АМВ. ДЗ на неделю 4.

Прохоров Юрий, 771

Задача 0

Постройте NP-сертификат простоты числа $p = 3911$, $g = 13$. Простыми в рекурсивном построении считаются только числа 2, 3, 5.

Решение:

$$p - \text{простое} \iff \left|\left(\mathbb{Z}/p\mathbb{Z}\right)^*\right| = \phi(p) = p - 1,$$

где $\phi(p)$ — функция Эйлера, число взаимно простых с p чисел, меньших p. То есть в мультипликативной группе вычетов по модулю p ровно $p-1$ элементов.

Это эквивалентно тому, что в группе $G = \left(\mathbb{Z}/p\mathbb{Z}\right)^*$ есть порождающий элемент g, $ord(g) = p - 1$. Чтобы его порядок был равен $p-1$ необходимо и достаточно, по определению, чтобы

$$\forall k\ (0 < k < p-1)\ :\ g^k \not\equiv 1 \mod p$$

А для этого достаточно проверить, что $g^k \not\equiv 1 \mod p$ при $k = \frac{p-1}{p_k}$ для всех простых делителей $p-1$.

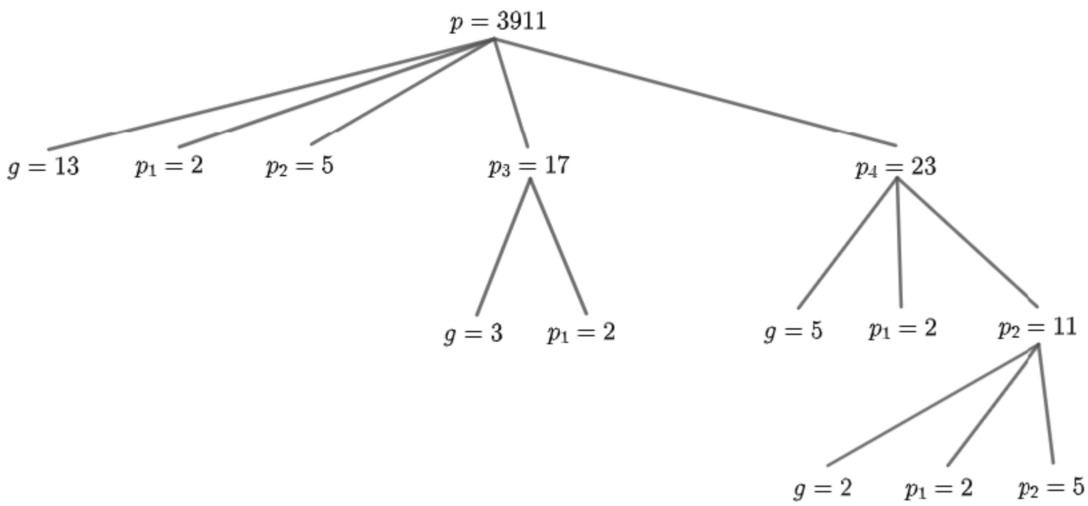
Итак, сертификат простоты числа p состоит из:

1. Генератора g группы G.
2. Простых делителей $p_1, \ldots, p_s$ числа $p-1$.
3. Сертификатов простоты для $p_1, \ldots, p_s$.

В общей сложности можно показать, что длина записи такого сертификата будет полиномиальной.

Верификатор выполняет следующие действия:

1. Проверяет, что $p_1, \ldots, p_s$ — делители $p-1$.
2. Проверяет, что $g^{\frac{p-1}{p_k}} \not\equiv 1 \mod p,\ \forall k = 1, \ldots, s$
3. Проверяет числа $p_1, \ldots, p_s$ на простоту.



Генераторы групп меньших порядков можно найти перебором.

Задача 1

(i) Докажите, что в Σ_2 лежит язык булевых формул от двух наборов переменных $\varphi(x_1, \ldots, x_n, y_1 \ldots y_n) = \varphi(\vec{x}, \vec{y})$ таких, что при некоторых значениях $\vec{x}$ они справедливы вне зависимости от значений $y_1, \ldots, y_n$.

(ii) Придумайте какую-нибудь свою задачу из класса Σ_3 (или Π_3, на ваш вкус).

