Лекция 4. Задачи преобразования и задачи распознавания. Классы P и NP. Полиномиальные, NP-трудные и NP-полные задачи. NP-полнота задачи k-раскраски графов при каждом заданном числе $k \geqslant 3$. Задача обобщенной выполнимости.

Лектор — Селезнева Светлана Николаевна
selezn@cs.msu.ru

факультет ВМК МГУ имени М. В. Ломоносова

Лекции на сайте http://mk.cs.msu.ru

Задача преобразования

(*Общую*) *задачу преобразования* $\mathcal{Z}$ можно определить ее

1) **входом**,

2) **выходом** и

3) **отображением**.

Вход и **выход** — конечные списки параметров задачи $\mathcal{Z}$. Каждый параметр может принимать значения из некоторого заданного множества.

Отображение — закон, по которому входные параметры преобразуются в выходные.

Если каждому **входному** параметру придать какое-то определенное значение, то получим соответствующую **частную** задачу (или **пример** задачи) $z \in \mathcal{Z}$.

Задача сложения чисел

Например, рассмотрим общую задачу **сложения** целых неотрицательных чисел:

1) **вход**: числа $x, y \in \mathbb{Z}_+$;

2) **выход:** число $u \in \mathbb{Z}_+$;

3) **отображение**: $u = x + y$.

Частная задача сложения чисел (*пример* задачи): $x = 5, y = 9$.

Задача распознавания

(*Общую*) *задачу распознавания* $\mathcal{Z}$ можно определить ее

1) **входом** (или **условием**) и

2) **вопросом** (или **свойством**).

Вход — конечный список параметров задачи $\mathcal{Z}$. Каждый параметр может принимать значения из некоторого заданного множества.

Если каждому параметру придать какое-то определенное значение, то получим соответствующую **частную** задачу (или **пример** задачи) $z \in \mathcal{Z}$.

Вопрос — **свойство** примеров задачи. Для каждого примера задачи требуется выяснить, обладает этот пример указанным **свойством** или нет. Поэтому в задачах распознавания ответом на вопрос может быть **«да»** или **«нет»**.

Задача проверки четности числа

Например, рассмотрим общую задачу **проверки четности** целых неотрицательных чисел:

1) **вход**: число $x \in \mathbb{Z}_+$;

2) **вопрос:** является ли число x четным?

Частная задача проверки четности числа (*пример* задачи): $x = 5$.

Задача выполнимости 3-КНФ

Например, задача **выполнимости 3-КНФ**:

1) **вход**: КНФ $K(x_1, \ldots, x_n) = D_1 \cdot \ldots \cdot D_m$, в которой каждая ЭД D_j содержит не более 3-х литералов, $j = 1, \ldots, m$;

2) **вопрос**: выполнима ли КНФ K, т. е. найдется ли такой набор $\alpha \in E_2^n$, что $K(\alpha) = 1$?

Пример задачи выполнимости 3-КНФ.

$$K(x_1, x_2, x_3, x_4) = (x_1 \vee \bar{x}_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee \bar{x}_4)(\bar{x}_2 \vee \bar{x}_3 \vee x_4).$$

Несложно проверить, что $K(1, 1, 0, 0) = 1$, т. е. КНФ K — выполнима.

Задача выполнимости 2-КНФ

Например, задача **выполнимости 2-КНФ**:

1) **вход**: КНФ $K(x_1, \ldots, x_n) = D_1 \cdot \ldots \cdot D_m$, в которой каждая ЭД D_j содержит не более 2-х литералов, $j = 1, \ldots, m$;

2) **вопрос**: выполнима ли КНФ K, т. е. найдется ли такой набор $\alpha \in E_2^n$, что $K(\alpha) = 1$?

Пример задачи выполнимости 2-КНФ:

$$K(x_1, x_2) = (\bar{x}_1 \vee \bar{x}_2)(\bar{x}_1 \vee x_2)(x_1 \vee \bar{x}_2)(x_1 \vee x_2).$$

Можно проверить, что $K = 0$, т. е. КНФ K не является выполнимой.

Задача k-раскраски графов

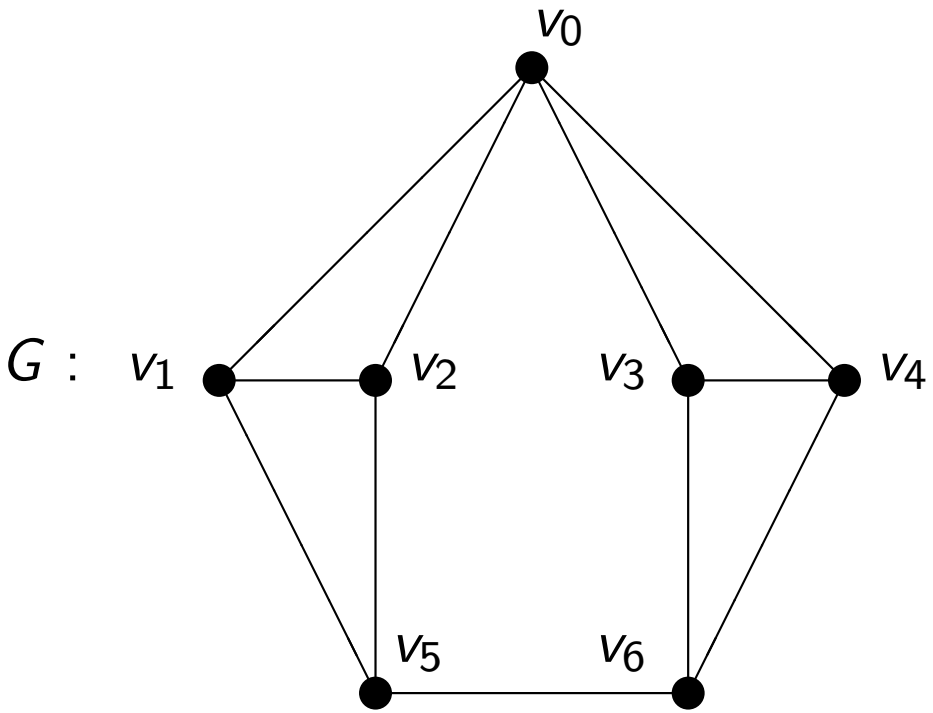
Например, задача k-**раскраски графов**:

