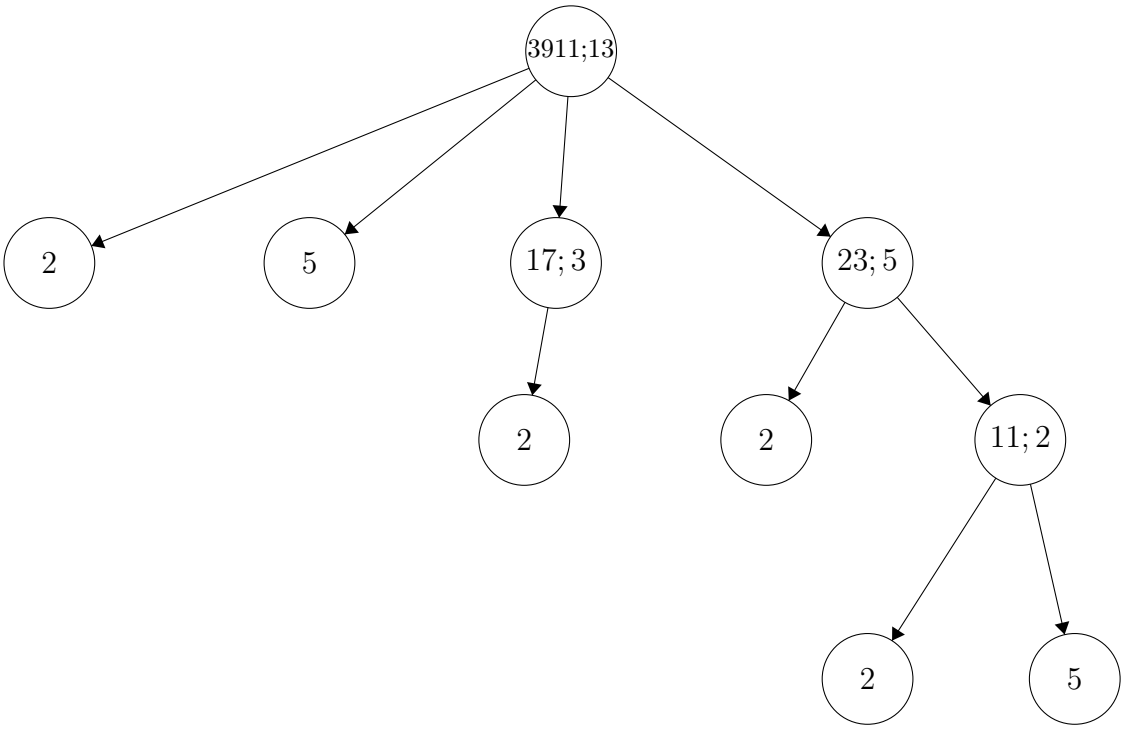
Задача 0

Пары указаны в порядке $(p; g)$



Задача 1

(i) Используя описание языка из условия (обозначим его L), запишем определение $\sum_2$:
$\varphi \in L \Leftrightarrow \ \exists\, \bar{x}\ \forall\, \bar{y} \hookrightarrow R(\varphi, \bar{x}, \bar{y}) = \varphi(\bar{x}, \bar{y}) = 1$
Это действительно $\sum_2$, так как посимвольно $|\varphi(\bar{x}, \bar{y})| > 2n$, а $|\bar{x}| = |\bar{y}| = n$, то есть $|\bar{x}|, |\bar{y}| = poly(|\varphi|)$. Подставив $\bar{x}$ в исходную формулу и выполнив её упрощения, которые полиномиальны от её длины, так как укорачивают её длину, мы получим 1 или что-то иное, значит, $R \in P$.