(iii) Докажите, что $\Sigma_k \subset \Sigma_{k+1} \cap \Pi_{k+1}$.

(iv) Докажите, что $\mathcal{NP} \subset \mathcal{PSPACE} \subset \mathcal{EXPTIME}$.

Решение:

(i) Запишем определение Σ_2, обозначив за L данный язык:

$$x \in L \iff \exists s_1 \, \forall s_2 \; R(x, s_1, s_2) = 1, \qquad R \text{ считается за } \mathcal{P}$$

В нашем случае, x — формула ϕ, $s_1 = \vec{x}$ — выполняющий набор, $s_2 = \vec{y}$ — произвольный набор переменных $\vec{y}$ или мусор, который можно отсеять за полиномиальной время, R — полиномиально вычислимый предикат, подставляющий значения $\vec{x}, \vec{y}$ в формулу ϕ.

Видно, что это определение совпадает с определением языка L, значит, $L \in \Sigma_2$.

(ii) Рассмотрим шашечную доску размера $n \times n$. Пусть у каждого игрока $\Theta(n^2)$ шашек. Шашки могут ходить на любое количество клеток по диагонали (как дамки). За один ход каждый игрок может выбрать любое количество своих шашек (какое-то их подмножество) и подвинуть их. Цель: съесть все шашки соперника.

Рассмотрим язык L, состоящий из всех таких конфигураций доски, когда текущий ход у белых, что белые могут выиграть ровно через один ход. Покажем, что $L \in \Sigma_3$. Запишем формальное определение языка L:

$$x \in L \iff \exists s_1 \, \forall s_2 \, \exists s_3 \; R(x, s_1, s_2, s_3) = 1,$$

где x — состояние доски, s_1 — текущий ход белых, s_2 — произвольный корректный ход черных, s_3 — победный ход белых.

Это совпадает с определением Σ_3. Скорее всего, задача не лежит в Σ_2 или Π_2, потому что тогда придется перебирать все возможные ходы какого-то игрока. А число таких ходов по порядку больше числа всех подмножеств $\Theta(n^2)$-элементного множества. Поэтому такая процедура не будет полиномиальной.

(iii)

$$L \in \Sigma_k : \qquad x \in L \iff \exists s_1 \, \forall s_2 \, \ldots \, Q_k s_k \; R(x, s_1, \ldots, s_k) = 1$$

$$M \in \Sigma_{k+1} : \qquad x \in M \iff \exists s_1 \, \forall s_2 \, \ldots \, Q_k s_k \; Q_{k+1} s_{k+1} \; R(x, s_1, \ldots, s_k, s_{k+1}) = 1$$

Чтобы L удовлетворял определению Σ_{k+1} возьмем в качестве R такую функцию, которая игнорирует сертификат s_{k+1} независимо от квантора перед ним. В качестве такой функции, ясно, подходит функция R из определения принадлежности $L \in \Sigma_k$.

$$M \in \Pi_{k+1} : \qquad x \in M \iff \forall s_{k+1} \, \exists s_1 \, \forall s_2 \, \ldots \, Q_k s_k \; R(x, s_1, \ldots, s_k, s_{k+1}) = 1$$

Аналогично, если R будет игнорировать s_{k+1}, то будет выполнено определение принадлежности Π_{k+1}.

Итак,

$$\Sigma_k \subseteq \Sigma k + 1, \; \Sigma_k \subseteq \Pi_{k+1} \qquad \Longrightarrow \qquad \Sigma_k \subseteq \Big(\Sigma_{k+1} \cap \Pi_{k+1}\Big)$$

(iv) • $\mathcal{NP} \subseteq \mathcal{PSPACE}$

Пусть $L \in \mathcal{NP}$. Тогда существует распознающая его недетерминированная полиномиальная МТ M. Ясно, что M использует полиномиальное число ячеек $p_1(n)$, потому что иначе на их заполнение ушло бы неполиномиальное время.

Описание M конечно, поэтому существует такое число k, что из каждой конфигурации у M не более k возможных вариантов, как попасть в следующую конфигурацию. Таким образом, работу МТ можно представить в виде k-арного дерева глубиной $p_2(n)$, так как МТ полиномиальна.