1) **вход**: граф $G = (V, E)$, где V — конечное множество вершин и E — множество ребер, т. е. неупорядоченных пар различных вершин из V;

2) **вопрос**: можно ли вершины графа G раскрасить в k цветов, т. е. найдется ли такое отображение $\rho : V \to \{0, 1, \ldots, k-1\}$, что если $(v, w) \in E$, где $v, w \in V$, то $\rho(v) \neq \rho(w)$?

Задача 3-раскраски графов

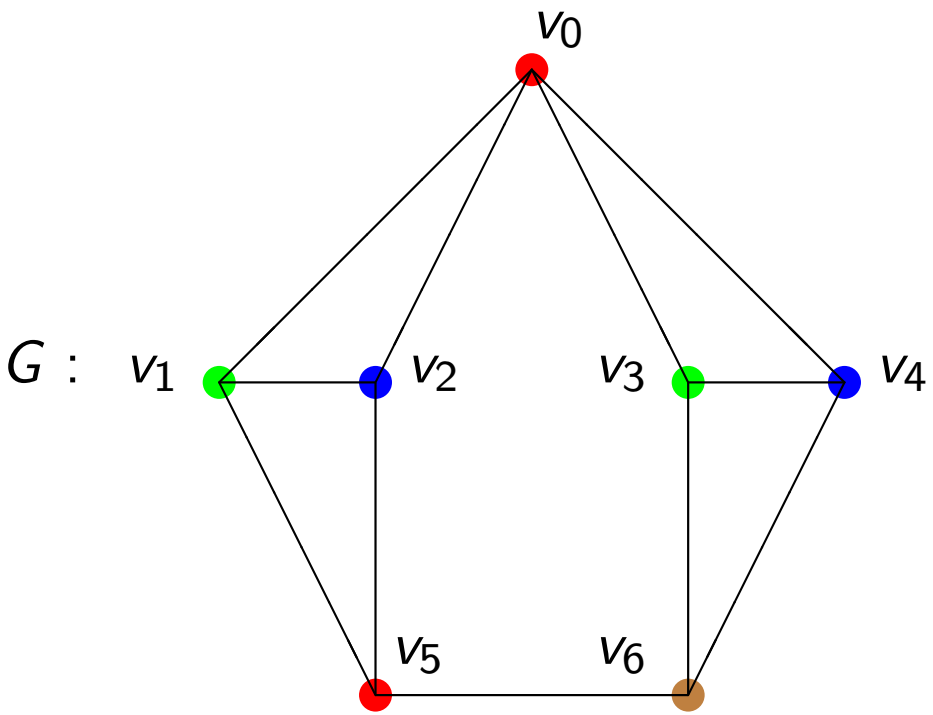
Пример задачи 3-раскраски графов:



Можно проверить, что для графа G ответ «нет», т. е. G не является 3-**раскрашиваемым** графом.

Задача 4-раскраски графов

Пример задачи 4-раскраски графов:



Несложно увидеть, что для графа G ответ «да», т. е. G является 4-**раскрашиваемым** графом.

Язык, соответствующий задаче

С задачей распознавания связывают соответствующий **язык**.

Язык — множество примеров задачи, для которых ответ «да».

Если $\mathcal{Z}$ — задача распознавания, то соответствующий ей язык будем обозначать $L_{\mathcal{Z}}$.

Язык задачи выполнимости 2-КНФ

Например, если $\mathcal{Z}$ — задача **выполнимости 2-КНФ**, то $L_{\mathcal{Z}}$ — множество всех выполнимых 2-КНФ, т. е. таких КНФ с не более двумя литералами в каждой ЭД, которые определяют функцию, не равную тождественно нулю.

Представление примеров задачи

Пусть $\mathcal{Z}$ — (общая) задача (преобразования или распознавания).

Считаем, что все возможные значения параметров задачи $\mathcal{Z}$ записаны **известным образом конечными** словами в некотором **конечном** алфавите A.

Поэтому каждый пример z задачи $\mathcal{Z}$ является **конечным словом** в алфавите A (т. е. принадлежит множеству A^*).

Определим **длину** $|z|$ примера z задачи $\mathcal{Z}$ как число букв алфавита A в записи этого примера.

Задача сложения чисел

Например, для задачи $\mathcal{Z}$ **сложения чисел** все параметры (числа) x, y, u можно записать словами в алфавите $\{0, 1\}$, а именно, соответствующими числами в двоичной системе счисления.