(ii) Пример задачи из $\sum_3$ – язык графов:

$\exists$ вершинное покрытие графа $u : \forall\, u'-$отличного от u вершинного покрытия

$\exists\, p-$простое число: $\hookrightarrow (|u| = m) \wedge (|u'| < m \Rightarrow u'-$не вершинное покрытие$) \wedge$

$$\wedge\ (m \vdots p^3)$$

Проверка каждого условия в предикате полиномиальна, поэтому сам он полиномиален. Каждая из величин $|u|, |u'|, p$ меньше по длине, чем описание графа, то есть каждая полиномиальна относительно длины графа.

(iii) Переделаем определение $\sum_k$ под $\sum_{k+1}$ и $\prod_{k+1}$, добавив новую переменную с квантором:
$\sum_k = \{L | x \in L \Leftrightarrow\ \exists\ y_1 \forall\ y_2 \cdots (\exists\ or\ \forall) y_k \hookrightarrow R(y_1, \cdots, y_k) = 1\}$

$\sum_{k+1} = \{L | x \in L \Leftrightarrow\ \exists\, y_1 \forall\, y_2 \cdots (\exists\, or\, \forall) y_k (\forall\, or\, \exists) y_{k+1} \hookrightarrow P(y_1, \cdots, y_k, y_{k+1}) =$
$= R(y_1, \cdots, y_k) = 1\}$

$\prod_{k+1} = \{L | x \in L \Leftrightarrow\ \ \forall y_0\ \exists\ y_1 \forall\ y_2 \cdots (\exists\ or\ \forall) y_k \hookrightarrow P(y_0, y_1, \cdots, y_k) =$
$= R(y_1, \cdots, y_k) = 1\}$

Значит, $\sum_k \subset \sum_{k+1} \cap \prod_{k+1}$

(iv) $NP \subset PSPACE$
Рассмотрим язык L лежащий в NP. Тогда для него есть полином $poly()$ и верификатор, принимающий на вход изучаемое слово x и сертификат длины $poly(|x|)$. Алгоритм, разрешающий L составим из итераций верификатора:

Пусть нам подано на вход слово $x, |x| = n$. Вместо сертификата будем подавать на вход верификатору всевозможные слова, составленные из символов алфавита, длины не большей $poly(n)$. В силу конечности алфавита, в котором задан язык, и конечности величины $poly(n)$, всего будет подано конечное число пар (слово; предполагаемый сертификат), по одному за итерацию, на вход верификатору, который работает $poly(n)$ на каждой паре, значит, всего — конечное время. Из полиномиальности работы верификатора по времени следует также, что будет задействовано еще (кроме занятых входом) не более $poly(n)$ ячеек памяти. На каждой итерации опустошаем данные после предыдущего шага и запускаем верификатор со следующим возможным сертификатом (их можно подавать в алфавитном порядке). Если на какой-то итерации верификатор вернул «да», то останавливаемся и возвращаем «да», иначе после проверки всех верификаторов вернем «нет». Во время работы данного алгоритма количество занятой памяти всегда не более полинома от длины исследуемого слова.

$PSPACE \subset EXPTIME$

Всякий алгоритм реализуется на deterministic-TM $= (A, Q, Q_f, \delta, q_0, \Lambda)$ (учебник Журавлева, Флерова, Вялого, стр. 175), процесс работы которой описывается последовательностью конфигураций (пар из слова на ленте в данный момент и состояния машины) с использованием конечного числа состояний и конечного числа символов алфавита. Из того, что каждая конфигурация встречается не более раза, а их общее число при входе x равно $|Q||A|^{poly(|x|)}$ в силу полиномиальной ограниченности памяти, получаем, что временная асимптотика работы экспоненциальна от длины входа.

Задача 2

1) Количество перестановок из условия находим по формуле включений-исключений (у нас есть k свойств $\{a_p\}$, $p = 1 \cdots k$, p-тое свойство — элемент i_p стоит на позиции j_p):

$$N(\bar{a}_1, \cdots, \bar{a}_k) = \sum_{i=0}^{k} (-1)^i C_k^i (n-i)! = \sum_{i=0}^{k} (-1)^i \frac{k!}{i!(k-i)!}(n-i)! \qquad (*)$$

Нахождение этого N сводится к вычислению $4k$ перманентов квадратных матриц, составленных из единиц, размер которых соответствует числам под знаком факториала во всей развернутой сумме $(*)$, т. к. перманент матрицы такого вида размера $q \times q$ равно $(q!)$.

