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 №5
дисциплины
«Искусственный интеллект и машинное обучение»
Вариант

Выполнил:
Степанов Леонид Викторович
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:
Богданов Сергей Сергеевич,
Ассистент департамента
цифровых, робототехнических
систем и электроники

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 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

Порядок выполнения работы:

Задания:

Поиск элемента в дереве с использованием алгоритма итеративного углубления

```python
# Представьте себе систему управления доступом, где каждый пользователь
# представлен узлом в дереве. Каждый узел содержит уникальный идентификатор
# пользователя. Ваша задача – разработать метод поиска, который позволит
# проверить существование пользователя с заданным идентификатором в системе,
# используя структуру дерева и алгоритм итеративного углубления.


class BinaryTreeNode:
    def __init__(self, value, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right

    def add_children(self, left, right):
        self.left = left
        self.right = right

    def __repr__(self):
        return f"<{self.value}>"
```

Рисунок 1 – Условие задания

```python
def depth_limited_search(node, target_id, limit):
    """Ищет узел с заданным идентификатором target_id на ограниченной глубине limit."""
    if node is None:
        return False
    if node.value == target_id:
        return True
    if limit <= 0:
        return False

    # Рекурсивно ищем в дочерних узлах, уменьшая лимит глубины на 1
    return depth_limited_search(
        node.left, target_id, limit - 1
    ) or depth_limited_search(node.right, target_id, limit - 1)


def iterative_deepening_search(root, target_id):
    """Основной метод для поиска узла с идентификатором target_id с итеративным углублением."""
    depth = 0
    while True:
        # Запускаем поиск с ограничением глубины depth
        found = depth_limited_search(root, target_id, depth)
        if found:
            return True
        depth += 1  # Увеличиваем глубину для следующей итерации
        # При желании можно установить максимальный лимит для depth, чтобы избежать бесконечного цикла
```

Рисунок 2 – Реализация итеративного поиска в глубину

```
53
54  def main():
55      root = BinaryTreeNode(1)
56      root.add_children(BinaryTreeNode(2), BinaryTreeNode(3))
57      root.left.add_children(BinaryTreeNode(4), BinaryTreeNode(5))
58      root.right.add_children(BinaryTreeNode(6), BinaryTreeNode(7))
59
60      # Проверяем существование узлов с идентификаторами 5 и 10
61      print(
62          iterative_deepening_search(root, 5, 5)
63      )  # Ожидаемый результат: True (узел существует)
64      print(
65          iterative_deepening_search(root, 10, 5)
66      )  # Ожидаемый результат: False (узел не существует)
67
68
69  if __name__ == "__main__":
70      main()
71
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) C:\Users\Лео\Desktop\3 Курс\ИИ\ii-5>python .\prog\task1.py
True
False

(base) C:\Users\Лео\Desktop\3 Курс\ИИ\ii-5>
```

Рисунок 3 – Результат работы

Поиск с файловой системы

```
4   # Рассмотрим задачу поиска информации в иерархических структурах данных,
5   # например, в файловой системе, где каждый каталог может содержать подкаталоги
6   # и файлы. Алгоритм итеративного углубления идеально подходит для таких задач,
7   # поскольку он позволяет исследовать структуру данных постепенно, углубляясь на
8   # один уровень за раз и возвращаясь, если целевой узел не найден. Для этого
9   # необходимо:
10  # Построить дерево, где каждый узел представляет каталог в файловой
11  # системе, а цель поиска — определенный файл.
12  # Найти путь от корневого каталога до каталога (или файла), содержащего
13  # искомый файл, используя алгоритм итеративного углубления.
14
15
16  class TreeNode:
17      def __init__(self, value):
18          self.value = value
19          self.children = []
20
21      def add_child(self, child):
22          self.children.append(child)
23
24      def add_children(self, *args):
25          for child in args:
26              self.add_child(child)
27
28      def __repr__(self):
```