Значит, $x = 5$ записывается как 101, $y = 9$ — как 1001, а их сумма $u = 14$ — как 1110.

Поэтому пример $x = 5$, $y = 9$ можно записать словом $\alpha(z) = 101 + 1001$ в алфавите $A = \{0, 1, +\}$.

Задача выполнимости 2-КНФ

Например, для задачи **выполнимости 2-КНФ** все примеры можно задать соответствующими 2-КНФ в конечном алфавите

$$A = \{(,), \vee, \bar{}, x, 0, 1\}.$$

При этом индексы переменных записываем числами в двоичной системе счисления.

Значит, 2-КНФ

$$K = (x_1 \vee \bar{x}_3)(\bar{x}_2 \vee x_4)$$

задаем словом

$$(x1 \vee \bar{}x11)(\bar{}x10 \vee x100)$$

длины 20, т. е. $|K| = 20$.

Строгие определения

Можно дать строгие определения.

(Общей) **задачей преобразования** $\mathcal{Z}$ называется произвольное отображение

$$f_{\mathcal{Z}} : A^* \to B^*,$$

где A, B — конечные алфавиты.

(Общей) **задачей распознавания** $\mathcal{Z}$ называется произвольное отображение

$$g_{\mathcal{Z}} : A^* \to \{0, 1\},$$

где A — конечный алфавит.

Язык, соответствующий задаче распознавания $\mathcal{Z}$, — множество

$$L_{\mathcal{Z}} = \{\alpha \in A^* \mid g_{\mathcal{Z}}(\alpha) = 1\}.$$

Алгоритм

Алгоритм получает **входные данные**, шаг за шагом выполняет какие-то **действия** в соответствии со своим устройством и выдает **выходные данные**.

Входные и **выходные данные** являются **конечными словами** (т. е. конечными последовательностями букв) в некоторых **конечных** алфавитах A и B соответственно.

Входные данные будем называть **входом**, а выходные данные — **выходом** алгоритма.

Простейшие действия алгоритма

Алгоритм шаг за шагом выполняет какие-то **действия**.

Некоторые из действий над буквами выделяются как **простейшие действия**.

Любое действие алгоритма является последовательностью простейших действий.

Строгие определения

Под **алгоритмом** понимаем *детерминированную машину Тьюринга* (МТ).

Говорим, что алгоритм (МТ) $\mathcal{A}$ **осуществляет отображение** $f_{\mathcal{A}} : A^* \to B^*$, если

1) заданы **входной** и **выходной** алфавиты A и B, где $A, B \subseteq C \setminus \{c_0\}$, C — алфавит букв ленты МТ, c_0 — пустая буква;

2) для каждого слова $\alpha \in A^*$ алгоритм $\mathcal{A}$ **останавливается** на слове α;

3) после остановки на ленте **остается** некоторое слово $\beta = f_{\mathcal{A}}(\alpha) \in B^*$ (т. е слева и справа от этого слова на ленте только c_0).

Сложность алгоритма

Сложность $L_{\mathcal{A}}(N)$ алгоритма $\mathcal{A}$ — наибольшее число простейших действий (тактов работы МТ), которое выполнит алгоритм $\mathcal{A}$ до выдачи выхода, среди всех входов **длины** N (под **длиной** конечного слова понимаем число букв в этом слове).

Алгоритм $\mathcal{A}$ называется **полиномиальным** (или **с полиномиальной сложностью**), если

$$L_{\mathcal{A}}(N) = O(N^s)$$

для некоторого числа s, $s > 0$, постоянного для этого алгоритма $\mathcal{A}$.

Алгоритм, решающий задачу

Алгоритм $\mathcal{A}$ **решает** задачу $\mathcal{Z}$ (*преобразования* или *распознавания*), если для любого примера $z \in \mathcal{Z}$ выполняется следующее: получив на вход этот пример z, алгоритм через конечное число шагов останавливается и выдает **правильный ответ** об этом примере.

Другими словами,

1) для задачи *преобразования* $f_{\mathcal{Z}} : A^* \to B^*$: получив на вход слово $\alpha \in A^*$, алгоритм через конечное число шагов останавливается и выдает $\beta = f_{\mathcal{Z}}(\alpha) \in B^*$;

2) для задачи *распознавания* $g_{\mathcal{Z}} : A^* \to \{0, 1\}$: получив на вход слово $\alpha \in A^*$, алгоритм через конечное число шагов останавливается и при $\alpha \in L_{\mathcal{Z}}$ выдает 1, иначе — выдает 0.

Строгие определения

Пусть $\mathcal{A}$ — алгоритм, осуществляющий преобразование

$$f_{\mathcal{A}} : A^* \to B^*,$$

где $A, B \subseteq C \setminus \{c_0\}$, C — алфавит букв ленты МТ и c_0 — пустая буква.

Алгоритм $\mathcal{A}$ **решает** задачу $\mathcal{Z}$, $f_{\mathcal{Z}} : A^* \to B^*$, если $f_{\mathcal{A}} = f_{\mathcal{Z}}$.

Другими словами, для любого слова $\alpha \in A^*$ выполняется, что

$$f_{\mathcal{A}}(\alpha) = \beta = f_{\mathcal{Z}}(\alpha) \in B^*.$$

Класс P

Задача (*преобразования* или *распознавания*) $\mathcal{Z}$ называется **полиномиальной**, если найдется *полиномиальный* алгоритм, **решающий** эту задачу.

