Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №3 по дисциплине "Проектирование экспертных систем"

Тема Алгоритм обратного метода поиса в глубину в графах И/ИЛИ

Студент Варламова Е. А.

Группа ИУ7-33М

Оценка (баллы) _____

Преподаватели Русакова З.Н.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – реализовать алгоритм обратного метода поиса в глубину в графах И/ИЛИ.

Для достижения поставленной цели потребуется:

— описать используемые структуры данных;

— описать алгоритм обратного метода поиса в глубину в графах И/ИЛИ;

— привести реализацию алгоритма;

— привести примеры работы алгоритма.

1 | Используемые структуры данных

Разработаем класс вершины графа. Поля класса:

— номер вершины;

— флаг вершины – открыта/закрыта/просмотрена/запрещена.

Разработаем класс правила. Поля класса:

— список входных вершин;

— целевая вершина;

— номер правила;

— флаг правила – открыта/закрыта/просмотрена/запрещена.

Разработаем класс поиска. Поля класса:

— список правил;

— 2 флага: есть решение, нет решения, поставим их в 1;

— целевая вершина;

— стек открытых вершин;

— список открытых правил;

— список закрытых вершин;

— список закрытых правил;

— список запрещенных вершин;

— список запрещенных правил.

2 | Алгоритм поиска по графу в ширину от данных

Вход:

— доказанные вершины;

— целевая вершина.

Выход:

— информация о решении.

2.1 Основной метода поиска

Поместить все доказанные вершины в список закрытых вершин. Добавить целевую вершину в стек открытых вершин. Поа флаги решения истинны, выполняем:

— вызвать метод потоми, который возвращает количество закрытых правил;

— если флаг решение найдено, то выход с сообщением об успешном поиске;

— если количество закрытых правил равно 0 и стек содержит не более одной вершины, то выход с сообщением о неудачном поиске;

— если количество закрытых правил равно 0, то вызывать метод обратного хода (backtracking).

2.2 Метод потомки

В цикле по списку правил выполнить.

— Если правило не открыто, то пропустить его.

— Если выходная вершина текущего правила совпадает с текущей вершиной стека, то отметить правило как посещённое, добавить его в список открытых правил и записать в стек все вершины левой части правила, которые не закрыты. Если таких нет, то вызывать метод маркирования. Увеличить количество закрытых правил.

Вернуть количество закрытых правил.

2.3 Метод маркирования (label)

— Взять последнее правило из списка открытых правил (удалив его) и поместить его в список закрытых.

— Взять верхний элемент N из стека открытых вершин (удалив его) и поместить его в список закрытых вершин.

— Если вершина N равна целевой, то изменить флаг наличия решения.

— Получить последний элемент списка открытых правил M (без удаления) и верхний элемент стека открытых вершин K (без удаления). Если целевая вершина правила M совпадает с вершиной K, то вернуться к первому шагу (так как в таком случае очевидно, что в стеке открытых вершин нет вершин из левой части этого правила, а значит оно доказано и может быть перемещено в закрытые).

2.4 Метод обратного хода (backtracking)

— Взять верхний элемент стека открытых вершин (с удалением) и поместить его в список запрещенных вершин.

— Взять последний элемент списка открытых правил (с удалением) и поместить его список запрещённых правил.

— Удалить из стека открытых вершин все вершины, которые находятся в левой части запрещённого правила.

3 | Реализация

Листинг 3.1: Структуры данных

```
class Label(Enum):
    OPEN = 0
    CLOSE = 1
    FORBIDDEN = -1
    VIEWED = 2


class Node:
    def __init__(self, number: int, flag: int = Label.OPEN):
        self.number = number
        self.flag = flag

    def __str__(self):
        res = '' + f'{self.number}'
        return res

    def __repr__(self):
        res = '' + f'{self.number}'
        return res


class Rule:
    def __init__(self, number: int, out_node: Node, node_arr: List[Node],
       label=Label.OPEN):
        self.number = number
        self.out_node = out_node
        self.node_arr = node_arr
        self.label = label
```