Будем кодировать каждый возможный путь в этом дереве от корня до листа (он начала работы МТ до конца) с помощью последовательности $p_2(n)$ чисел от 1 до k. Построим детерминированную МТ M'. Дадим ей $\Theta(p_1(n) + p_2(n))$ ячеек памяти, разделенных на 2 блока. В первом блоке длины $\Theta(p_1(n))$ будут производиться вычисления машины M. Во втором блоке длины $\Theta(p_2(n))$ будет записан путь, по которому этой машине M надо производить вычисления, то есть машина M' является детерминированной.

Изначально, во втором блоке стоят все единицы, то есть мы идем по самому левому пути в дереве конфигураций. Затем, после одного вычисления, в последнюю ячейку второго ставится 2 и снова производится детерминированное вычисление по этому пути. Если, хотя бы один путь привел МТ в финальное состояние, то слово принимается.

- $\mathcal{PSPACE} \subseteq \mathcal{EXPTIME}$

Пусть $L \in \mathcal{PSPACE}$. Тогда существует детерминированная МТ M, использующая полиномиальное число $p(n)$ ячеек ленты. Описание M конечно, поэтому для описания одной конфигурации (все, что записано на ленте, положение головки, состояние) нужно слово длины $\Theta(p(n)) = P(n)$ в некотором конечном алфавите Σ мощности k. Тогда описаний всех возможных конфигураций МТ не более $\Theta\big(k^{P(n)}\big)$.

Составим их этих описаний ориентированный граф, в котором вершинами будут конфигурации, а ребра будут между между такими конфигурациями, что из одной можно за 1 такт МТ попасть в другую. Граф составляется за экспоненциальное время. Слово x принимается машиной M тогда и только тогда, когда в этом графе есть путь из начальной конфигурации в какую-то конечную. Для этого достаточно сделать обход в ширину за время $\Theta\Big(k^{\Theta(P(n))}\Big)$, что тоже является экспоненциальным.

Задача 2

Покажите, как свести следующую задачу к вычислению некоторого перманента: найти количество перестановок n элементов, в которых части элементов (с номерами $i_1, i_2, \dots i_k$) запрещено занимать позиции $j_1, \dots j_k$ соответственно.

Решение:

Переобозначим элементы так, что $i_s = s$, $\forall s$. Пусть также $j_s = s$, $\forall s$, на количество корректных перестановок это не повлияет. Составим матрицу

$$A = \left(\begin{array}{ccccc|cc}
0 & 1 & 1 & \dots & 1 & \dots & 1 \\
1 & 0 & 1 & \dots & 1 & \dots & 1 \\
1 & 1 & 0 & \dots & 1 & \dots & 1 \\
\vdots & \vdots & \vdots & \ddots & 1 & \dots & \vdots \\
1 & 1 & 1 & \dots & 0 & \dots & 1 \\
\cline{1-5}
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & 1 & \dots & 1 & \dots & 1
\end{array}\right)_{n\times n},$$

в которой в подматрице размера $k \times k$ на диагонали стоят все нули, а все остальные элементы в матрице A равны единице. Итак, матрица обладает свойством

$$a_{ij} = \begin{cases} 1, & \text{элемент } i \text{ может стоят на позиции } j \\ 0, & \text{иначе} \end{cases}$$

Тогда перестановка $j_1, j_2, \ldots, j_n$ является корректной тогда и только тогда

$$a_{1j_1} a_{2j_2} \ldots a_{nj_n} = 1$$

Число таких перестанок как раз и есть перманент матрицы A:

$$\operatorname{perm}(A) = \sum_{(j_1,\ldots,j_n)} a_{1j_1} \ldots a_{nj_n}$$

Задача 3

Докажите, что язык выполнимых ДНФ $C_1 \vee C_2 \vee \ldots \vee C_m$, где $C_i = (l_{i1} \wedge l_{i2} \wedge \ldots \wedge l_{ik_i})$, l_{ij} — литералы, принадлежит $\mathcal{P}$. Найдите ошибку или пробел в рассуждении: любую КНФ можно преобразовать в эквивалентную ДНФ, поэтому задача выполнимости КНФ сводится к задаче выполнимости ДНФ и лежит в $\mathcal{P}$.