Если задача *распознавания* $\mathcal{Z}$ — полиномиальна, то говорят, что она принадлежит **классу** P, и пишут

$$\mathcal{Z} \in P.$$

Класс P — множество всех задач *распознавания*, которые можно **решить** *полиномиальным* алгоритмом.

Алгоритм, проверяющий задачу

Пусть $\mathcal{Z}$ — задача *распознавания*.

Алгоритм $\mathcal{C}$ с входами z, u **проверяет** с параметрами $c, t > 0$, задачу $\mathcal{Z}$, если для любого примера $z \in \mathcal{Z}$ выполняется следующее: получив на вход этот пример z,

1) в случае $z \in L_{\mathcal{Z}}$ **для каких-то** дополнительных входных данных u_z, $|u_z| \leqslant c \cdot |z|^t$, он через конечное число шагов останавливается и выдает «да»;

2) в случае $z \notin L_{\mathcal{Z}}$ **ни для каких** дополнительных входных данных u, $|u| \leqslant c \cdot |z|^t$, он не может выдать «да».

Для каждого примера $z \in L_{\mathcal{Z}}$ дополнительные входные данные u_z называются **сертификатом** примера z, а алгоритм $\mathcal{C}$ называется **проверкой сертификата**.

Строгие определения

Пусть $\mathcal{C}$ — алгоритм, осуществляющий преобразование

$$f_{\mathcal{C}} : A^* \times B^* \to \{0, 1\},$$

где $A, B, \{0, 1\} \subseteq C \setminus \{c_0\}$, C — алфавит букв ленты МТ и c_0 — пустая буква.

Алгоритм $\mathcal{C}$ **проверяет** с параметрами $c, t > 0$, задачу распознавания $\mathcal{Z}$, $g_{\mathcal{Z}} : A^* \to \{0, 1\}$, если для любого $\alpha \in A^*$ выполняется следующее:

1) если $g_{\mathcal{Z}}(\alpha) = 1$, то **для некоторого** слова $\beta \in B^*$, $|\beta| \leqslant c \cdot |\alpha|^t$, верно $f_{\mathcal{C}}(\alpha, \beta) = 1$;

2) если $g_{\mathcal{Z}}(\alpha) = 0$, то **для любого** слова $\beta \in B^*$, $|\beta| \leqslant c \cdot |\alpha|^t$, верно $f_{\mathcal{C}}(\alpha, \beta) = 0$.

Для каждого слова $\alpha \in L_{\mathcal{Z}}$ соответствующее слово $\beta \in B^*$ называется его **сертификатом**, а алгоритм $\mathcal{C}$ называется **проверкой сертификата**.

Класс NP

Если для задачи *распознавания* $\mathcal{Z}$ найдется **полиномиальный** алгоритм с некоторыми параметрами c, t, $c, t > 0$, **проверяющий** эту задачу, то говорят, что она принадлежит **классу** NP, и пишут

$$\mathcal{Z} \in NP.$$

Класс NP — множество всех задач *распознавания*, которые можно **проверить** *полиномиальным* алгоритмом.

Задача выполнимости 3-КНФ

Задача **выполнимости 3-КНФ**.

Сертификат для выполнимой 3-КНФ $K(x_1, \ldots, x_n)$ — набор $\alpha \in E_2^n$, для которого $K(\alpha) = 1$.

Проверка сертификата — алгоритм, который получает 3-КНФ $K(x_1, \ldots, x_n)$ и набор $\alpha \in E_2^n$, находит значение $K(\alpha)$ и выдает «да» при $K(\alpha) = 1$ и «нет» при $K(\alpha) = 0$.

Задача 3-раскраски графов

Задача 3-**раскраски графов**.

Сертификат для 3-раскрашиваемого графа $G = (V, E)$ — такое отображение $\rho : V \to \{0, 1, 2\}$, что для каждого ребра $(v, w) \in E$ верно $\rho(v) \neq \rho(w)$.

Проверка сертификата — алгоритм, который получает граф $G = (V, E)$ и отображение $\rho : V \to \{0, 1, 2\}$, проверяет значения ρ на концах каждого ребра из E и выдает «да», если для каждого ребра они различны, и «нет», если найдется ребро, для которого они одинаковы.

NP-трудная задача

Задача *распознавания* $\mathcal{Z}_0$ называется NP-**трудной**, если при помощи **полиномиального** алгоритма ее решения для каждой задачи $\mathcal{Z}$ из класса NP можно построить **полиномиальный** алгоритм, решающий задачу $\mathcal{Z}$.

P-сводимость

Пусть $\mathcal{Z}_1$, $\mathcal{Z}_2$ — задачи *распознавания*.

Задача $\mathcal{Z}_1$, $g_{\mathcal{Z}_1} : A_1^* \to \{0,1\}$, **полиномиально сводится** (P-**сводится**) к задаче $\mathcal{Z}_2$, $g_{\mathcal{Z}_2} : A_2^* \to \{0,1\}$, если найдется такой **полиномиальный алгоритм** $\mathcal{A}$, осуществляющий преобразование $f_{\mathcal{A}} : A_1^* \to A_2^*$ примеров задачи $\mathcal{Z}_1$ в примеры задачи $\mathcal{Z}_2$, что для любого слова $z_1 \in A_1^*$ выполняется:

$z_1 \in L_{\mathcal{Z}_1}$ тогда и только тогда, когда $f_{\mathcal{A}}(z_1) = z_2 \in L_{\mathcal{Z}_2}$.

