Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**дисциплины «Искусственный интеллект в профессиональной сфере»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Кожуховский Виктор Андреевич  3 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем  », очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  Воронкин Роман Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

Тема: Исследование поиска с ограничением глубины

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

Порядок выполнения работы:

1. Изучил теоретический материал работы.

2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал примеры лабораторной работы.

8. Решите задания лабораторной работы с помощью языка программирования Python и элементов программного кода лабораторной работы 1 (имя файла начинается с PR.AI.001.). Проверьте правильность решения каждой задачи на приведенных тестовых примерах.

Вы работаете над разработкой системы навигации для робота-пылесоса. Робот способен передвигаться по различным комнатам в доме, но из-за ограниченности ресурсов (например, заряда батареи) и времени на уборку, важно эффективно выбирать путь. Ваша задача - реализовать алгоритм, который поможет роботу определить, существует ли путь к целевой комнате, не превышая заданное ограничение по глубине поиска.

Дано дерево, где каждый узел представляет собой комнату в доме. Узлы связаны в соответствии с возможностью перемещения робота из одной комнаты в другую. Необходимо определить, существует ли путь от начальной комнаты (корень дерева) к целевой комнате (узел с заданным значением), так, чтобы робот не превысил лимит по глубине перемещения.

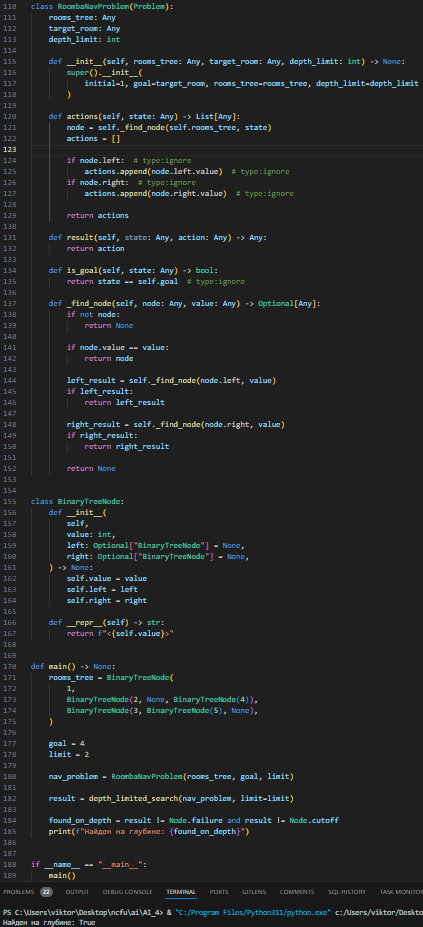


Рисунок 1. Решение задачи о навигации робота-пылесоса

Представьте, что вы разрабатываете систему для управления складом, где товары упорядочены в структуре, похожей на двоичное дерево. Каждый узел дерева представляет место хранения, которое может вести к другим местам хранения (левому и правому подразделу). Ваша задача — найти наименее затратный путь к товару, ограничив поиск заданной глубиной, чтобы гарантировать, что поиск займет приемлемое время.

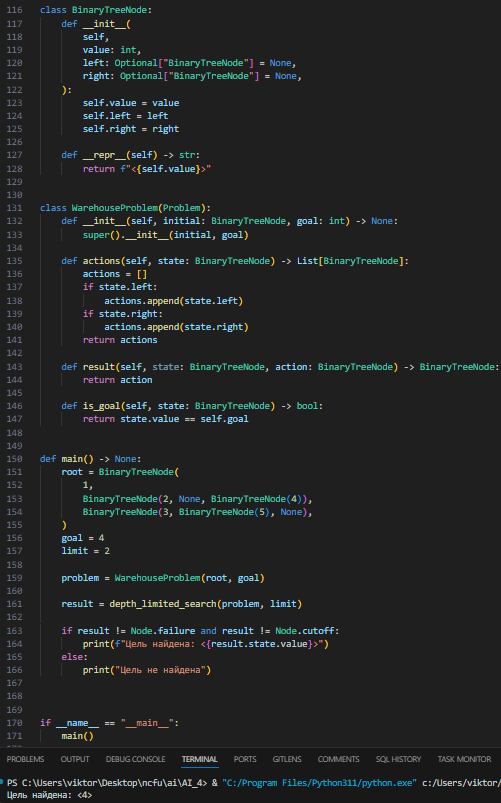


Рисунок 2. Решение задачи управления складом

Представьте, что вы разрабатываете систему для автоматического управления инвестициями, где дерево решений используется для представления последовательности инвестиционных решений и их потенциальных исходов. Цель состоит в том, чтобы найти наилучший исход (максимальную прибыль) на определённой глубине принятия решений, учитывая ограниченные ресурсы и время на анализ.