Рисунок 4 – Условие задания

```
31
32  def iterative_deepening_search(root, target):
33      depth = 0
34      while True:
35          found = depth_limited_search(root, target, depth)
36          if found is not None:
37              return found
38          depth += 1
39
40
41  def depth_limited_search(node, target, depth):
42      if depth == 0 and node.value == target:
43          return node.value
44      if depth > 0:
45          for child in node.children:
46              result = depth_limited_search(child, target, depth - 1)
47              if result is not None:
48                  return result
49      return None
50
```

Рисунок 5 – Реализация итеративного поиска в глубину

Рисунок 7 – Результат работы

Индивидуальное задание:

Поиск файлов с определённым типом данных. В файловой системе есть как текстовые, так и бинарные файлы. Найдите все текстовые файлы (ASCII или UTF-8), начиная с уровня 3, и проверьте их содержимое. Поиск ограничен глубиной 20 уровней

Рисунок 8 – Реализация итеративного поиска в глубину

```
def main():
    # Создание дерева с глубиной и разнообразным содержимым
    root = TreeNode(b"Root data")
    # Уровень 1
    child1 = TreeNode(b"This is a text file.")
    child2 = TreeNode(b"\x89PNG some binary content")
    child3 = TreeNode(b"Binary data here")
    # Уровень 2
    child1_1 = TreeNode(b"Another text file at level 2.")
    child1_2 = TreeNode(b"Yet another text file.")
    child2_1 = TreeNode(b"\x00\xFF Binary file again")
    child3_1 = TreeNode(b"Level 2 text data.")

    # Уровень 3 (глубина начала анализа)
    child1_1_1 = TreeNode(b"This text is at level 3.")
    child1_1_2 = TreeNode(b"\xDE\xAD\xBE\xEF Still binary content")
    child1_2_1 = TreeNode(b"Level 3 valid UTF-8 text.")
    child3_1_1 = TreeNode(b"Text deep in level 3.")

    # Уровень 4
    child1_1_1_1 = TreeNode(b"Deeper text in level 4.")
    child1_2_1_1 = TreeNode(b"Some more text at level 4.")
    child3_1_1_1 = TreeNode(b"Binary\x00 data in level 4")

    # Построение дерева
    root.add_children(child1, child2, child3)
    child1.add_children(child1_1, child1_2)
    child2.add_child(child2_1)
    child3.add_child(child3_1)
    child1_1.add_children(child1_1_1, child1_1_2)
    child1_2.add_child(child1_2_1)
    child3_1.add_child(child3_1_1)
    child1_1_1.add_child(child1_1_1_1)
    child1_2_1.add_child(child1_2_1_1)
    child3_1_1.add_child(child3_1_1_1)

    # Поиск текстовых файлов с уровня 3
    text_files = search_text_files(root, start_level=3)

    # Печать результатов
    print("Найденные текстовые файлы с уровня 3:")
    for node, content in text_files:
        print(f"Node: {node}, Content: {content}")


if __name__ == "__main__":
    main()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

(base) C:\Users\Лео\Desktop\3 Курс\ИИ\ii-5>python .\prog\idz.py
Найденные текстовые файлы с уровня 3:
Node: <b'This text is at level 3.'>, Content: This text is at level 3.
Node: <b'Deeper text in level 4.'>, Content: Deeper text in level 4.
Node: <b'Level 3 valid UTF-8 text.'>, Content: Level 3 valid UTF-8 text.
Node: <b'Some more text at level 4.'>, Content: Some more text at level 4.
Node: <b'Text deep in level 3.'>, Content: Text deep in level 3.
Node: <b'Binary\x00 data in level 4'>, Content: Binary data in level 4
```

Рисунок 9 – Результат работы

Ссылка на github: https:\\github.com/FiaLDI/ii-5

Контрольные вопросы:

Ответы на вопросы:

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

Параметр n обычно обозначает текущую глубину или уровень ограничений в поиске. Он определяет, насколько глубоко алгоритм может

исследовать дерево. Чем больше значение n, тем глубже осуществляется поиск, но это также увеличивает затраты по времени и памяти.