NP-трудная задача

Пусть $\mathcal{Z}_0$ — задача *распознавания*.

Задача $\mathcal{Z}_0$ называется NP-**трудной**, если **любая** задача из NP полиномиально сводится к ней.

Т. е. $\mathcal{Z}_0$ — NP-**трудная задача**, если для любой задачи $\mathcal{Z} \in NP$ выполняется:

$$\text{задача } \mathcal{Z} \ P\text{-сводится к задаче } \mathcal{Z}_0.$$

Любую NP-трудную задачу назовем **труднорешаемой**.

NP-полная задача

Задача *распознавания* $\mathcal{Z}_0$ называется NP-**полной**, если

1) $\mathcal{Z}_0 \in NP$ и

2) $\mathcal{Z}_0$ является NP-трудной.

Задача выполнимости 3-КНФ

Задачу **выполнимости 3-КНФ** обозначим 3-*ВЫП*.

Задача 3-*ВЫП* является NP-полной.

Задача 3-раскраски графов

Задачу k-**раскраски графов** обозначим k-BP.

Теорема 1. *Задача* 3-BP *является* NP-*полной*.

Задача 3-раскраски графов

Доказательство. 1. Сначала покажем, что 3-$BP \in NP$.

Сертификат для 3-раскрашиваемого графа $G = (V, E)$ — такое отображение $\rho : V \to \{0, 1, 2\}$, что для каждого ребра $(v, w) \in E$ верно $\rho(v) \neq \rho(w)$.

Проверка сертификата — алгоритм, который получает граф $G = (V, E)$ и отображение $\rho : V \to \{0, 1, 2\}$, проверяет значения ρ на концах каждого ребра из E и выдает «да», если для каждого ребра они различны, и «нет», если найдется ребро, для которого они одинаковы.

Задача 3-раскраски графов

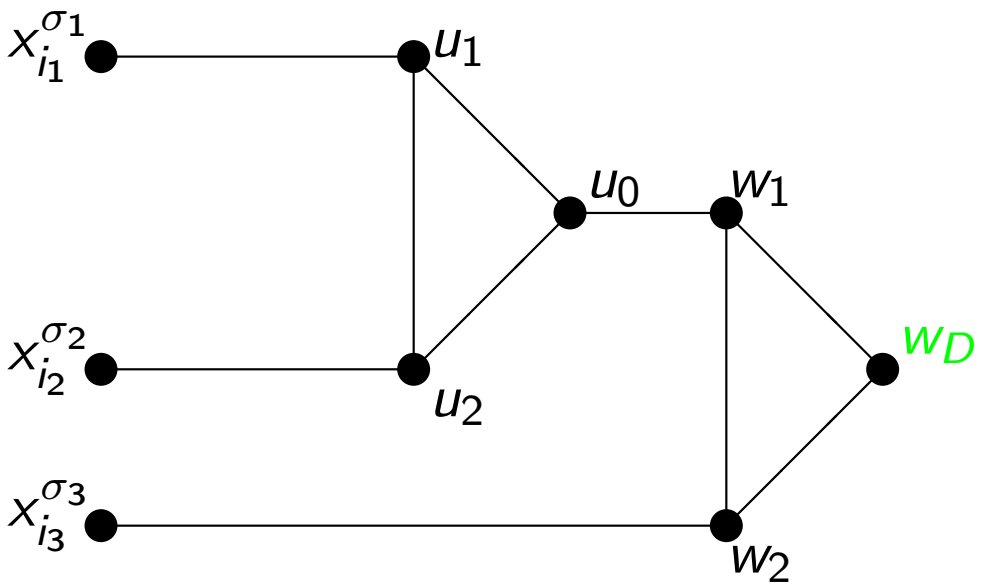
2. Теперь докажем, что 3-BP является NP-трудной.

Для этого полиномиально сведем NP-полную задачу 3-$ВЫП$ выполнимости 3-КНФ к задаче 3-BP.

Задача 3-раскраски графов

Пусть $K(x_1, \ldots, x_n)$ — вход задачи 3-*ВЫП* и $D = x_{i_1}^{\sigma_1} \vee x_{i_2}^{\sigma_2} \vee x_{i_3}^{\sigma_3}$ — произвольная ее элементарная дизъюнкция (ЭД), $\sigma_1, \sigma_2, \sigma_3 \in E_2$.

ЭД D сопоставим граф G_D с множеством вершин $V_D = \{x_{i_1}^{\sigma_1}, x_{i_2}^{\sigma_2}, x_{i_3}^{\sigma_3}, u_0, u_1, u_2, w_1, w_2, w_D\}$:



Задача 3-раскраски графов

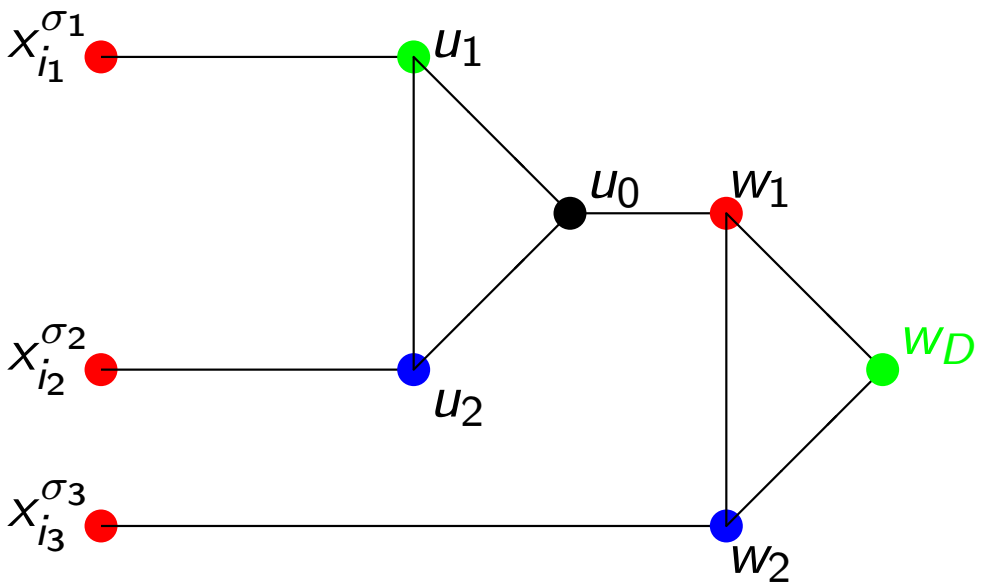
Подграф графа G_D с множеством вершин

$$V_H = \{u_0, u_1, u_2, w_1, w_2, w_D\}$$

и всеми ребрам, соединяющими эти вершины в G_D, обозначим H_D.

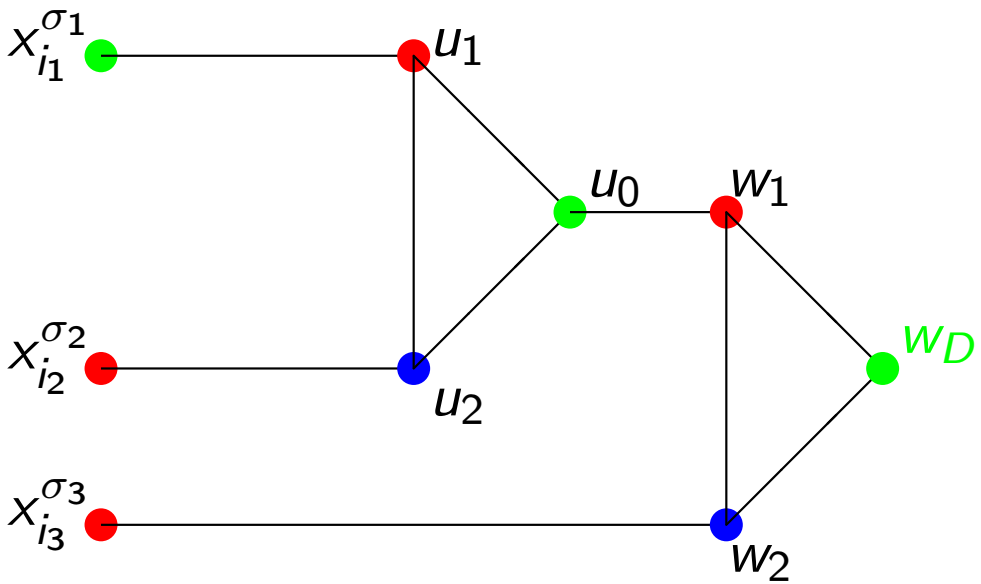
Задача 3-раскраски графов

Лемма 1. *Не существует такой раскраски* $\rho : V_D \to \{0, 1, 2\}$ *графа* G_D, *что* $\rho(x_{i_1}^{\sigma_1}) = \rho(x_{i_2}^{\sigma_2}) = \rho(x_{i_3}^{\sigma_3}) = 0$ *и* $\rho(w_D) = 1$.