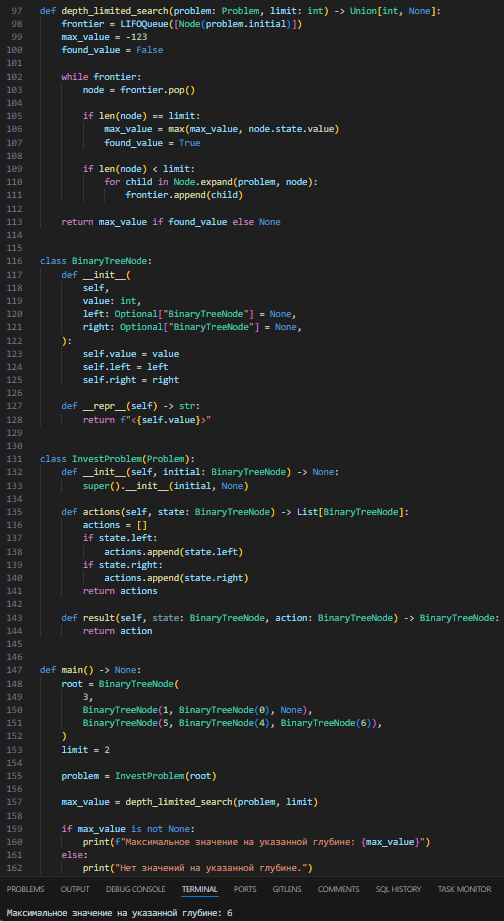


Рисунок 3. Решение задачи о разработке системы управления инвестициями

10. Для построенного графа лабораторной работы 1 (имя файла начинается с PR.AI.001.) напишите программу на языке программирования Python, которая с помощью алгоритма поиска с ограничением глубины находит минимальное расстояние между начальным и конечным пунктами. Определите глубину дерева поиска, на которой будет найдено решение. Сравните найденное решение с решением, полученным вручную.



Рисунок 4. Код программы решения задания и его выполнение

11. Зафиксировал сделанные изменения в репозитории.

12. Выполнил слияние ветки для разработки с веткой master/main.

13. Отправил сделанные изменения на сервер GitHub.

Ссылка: https://github.com/Viktorkozh/AI\_4

Контрольные вопросы:

1. Что такое поиск с ограничением глубины, и как он решает проблему бесконечных ветвей?

Поиск с ограничением глубины — это модификация поиска в глубину, которая исследует дерево лишь до определённого уровня глубины. Он решает проблему бесконечных ветвей, возникающую, когда дерево расширяется бесконечно, ограничивая исследование каждой ветви до установленного уровня.

2. Какова основная цель ограничения глубины в данном методе поиска?

Основная цель ограничения глубины в данном методе поиска — предотвратить бесконечное расширение дерева и обеспечить, что если решение существует на глубине, не превышающей максимально установленный уровень, оно будет найдено.

3. В чем разница между поиском в глубину и поиском с ограничением глубины?

Разница между поиском в глубину и поиском с ограничением глубины заключается в том, что поиск с ограничением глубины прекращает исследование, когда достигнут узел, генерирующий дочерний узел на чрезмерной глубине, в то время как стандартный поиск в глубину может продолжать углубляться бесконечно.

4. Какую роль играет проверка глубины узла в псевдокоде поиска с ограничением глубины?

Проверка глубины узла в псевдокоде поиска с ограничением глубины позволяет определить, достиг ли узел максимально допустимого уровня глубины, и, если да, прекратить дальнейшее расширение этого узла.

5. Почему в случае достижения лимита глубины функция возвращает «обрезание»?

В случае достижения лимита глубины функция возвращает «обрезание», чтобы указать, что дальнейший поиск в этом направлении прекращается из-за ограничения глубины.

6. В каких случаях поиск с ограничением глубины может не найти решение, даже если оно существует?

Поиск с ограничением глубины может не найти решение, если целевой узел расположен на уровне глубины, превышающем установленное ограничение.

