

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

Искусственный интеллект в профессиональной сфере
Отчет по лабораторной работе №5
Исследование поиска с интерактивным углублением

Выполнил:
Тихоненко Борис Витальевич
3 курс, группа ИТС-б-з-22-1,
11.03.02
«Инфокоммуникационные
технологии и системы связи»,
заочная форма обучения

(подпись)

Проверил: доцент департамента
цифровых робототехнических систем и
электроники
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

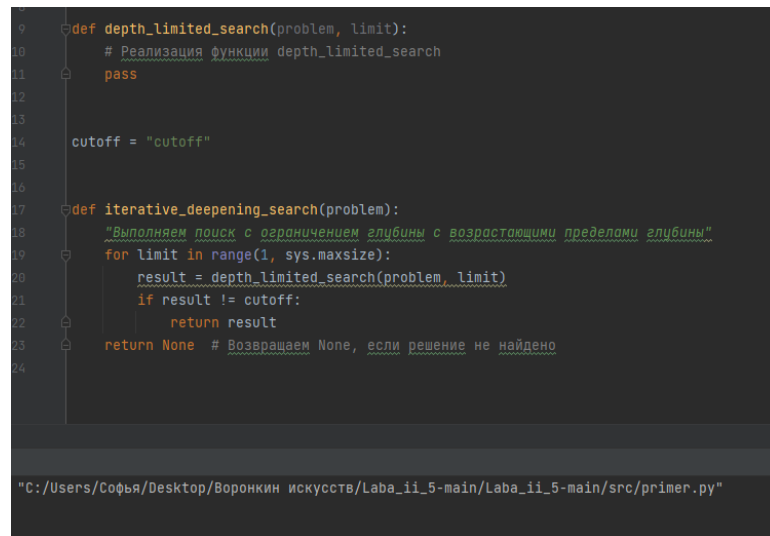
Ставрополь, 2025 г.

Тема: исследование поиска с итеративным углублением.

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

Ход работы:

Задание 1. Создал проект в папке репозитория. Приступил к работе с примером. Добавил новый файл lab5-prim.py. Рассмотрим реализацию алгоритма поиска с итеративным углублением на практике, в программном коде:



```
17 def depth_limited_search(problem, limit):
18     # Реализация функции depth_limited_search
19     pass
20
21     cutoff = "cutoff"
22
23
24 def iterative_deepening_search(problem):
25     """Выполняем поиск с ограничением глубины с возрастающими пределами глубины"""
26     for limit in range(1, sys.maxsize):
27         result = depth_limited_search(problem, limit)
28         if result != cutoff:
29             return result
30     return None # Возвращаем None, если решение не найдено
```

"C:/Users/Софья/Desktop/Воронкин искусств/Laba_ii_5-main/Laba_ii_5-main/src/prim.py"

Рисунок 1. Выполнение lab5-prim.py

Задание 2. Необходимо написать программу для поиска элемента с использованием алгоритма итеративного углубления.

Цель данного примера научиться применять алгоритм итеративного углубления для решения задачи поиска заданного элемента в дереве. Этот метод поиска эффективен для структур данных с неопределенной или большой глубиной, позволяя находить элементы, минимизируя время поиска и используемый объем памяти.

Необходимо разработать метод поиска, который позволит проверить существование пользователя с заданным идентификатором в системе, используя структуру дерева и алгоритм итеративного углубления.

ограничить поиск 20 уровнями. Алгоритм должен обработать дерево эффективно, избегая чрезмерного потребления ресурсов.

```
73
74     return deepest_file_overall, deepest_level_overall
75
76
77 if __name__ == "__main__":
78     synthetic_tree_path = "synthetic_tree"
79     generate_synthetic_tree(synthetic_tree_path, depth=5, breadth=3)
80
81     max_depth_to_search = 20
82     deepest_file, deepest_level = iterative_deepening_search(synthetic_tree_path, max_depth_to_search)
83
84     print(f"Самый глубокий файл: {deepest_file}, Глубина: {deepest_level}")
85
exe "C:/Users/Софья/Desktop/Воронкин Искусств/Laba_ii_5-main/Laba_ii_5-main/src/idz3.py"
```

Рисунок 4. Выполнение написанной программы

Ответы на 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 определяет текущую глубину узла. Он влияет на поиск, задавая ограничение на количество уровней, которые может пройти алгоритм, предотвращая исследование узлов, выходящих за пределы этой глубины.