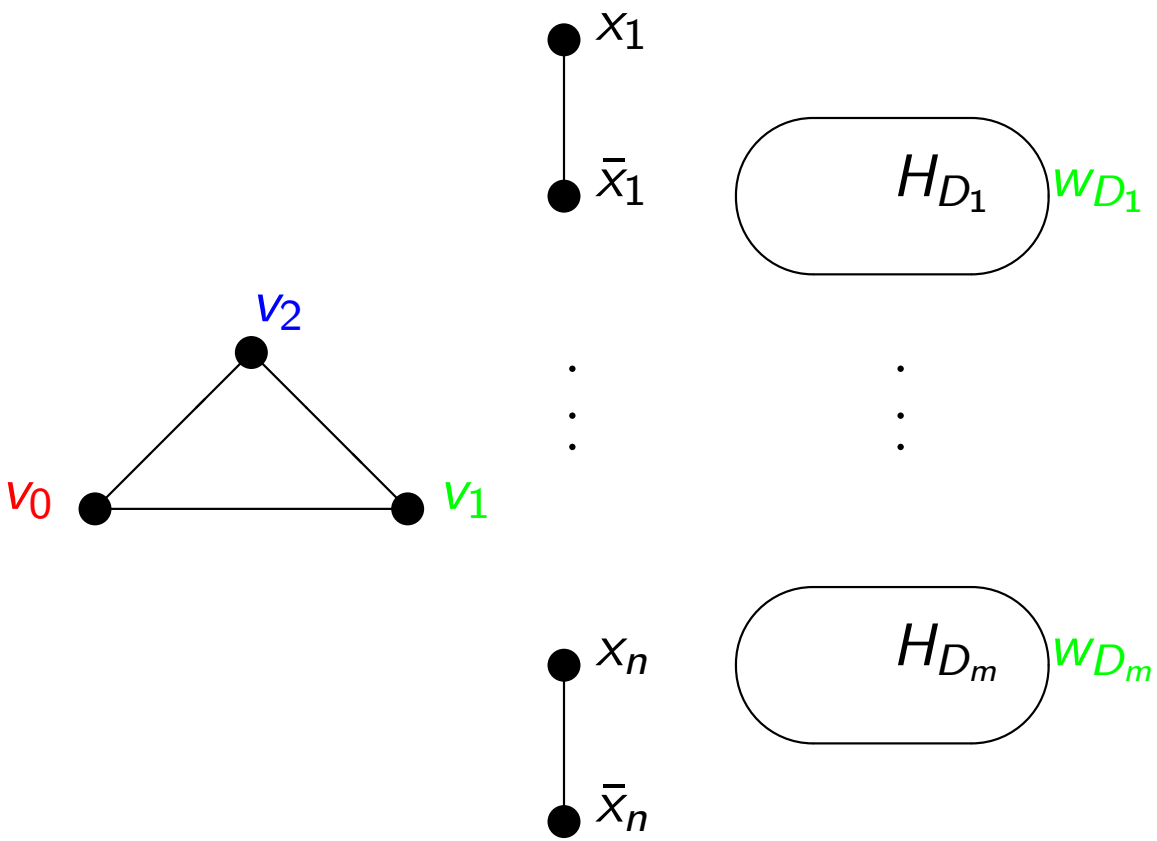
Задача 3-раскраски графов

Лемма 2. *Для любых $a, b, c \in \{0, 1\}$, не равных одновременно 0, существует такая раскраска $\rho : V_D \to \{0, 1, 2\}$ графа G_D, что $\rho(x_{i_1}^{\sigma_1}) = a$, $\rho(x_{i_2}^{\sigma_2}) = b$, $\rho(x_{i_3}^{\sigma_3}) = c$ и $\rho(w_D) = 1$.*



Задача 3-раскраски графов

КНФ $K(x_1, \ldots, x_n) = D_1 \cdot \ldots \cdot D_m$ сопоставим граф G_K:



Задача 3-раскраски графов

В подграфах H_{D_j} каждой вершине, кроме w_{D_j}, припишем верхний индекс (j), $j = 1, \ldots, m$.

Граф $G_K = (V_K, E_K)$ содержит ребра:

1) (v_0, v_1), (v_0, v_2), (v_1, v_2);

2) $(x_i, \bar{x}_i)$ для всех $i = 1, \ldots, n$;

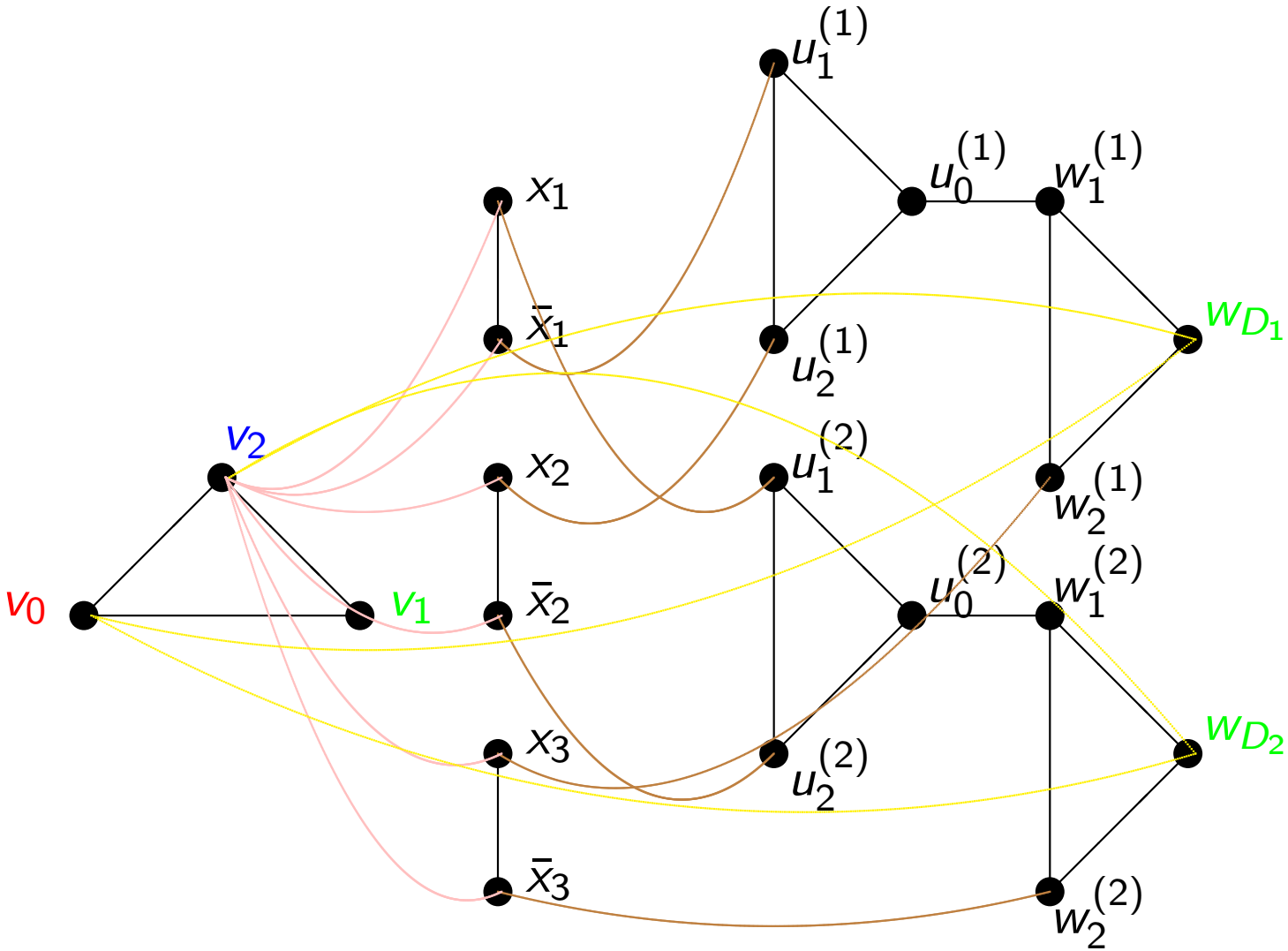
3) (v_2, x_i), $(v_2, \bar{x}_i)$ для всех $i = 1, \ldots, n$;