7. Как поиск в ширину и в глубину отличаются при реализации с использованием очереди?

Поиск в ширину и в глубину отличаются при реализации с использованием очереди тем, что в поиске в ширину новые узлы добавляются в конец очереди, а в поиске в глубину новые дочерние узлы помещаются в начало очереди.

8. Почему поиск с ограничением глубины не является оптимальным?

Поиск с ограничением глубины не является оптимальным, так как он может находить решения на глубинах n или n − 1, в то время как более короткое решение может существовать на глубине 2, но оно не было достигнуто из-за ограничения глубины.

9. Как итеративное углубление улучшает стандартный поиск с ограничением глубины?

Итеративное углубление улучшает стандартный поиск с ограничением глубины, сочетая лучшие качества поиска в глубину и ширину, итеративно увеличивая глубину поиска и обеспечивая полноту и оптимальность.

10. В каких случаях итеративное углубление становится эффективнее простого поиска в ширину?

Итеративное углубление становится эффективнее простого поиска в ширину в случаях деревьев с высоким коэффициентом ветвления, так как оно избегает хранения большого числа узлов в памяти.

11. Какова основная цель использования алгоритма поиска с ограничением глубины?

Основная цель использования алгоритма поиска с ограничением глубины — найти решение, если оно существует на глубине, не превышающей установленный уровень.

12. Какие параметры принимает функция depth\_limited\_search, и каково их назначение?

Функция depth\_limited\_search принимает два аргумента: problem (задача, которую нужно решить) и limit (максимальная глубина поиска). Параметр limit определяет, до какой глубины будет производиться поиск.

13. Какое значение по умолчанию имеет параметр limit в функции depth\_limited\_search ?

Значение по умолчанию параметра limit в функции depth\_limited\_search равно 10.

14. Что представляет собой переменная frontier, и как она используется в алгоритме?

Переменная frontier представляет собой стек (LIFO-очередь), содержащий узлы для дальнейшего рассмотрения в алгоритме.

15. Какую структуру данных представляет LIFOQueue, и почему она используется в этом алгоритме?

LIFOQueue представляет собой структуру данных, которая работает по принципу "последний пришёл — первый вышел" (LIFO), и она используется в этом алгоритме для управления узлами, которые нужно обработать.

16. Каково значение переменной result при инициализации, и что оно означает?

Значение переменной result при инициализации установлено в значение failure, что означает неудачу поиска.

17. Какое условие завершает цикл while в алгоритме поиска?

Цикл while в алгоритме поиска завершается, когда переменная frontier становится пустой, то есть когда больше нет узлов для рассмотрения.

18. Какой узел извлекается с помощью frontier.pop() и почему?

С помощью frontier.pop() извлекается последний добавленный узел, чтобы продолжить его обработку в алгоритме.

19. Что происходит, если найден узел, удовлетворяющий условию цели (условие problem.is\_goal(node.state))?

Если найден узел, удовлетворяющий условию цели (условие problem.is\_goal(node.state)), поиск завершается успешно, и текущий узел возвращается как результат.

20. Какую проверку выполняет условие elif len(node) >= limit, и что означает его выполнение?

Условие elif len(node) >= limit проверяет, достиг ли текущий узел ограничения по глубине. Если это так, дальнейший поиск в этом направлении прекращается.

21. Что произойдет, если текущий узел достигнет ограничения по глубине поиска?

Если текущий узел достигнет ограничения по глубине поиска, функция возвращает значение cutoff, указывая на то, что лимит глубины был достигнут.

22. Какую роль выполняет проверка на циклы elif not is\_cycle(node) в алгоритме?

Проверка на циклы elif not is\_cycle(node) в алгоритме выполняет роль предотвращения повторного посещения узлов, что может привести к бесконечным циклам.

23. Что происходит с дочерними узлами, полученными с помощью функции expand(problem, node) ?

Дочерние узлы, полученные с помощью функции expand(problem, node), добавляются в переменную frontier для дальнейшей обработки.

24. Какое значение возвращается функцией, если целевой узел не был найден?

Если целевой узел не был найден, функцией возвращается значение result, которое может быть либо failure, либо cutoff.

25. В чем разница между результатами failure и cutoff в контексте данного алгоритма?

Разница между результатами failure и cutoff в контексте данного алгоритма заключается в том, что failure указывает

Вывод: приобрел навыки по работе с поиском с ограничением глубины с помощью языка программирования Python версии 3.x