Решение:

• ДНФ выполнима тогда и только тогда, когда выполним хотя бы один ее конъюнкт. Конъюнкт выполним тогда и только тогда в нем нет противоречия вида $x \wedge \neg x$. Пусть в ДНФ n конъюнктов, в каждом конъюнкте не более m литералов.

Таким образом, за $O(m^2)$ мы можем проверить попарно все литералы одного конъюнкта на непротиворечивость. За $O(nm^2)$ мы можем сделать это для всех конъюнктов. Длина входа есть $\Theta(nm)$, поэтому приведенных алгоритм работает за квадратичное время.

• Преобразование КНФ в ДНФ нельзя (если $\mathcal{P} \neq \mathcal{NP}$) произвести за полиномиальное время. Известный нам алгоритм построения полной ДНФ заключается в переборе всех значений булевой функции, что требует экспоненциального числа операций.

Задача 3.5

Расставьте и обоснуйте $\mathcal{P}$, $\mathcal{NP} - complete$, $co - \mathcal{NP} - complete$:

	Выполнимость	Тавтологичность
КНФ	$\mathcal{NP} - complete$	$\mathcal{P}$
ДНФ	$\mathcal{P}$	$co - \mathcal{NP} - complete$

Под выполнимостью понимается задача проверки наличия набора значений переменных, на котором формула равна 1. Под тавтологичностью понимается задача проверки свойства формулы принимать значение 1 на всех наборах.

Решение:

• $CNF - SAT \in \mathcal{NP} - complete$ (выполнимость КНФ)
Это уже известный нам факт.

• $DNF - SAT \in \mathcal{P}$
Доказано в задаче 3.

• $CNF - TAUT \in \mathcal{P}$

Покажем, что $CNF - TAUT \in co - \mathcal{P} = \mathcal{P}$. Пусть КНФ $\phi \notin CNF - TAUT$. По законам Де-Моргана, формула $\neg\phi$ является ДНФ. Если ϕ нетавтологична, то $\neg\phi$ выполнима, то есть $\neg\phi \in DNF - SAT \in \mathcal{P}$.

Более формально, мы построили полиномиальную сводимость $\overline{CNF - TAUT} \leq_p DNF - SAT$, заключающуюся в замене все символов $\vee$ на $\wedge$ и наоборот и добавлении $\neg$ ко всем литералам.

- $DNF - TAUT \in co - \mathcal{NP} - complete$

Лемма.
$$L \in \mathcal{NP} - complete \qquad \Longleftrightarrow \qquad \overline{L} \in co - \mathcal{NP} - complete$$

Доказательство:
Пусть $L \in \mathcal{NP} - complete$. Пусть $\overline{M} \in co - \mathcal{NP}$ — произвольный язык. Тогда $M \in \mathcal{NP}$ и $M \leq_p L$ с помощью функции f.

$$\Big[x \in M \iff f(x) \in L\Big] \qquad \Longleftrightarrow \qquad \Big[x \in \overline{M} \iff f(x) \in \overline{L}\Big]$$

Таким образом, та же функция f осуществляет сводимость $\overline{M} \leq_p \overline{L}$. В силу произвольности $\overline{M}$ заключаем, что $\overline{L} \in co - \mathcal{NP} - complete$.

В обратную сторону абсолютно аналогично. □

Покажем, что $\overline{DNF - TAUT} \in \mathcal{NP} - complete$.

ДНФ нетавтологична тогда и только тогда существует набор, на котором она равна 0. Тогда этот набор можно взять в качестве сертификата, поэтому $\overline{DNF - TAUT} \in \mathcal{NP}$.

Сведем $CNF - SAT \leq_p DNF - TAUT$. Аналогично предыдущему пункту,

$$\phi \in CNF - SAT \iff \neg\phi \in \overline{DNF - TAUT}$$

Это и есть нужная полиномиальная сводимость.

Задача 4

Найдите Θ-асимптотику суммы $\displaystyle\sum_{k=1}^{n} \sqrt{k}$, оценив её с помощью интеграла $\displaystyle\int_1^n \sqrt{x}dx$ сверху и снизу. Выведите аналогичную формулу для асимптотики $\displaystyle\sum_{k=1}^{n} k^\alpha$ для $\alpha > 0$.