4) (v_0, w_{D_j}), (v_2, w_{D_j}) для всех $j = 1, \ldots, m$;

5) все ребра подграфов H_{D_j} для всех $j = 1, \ldots, m$;

6) если $D_j = x_{i_1}^{\sigma_1} \vee x_{i_2}^{\sigma_2} \vee x_{i_3}^{\sigma_3}$, где $\sigma_1, \sigma_2, \sigma_3 \in E_2$, и $u_1^{(j)}, u_2^{(j)}, w_2^{(j)}$ — вершины в подграфе H_{D_j}, то $(x_{i_1}^{\sigma_1}, u_1^{(j)})$, $(x_{i_2}^{\sigma_2}, u_2^{(j)})$, $(x_{i_3}^{\sigma_3}, w_2^{(j)})$ для всех $j = 1, \ldots, m$.

Граф G_K для $K = (\bar{x}_1 \vee x_2 \vee x_3)(x_1 \vee \bar{x}_2 \vee \bar{x}_3)$



Задача 3-раскраски графов

1. Отметим что при любой раскраске вершин графа G_K в цвета $\{0, 1, 2\}$ вершины v_0, v_1, v_2 получат разные цвета.

2. Пусть, для определенности, вершина v_i получает цвет i, $i = 1, 2, 3$.

3. Тогда вершины w_{D_j} получают цвет 1.

4. Вершины x_i и $\bar{x}_i$ получают **разные** цвета из 0 и 1.

5. Из лемм 1 и 2 следует, что вершины графа G_K можно раскрасить в 3 цвета тогда и только тогда, когда найдется такой набор $\alpha \in E_2^n$, что $K(\alpha) = 1$.

□

Задача k-раскраски графов

Теорема 2. *При каждом заданном числе $k \geqslant 3$ задача k-BP является NP-полной*.

S-КНФ

Пусть $S \subseteq P_2$ и каждая функция из S имеет свое, отличное от других функций, обозначение.

Определим по индукции S-**КНФ**.

Базис индукции. Если g — обозначение n-местной функции из S и $x_{i_1}, \ldots, x_{i_n}$ — (не обязательно различные) переменные, то выражение $g(x_{i_1}, \ldots, x_{i_n})$ — S-КНФ.

Индуктивный переход. Если $K_1, \ldots, K_m$ — уже построенные S-КНФ, то выражение $K_1 \cdot \ldots \cdot K_m$ — S-КНФ.

S-КНФ

Т. е. если $S \subseteq P_2$, то S-КНФ называем выражение

$$g_1 \cdot \ldots \cdot g_m,$$

где $g_j \in S$, $g_j = g_j(x_{j_1}, \ldots, x_{j_{n_j}})$, $j = 1, \ldots, m$.

Задача обобщенной выполнимости

Задача *распознавания S-ВЫП*, где $S \subseteq P_2$.

Вход: S-КНФ $K(x_1, \ldots, x_n) = g_1 \cdot \ldots \cdot g_m$, где $g_j \in S$, $j = 1, \ldots, m$.

Вопрос: выполнима ли S-КНФ K, т. е. найдется ли такой набор $\alpha \in E_2^n$, что $K(\alpha) = 1$?

Задача обобщенной выполнимости

1. Задача 3-*ВЫП* является задачей S-*ВЫП* с

$$S = \{x_1^{\sigma_1} \vee x_2^{\sigma_2} \vee x_3^{\sigma_3} \mid \sigma_1, \sigma_2, \sigma_3 \in E_2\} \subseteq P_2.$$

2. Задача 2-*ВЫП* является задачей S-*ВЫП* с

$$S = \{x_1^{\sigma_1} \vee x_2^{\sigma_2} \mid \sigma_1, \sigma_2 \in E_2\} \subseteq P_2.$$

3. Задача 2-*BP* является задачей S-*ВЫП* с

$$S = \{x_1 \oplus x_2\} \subseteq P_2.$$

Конец лекции