2) Если же решать, вычисляя только один перманент, то вычислим перманент матрицы, элементы которой равны 1, кроме элементов с индексами $(a, b) : a = i_n \in \{i_1, i_2, \cdots, i_k\}, b = j_n \in \{j_1, j_2, \cdots, j_k\}$, которые приравняем к 0. Тогда каждой перестановке в перманенте будет соответствовать единица, кроме тех, где индексы какого -то элемента в произведении равны (i_n, j_n).

Задача 3

ДНФ выпонима $\Leftrightarrow$ выполним хотя бы один из C_i. Пройдемся один раз по поданной на вход ДНФ. При этом для каждого C_i проверим наличие пары литералов вида x и $\bar{x}$ (по мере прочтения будем отмечать для

встречаемых переменных наличие в данном дизьюнкте их самих или их отрицания). Если такой нет в данном дизьюнкте, то ДНФ является выполнимой, иначе переходим к следующему дизьюнкту. Если мы прочли всю ДНФ, не вернув положительный ответ, то после протения возвращаем отрицательный ответ: ДНФ невыполнима.

Пробел: преобразование КНФ к эквивалентной ДНФ должно быть полиномиальным от длины КНФ, но не показано, есть ли такое.

Задача $3\frac{1}{2}$

	Выполнимость	Тавтологичность
КНФ	NPc	P
ДНФ	P	*co-*NPc

1,1) Выполнимые КНФ — с семинара.
2,1) Выполнимые ДНФ — из номера 3.
1,2) Тавтологичность отрицания формулы равносильна невыполнимости самой формулы. По КНФ строим отрицание за линейное время по законам де Моргана, получаем ДНФ. Узнав за полиномиальное время, выполнима ли эта ДНФ, узнаем, тавтологична ли исходная КНФ.
2,2) Т. к. дополнение языка тавтологичных ДНФ есть обьединение множества слов, не являющихся булевыми формулами, распознаемых точно за полином, и множества ДНФ, отрицания которых (т. е. КНФ) выполнимы. Поскольку отрицание ДНФ выполнимо за линейное время (всякий знак $\vee$ меняем на $\wedge$ и наоборот, всякому литералу без отрицания добаляем отрицание и наоборот), дополнение языка тавтологичных ДНФ сводится за полином к NPc языку выполнимых КНФ. Получаем, что дополнение языка тавтологичных ДНФ есть NPc, тогда сам язык *co-*NP

Задача 4

$\sum_{k=1}^{n} \sqrt{k} = 1 \cdot \sqrt{1} + 1 \cdot \sqrt{2} + 1 \cdot \sqrt{3} + \cdots + 1 \cdot \sqrt{n}$

Нарисуем данную ступенчатую фигуру и впишем в нее снизу функцию $\sqrt{x}$ и сверху опишем ее функцией $\sqrt{x+1}$. Тогда

$$\int_0^n \sqrt{x}dx \leq \sum_{k=1}^{n} \sqrt{k} \leq \int_{-1}^{n} \sqrt{x+1}dx \Leftrightarrow$$

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

Отсюда получаем асимптотику $\Theta(n\sqrt{n})$

По аналогичным шагам получаем

$$\int_0^n x^\alpha dx \leq \sum_{k=1}^{n} k^\alpha \leq \int_0^n (x^\alpha + 1)dx \Leftrightarrow$$

$$\frac{1}{\alpha+1}n^{\alpha+1} \leq \sum_{k=1}^{n} k^\alpha \leq \frac{1}{\alpha+1}n^{\alpha+1} + n \leq \frac{2}{\alpha+1}n^{\alpha+1}$$

при больших n. Отсюда асимптотика $\Theta(n^{\alpha+1})$.

