

Лабораторная работа 1

Тема: Исследование методов поиска в пространстве состояний

Цель работы: приобретение навыков по работе с методами поиска в пространстве состояний с помощью языка программирования Python версии 3.x

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 *

 RomanGorchakov ▾

Repository name *

/ 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 (#3261)	3 years ago
	Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
	PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
	Python.gitignore	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 #3792 from jl5000/patch-1	2 years ago
	README.md	Merge pull request #3854 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 (#3312)	3 years ago
	RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
	Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	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».

1 Горчаков Роман Владимирович. Вариант 2

2 # Лабораторная работа 1. Исследование методов поиска в пространстве состояний

3

4 Искусственный интеллект (ИИ) представляет собой увлекательное переплетение различных поисковых методологий. Среди базовых из них является "слепой поиск". Этот метод можно представить как попытку найти выход из затерянного лабиринта, ощупывая каждую стену, каждый угол, без заранее известного плана или карты. Он исследует пространство возможных решений методом проб и ошибок, не обладая информацией о том, насколько близко к каждой принятое решение к финальной цели. Такой подход, хотя и элементарен, является основой для разработки более сложных алгоритмов, включая эвристический поиск.

5

6 Эвристический поиск, в отличие от слепого, использует дополнительные знания или "эвристики" для направления процесса поиска, подобно тому, как путешественник использует компас в неизведанных землях. Один из наиболее известных примеров такого поиска - алгоритм A*, который находит наиболее оптимальный путь к цели, основываясь на заранее заданных критериях и предположениях.

7

8 Многие задачи в искусственном интеллекте, будь то шахматы, планирование маршрута или доказательство математических теорем, сводятся к поиску эффективного пути в огромном пространстве возможных решений. ИИ должен найти способ эффективно исследовать это пространство, балансируя между доступными ресурсами и желанием достичь решения, что делает его уникальным и незаменимым инструментом в современном мире.

9

10 Разнообразие задач обладает уникальными характеристиками. Некоторые из них являются детерминированными и полностью наблюдаемыми, что соответствует так называемым задачам одного состояния. В таких задачах агент полностью осведомлен о своем текущем положении и задача сводится к определению последовательности действий для достижения цели.

11

12 Основным методом решения задач поиска такого типа является алгоритм поиска по дереву. Суть метода заключается в автоматическом моделировании процесса исследования пространства состояний путём генерации преемников уже изученных состояний, что известно как расширение состояний.

13

14 Алгоритм поиска по дереву - алгоритм, принимающий на вход задачу и стратегию (о последней мы поговорим позднее) и возвращающий решение в виде последовательности действий, переводящей из начального состояния в целевое, либо заканчивающийся неудачей, если такая последовательность отсутствует или не может быть найдена. Алгоритм действует следующим образом: если нет кандидатов для расширения, алгоритм возвращает неудачу; в противном случае выбирается листовый узел для расширения в соответствии со стратегией, определяющей, какой из листовых узлов будет расширен; если узел содержит целевое состояние, возвращается соответствующее решение - путь в дереве, приводящий к этому листовому узлу с целевым состоянием; если нет, узел расширяется, и полученные узлы добавляются к дереву. Этот процесс и называется поиском по дереву.

15

16 Край (Frontier) представляет собой очередь узлов, из которой мы берём узлы с переднего края. Если край пуст, это означает отсутствие узлов для расширения и неудачу. Если край не пуст, мы берём узлы с переднего края, применяем тест на цель, предоставленный задачей, и если это целевое состояние, возвращаем решение.

17

18 Цель и решение (Goal & Solution) - разные понятия: цель - нахождение в определённой точке, а решение - последовательность действий от начального состояния до цели. Если это не цель, мы расширяем узел, применяя функцию расширения, что даёт набор узлов-преемников и вставляем их в край.

