

Лабораторная работа 5

Тема: Исследование поиска с ограничением глубины.

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

Ссылка на GitHub: https://github.com/RomanGorchakov/AI_5

Порядок выполнения работы

1. Создаём новый общедоступный репозиторий в Github, в котором будет использована лицензия MIT и язык программирования Python.

<>

 Start writing code

...

Start a new repository for RomanGorchakov

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

AI_5

✓ AI_5 is available.

☒ Public

Anyone on the internet can see this repository

☐ Private

You choose who can see and commit to this repository

Create a new repository

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

RomanGorchakov / README.md

Create

1 - 👋 Hi, I'm @RomanGorchakov

2 - 👀 I'm interested in ...

3 - 🌱 I'm currently learning ...

4 - ❤️ I'm looking to collaborate on ...

5 - 💻 How to reach me ...

6 - 🗨️ Pronouns: ...

7 - ⚡ Fun fact: ...

8

Add a license to your project

License	Permissions	Limitations	Conditions
Apache License 2.0	A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.		
GNU General Public License v3.0			
MIT License	<ul style="list-style-type: none"> ✓ Commercial use ✓ Modification ✓ Distribution ✓ Private use 	<ul style="list-style-type: none"> ✗ Liability ✗ Warranty 	<ul style="list-style-type: none"> ④ License and copyright notice
BSD 2-Clause "Simplified" License			
BSD 3-Clause "New" or "Revised" License	This is not legal advice. Learn more about repository licenses.		
Boost Software License 1.0			
Creative Commons Zero v1.0 Universal			
Eclipse Public License 2.0			
GNU Affero General Public License			

To adopt MIT License, enter your details. You'll have a chance to review before committing a `LICENSE` file to a new branch or the root of your project.

Year ④

Full name ④

Review and submit

MIT License

Copyright (c) Year Full name

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge,

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».

RomanGorchakov / AI_5

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Your license is ready. Please review it below and either commit it to the main branch or to a new branch.

AI_5 / .gitignore in main

Cancel changes Commit changes

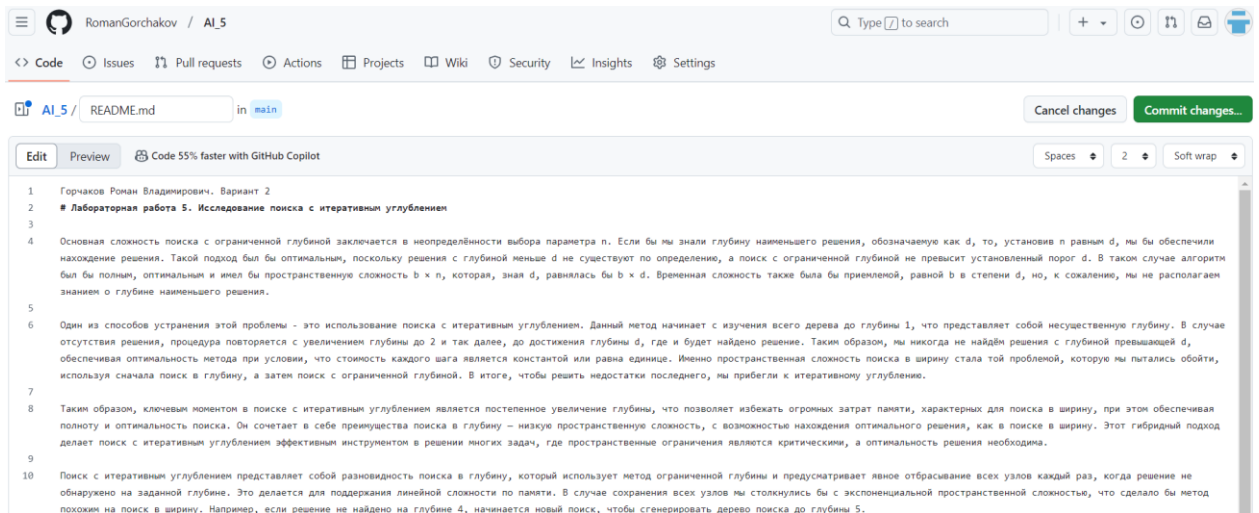
Edit Preview Choose .gitignore template: Python Code 55% faster with GitHub Copilot Spaces 2 No wrap

```

1 # Byte-compiled / optimized / DLL files
2 __pycache__/
3 *.py[co]
4 *.pyc
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/

```

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.