Задача 5

а) Обозначим язык из условия 3-SAT^*. Покажем 3-SAT $\leq_p$ 3-SAT^* $\Leftrightarrow \exists\ f_p(x) : \forall x \hookrightarrow x \in$ 3-$SAT \Leftrightarrow f_p(x) \in$ 3-SAT^*

Алгоритм, вычисляющий f_p, будет преобразовывать поданную на вход КНФ $x \in$ 3-SAT к тождественной формуле, описанной в условии (т. е. возвращается формула, где каждая переменная входит не более трех раз, каждый литерал — не более 2). Для каждой переменной будем хранить число вхождений в формулу. Проходясь по исходной формуле, каждое третье вхождение для любой переменной будем обрабатывать следующим образом: вместо дизьюнкта, содержащего третье вхождение данной переменной q_1, запишем тот же дизьюнкт, но заменив данную переменную q_1 на новую a_2, не содержащуюся в исходной КНФ (выдуманную), и добавим «связочный» дизьюнкт $a_2 \equiv q_1$ или $a_2 \equiv \bar{q}_1$, в зависимости от того, равны ли два других литерала с данной переменной. Далее все вхождения переменной q_1 считаем в исходной формуле вхождениями a_2 (это можно отмечать в матрице с числом входов переменных) и при дальнейшем чтении входной КНФ повторяем данную операцию для каждого третьего повтора какой-либо переменной. В итоге, прочитывая входную

КНФ из n переменных, мы максимум $\frac{n}{3}$ раз повторим одно и то же константное действие обновления переменной и получим формулу, тождественную исходной, число переменных в которой больше максимум в два раза в сравнении с исходным. Алгоритм полиномиален по времени и использованной памяти. Формулы $x \equiv f_p(x)$, поэтому верна сводимость, т. к. каждый из входов удовлетворяет своему языку и принадлежность входов соответствующему языку равносильна.

б) Пусть есть невыполнимая КНФ из 3-SAT^* — языка РОВНО-3-SAT. По теоремам 1.43 и 1.45 из курса ТФС (книга Журавлева Ю. И. «Дискретный анализ. Формальные системы и алгоритмы.», стр. 64-65) имеем:

$(*)$ КНФ невыполнима $\Leftrightarrow$ существует резолютивный вывод пустого дизъюнкта из дизъюнктов КНФ

Значит, методом резолюций должны получить противоречие. Для этого необходимо методом резолюций получить дизъюнкт из одной переменной и дизъюнкт из отрицания этой переменной. Рассмотрим варианты получения одного из таких дизъюнктов (назовем эту переменную c) с конца к началу, так как получение другого аналогично. Чтобы получить дизъюнкт c, нужно чтобы до этого были получены дизъюнкты $b \vee c$ и $\bar{b} \vee c$. В силу симметричности рассмотрим получение одного из них. Дизъюнкт $b \vee c$ можно получить из пар дизъюнктов вида: (1) $a \vee b \vee c$ и $\bar{a} \vee b \vee c$; (2) $a \vee b \vee c$ и $\bar{a} \vee b$ и аналогичные (то есть длины 3 и 2); (3) $a \vee b$ и $\bar{a} \vee b$ (длины два каждый); $a \vee b \vee c$ и $\bar{a}$ (то есть длины три и один, но этот вариант не рассматриваем, т.к. тут необходимо иметь дизъюнкт длины один, а значит, для его получения необходимо повторить все те же шаги). В случае (1) получаем, что переменная c должна присутствовать хотя бы в двух разных исходных дизъюнктах («исходных», то есть тех, что образуют КНФ). В случае (2), повторив для дизъюнкта $\bar{a} \vee b$ аналогичные излагаемые рассуждения, получаем, что необходимо хотя бы два исходных дизъюнкта с переменной c для его получения, значит, всего хотя бы три исходных дизъюнкта с c для получения $b \vee c$. В случае (3) аналогичным образом получаем, что нужно хотя бы четыре исходных дизъюнкта с переменной c. Т.е. для получения методом резолюций дизъюнкта из одного литерала нужно минимум два дизъюнкта из КНФ, содержащих эту переменную, тогда для получения противоречия нужно четыре таких дизъюнкта, но этого не может быть в нашем языке 3-SAT^*. Тогда по теореме $(*)$, получаем что все формулы выполнимы или что не

бывает невыполнимых формул в 3-SAT^*.