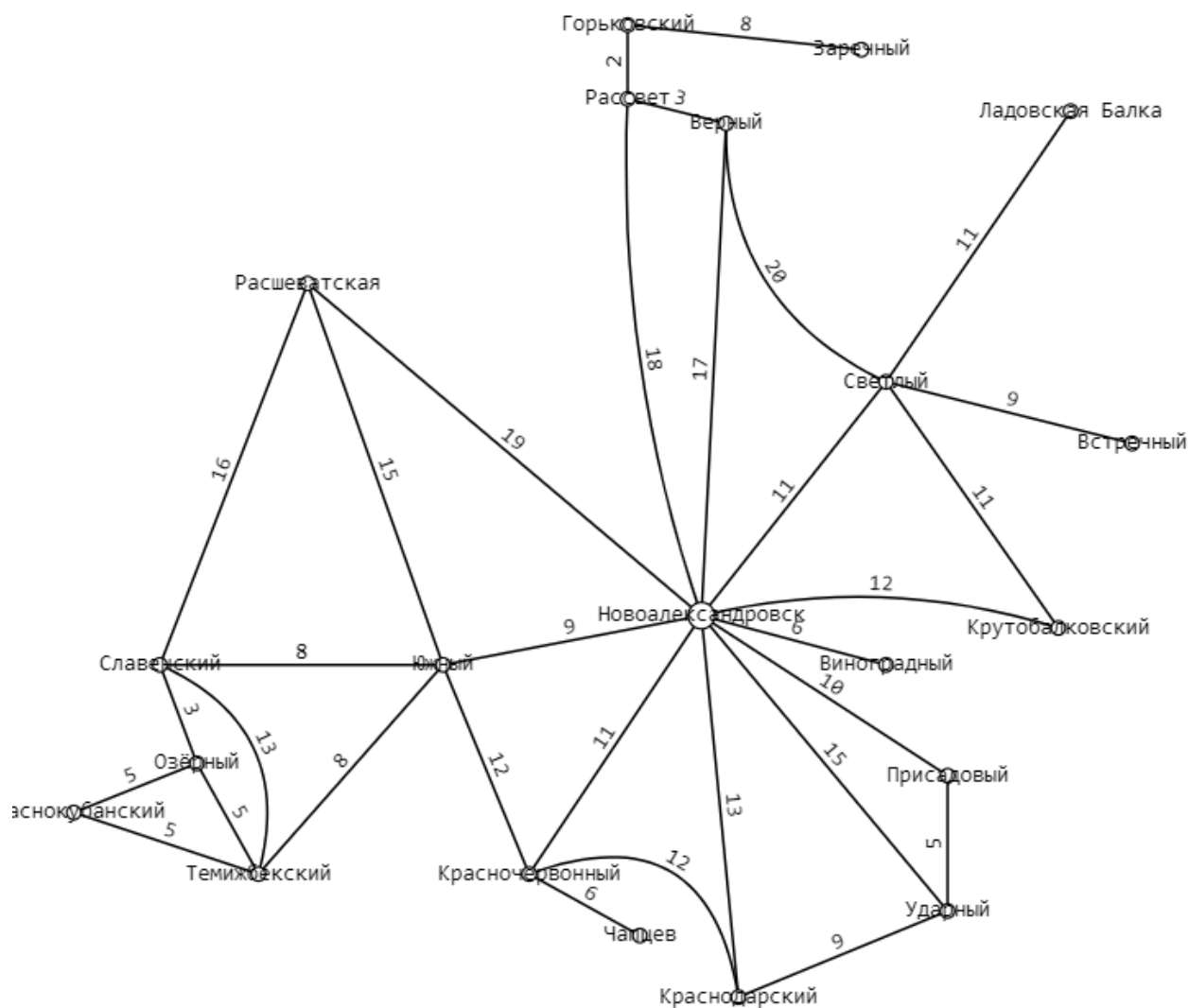
19

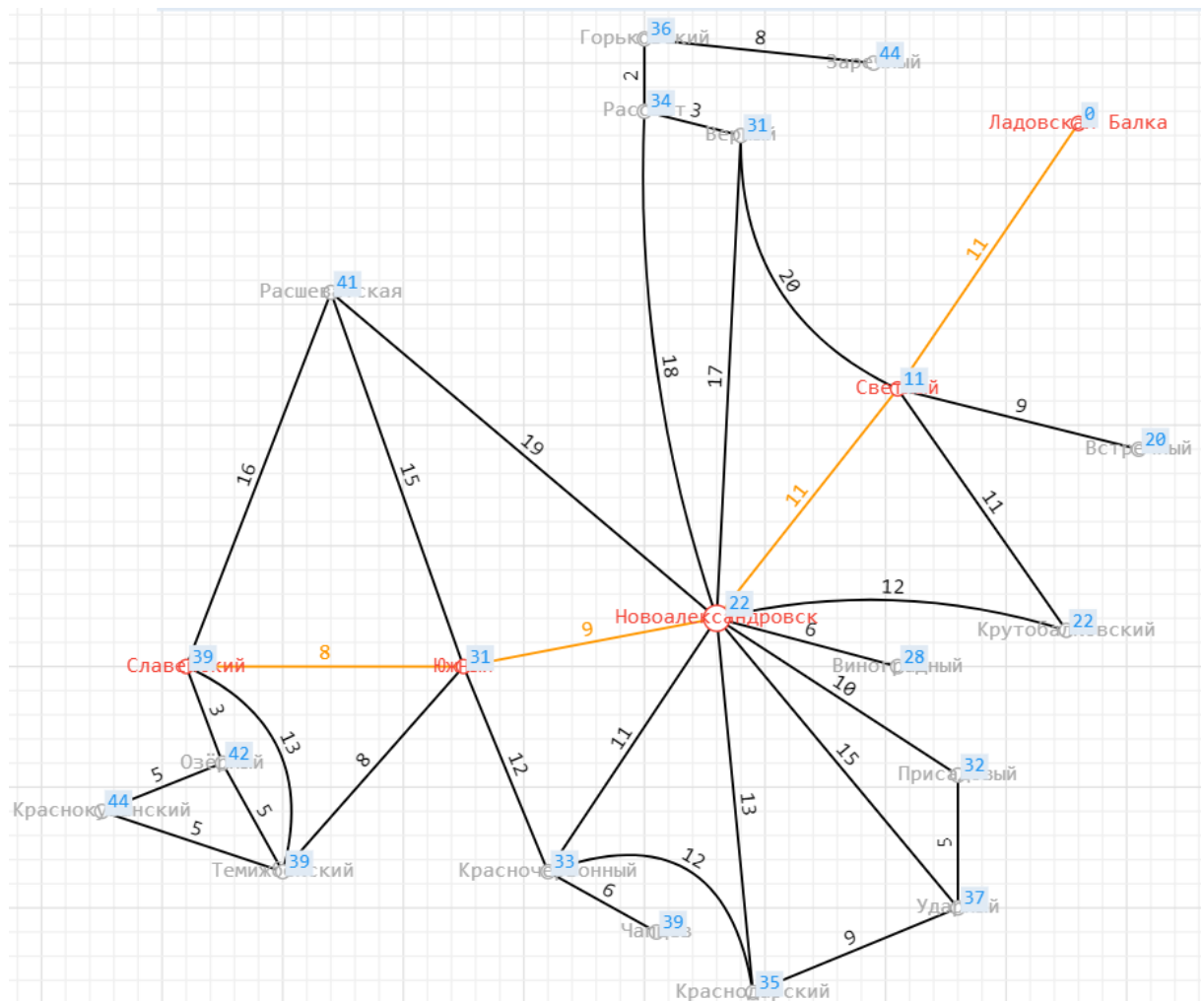
4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствие с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout -b develop» для 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_1 (main) $ git checkout -b develop
Switched to a new branch 'develop'
• @RomanGorchakov →/workspaces/AI_1 (develop) $ git branch feature_branch
• @RomanGorchakov →/workspaces/AI_1 (develop) $ git branch release/1.0.0
• @RomanGorchakov →/workspaces/AI_1 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @RomanGorchakov →/workspaces/AI_1 (main) $ git branch hotfix
