного поиска в глубину?

Основное отличие в том, что рекурсивная реализация использует рекурсию для повторения действий на каждом шаге, а итеративная предполагает обход графа с использованием стека для задержки обработки некоторых узлов.

25. Какие потенциальные проблемы могут возникнуть при использовании этого алгоритма для поиска в бесконечных деревьях?

1) Невозможность посетить все узлы. Например, в двоичном дереве бесконечной глубины поиск в глубину будет двигаться вдоль одной стороны дерева, никогда не посетив остальные вершины.

2) Бесконечная рекурсия. Если из вершины А можно перейти в вершину В, то из вершины В можно перейти в вершину А, и рекурсия станет бесконечной.