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

Оптимальная глубина зависит от расположения решения в дереве. Без знания структуры дерева или местоположения решения невозможно определить, на каком уровне оно находится. Установка слишком большого значения увеличивает издержки, а слишком маленького — может упустить решение.

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

Меньшее использование памяти: Итеративное углубление использует память, пропорциональную глубине текущего уровня, в отличие от поиска в ширину, который требует хранения всех узлов текущего уровня. Находит оптимальное решение в условиях равных затрат на переходы.

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

Алгоритм повторяет поиск с ограничением глубины, постепенно увеличивая это ограничение. На каждой итерации он выполняет поиск заново, начиная с корня, что позволяет использовать только стек вызовов вместо хранения всех узлов. Это экономит память, но за счет увеличения времени работы.

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

Это связано с принципом ограничения глубины: алгоритм должен исследовать все пути на заданной глубине перед тем, как углубиться дальше. Начало поиска с корня гарантирует, что все узлы на новых уровнях будут проверены.

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

Временная сложность: O(b^d), где b — коэффициент разветвления, d — глубина решения. Пространственная сложность: O(d), так как требуется только стек вызовов.

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

Сохраняет память, как поиск в глубину. Находит оптимальные решения, как поиск в ширину (при равных затратах переходов).

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

Большая часть работы выполняется на последних уровнях. Повторное исследование верхних уровней занимает небольшую часть времени по сравнению с затратами на исследование глубоких уровней.

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

Общее число узлов приближается к b^d. Повторное исследование верхних уровней увеличивает суммарное количество узлов лишь на коэффициент порядка b·d

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

Если дерево имеет очень высокий коэффициент разветвления. Если решение находится на большой глубине, и дерево имеет значительную избыточность.

11. Какую задачу решает функция iterative_deepening_search?

Она выполняет поиск решения в дереве или графе, постепенно увеличивая предел глубины до нахождения решения.

12. Каков основной принцип работы поиска с итеративным углублением?

Постепенное увеличение максимальной глубины поиска, начиная с корня, пока не будет найдено решение.

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

Problem — это объект, описывающий задачу поиска. Обычно он включает стартовый узел, функцию проверки решения и функцию генерации следующих состояний.

14. Какова роль переменной limit в алгоритме?

Limit задает максимальную глубину, до которой алгоритм может исследовать дерево на текущей итерации.

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

Это способ задания "бесконечного" цикла, где глубина будет постепенно увеличиваться с каждым шагом, пока не будет найдено решение.

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

Это позволяет избежать излишнего расхода ресурсов и обеспечивает нахождение оптимального решения на минимальной глубине.

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

Функция depth_limited_search. Она выполняет поиск с ограничением по глубине и возвращает либо найденное решение, либо статус "обрезание".

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

Она проверяет все узлы до заданной глубины. Возвращает:

1) Найденное решение.
2) Статус "обрезание" (если глубина недостаточна).
3) Пустой результат (если решение отсутствует).

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

Cutoff указывает, что поиск был прерван из-за ограничения глубины, но потенциально решение может находиться глубже.

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

Это позволяет алгоритму решить, нужно ли увеличивать глубину на следующей итерации.

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

Алгоритм сразу завершится, вернув найденное решение.

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

Если решение не найдено, алгоритм продолжает увеличивать глубину, пока не будет исчерпана доступная память или время выполнения.

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

1) Избегает бесконечных циклов.
2) Находит оптимальное решение.
3) Использует минимальную память.

24. Какие потенциальные недостатки может иметь этот подход?

Повторное исследование верхних уровней увеличивает временные затраты. Невозможность применения в условиях с высокой сложностью пересчета.

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

Использовать эвристики для направления поиска. Применить алгоритмы с памятью, такие как IDA.

Вывод: в ходе работы были приобретены навыки по работе с поиском с итеративным углублением с помощью языка программирования Python версии 3.x