• @RomanGorchakov →/workspaces/AI_1 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/AI_1 (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) (4.2.2)
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 6.0 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_1 (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_1 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
• @RomanGorchakov →/workspaces/AI_1 (develop) $ conda env export > environment.yml
```

5. Построим граф, где узлы будут представлять населённые пункты, а рёбра – дороги, соединяющие их. Вес каждого ребра соответствует расстоянию между этими пунктами. Выбираем начальный и конечный пункты на графе. Определяем минимальный маршрут между ними, который должен проходить через три промежуточных населённых пункта.





6. Методом полного перебора решаем задачу коммивояжёра (найти самый выгодный маршрут, проходящий по всем городам хотя бы один раз, с возвращением в исходный город) для построенного графа.



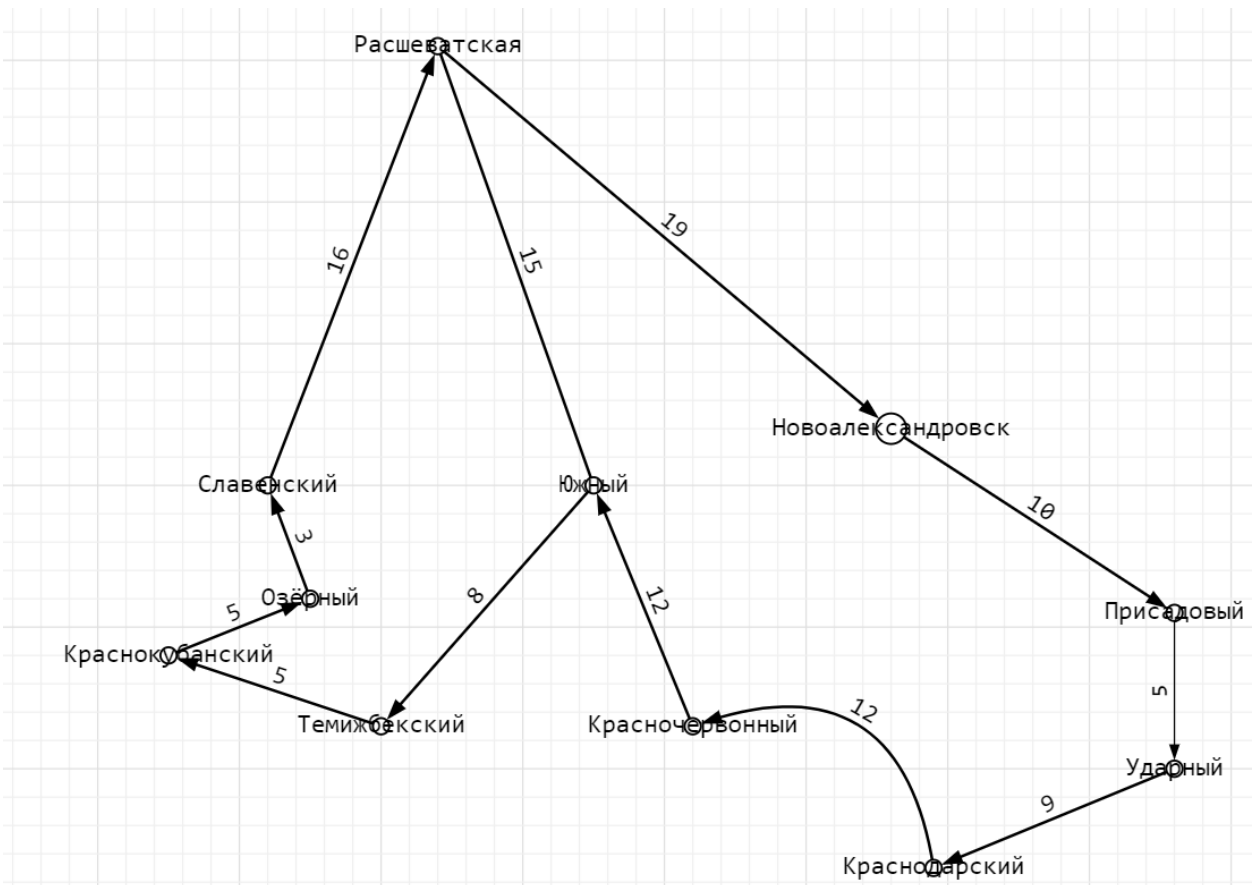
```
Командная строка
Microsoft Windows [Version 10.0.22631.4317]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Admin>D:

D:>cd D:\курс\Искусственный интеллект в профессиональной сфере\1\Индивидуальные задания\Задание 2

D:\курс\Искусственный интеллект в профессиональной сфере\1\Индивидуальные задания\Задание 2>python commivoyager.py
['Новоалександровск', 'Присадовый', 'Ударный', 'Краснодарский', 'Красночервоный', 'Южный', 'Темижбекский', 'Краснокубанский',
'Озёрный', 'Славенский', 'Расшеватская', 'Новоалександровск'] 104

D:\курс\Искусственный интеллект в профессиональной сфере\1\Индивидуальные задания\Задание 2>
```



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

```

@RomanGorchakov → /workspaces/Py21 (develop) $ git add .
@RomanGorchakov → /workspaces/Py21 (develop) $ git commit -m "SQL databases"
[develop cc32d83] SQL databases
5 files changed, 617 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 environment.yml
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\individual.py"
create mode 100644 ".../320\236\321\202\321\207\321\221\321\202\320\233\320\2402.19_\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
create mode 100644 ".../320\237\321\200\320\270\320\274\320\265\321\200\example.py"
@RomanGorchakov → /workspaces/Py21 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@RomanGorchakov → /workspaces/Py21 (main) $ git merge develop
Updating 41dd812..cc32d83
Fast-forward
 .pre-commit-config.yaml
 environment.yml
 .../individual.py
 .../320\233\320\2402.19_\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
 "\320\237\321\200\320\270\320\274\320\265\321\200\example.py"
5 files changed, 617 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 environment.yml
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\individual.py"
create mode 100644 ".../320\236\321\202\321\207\321\221\321\202\320\233\320\2402.19_\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
create mode 100644 ".../320\237\321\200\320\270\320\274\320\265\321\200\example.py"
@RomanGorchakov → /workspaces/Py21 (main) $ git push -u
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (10/10), 554.90 KiB | 19.82 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/RomanGorchakov/Py21
 41dd812..cc32d83 main -> main
branch 'main' set up to track 'origin/main'.
@RomanGorchakov → /workspaces/Py21 (main) $
  
```

main
1 Branch
0 Tags
Go to file
Add file
Code

RomanGorchakov SQL databases
cc32d83 · 1 minute ago
5 Commits

Индивидуальное задание	SQL databases	1 minute ago
Отчёт	SQL databases	1 minute ago
Пример	SQL databases	1 minute ago
.gitignore	Create .gitignore	3 days ago
.pre-commit-config.yaml	SQL databases	1 minute ago
LICENSE	Create LICENSE	3 days ago
README.md	Update README.md	14 hours ago
environment.yml	SQL databases	1 minute ago

README
MIT license

Горчаков Роман Владимирович. Вариант 4

Лабораторная работа 2.21. Взаимодействие с базами данных SQLite3 с помощью языка программирования Python.

Сами по себе СУБД редко используются для работы с базами данных. В том смысле, что в реальных проектах связи БД + СУБД бывает недостаточно. Обычно с СУБД работают через какой-либо язык программирования. Это позволяет более гибко принимать запросы, обрабатывать ответы перед передачей их куда-либо далее. Ведь у императивного, а не декларативного как SQL, языка программирования средств для работы с данными больше, да и логика богаче.

Чтобы использовать SQLite3 в Python, прежде всего, вам нужно будет импортировать модуль sqlite3, а затем создать объект соединения, который соединит нас с базой данных и позволит нам выполнять операторы SQL. Объект соединения создается с помощью функции connect(). Будет создан новый файл под названием

Контрольные вопросы

1. Что представляет собой метод «слепого поиска» в искусственном интеллекте?

