Московский физико-технический институт
Физтех-школа прикладной математики и информатики

ОСНОВЫ КОМБИНАТОРИКИ И ТЕОРИИ ЧИСЕЛ

II СЕМЕСТР

Лектор: *Райгородский*

Автор: *Киселев Николай*
Репозиторий на Github

весна 2025

Содержание

1 Продолжаем графы

1.1 2-КНФ

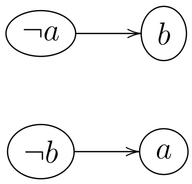
Дана 2-КНФ. Есть ли у нее решение?

Кто не помнит, что такое 2-КНФ, вот пример: $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_4) = 1$
Понятно, что решение найдется только в случае равенства всех скобок.

$$a \vee b = (\neg a \to b) = (\neg b \to a)$$

Перепишем все дизъюнкции через импликации по правилу выше

1. Введем граф: в нем $2n$ вершин (для каждой переменной будет она и ее отрицание) и $2m$ ребер, соответствующие импликациям.



Утверждение 1.1. *Формула выполнима $\iff \forall p$ вершины p и $\neg p$ лежат в разных КСС*

Доказательство. Пусть φ выполнима, но p и $\neg p$ лежат в одной КСС. Переходя по стрелкам, мы однозначно определяем, что $p = \neg p = 1$

В обратную сторону докажем, запустив алгоритм Косарайю:

1. Положим $p = 1$, если $id[p] > id[\neg p]$, и $p = 1$, если $id[\neg p] > id[p]$

Докажем, что это - выполняющий набор. Пойдем от противного. Есть скобка $(x \vee y) = 0$

$$(x \vee y) = 0 \Rightarrow x = y = 0 \Rightarrow id[\neg x] > id[x], id[\neg y] > id[y]$$

Как мы знаем по прошлой лекции, из наличия ребра $u \to v \Rightarrow id[u] \leqslant id[v]$. Получаем:

$$id[\neg x] > id[x] \geqslant id[\neg y] > id[y] \geqslant id[\neg x]$$

□

1.2 Эйлеровость

Определение 1.1. Эйлеровым циклом в графе называется цикл, проходящий по всем ребрам ровно по 1 разу.

Теорема 1.1. *В ориентированном графе G существует эйлеровый цикл тогда и только тогда, когда после удаления всех изолированных вершин G становится сильно связным и $\forall v$ $indeg(v) = outdeg(v)$*

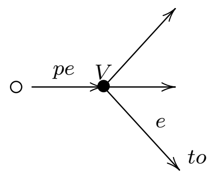
Доказательство. $\Rightarrow$ Удалили все все изолированных вершины, получили граф, где есть цикл по всем ребрам. Тогда понятно, что количество вхождений в вершину равно количеству выходов из нее

$\Leftarrow$ Алгоритм. Пусть стек S - последовательность ребер на эйлеровом цикле.

Напишем **Псевдокод:**

```
def euler(v, pe):
  while из v есть неиспользованные исходящие ребра:
    пусть e - произвольное из них
    e - помечаем использованным
    e = (v, to); ++ptr[v]
    euler(to, e)
  S push(pe)
```

Дальше идет индукция по размеру стека, ограничимся рисунком происходящего



$\square$

Как восстанавливать этот путь?

Лемма 1.1. *Пусть граф G удовлетворяет условию на равенство количества входящих и исходящих ребер, тогда $euler(v, pe)$ первым положит ребро на стек, когда будет в вершине v.*

Доказательство. Понятно из картинки. Мы кладем ребро, когда возвращаемся в v. $\square$

Замечание. Как проверить наличие эйлерова пути? Найти 2 вершины, являющиеся началом и концом потенциального пути, провести ребро и запустить алгоритм.

У конечных вершин indeg и outdeg отличаются на 1, причем в разные стороны

Утверждение 1.2. *Переложить описанный результат на неориентированный граф.*

1.3 DFS на Неориентированных графах

Назовем дерево ребер графа, вдоль которых ходит алгоритм dfs, деревом dfs.

Замечание. Ребра в неориентированном графе могут быть 2 типов:

1. Прямые или древесные - ребра, по которым мы переходим в вершину в алгоритме

2. Обратные (остальные)

Определение 1.2. G - Неориентированный граф, e - ребро в нем. Ребро e называется мостом в нем, если при удалении e в нем количество компонент связности возрастет.

Определение 1.3. V - точка сочленения, если при удалении V количество компонент связности возрастет.

Утверждение 1.3. *Обратные ребра не являются мостами*

Пусть в dfs поддерживает tin. Обозначим $ret[v] = min(tin[v], tin[u])$, где последнее выбирается из тех вершин u, не лежащих в поддереве v, для которых существует обратное ребро $w \to u,\ w \in$ поддереву v

Утверждение 1.4. $ret[v]$ *- минимум из следующих значений:*

1. $tin[v]$
2. $tin[p]$ *по всем обратным ребром* (v, p)
3. $ret[to]$ *по всем прямым ребрам* (v, to)

Утверждение 1.5. *Ребро* (u, v) *является мостом* $\Leftrightarrow$ (u, v) *- древесное и* $ret[v] = tin[v]$