Задача 1

f_p — вычислимая за полиномиальное время функция f
(1) Рефлексивность $A \leq_p A \Leftarrow \exists f_p(x) = x$ (функция линейная) : $\forall x \hookrightarrow x \in A \Leftrightarrow f(x) \in A$

Транзитивность $A \leq_p B \overset{def}{\Leftrightarrow} \exists f_p(x) : \forall x \hookrightarrow x \in A \Leftrightarrow f_p(x) \in B$
$B \leq_p C \overset{def}{\Leftrightarrow} \exists g_p(x) : \forall x \hookrightarrow x \in B \Leftrightarrow g_p(x) \in C$

Тогда $\exists h_p(x) = g_p(f_p(x))$ (композиция двух полиномов — полином) : $\forall x \hookrightarrow x \in A \Leftrightarrow h_p(x) = g_p(f_p(x)) \in C \overset{def}{\Leftrightarrow} A \leq_p C$

(2) $A \leq_p B \overset{def}{\Leftrightarrow} \exists f_p(x) : \forall x \hookrightarrow x \in A \Leftrightarrow f_p(x) \in B \quad (*)$

Вычислим для $\forall x \in A$ за полиномиальное время $f_p(x)$. В силу $(*)$, ответ (да или нет) полиномиального (т. к. $B \in P$) алгоритма проверки принадлежности $f_p(x) \in B$, который запускаем далее, есть ответ для алгоритма проверки принадлежности $x \in A$. Т. к. композиция полиномов есть полином, то $A \in P$
(3) $B \in NP \overset{def}{\Leftrightarrow} \exists\ poly(|x|), \exists\ s : |s| \leq poly(|x|), \exists\ V(x,s) : V$ работает $\leq poly(|x|) \hookrightarrow V(x,s) = 1 \Leftrightarrow x \in B$
$A \leq_p B \overset{def}{\Leftrightarrow} \exists f_p(x) : \forall x \hookrightarrow x \in A \Leftrightarrow f_p(x) \in B$

Из этого получаем: $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow V(f(x), s) = 1 = F(x, s)$, $\exists\, poly_2(|x|) : poly(|x|) \leq poly_2(|x|), F$ работает $\leq poly_2(|x|)$ (т. к. новый верификатор F есть композиция полиномиальных V и f. Тогда верно $|s| \leq poly_2(|x|)$, возьмем тот же сертификат s и получим по определению $A \in NP$.

Задача 2

Длиной входа считается количество чисел в десятичной системе.
(1) Т. к. в двудольном графе не может быть ни одного треугольника, то пусть наш алгоритм возвращает ответ «нет» (описание графа не принадлежит указанному языку) для любых входных данных. Их нужно прочесть, поэтому данный алгоритм линейный.

(2) Пусть в графе n вершин. Запускаем DFS (в худшем случае n^2 итераций) из любой вершины. Если при этом будут посещены все вершины графа, то он связен, возвращаем «нет», иначе снова запускаем DFS из

любой непосещенной вершины. В течение всего этого, если встретится раннее посещенная вершина, значит есть цикл, возвращаем «нет», иначе в конце возвращаем «да». В итоге, задача решена за полиномиальное от длины входа (n^2 — количество чисел в матрице смежности).

(3) Алгоритм: в первой паре вложенных циклов пройдемся по ячейкам матрицы от (1,1) до (2018,2018) — это левая верхняя клетка искомой подматрицы, затем во второй паре вложенных циклов пройдемся по данной подматрице. В итоге, сделаем $2018^2(n-2018)^2$ итераций — полином от длины входа, равной n^2.

Задача 3

Пример неполиномиального роста при непосредственном применении метода Гаусса для матрицы системы:

$$\begin{pmatrix} \frac{1}{h} & 1 & \cdots & 1 & 1 & 1 \\ 1 & \frac{1}{h+1} & 1 & \cdots & 1 & 1 \\ 1 & 1 & \frac{1}{h+2} & \cdots & 1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & 1 & 1 & \frac{1}{h+m-1} \end{pmatrix}$$

(тут матрица $m \times n$)
Умножаем вторую строку на ведущий коэффициент из первой строки и вычитаем. Это повторяем для пар соселних строк дальше. Получаем на первом шаге знаменатель $h(h+1)$, на втором — $h^2(h+1)(h+2)$. Получаем на посленем шаге $h^{2^{(m-1)}}$. В битовой арифметике длина равна $2^{(m-1)} \log h$, что неполиномиально относительно $\log(m(n+1))$