Решение:

Получим сразу оценку в общем случае.

Заметим, что
$$\sum_{k=1}^{n} k^\alpha = \int_0^n \lceil x \rceil^\alpha dx$$

Рассмотрим функцию $f(x) = x^\alpha,\ \alpha > 0$. При $x \geq 0:$

$$x \leq \lceil x \rceil \qquad \Longrightarrow \qquad x^\alpha \leq \lceil x \rceil^\alpha \qquad \Longrightarrow \qquad \int_0^n x^\alpha dx \leq \int_0^n \lceil x \rceil^\alpha dx$$

$$\frac{1}{\alpha+1} x^{\alpha+1}\Big|_0^n = \frac{n^{\alpha+1}}{\alpha+1} \leq \sum_{k=1}^{n} k^\alpha$$

Теперь пусть $g(x) = (1+x)^\alpha$. Аналогично, при $x \geq 0:$

$$1 + x \geq \lceil x \rceil \qquad \Longrightarrow \qquad (1+x)^\alpha \geq \lceil x \rceil^\alpha \qquad \Longrightarrow \qquad \int_0^n (1+x)^\alpha dx \geq \int_0^n \lceil x \rceil^\alpha dx$$

$$\frac{1}{\alpha+1}(1+x)^{\alpha+1}\Big|_0^n = \frac{(1+n)^{\alpha+1} - 1}{\alpha+1} \geq \sum_{k=1}^{n} k^\alpha$$

Итак, имеем

$$\Theta(n^{\alpha+1}) = \frac{n^{\alpha+1}}{\alpha+1} \le \sum_{k=1}^{n} k^\alpha \le \frac{(1+n)^{\alpha+1}-1}{\alpha+1} = \Theta(n^{\alpha+1})$$

$$\sum_{k=1}^{n} k^\alpha = \Theta(n^{\alpha+1})$$

В частности, при $\alpha = \frac{1}{2}$:

$$\sum_{k=1}^{n} \sqrt{k} = \Theta(n^{\frac{3}{2}}).$$

Задача 5

Останется ли $3 - SAT$ полной, если ограничиться формулами, в которых каждая переменная входит не более 3 раз, а каждый литерал — не более 2 раз?

а) Под $3 - SAT$ понимается НЕ-БОЛЕЕ-$3 - SAT$.

б) **(Бонусная задача)** Покажите, что если имеется в виду РОВНО-$3-SAT$, то не бывает невыполнимых формул указанного вида.

Решение:

(а) Да, останется.

Обозначим такой язык за L. Построим полиномиальную сводимость $3 - SAT \le_p L$. Считаем, что $3 - SAT$ состоит из выполнимых РОВНО-3-КНФ таких, что в каждом дизъюнкте все переменные различны.

Пусть $\phi \in 3 - SAT$:

$$\phi = D_1 \wedge D_2 \wedge \ldots \wedge D_m, \qquad D_i = a_{i1} \vee a_{i2} \vee a_{i3}, \qquad a_{ij} - \text{ литералы}$$

Алгоритм f преобразования РОВНО-3-КНФ ϕ в 3-КНФ ψ:

1. Введем $3m$ новых переменных $y_{ij},\ i = 1, \ldots, m,\ j = 1, 2, 3$:

$$y_{ij} = a_{ij}, \qquad \text{т.е. } \ y_{ij} = x_k \ \text{ или } \ y_{ij} = \neg x_k$$

2. Построим КНФ ψ, состоящую из $2 \cdot 3m$ 2-дизъюнктов и m 3-дизъюнктов:

   - Для каждой из $3m$ новых переменных добавим 2-дизъюнкты:

   $$(\neg y_{ij} \vee a_{ij})(y_{ij} \vee \neg a_{ij}) = (y_{ij} \equiv a_{ij})$$

   - Для каждого 3-дизъюнкта $(a_{i1} \vee a_{i2} \vee a_{i3})$ в ϕ добавляем 3-дизъюнкт

   $$(y_{i1} \vee y_{i2} \vee y_{i3})$$

   Теперь все переменные y_{ij} встречаются в ψ не более трех раз, а каждый литерал с y_{ij} — не более двух раз. Однако переменные x_k до сих пор могут встречаться более трех раз. Эти переменные играют роль связей между y_{ij}. На данный момент имеем формулу, эквивалентную исходной по выполнимости.

