Лекция 4

Обратный поиск в графах И ИЛИ

Задано:

- Целевая вершина
- Исходные данные

Исходные данные помещаем в список закрытых вершин (они известны). Целевую помещаем в список открытых. Список открытых вершин используется как стек в методе поиска в глубину.

Вводим списки (рабочая память):

- Открытых, закрытых правил
- Открытых, закрытых вершин

Идея алгоритма: использовать стек для раскрытия подцелей. Параллельно формировать список открытых правил. Вершина будет закрыта, если она в исходных данных или будет являться выходом модуля правила, который доказан в процессе поиска. Доказан значит что все его входные вершины покрываются закрытыми вершинами. В список запрещённых вершин помещаются вершины, которые на раскрываются на текущем шаге поиска. Список запрещённых правил, которые включают правила, в которых хотя бы одна входная вершина запрещена.

Методы:

- обратный метод поиска.
- определение потомков текущей подцели
- Backtracking или возврат. Формируются списки запрещённых вершин или правил
- Метод, который выполняет разметку правил.
- Метод, который строит дерево поиска по списку закрытых правил.

Флаги:

- есть решение
- нет решения
- флаг разметки

Метод поиска потомков в графах И/ИЛИ

```
j (счётчик правил) := 0
указатель правил := 0

пока не конец списка правил и j == 0 выполнить:
// цель - найти текущий модуль правила, который раскрывает ...
// на первом шаге это целевая заданная вершина
// наша цель это просмотреть базу правил, найти первый модуль, который раскрывает
// текущую подцель, выбирая правила, у которых метка 0 (не просмотрены)
// если такой модуль нашли, то j += 1, правила записываем в список открытых правил
// и определяем новые подцели, которые являются входными вершинами правила,
// которые не покрываются исходыми данными.
// Мы их записываем в голову стека, то есть списка открытых вершин
// Если новые поцели есть, то модуль ещё не доказан
```

Если все входные вершины покрываются закрытыми (исходными данными), то можно

- вызывать метод разметки
- или сбросить флаг разметки в 0, а вызвать его уже из поиска

Код метода потомки

… пишутся в голову и …

В цикле пока не конец базы правил и j == 0:

…

Выход из цикла при j == 1 или не нашли, но дошли до конца базы правил.

1 вариант: если все входные вершины найденного правила закрыты, то можно отсюда вызвать метод разметки.

2 вариант:

Метод backtracking

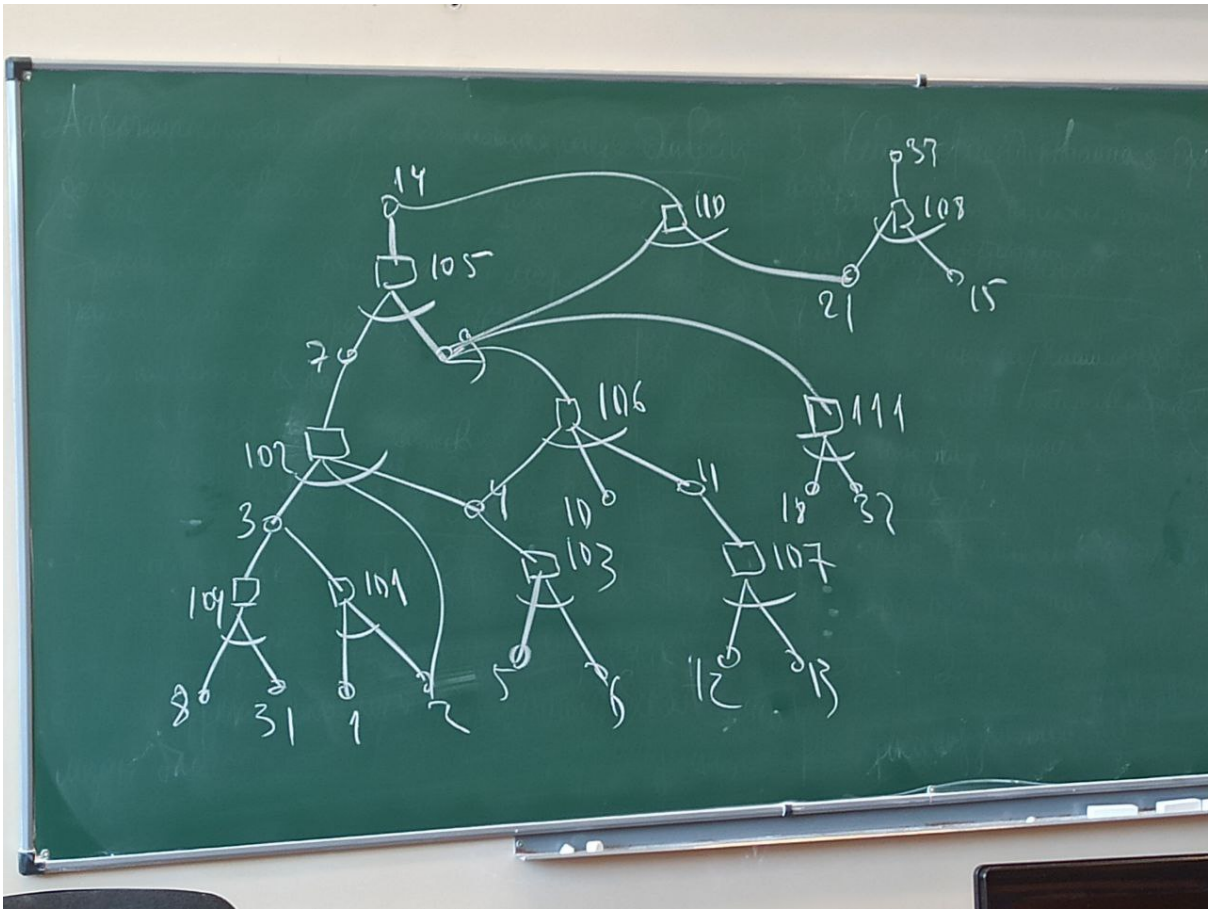
Метод поиска

Который организует вызов других методов

1 случай - рассматриваем пока нет решения.

Алгоритм

В цикле пока fy == 1 и fn == 1. Вызываем метод потомков k = … . Если флаг решения 0 (устанавливаем в методе потомков) тогда return (решение найдено). Иначе если число потомков 0 и список открытых содержит только одну вершину (целевую), то флаг fn (нет решения) сбрасываем в 0. Если j == 0, а стек содержит больше одной вершины, то вызываем метод возврата.



Рассмотрим часть алгоритма

вх. [18, 32, 21]

SOW	SOP	SZW	SZP

[14]	[]	[]	[]
[7, 9, 14]	[105]	[]	[]
[3, 2, 4, 7, 9, 14]	[102, 105]	[]	[]
[8, 31, 3, 2, 4, 7, 9, 14]	[104, 102, 105]	[]	[]
[3, 2, 4, 7, 9, 14]	[102, 105]	[8]	[104]
[1, 2, 3, 2, 4, 7, 9, 14]	[101, 102, 105]	[8]	[104]
[3, 2, 4, 7, 9, 14]	[102, 105]	[8, 1]	[101, 104]
[7, 9, 14]	[105]	[3, 8, 1]	[102, 101, 104]

Порядок выбора:

105

102

104

101

103

106

110

111

107

112

108

109

112

В методе поиска в глубину вы находим самое левое дерево решений.

Если нет раскрывающего модуля. В этом случае j =0, список не пуст, то идёт организация возврата

Текущую подцель пишем в запрещённые. Также из стека открытых вершин удалить все подцели соответствующие входным вершинам запрещённого правила. Количество этих подцелей проверяется флагом входной вершины.

Переходим на следующий шаг. Из головы извлекаем подцель. Эта подцель вход последнего модуля в списке правил.