Задача 4

(1) $L \in P \overset{def}{\Leftrightarrow}$ существует полиномиальный от длины входа алгоритм, распознающий L $(*)$. Пусть нам дано слово из L^* длины n. Будем отмечать позиции в нем, на которых могут кончаться слова из L, итерируясь по слову и проверяя на принадлежность L отрезок слова от каждой уже имеющейся отметки до текущей позиции итерации. Если n-я позиция отмечена после завершения программы, то возвращаем «да», иначе

— «нет». Отметок максимум n, значит, для L^* получили алгоритм, работающий за полином степени на два большей, чем для $(*)$.

(2) $L \in NP \overset{def}{\Leftrightarrow} \exists\ poly(|x|), \exists\ s : |s| \leq poly(|x|), \exists\ V(x,s) : V$ работает $\leq poly(|x|) \hookrightarrow V(x,s) = 1 \Leftrightarrow x \in L$
Для слова из L^* сертификатом будет его разбиение на слова из L (не превосходящее по длине само слово) и соответсвующее этому разбиению множество сертификатов для каждого куска. Кусков конечное число. Для каждого из них запускаем полиномиальный от длины куска верификатор V с соответсвующим сертификатом. Так получаем для всего слова новый верификатор, полиномиальный от длины слова.

Задача 5

Теорема Кронекера-Капелли: Система линейных алгебраических уравнений совместна тогда и только тогда, когда ранг матрицы системы равен рангу расширенной матрицы системы.
Предположим расширеннная система приведена к верхне-треугольному виду. Если ранги расширенной матрицы и матрицы системы не равны, т. е. в приведенной матрице системы есть нулевая строка, а в расширеннной матрице ей соответствует ненулевое число, то система несовместна. Нулевая строка в приведенной матрице системы есть линейная комбинация строк этой матрицы, т. к. приведение матрицы к верхнетреугольному виду есть последовательность элементарных преобразований строк матрицы.

Первой частью сертификата будет набор коэффициентов (с хотя бы одним ненулевым элементом) для строк расширенной матрицы такой, что линейная комбинация строк матрицы системы равна нулю, а линейная комбинация соответствующих элементов столбца свободных членов не равна нулю (если такой набор есть). Вторая часть сертификата будет содержать решение системы уравнений, если таковое имеется. Первая часть нужна для доказательства несовместности, а вторая — совместности (просто подставим решение из сертификата), т. е. для конкретной матрицы одна из частей сертификата мусор, а другая необходима, но проверку выполняем для обеих частей (понятно, что результаты этих двух проверок взаимоисключающие). Для матрицы $m \times n$ длина сертификата $m + n$ меньше длины входных данных (коэффициентов уравнений). Используя полиномиальный модифицированный алгоритм Ев-

клида, можем оценить порядок чисел в сертификате как полиномы от коэффициентов матрицы. Верификатор выполняет простые арифметические действия, поэтому его асимптотика удовлетворяет определению принадлежности языка классу NP.

Матрица $A = \{a_{ij}\}$, столбец свободных членов $B = \{b_i\}, i = 1 \cdots m, j = 1 \cdots n$, сертификат $y = \{y_1; y_2\} = \{q_1, q_2 \cdots q_m; x_1, x_2 \cdots x_n\}$

$$R(x, y) = (y_1 A = \bar{0})\ and\ (y_1 B! = 0)\ and\ not(A y_2! = B)$$

Задача 6

Язык лежит в NP: сертификат — это делитель d. Проверить делимость можно Алгоритмом Евклида за полином.
Язык лежит в $co - NP$: сертификат — простые делителей, сертификат их простоты, и их степени в разложении числа. Длина полиномиальна, т. к. если $N = \prod p_i^{n_i}$, то $\sum \log p_i + \sum n_i \leq 2 \sum n_i \log p_i \leq 2 \log N \leq 2 \log (N + M)$, а сертификаты простоты полиномиальной длины относительно самого простого числа. Т. е. в итоге длина сертификата полиномиальна.
Алгоритм будет сравнивать M с простыми делителями (все они больше M или есть хотя бы один меньше M), проверит, что их произведение равно N, и проверит их простоту с сертификатов. Каждое из действий работает за полиномиальное время.