Задача 6

Пусть дано слово x — описание графа на m вершинах. При нечетных m будем считать $\left\lfloor \frac{m}{2} \right\rfloor$ половиной вершин.
1) Если $k \geq \left\lfloor \frac{m}{2} \right\rfloor$, то $f_p(x) = x$. Иначе алгоритм, вычисляющий описание нового графа $f_p(x)$ на n вершинах, добавит к исходному набору вершин $n - m$ новых вершин и соединит каждую новую вершину с каждой вершиной нового графа, затем проведет все ребра из старого графа меж старыми вершинами. Найдем сколько вершин нужно добавить:

$$\left\lfloor \frac{n}{2} \right\rfloor \leq k + n - m \Leftrightarrow \left\lfloor \frac{n}{2} \right\rfloor \geq m - k \Leftrightarrow n \geq 2(m - k)$$

Будем считать $n = 2(m-k+1)$. Значит, добавить надо $n-m = m-2k+2$ новых вершин. Заметим, что при этом построении, в исходном графе есть клика размера $k \Leftrightarrow$ в новом графе $f_p(x)$ есть клика хотя бы на половине вершин.

Задача 7

а) $f(n) = n^{\log n}$
(1) $f(n) = \omega(n^c) \Leftrightarrow n^c = o(f(n)).$ $\lim_{n\to\infty} \frac{n^c}{n^{\log n}} = 0$

(2) $f(n) = o(2^{nd}) \Leftrightarrow \lim_{n\to\infty} \left| \frac{f(n)}{2^{nd}} \right| = 0$

$\lim_{n\to\infty} \frac{n^{\log n}}{2^{nd}} = \lim_{n\to\infty} \exp(\ln n \log n - nd \ln 2) = 0$, т. к. любая степень логарифма растет медленнее полинома любой степени, большей 1.

б) Так как отрицание 4-ДНФ есть 4-КНФ и за один проход по ДНФ, заменяя каждый знак $\vee$ на $\wedge$ и наоборот, а также накладывая отрицание на каждый литерал (двойное отрицание не пишем), получаем отрицание ДНФ, то из того, что не существует алгоритма, разрешающего язык тавтологий вида 4-ДНФ быстрее, чем за $const \cdot n \log_2^{\log_2 n} n$, получаем, что нет алгоритма разрешающего язык L невыполнимых 4-КНФ быстрее, чем за $const \cdot n \log_2^{\log_2 n} n$. Если бы для второго языка был более быстрый алгоритм, то, добавив к нему линейный алгоритм отрица-

ния формул, получили бы более быстрый алгоритм для первого языка. $\bar{L} = \{(\text{не 4-КНФ}) \cup 4\text{-}CNF\text{-}SAT\}$. $\bar{L} \in NP$, так как формулы, не являющиеся 4-КНФ, можно отсеять за один проход по ним. Тогда $L \in co\text{-}NP$. Покажем, что $L \notin P \Leftrightarrow \forall C > 0 \hookrightarrow n \log_2^{\log_2 n} n = \Omega(n^c)$. Логарифмируем последнее равенство: $\exists\, C > 0,\ \exists n_0 > 0 : \forall n > n_0 \hookrightarrow (n \log_2^{\log_2 n} n \geq Cn^c \Leftrightarrow \log n + \log n \cdot \log\log n \geq c \log C \cdot \log n \Leftrightarrow \log\log n \geq c \log C - 1)$, что верно. Поскольку $L \notin P \wedge L \in co\text{-}NP \Rightarrow P \neq co\text{-}NP$.