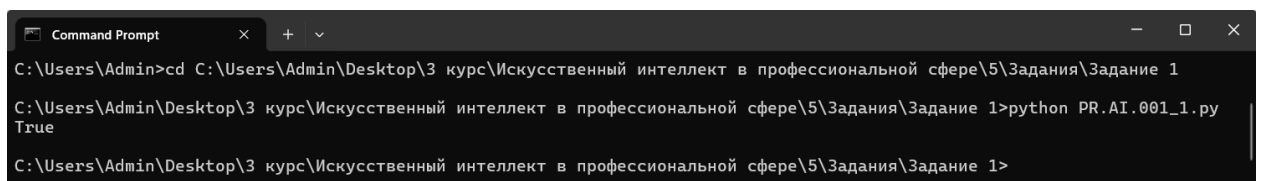
```
• @RomanGorchakov →/workspaces/AI_5 (main) $ git checkout -b develop
Switched to a new branch 'develop'
• @RomanGorchakov →/workspaces/AI_5 (develop) $ git branch feature_branch
• @RomanGorchakov →/workspaces/AI_5 (develop) $ git branch release/1.0.0
• @RomanGorchakov →/workspaces/AI_5 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @RomanGorchakov →/workspaces/AI_5 (main) $ git branch hotfix
• @RomanGorchakov →/workspaces/AI_5 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/AI_5 (develop) $
```

```

• @RomanGorchakov →/workspaces/AI_5 (develop) $ pip install black
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-8.1.8-py3-none-any.whl.metadata (2.3 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) (24.2)
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 60.5 MB/s eta 0:00:00
Downloading click-8.1.8-py3-none-any.whl (98 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-8.1.8 mypy_extensions-1.0.0 pathspec-0.12.1
• @RomanGorchakov →/workspaces/AI_5 (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_5 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
• @RomanGorchakov →/workspaces/AI_5 (develop) $ conda env export > environment.yml

```

5. Создаём файл «PR.AI.001_1.py», в котором нужно разработать метод поиска, который позволит проверить существование пользователя с заданным идентификатором в системе, используя структуру дерева и алгоритм итеративного углубления.



```

Command Prompt
C:\Users\Admin>cd C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Задания\Задание 1
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Задания\Задание 1>python PR.AI.001_1.py
True
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Задания\Задание 1>

```

6. Создаём файл «PR.AI.001_2.py», в котором нужно построить дерево, где каждый узел представляет каталог в файловой системе, а цель поиска — определенный файл, и найти путь от корневого каталога до каталога (или файла), содержащего искомый файл, используя алгоритм итеративного углубления.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Задания\Задание 1>cd C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Задания\Задание 2
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Задания\Задание 2>python PR.AI.001_2.py
dir -> dir3 -> files
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Задания\Задание 2>
```

7. Создаём файл «PR.AI.001_individual.py», в котором нужно найти все файлы, которые содержат в своём названии строку «backup», в большом файловом дереве, используя итеративное углубление и минимальную глубину для каждого нового уровня.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Индивидуальное задание>python PR.AI.001_individual.py
Введите корневую директорию для поиска: C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Индивидуальное задание
Найдены файлы:
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Индивидуальное задание\backup.txt
C:\Users\Admin\Desktop\3 курс\Искусственный интеллект в профессиональной сфере\5\Индивидуальное задание>
```