Задача 7

$\text{ГП} \leq_p \text{ГЦ} \overset{def}{\Leftrightarrow} \exists\ f_p(x) : \forall x \hookrightarrow x \in \text{ГП} \Leftrightarrow f_p(x) \in \text{ГЦ}$
Пусть f_p добавляет в граф вершину и соединяет ее со всеми другими вершинами. Тогда, если в исходном графе x есть Г-путь, то в графе $f_p(x)$ есть Г-цикл. И обратно, если в графе $f_p(x)$ есть Г-цикл, то в исходном графе x есть Г-путь. Т. к. число новых ребер равно числу вершин в графе, то f_p действительно полиномиальна от входа.

$\text{ГЦ} \leq_p \text{ГП} \overset{def}{\Leftrightarrow} \exists\ g_p(x) = x : \forall x \hookrightarrow x \in \text{ГЦ} \Leftrightarrow g_p(x) \in \text{ГП}$
Одну вершину в гамильтоновом цикле сделаем множеством нескольких вершин, связанных меж собой и со всеми вершинами, с которыми связана выбранная вершина в исходном графе. Если был цикл, то путь точно

будет, например, путь с началом в одной вершине созданного множества и концом в другой. Если в графе есть путь, то сольем обратно вершины множества в одну. Чтобы при этом получался цикл, изначально кроме превращения вершины в множество нужно заставить путь начинаться и кончаться на вершинах данного множества, добавив, например, по одиночной вершине к двум любым вершинам нашего добавленного множества.

Задача 8

Построим ДКА, принимающий данный язык, по РВ. Для этого по алгоритму из книги ТРЯП Серебрякова построим НКА (1) (страницы 55-57), а по нему ДКА (2)(страница 58). Вход алгоритма состоит из РВ длиной q символов и слова длиной a, длина входа $d = a + q$. Шаг (1) алгоритма и число состояний n НКА линейны относительно q. На шаге (2) проверяем переходы из каждого состояния в подмножестве состояний НКА (вначале это ϵ-замыкание стартового состояния) по кадому символу алфавита, которых конечное число, и по ϵ в каждое состояние другого подмножества (в которые осуществляются переходы). Так просмотрим все пары элементов двух подмножеств. Всего операций порядка n^2a. Далее повторяем описанное для каждого нового подмножества и предыдущего. В худшем случае этих состояний будет столько же, т. е. n. Тогда шаг (2) работает в худшем случае за n^3. В итоге, за $O(d^3)$ построили ДКА из РВ. Далее он работает на слове линейно от его длины a. В результате, получен полиномиальный алгоритм. Если ДКА попадет в финальное состояние после прочтения слова, то алгоритм вернет «нет», иначе «да».

Задача 9

(1)Строим по РВ ДКА как было описано в задаче 8 для каждого языка (за $O(q^3)$, где q — максимальная длина РВ). Далее строим ДКА P для пересечения языков двух ДКА с помощью конструкции произведения: состояния в P есть пары состояний, по одному из исходных ДКА. Для таких состояний установим переход по символу, если такой переход есть для соответсвующих состояний в каждом ДКА. Повторяем это для каждого символа алфавита, которых константное число, а затем пересекаем полученный на текущий момент ДКА с следующим из 2019 данных.

Первое пересечение осуществимо за $O(q^2)$,т. к. число состояний линейно ДКА относительно длины РВ, второе за $O(q^3)$, в итоге получим ДКА перечения языков за $O(q^{2019})$, т. к. при каждом пересечении количество состояний в результирующем ДКА есть произведение количеств состояний пересекаемых ДКА.
Слово, лежащее в пересечении языков будет принято полученным ДКА, а значит есть путь от начального состояния к конечному, если такое слово существсвует. С помощью DFS из начального состояния, работающего в худшем случае за квадрат числа состояний, проверим наличие такого пути и дадим ответ. В итоге получен алгоритм полиномиальный от длины всех РВ.

(2)Если в предыдущей задаче заменить 2019 на n получаем асимптотику $O(nq^3 + q^n + q^{2n})$, где q есть максимальная длина РВ из входных данных. Такая асимптотика не полеиомиальна от длины входа. Задача разрешима, поскольку предоствален разрешающий алгоритм.