«Слепой поиск» в искусственном интеллекте можно представить как попытку найти выход из затемненного лабиринта, ощупывая каждую стену, каждый угол, без заранее известного плана или карты. Он исследует пространство возможных решений методом проб и ошибок, не обладая информацией о том, насколько близко каждое принятое решение к финальной цели.

2. Как отличается эвристический поиск от слепого поиска?

Эвристический поиск, в отличие от слепого, использует дополнительные знания или «эвристики» для направления процесса поиска, подобно тому, как путешественник использует компас в неизведанных землях.

3. Какую роль играет эвристика в процессе поиска?

Эвристика играет в процессе поиска следующую роль:

1) направляет поиск к более перспективным областям поискового пространства. Это важно в задачах, где пространство поиска слишком велико для исчерпывающего изучения или где ландшафт решения неровный со множеством локальных оптимумов;

2) повышает эффективность поиска. Эвристики предлагают разумные предположения о перспективных направлениях, в которых следует двигаться. Это часто приводит к более быстрой конвергенции к хорошему решению, сокращая время вычислений и ресурсы;

3) избегает локальных оптимумов. Эвристики вносят хаотичность в процесс поиска, позволяя ему иногда принимать худшие решения, чтобы избежать локальных оптимумов и исследовать большую часть пространства поиска;

4) сбалансирует исследование новых областей в пространстве поиска с использованием известных хороших областей. Например, запретный поиск использует эвристику, основанную на памяти, которая помогает алгоритму запоминать, какие области были исследованы, чтобы предотвратить циклическое поведение и стимулировать исследование новых областей.

4. Приведите пример применения эвристического поиска в реальной задаче.

Пример применения эвристического поиска в реальной задаче — нахождение оптимальной рассадки туристических групп по автобусам во время круиза. Для этого используется эвристика случайного поиска. Алгоритм рассадки включает следующие шаги:

- 1) выбор случайной группы и случайного автобуса;
- 2) проверка, можно ли посадить выбранную группу в выбранный автобус (не сидит ли группа в другом автобусе и достаточно ли мест). Если нет, то

увеличивается счётчик неудачных попыток на единицу. Если да, то счётчик обнуляется;

3) если количество неудачных попыток превысило 50, выход из цикла, иначе повторение шага 1;

4) проверка, все ли группы рассажены. Если да, то считается максимальная разница между количеством пустых мест в автобусах (это число будет считаться оценкой).

5. Почему полное исследование всех возможных ходов в шахматах затруднительно для ИИ?

В идеальном мире, программа могла бы анализировать каждый возможный ход и его последствия, строить огромное дерево всех возможных комбинаций и выбирать наиболее перспективные варианты. Однако, учитывая огромное количество возможных ходов в шахматах, полное исследование всех вариантов становится практически невозможным даже для современных суперкомпьютеров.

6. Какие факторы ограничивают создание идеального шахматного ИИ?

Даже если технически возможно построить структуру данных для представления всех возможных ходов в шахматах, встает вопрос о ресурсах – времени и памяти. Для многих задач эти ограничения могут быть преодолены, но в контексте шахмат они становятся критическими факторами. Необходимость быстро обрабатывать информацию и принимать решения в условиях когда время может быть ограничено делает задачу создания эффективного шахматного ИИ особенно сложной.

7. В чем заключается основная задача искусственного интеллекта при выборе ходов в шахматах?