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

```
pyproject.toml
setup.cfg
uv.lock
.../\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"
.../PR.AI.001_individual.py"
.../backup.txt"
...33\320\2404\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
12 files changed, 755 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 .python-version
create mode 100644 edu.pyoop.code-workspace
create mode 100644 environment.yml
create mode 100644 pyproject.toml
create mode 100644 setup.cfg
create mode 100644 uv.lock
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\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\320\276\320\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265/PR.AI.001_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\320\276\320\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265/backup.txt"
create mode 100644 "\320\236\321\202\321\207\321\221\321\202\321\202\320\233\320\2404\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
Bin 0 -> 1147567 bytes
@RomanGorchakov -> /workspaces/AI_5 (main) $ git push -u
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (19/19), 1.05 MiB | 3.58 MiB/s, done.
Total 19 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/RomanGorchakov/AI_5
 1981d9b..e094d99 main -> main
branch 'main' set up to track 'origin/main'.
@RomanGorchakov -> /workspaces/AI_5 (main) $
```

The screenshot shows a GitHub repository interface. At the top, the repository name 'RomanGorchakov Iterative Deepening Search' is displayed with a commit hash 'e094d99' and '4 Commits'. Below this is a table of files and folders. The files include 'Задания', 'Индивидуальное задание', 'Отчёт', '.gitignore', '.pre-commit-config.yaml', '.python-version', 'LICENSE', 'README.md', 'edu.pyoop.code-workspace', 'environment.yml', 'pyproject.toml', 'setup.cfg', and 'uv.lock'. The right sidebar contains sections for 'About', 'Releases', 'Packages', 'Languages' (showing Python at 100.0%), and 'Suggested workflows' (including 'Publish Python Package' and 'Python application'). The main content area shows the README file, which contains the text: 'Горчаков Роман Владимирович. Вариант 2' and 'Лабораторная работа 5. Исследование поиска с итеративным углублением'.

File/Folder	Commit	Time
Задания	Iterative Deepening Search	now
Индивидуальное задание	Iterative Deepening Search	now
Отчёт	Iterative Deepening Search	now
.gitignore	Create .gitignore	3 hours ago
.pre-commit-config.yaml	Iterative Deepening Search	now
.python-version	Iterative Deepening Search	now
LICENSE	Create LICENSE	3 hours ago
README.md	Create README.md	3 hours ago
edu.pyoop.code-workspace	Iterative Deepening Search	now
environment.yml	Iterative Deepening Search	now
pyproject.toml	Iterative Deepening Search	now
setup.cfg	Iterative Deepening Search	now
uv.lock	Iterative Deepening Search	now

Контрольные вопросы

1. Что означает параметр n в контексте поиска с ограниченной глубиной, и как он влияет на поиск?

Параметр n в контексте поиска с ограниченной глубиной – число вершин в графе, которые нумеруются числами от 1 до n . Влияние параметра n на поиск заключается в том, что при использовании алгоритма поиска в глубину вершины нумеруются таким образом, что в любой момент исполнения алгоритма обрабатывается только одна вершина.

2. Почему невозможно заранее установить оптимальное значение для глубины d в большинстве случаев поиска?

Если бы мы знали глубину наименьшего решения, обозначаемую как d , то, установив n равным d , мы бы обеспечили нахождение решения. Такой подход был бы оптимальным, поскольку решения с глубиной меньше d не существуют по

определению, а поиск с ограниченной глубиной не превысит установленный порог d . В таком случае алгоритм был бы полным, оптимальным и имел бы пространственную сложность $b \times n$, которая, зная d , равнялась бы $b \times d$. Временная сложность также была бы приемлемой, равной b в степени d , но, к сожалению, мы не располагаем знанием о глубине наименьшего решения.

3. Какие преимущества дает использование алгоритма итеративного углубления по сравнению с поиском в ширину?

Использование алгоритма итеративного углубления по сравнению с поиском в ширину даёт следующие преимущества:

- сочетание преимуществ поиска в глубину и поиска в ширину. Алгоритм позволяет найти наилучший предел глубины поиска, постепенно увеличивая его до тех пор, пока не будет найдена цель;

- меньшие затраты на память. Поиск с итеративным углублением характеризуется скромными требованиями к памяти;

- более быстрое осуществление поиска. Несмотря на повторное формирование состояний, поиск с итеративным углублением фактически осуществляется быстрее, чем поиск в ширину.

4. Опишите, как работает итеративное углубление и как оно помогает избежать проблем с памятью.