   - Для каждой переменной x_k исходной КНФ рассмотрим эквиваленции, содержащую эту переменную. Другими словами, представим текущую формулу ψ в виде:

   $$\psi = (y_1 \equiv x_k)(y_2 \equiv x_k) \ldots (y_s \equiv x_k)(y_{s+1} \equiv \neg x_k) \ldots (y_t \equiv \neg x_k) \ldots$$

   По выполнимости это эквивалентно:

   $$\psi = (y_1 \equiv y_2) \ldots (y_{s-1} \equiv y_s)(y_s \equiv \neg y_{s+1})(y_{s+1} \equiv y_{s+2}) \ldots (y_{t-1} \equiv y_t) \ldots$$

   Если все эквиваленции развернуть в 2-дизъюнкты, то получится, что некоторые переменные y_i встречаются 5 раз: по 2 раза в каждой из двух эквиваленций и в 1 раз в 3-дизъюнкте.

- Заменим все эти эквиваленции на следующие выражения:

  $$\psi = (y_1 \vee \neg y_2)(y_2 \vee \neg y_3)\dots(y_{s-1} \vee \neg y_s)\dots(y_s \vee y_{s+1})(\neg y_{s+1} \vee y_{s+2})\dots(\neg y_{t-1} \vee y_t)(\neg y_t \equiv \neg y_1)\dots$$

  Здесь каждый литерал встречается ровно один раз, а каждая переменная — ровно два раза. Третье вхождение — в исходном 3-дизъюнкте.

Видно, что преобразование заключается в переборе всех дизъюнктов и переменных, поэтому оно требует полиномиального времени. Похожем корректность последнего шага второго пункта.

Достаточно показать, что

$$(z_1 \equiv z_2)(z_2 \equiv z_3)\dots(z_{p-1} \equiv z_p) = (z_1 \vee \neg z_2)(z_2 \vee \neg z_3)\dots(z_{p-1} \vee \neg z_p)(z_p \vee \neg z_1)$$

Из этого дописыванием отрицаний в нужных местах будет следовать корректность перехода. Обозначим за ψ левую часть равенства выше, а за ψ' — правую часть.

Пусть $\psi = 1$ при некотором наборе. Тогда из структуры ψ видно, что все переменные принимают одно и то же значение. Тогда каждый дизъюнкт ψ' обратится в 1, и $\psi' = 1$.

Пусть $\psi = 0$. Тогда существуют две переменные z_k и z_{k+1}, принимающие разные значения. Пусть $z_k = \neg z_{k+1} = 0$. Тогда дизъюнкт в ψ' : $(z_k \vee \neg z_{k+1}) = 0$ и $\psi' = 0$. Пусть $z_k = \neg z_{k+1} = 1$. Рассмотрим значения z_l и z_{l+1}, $l = 1, \dots, k-1$. Допустим, мы нашли такое l, что $z_l = \neg z_{l+1} = 0$. Тогда реализуется первый случай. Пусть мы не нашли такое l. Тогда

$$z_1 = z_2 = \dots = z_k = \neg z_{k+1} = \dots = \neg z_p = 1$$

Но тогда в ψ' дизъюнкт $(z_p \vee \neg z_1)$ обратится в 0, и $\psi' = 0$.

Задача 6

Постройте сводимость по Карпу языка (G, k) графов, в которых есть k-клика к языку графов, в которых есть клика хотя бы на половине вершин.

Решение:

В этой задаче *максимальным подграфом* будем называть такой подграф графа G, что в нем есть все возможные ребра графа G. Также будем считать, что граф G состоит из n вершин и m ребер. Язык из графов, в которых есть клика на половине вершин назовем L.

Алгоритм f преобразования пары (G, k) в граф G':

1. $G' = G$.
2. Добавить в G' $(n - 2k)$ новых вершин.
3. Соединить все новые вершины со всеми вершинами графа G' (в том числе и между собой).

Пусть $(G, k) \in CLIQUE$. Граф G' имеет $2n - 2k$ вершин. Рассмотрим k вершин, на которых в G была клика, и $n - 2k$ новых вершин. Новые вершины соединены со всеми, а старые k вершин образуют клику. Поэтому вместе эти $n - k$ вершин образуют клику на половине вершин. Значит, $G' \in L$.