Листинг 3.2: Класс поиска

```

class Search:
    def __init__(self, rule_arr: [Rule]):
        self.rule_arr = rule_arr  #
        self.open_node_st = Stack()
        self.open_rule_lst = []
        self.close_node_lst = []
```

```
        self.close_rule_lst = []
        self.prohibited_node_lst = []
        self.prohibited_rule_lst = []

        self.goal_node = None
        self.solution_flg = 1
        self.no_solution_flg = 1
        self.no_label = 1

    def run(self, goal_node: Node, in_node_arr: [Node]):
        self.goal_node = goal_node
        self.open_node_st.push(goal_node)
        self.close_node_lst = in_node_arr

        while self.solution_flg and self.no_solution_flg:
            rule_cnt = self.child_search()

            # solution was found
            if self.solution_flg == 0:
                print("Solution was found")
                return

            if rule_cnt == 0 and self.open_node_st.length() < 2:
                self.no_solution_flg = 0
                print("Solution was not found")
            elif rule_cnt == 0:
                print("Backtracking process is going to be launched")
                self.backtracking()

    def child_search(self):
        cnt_rules = 0

        for rule in self.rule_arr:
            print(f'[Rule {rule.number}] Current rule')

            current_node = self.open_node_st.peek()
            print(f'[Node {current_node.number}]  Current node')

            if rule.label != Label.OPEN: #
                print(f'[Rule {rule.number}] was already processed')
                print('-' * 128 + '\n')
                continue

            if rule.out_node == current_node:
                print(f'[Rule {rule.number}] has out node that equals goal
                    one')

                rule.label = Label.VIEWED
                self.open_rule_lst.append(rule)
                is_new_goal_added = self.add_new_goal(rule.node_arr)
```

```
57                  if not is_new_goal_added:
58                      print("Label process is going to be launched")
59                      self.label()
60
61                  cnt_rules += 1
62                  self.print_info(rule)
63                  break
64
65              if self.is_prohibited_node_exist(rule.node_arr):
66                  self.prohibited_rule_lst.append(rule)
67                  rule.label = Label.FORBIDDEN
68
69                  self.print_info(rule)
70                  continue
71
72              self.print_info(rule)
73
74          return cnt_rules
75
76      def label(self):
77          while True:
78              rule = self.open_rule_lst.pop()
79              self.close_rule_lst.append(rule)
80
81              node = self.open_node_st.pop()
82              self.close_node_lst.append(node)
83
84              print(f'[Labelling] Rule {rule.number} was added to close rules
                    ')
85              print(f'[Labelling] Node {node.number} was added to close nodes
                    ')
86
87              if node == self.goal_node:
88                  self.solution_flg = 0
89                  return
90
91              current_node = self.open_node_st.peek()
92              current_rule = self.open_rule_lst[-1]
93              if current_rule.out_node != current_node:
94                  return
95
96      def backtracking(self):
97          current_goal = self.open_node_st.pop()
98          rule = self.open_rule_lst.pop()
99
100         current_goal.flag = Label.FORBIDDEN
101         self.prohibited_node_lst.append(current_goal)
102
103         rule.label = Label.FORBIDDEN
104         self.prohibited_rule_lst.append(rule)
```

```
105
106         print(f'[Backtrack] Rule {rule.number} was added to prohibited rules
                ')
107         print(f'[Backtrack] Node {current_goal.number} was added to
                prohibited nodes')
108
109         for node in rule.node_arr:
110             print(f'[Backtrack] Node {node.number} should be removed from
                    opened nodes')
111             self.open_node_st.remove_element(node)
112         print()
113
114     def add_new_goal(self, node_arr: [Node]):
115         new_goal_flg = False
116
117         for node in node_arr[::-1]:
118             if node not in self.close_node_lst:
119                 self.open_node_st.push(node)
120                 new_goal_flg = True
121         return new_goal_flg
122
123     def is_prohibited_node_exist(self, node_arr: [Node]):
124         for node in node_arr:
125             if node in self.prohibited_node_lst:
126                 return True
127         return False
```

4 | Пример работы

Входной граф:

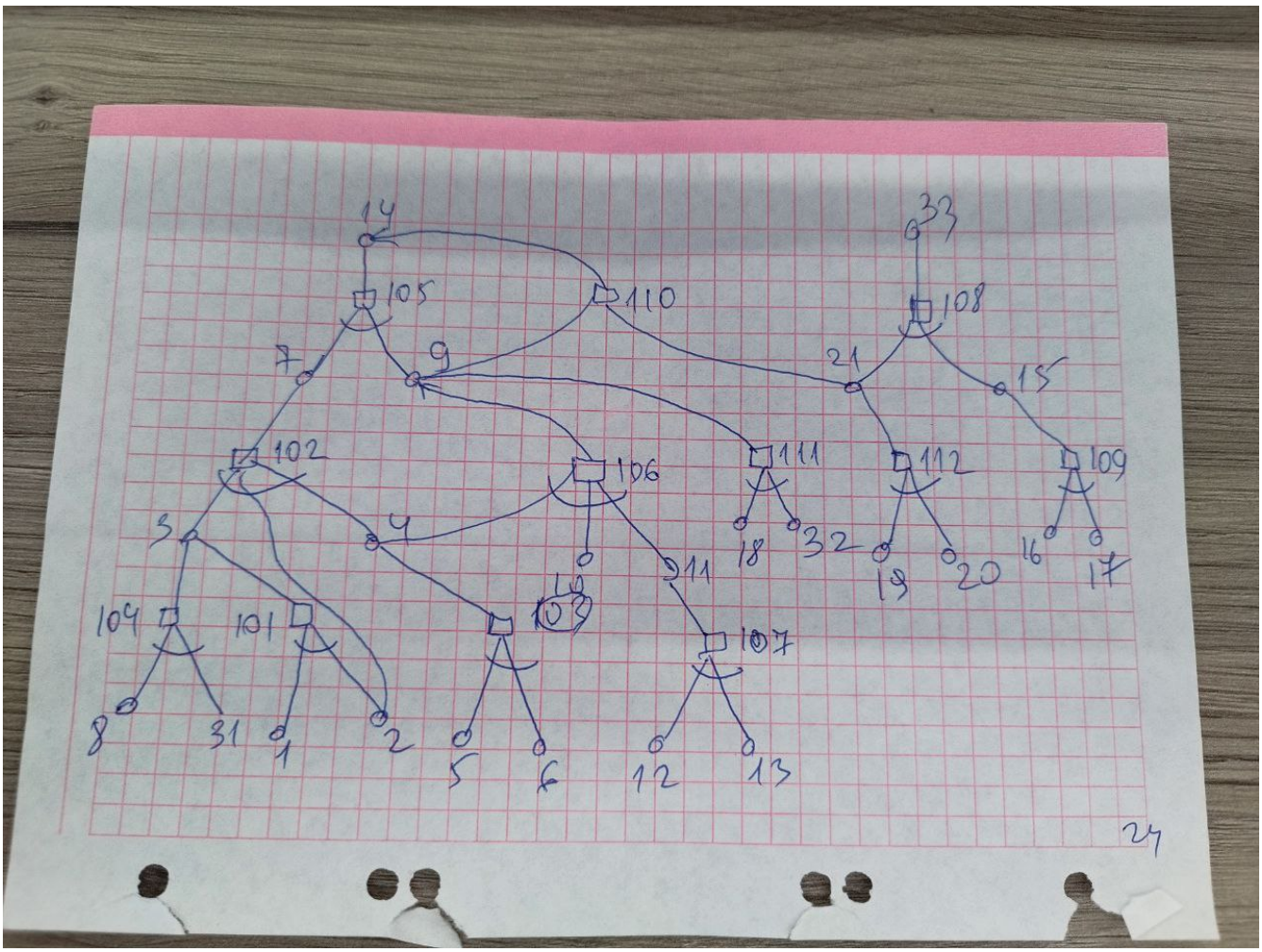

Рис. 4.1: Граф

Результат работы программы:

Листинг 4.1: Результат работы программы

```
1
2 [Rule 105] has out node that equals goal one
3 [Rule 105] list of opened nodes:     14 7
4 [Rule 105] list of closed nodes:     3 4 9 21
```

```
[Rule 105] list of prohibited nodes:
[Rule 105] list of opened rules:     105
[Rule 105] list of closed rules:
[Rule 105] list of prohibited rules:
-----------------------------------------

[Rule 102] has out node that equals goal one
[Rule 102] list of opened nodes:     14 7 2
[Rule 102] list of closed nodes:     3 4 9 21
[Rule 102] list of prohibited nodes:
[Rule 102] list of opened rules:     105 102
[Rule 102] list of closed rules:
[Rule 102] list of prohibited rules:
-----------------------------------------

Backtracking process is going to be launched
[Backtrack] Rule 102 was added to prohibited rules
[Backtrack] Node 2 was added to prohibited nodes
[Backtrack] Node 3 should be removed from opened nodes
[Backtrack] Node 2 should be removed from opened nodes
[Backtrack] Node 4 should be removed from opened nodes

[Rule 101] list of opened nodes:     14 7
[Rule 101] list of closed nodes:     3 4 9 21
[Rule 101] list of prohibited nodes: 2
[Rule 101] list of opened rules:     105
[Rule 101] list of closed rules:
[Rule 101] list of prohibited rules: 102 101
-----------------------------------------

Backtracking process is going to be launched
[Backtrack] Rule 105 was added to prohibited rules
[Backtrack] Node 7 was added to prohibited nodes
[Backtrack] Node 7 should be removed from opened nodes
[Backtrack] Node 9 should be removed from opened nodes

[Rule 110] has out node that equals goal one
Label process is going to be launched
[Labelling] Rule 110 was added to close rules
[Labelling] Node 14 was added to close nodes
[Rule 110] list of opened nodes:
[Rule 110] list of closed nodes:     3 4 9 21 14
[Rule 110] list of prohibited nodes: 2 7
[Rule 110] list of opened rules:
[Rule 110] list of closed rules:     110
[Rule 110] list of prohibited rules: 102 101 105
-----------------------------------------

Solution was found
```

ЗАКЛЮЧЕНИЕ

В результаты работы цель была достигнута.
Для достижения поставленной цели потребовалось:

— описать используемые структуры данных;

— описать алгоритм обратного метода поиса в глубину в графах И/ИЛИ;

— привести реализацию алгоритма;

— привести примеры работы алгоритма.