Представим, что решение находится на глубине 6, а мы проводим поиск с итеративным углублением до глубины 4. Несмотря на то, что мы уже исследовали все дерево до уровня 4, для продолжения поиска нам необходимо углубиться до уровня 5. Кажется логичным просто продолжить с текущей глубины, однако это невозможно. Причина кроется в особенности поиска в глубину: при отсутствии решения на конкретной ветви мы отбрасываем эту ветвь и переходим к следующей.

При поиске с ограниченной глубиной мы действуем аналогично: если не находим решение на конце ветви, мы её исключаем из рассмотрения. Хотя кажется неэффективным не сохранять результаты до глубины 4, ведь это позволило бы исследовать следующий уровень без повторения работы, такой подход превратил

бы метод в поиск в ширину, который исследует все варианты до определённой глубины и сохраняет их. Именно пространственная сложность поиска в ширину стала той проблемой, которую мы пытались обойти, используя сначала поиск в глубину, а затем поиск с ограниченной глубиной.

5. Почему алгоритм итеративного углубления нельзя просто продолжить с текущей глубины, а приходится начинать поиск заново с корневого узла?

Это делается для поддержания линейной сложности по памяти. В случае сохранения всех узлов мы столкнулись бы с экспоненциальной пространственной сложностью, что сделало бы метод похожим на поиск в ширину.

6. Какие временные и пространственные сложности имеет поиск с итеративным углублением?

Временная сложность поиска с итеративным углублением имеет порядок $O(b^l)$, где b – это коэффициент ветвления. Пространственная сложность алгоритма равна $O(b * l)$.

7. Как алгоритм итеративного углубления сочетает в себе преимущества поиска в глубину и поиска в ширину?

Алгоритм итеративного углубления сочетает в себе преимущества поиска в глубину – низкую пространственную сложность, с возможностью нахождения оптимального решения, как в поиске в ширину. Этот гибридный подход делает поиск с итеративным углублением эффективным инструментом в решении многих задач, где пространственные ограничения являются критическими, а оптимальность решения необходима.

8. Почему поиск с итеративным углублением остается эффективным, несмотря на повторное генерирование дерева на каждом шаге увеличения глубины?

Поиск с итеративным углублением остаётся эффективным, несмотря на повторное генерирование дерева на каждом шаге увеличения глубины, потому что большинство узлов находится на нижнем уровне дерева поиска. В таком случае затратами времени на повторное развёртывание узлов обычно можно пренебречь.

9. Как коэффициент разветвления b и глубина d влияют на общее количество узлов, генерируемых алгоритмом итеративного углубления?

Коэффициент ветвления b влияет на то, что в дереве поиска с одним и тем же коэффициентом ветвления на каждом уровне большинство узлов находится на нижнем уровне. Поэтому не имеет большого значения то, что узлы на верхних уровнях формируются многократно.

Глубина d определяет, как часто формируются узлы на разных уровнях. Узлы на нижнем уровне формируются один раз, те узлы, которые находятся на уровне, предшествующем нижнему, формируются дважды, и так далее, вплоть до дочерних узлов корневого узла, которые формируются d раз.

10. В каких ситуациях использование поиска с итеративным углублением может быть не оптимальным, несмотря на его преимущества?

Поиск с итеративным углублением сочетает в себе лучшие характеристики поиска в ширину: он находит решение, если оно существует (при конечном коэффициенте разветвления), и обеспечивает нахождение кратчайшего решения при одинаковой стоимости каждого шага. Однако в ситуациях с различающейся стоимостью шагов, например, при измерении расстояний на карте, ни один из методов не будет оптимальным.

11. Какую задачу решает функция `iterative_deepening_search`?

Функция `iterative_deepening_search` решает задачу поиска с итеративным углублением (IDS).

12. Каков основной принцип работы поиска с итеративным углублением?

Дерево просматривается в глубину сначала до глубины 1. Затем поиск начинается заново и ведётся до глубины 2. Потом снова возвращаются в корень и спускаются до глубины 3 и так далее (шаг увеличения глубины можно сделать большим, чем 1). Процесс заканчивается, когда исчерпываются все узлы или достигается некоторое предельное значение глубины.