Пусть $G' \in L$. Тогда в нем есть клика на хотя бы $n - k$ вершинах, а значит, и клика на ровно $n - k$ вершинах. Рассмотрим ее. Заметим, что в алгоритме добавляются только такие ребра, что хотя бы один их конец является новой вершиной. Новых вершин всего $n - 2k$, поэтому среди выбранных вершин найдутся k старых.

Любой максимальный подграф клики — тоже клика, поэтому данные k старых вершин образуют клику. Ребра между ними в алгоритме не добавлялись, поэтому эта k-клика была в исходном графе. Значит, $(G, k) \in CLIQUE$.

Задача 7

а) Верно ли, что существует такая функция $f : \mathbb{N} \to \mathbb{N}$, для любых констант $\forall\, c, d > 0$ выполнено

$$f(n) = \omega(n^c),\ f(n) = o(2^{nd}),$$

т. е. функция $f(n)$ растет быстрее любого заданного полинома, но медленнее любой заданной экспоненты?

б) Некто анонсировал теорему (т. е. утверждение может быть и неверно), что любой МТ требуется $\Omega(n \log_2^{\log_2 n} n)$ тактов для того, чтобы проверять тавтологичность формул, заданных в формате 4-ДНФ, т. е. дизъюнктивных нормальных форм, в каждый конъюнкт которых входит не более четырех переменных (здесь n — длина входа). Считаем, что теорема верна. Верно ли, что из этого вытекает, что $\mathcal{P}$ не совпадает с $co-\mathcal{NP}$?

Решение:

(а) Да, верно.

Пусть $f(n) = n^{\log n}$. Пусть $g(n) = n^c,\ h(n) = 2^{nd},\ c > 0,\ d > 0$.

$$f(n) = w(g(n)) \qquad \Longleftrightarrow \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

$$\frac{f(n)}{g(n)} = \frac{n^{\log n}}{n^c} = n^{\log n - c} \longrightarrow \infty, \qquad n \to \infty$$

$$f(n) = o(h(n)) \qquad \Longleftrightarrow \qquad \lim_{n\to\infty} \frac{f(n)}{h(n)} = 0$$

$$\frac{f(n)}{h(n)} = \frac{n^{\log n}}{2^{nd}} = \frac{2^{(\log n)^2}}{2^{nd}} = 2^{(\log n)^2 - dn} \longrightarrow 0, \qquad n \to \infty$$

(б) Да, верно.

В задаче 3.5 было доказано, что $DNF - TAUT \in co - \mathcal{NP} - complete$. Аналогично доказывается, что $4 - DNF - TAUT \in co - \mathcal{NP} - complete$.

Допустим, $\mathcal{P} = co - \mathcal{NP}$. Тогда $4 - DNF - TAUT \in co - \mathcal{NP} = \mathcal{P}$. Это значит, что существует МТ, которая решает эту задачу за $O(n^k)$ тактов. То есть

$$\exists C_1 > 0,\ \exists n_1\ :\ \forall n > n_1\ :\ T(n) \le C_1 n^k$$

Такая оценка работы выполняется для любого входа длины n. Теорема, однако, утверждает, что

$$\exists C_2 > 0,\ \exists n_2\ :\ \forall n > n_2\ :\ T(n) \ge C_2 n(\log n)^{\log n}$$

То есть для любого достаточно большого n существует такой вход длины n, что МТ работает за указанное время (то есть это оценка в худшем случае). Из этих двух утверждений имеем:

$$\exists C > 0,\ \exists n_0\ :\ \forall n > n_0\ :\ n(\log n)^{\log n} \le Cn^k \qquad \Longleftrightarrow \qquad n(\log n)^{\log n} = O(n^k)$$

Тогда при любых достаточно больших n:

$$(\log n)^{\log n} \le Cn^{k-1} \implies \log n \cdot \log\log n \le \log C + (k-1)\log n \le 2k \log n \implies \log\log n \le 2k$$

Но последнее неравенство неверно для любых n, так как функция $\log\log n$ неограничена. Противоречие.