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

Оптимальное значение глубины d зависит от неизвестной заранее глубины решения. Если d слишком мало, решение может быть пропущено; если слишком велико, алгоритм может потреблять лишние ресурсы или изучать ненужные пути.

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

Итеративное углубление использует меньше памяти, так как оно работает как поиск в глубину, при этом сохраняя полноту и оптимальность поиска в ширину, если стоимости переходов равны.

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

Алгоритм выполняет поиск в глубину с увеличением лимита глубины на каждой итерации. Это помогает избежать экспоненциального роста памяти,

характерного для поиска в ширину, так как для поиска в глубину требуется только стек глубины.

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

Начинать поиск заново необходимо, потому что алгоритм в предыдущей итерации не сохраняет узлы, которые превышают текущий лимит глубины. Это позволяет экономить память.

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

Временная сложность: $O(bd)$, где b — коэффициент разветвления, d — глубина решения.

Пространственная сложность: $O(d)$, так как требуется память только для стека глубины.

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

Он использует память, как поиск в глубину, но сохраняет полноту и оптимальность, как поиск в ширину, постепенно исследуя узлы на увеличивающихся уровнях глубины.

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

Большая часть времени уходит на исследование узлов на самом глубоком уровне, так как число узлов растет экспоненциально. Повторное исследование более верхних уровней имеет относительно низкую стоимость.

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

Алгоритм генерирует порядка b_d узлов. Однако за счет повторения на малых глубинах суммарное количество сгенерированных узлов составляет $O(b_b)$.

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

Если генерация узлов на верхних уровнях дерева требует значительных вычислений, то повторное их исследование может быть дорогостоящим. Также он неэффективен, если стоимость переходов не одинакова.

11. Какую задачу решает функция `iterative_deepening_search`?

Она находит решение задачи методом итеративного углубления, возвращая путь к решению или указание на то, что решение отсутствует.

12. Каков основной принцип работы поиска с итеративным углублением?

Постепенное увеличение предела глубины и выполнение поиска в глубину на каждом уровне до тех пор, пока не будет найдено решение.

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

Это объект, описывающий задачу поиска, включающий начальное состояние, функцию определения целей и правила переходов.

14. Какова роль переменной `limit` в алгоритме?

Она задает текущую максимальную глубину поиска.

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

Диапазон задает последовательное увеличение предела глубины с минимального значения до максимально возможного в системе.

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

Это позволяет найти решение на минимальной глубине, избегая ненужных затрат ресурсов на более глубокие уровни.

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

Функция `depth_limited_search` выполняет поиск с ограничением глубины, проверяя, можно ли найти решение в пределах текущей глубины.

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

Она проверяет узлы до заданной глубины. Возвращает либо путь к решению, либо указание, что поиск был "обрезан" (`cutoff`), либо "неудачу".

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

`Cutoff` обозначает, что поиск достиг текущего предела глубины и не смог продолжить исследование.

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

Это позволяет алгоритму определить, нужно ли увеличить глубину на следующей итерации.

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

Алгоритм завершится, вернув найденное решение.

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

Это позволяет алгоритму исследовать все возможные уровни глубины, если решение расположено на большом расстоянии.

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

Он сохраняет полноту и оптимальность, чего не может гарантировать обычный поиск в глубину.

24. Какие потенциальные недостатки может иметь этот подход?

Повторное исследование узлов может быть дорогостоящим в терминах времени, особенно если дерево большое или генерация узлов сложна.

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

- Использование эвристик для пропуска заведомо нерелевантных узлов.
- Применение более эффективного способа хранения и повторного использования верхних уровней дерева.

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