13. Что представляет собой аргумент `problem`, передаваемый в функцию `iterative_deepening_search`?

Аргумент `problem` представляет собой задачу, для которой необходимо найти решение.

14. Какова роль переменной `limit` в алгоритме?

`limit` – это предел глубины поиска, который будет увеличиваться до тех пор, пока не будет найдено решение или пока не будет достигнут теоретический максимум глубины.

15. Что означает использование диапазона `range(1, sys.maxsize)` в цикле `for`?

Цикл `for` итеративно увеличивает предел глубины поиска, начиная с 1 и потенциально до максимально возможного размера целого числа в системе (`sys.maxsize`).

16. Почему предел глубины поиска увеличивается постепенно, а не устанавливается сразу на максимальное значение?

Предел глубины поиска с итеративным углублением увеличивается постепенно, а не устанавливается сразу на максимальное значение, для того чтобы найти наилучший предел глубины поиска.

17. Какая функция вызывается внутри цикла и какую задачу она решает?

На каждой итерации вызывается функция `depth_limited_search` с текущим пределом глубины `limit`.

18. Что делает функция `depth_limited_search`, и какие результаты она может возвращать?

Эта функция пытается найти решение задачи в пределах заданной глубины. Результат ее выполнения сохраняется в переменной `result`.

19. Какое значение представляет собой `cutoff`, и что оно обозначает в данном алгоритме?

Значение `cutoff` обозначает, что достигнут текущий предел глубины, но целевой узел так и не был найден.

20. Почему результат сравнивается с `cutoff` перед тем, как вернуть результат?

Если `result` не равен `cutoff`, это означает, что было найдено решение или что поиск завершился неудачей по другой причине (не из-за достижения предела глубины).

21. Что произойдет, если функция `depth_limited_search` найдет решение на первой итерации?

Если функция `depth_limited_search` найдёт решение на первой итерации, она вернёт решение.

22. Почему функция может продолжать выполнение до тех пор, пока не достигнет `sys.maxsize`?

Это означает, что предел глубины будет увеличиваться до тех пор, пока не будет найдено решение или пока не будет достигнут теоретический максимум глубины.

23. Каковы преимущества использования поиска с итеративным углублением по сравнению с обычным поиском в глубину?

- эффективность для глубоких деревьев. Число узлов с глубиной обычно существенно растёт, и наибольшая их доля приходится на нижний слой, где находится искомый узел. Поэтому повторные пробеги от корня дерева не столь затратны по времени, а память для хранения списка узлов радикально сокращается по сравнению с поиском в ширину;

- гарантированное нахождение самого близкого к начальному состоянию решения. Алгоритм позволяет избежать экспоненциальной сложности;

- сочетание преимуществ поиска в глубину и поиска в ширину. В поиске с итеративным углублением сочетаются преимущества пространственной сложности поиска в глубину и полноты и оптимальности поиска в ширину.

24. Какие потенциальные недостатки может иметь этот подход?

- повторное исследование узлов. Алгоритм не сохраняет посещённые узлы, что приводит к повторному исследованию;

- более высокие вычислительные затраты. Это может занять больше времени и потреблять больше вычислительной мощности по сравнению с другими алгоритмами;

- медленная скорость работы. Это происходит из-за многократного исследования узлов.

25. Как можно оптимизировать данный алгоритм для ситуаций, когда решение находится на больших глубинах?

- запоминать глубину каждого узла. Это позволит при извлечении узла из стека знать, сколько ещё потомков необходимо породить;

- использовать ранее найденные результаты. Если в позиции на глубине N найден лучший ход, то велика вероятность, что на глубине больше D этот же ход будет лучшим;

- определять, является ли найденный ход единственным. Если ход в позиции единственный, то не важно, на какой глубине он был найден, и его можно использовать вне зависимости от глубины;

- записывать в таблицу транспозиций позиции, проанализированные с глубиной «бесконечность». Если ход в позиции единственный, то её можно записывать в таблицу как проанализированную с такой глубиной.