Многие задачи в искусственном интеллекте сводятся к поиску эффективного пути в огромном пространстве возможных решений. ИИ должен найти способ эффективно исследовать это пространство, балансируя между доступными ресурсами и желаемым качеством решения, что делает его уникальным и незаменимым инструментом в современном мире.

8. Как алгоритмы ИИ балансируют между скоростью вычислений и нахождением оптимальных решений?

В мире искусственного интеллекта часто приходится находить баланс между скоростью вычислений, объемом используемой памяти и способностью находить оптимальные решения. Некоторые алгоритмы могут быстро находить решения, но они могут быть далеки от оптимальных. Другие, напротив, способны находить наилучшие решения, но требуют значительных временных и ресурсных затрат.

9. Каковы основные элементы задачи поиска маршрута по карте?

Основные элементы задачи поиска маршрута по карте:

- 1) изучение сети (графа) дорог. Определяется протяжённость каждого элемента графа, его класс и ширина проезжей части;
 - 2) изучение физико-географических и метеорологических условий в районе перемещения;
 - 3) анализ влияния дестабилизирующих факторов и степени внешнего воздействия на элементы графа дорог;
 - 4) изучение защитных свойств местности на каждом элементе;
 - 5) выбор показателей оценки эффективности поиска маршрутов с учётом дестабилизирующих факторов и защитных свойств местности;
 - 6) свёртка показателей и присвоение весов ветвям графа дорог;
 - 7) формирование маршрутов и выбор оптимального маршрута.
10. Как можно оценить оптимальность решения задачи маршрутизации на карте Румынии?

Оптимальность решения предполагает нахождение маршрута с минимальной стоимостью. Среди менее эффективных вариантов могут быть маршруты через Тимишоару и Лугож с нелогичными блужданиями по кругу. Важно, чтобы итоговый маршрут завершался в Бухаресте, даже если он включает неоптимальные повторения. Оптимальное решение – такое решение, которое достигается с минимальными затратами времени и расстояния.

11. Что представляет собой исходное состояние дерева поиска в задаче маршрутизации по карте Румынии?

12. Какие узлы называются листовыми в контексте алгоритма поиска по дереву?

13. Что происходит на этапе расширения узла в дереве поиска?

14. Какие города можно посетить, совершив одно действие из Арада в примере задачи поиска по карте?

15. Как определяется целевое состояние в алгоритме поиска по дереву?

16. Какие основные шаги выполняет алгоритм поиска по дереву?

17. Чем различаются состояния и узлы в дереве поиска?

18. Что такое функция преемника и как она используется в алгоритме поиска?

19. Какое влияние на поиск оказывают такие параметры, как b (разветвление), d (глубина решения) и m (максимальная глубина)?

20. Как алгоритмы поиска по дереву оцениваются по критериям полноты, временной и пространственной сложности, а также оптимальности?

21. Какую роль выполняет класс `Problem` в приведенном коде?

22. Какие методы необходимо переопределить при наследовании класса `Problem`?

23. Что делает метод `is_goal` в классе `Problem`?
24. Для чего используется метод `action_cost` в классе `Problem`?
25. Какую задачу выполняет класс `Node` в алгоритмах поиска?
26. Какие параметры принимает конструктор класса `Node`?
27. Что представляет собой специальный узел `failure`?
28. Для чего используется функция `expand` в коде?
29. Какая последовательность действий генерируется с помощью функции `path_actions`?
30. Чем отличается функция `path_states` от функции `path_actions`?
31. Какой тип данных используется для реализации `FIFOQueue`?
32. Чем отличается очередь `FIFOQueue` от `LIFOQueue`?
33. Как работает метод `add` в классе `PriorityQueue`?
34. В каких ситуациях применяются очереди с приоритетом?
35. Как функция `heappop` помогает в реализации очереди с приоритетом?