Домашняя работа 2 — Кузнецов

1 Задание 2

1. C14.а) Язык простых чисел лежит в coNP. Действительно, дополнение к этому языку — язык составных чисел (и 1), на лекции было показано, что он лежит в NP (доказательство того, что число составное, — разложение числа на мнжители $n = ab, a > 1, b > 1$.

   б) Теорема Люка-Лемера. Известно, что мультипликативная группа любого конечного поля циклическая (см. теорему 3.44 в книге Журавлёв, Флёров, Вялый "Дискретный анализ. Основы высшей алгебры", издание 2). Значит, если число p простое, то имеется g, такое, что $g, g^2, \ldots, g^{p-1}$ — все не равные 0 вычеты по модулю p. Тогда для любого простого q, делящего $p - 1$, имеет место

   $$g^{p-1} \equiv 1(\mod p), g^{(p-1)/q} \not\equiv 1(\mod p).$$

   Итак, если n простое, то условие выполнено. Покажем теперь, если выполнено приведённое условие, то n простое.

   Предположим противное, что n составное. Пусть $\phi(n)$ — функци Эйлера числа n, то есть количество чисел, меньших n и взаимно простых с n. Ясно, что если n не простое, то $\phi(n) < n - 1$. Отсюда следует, что существует простое число, которое в разложение на простые множители $\phi(n)$ входит в меньшей тепени, чем в $n - 1$. Пусть это число q. По условиию, существует a, такое, что
   $a^{n-1} \equiv 1(\mod n)$ и $a^{(n-1)/q} \not\equiv 1(\mod n)$.
   Ясно, что a взаимно просто с n, и потому $a^{\phi(n)} \equiv 1(\mod n)$. Значит, $a^{GCD(n-1,\phi(n))} \equiv 1(\mod n)$. Из условия на q следует, что $\frac{n-1}{q}$ делится на $GCD(n-1, \phi(n))$, и потому $a^{(n-1)/q} \equiv 1(\mod n)$. Противоречие. Всё доказано.

   в) Нам нужно предъвить свидетельство (когда я учился в ВУЗе, говорили "сертификат") простоты числа n полиномиального (от $\log_2 n$) размера. В качестве такого свидетельства мы предъявим

дерево. Корнем дерева будет число n. Потомками корня будут вершины со всеми простыми делителями q числа $n-1$ (по одной вершине на каждый делитель — будем называть эти простые делители q-метками вершины), и соответствующими числами $a < n$ (см. предыдущий пункт — то есть числа a будут такими, что
$a^{n-1} \equiv 1(\mod n), a^{(n-1)/q} \not\equiv 1(\mod n)$; эти числа a будем называть a-метками вершины). И для каждого из потомков числа n мы повторим ту же операцию, то есть для каждого простого числа q, делящего $n-1$, его потомками будут простые делители $q-1$ с соответствующими a-метками $a < q$. И т.д. Останавливаем этот процесс, когда доходим до вершин с метками 2 или 3, то есть для вершин с q-метками 2 и 3 мы не ищем потомков, и такие вершины становятся листьями нашего дерева (листьями мы называем вершины без потомков).

Также условимся, что для вершины с q-меткой $h = 2^m + 1, m > 1$ мы заводим двух потомков с одинаковыми q-метками 2 (это нужно, чтобы у каждой вершины, не являющейся листом, было не менее двух потомков).

Заметим (и это легко доказать индукцией), что произведение q-меток во всех листьях нашего дерева не превосходит n. Следовательно, колиество листьев не превосходит $\log_2 n$. Также индукцией по построению легко доказываем, что (на каждом этапе построени нашего дерева, а значит, и в конце) количество листьев не менее количества остальных вершин (здесь используется, что у каждой вершины, не являющейся листом, не менее двух потомков). Значит, всего вершин в нашем дереве не более $2\log_2 n$. Поскольку все q-метки и a-метки не превосходят n, то для кодирования нашей структуры требуется полиномиальное (от $\log_2 n$) количество битов.

А проверка, что наш сертификат правильный, тоже может быть осуществлена за полиномиальное от $\log_2 n$ время, поскольку она включает такие операции, как возведение неких чисел $a < n$ в степень не более n по модулю $m \leq n$, проверки, что число в вершине раскладывается в произведение неких степеней его потомков, и, вроде, всё. Всё это можно сделать за полиномиальное время.

Как ввозводить в степень по модулю m за полиномиальное время? Если возводим в степень k, надо последовательным возведением в квадрат (со взятием остатка по модулю) получить все степени вида $2^s < k$ и перемножить некоторые из этих степеней (умножение проводим по модулю m, иначе полиномиальное время не получится)

.

2. C15. $E = \cup_{c>0} DTime(2^{cn})$ не замкнут относительно сведений по Карпу. Построим эффективно вычислимую нумерацию машин Тьюринга $T_1, T_2, \ldots$, в которой каждая машина Тьюринга встретится бесконечное число раз. Здесь "эффективно вычислимая" нумерация означает, что по числу n мы можем восстановить T_n быстро, за полиномиальное время. Например, можно закодировать команды машины Тьюринга двоичным числом n и условиться, что любое число нулей в конце ничего не означает (например, кодировать все символы двоичными числами, заканчивающимися на 1). Тогда у нас каждое число n будет либо ничего не означать, либо кодировать какую-то машину Тьюринга, которую мы можем восстановить по нему за полиномиальное время. Из-за возможности откидывания нулей в конце каждая машина Тьюринга будет в такой нумерации встречатья бесконечное число раз. Теперь построим алгоритм, который будет работать следующим образом.

   Алгоритм 1.1. *Любой вход, который состоит не только из единиц, отвергаем. На входе из одних единиц* 1^k *действуем так: если* k *не кодирует никакую машину Тьюринга, то вход отвергаем (выводим 0). Если* k *кодирует некоторую машину Тьюринга* T_k*, то запускаем эту* T_k *на* 1^{k^2} *и делаем ею* 2^{k^2} *шагов. Если она за это время не остановится, то отвергаем вход (выводим 0), а если остановится — то выводим ответ, противоположный ответу машины* T_k *на входе* 1^{k^2}*.*

   Покажем, что принимаемый описанным алгоритмом язык H сводится к языку из E. Действительно, рассмотрим язык, решаемый таким алгоритмом:

   Алгоритм 1.2. *Любой вход, который состоит не только из единиц или из числа единиц, не являющегося полным квадратом, отвергаем. На входе* 1^{k^2} *действуем так: если* k *не кодирует никакую машину Тьюринга, то вход отвергаем (выводим 0). Если* k *кодирует некоторую машину Тьюринга* T_k*, то запускаем эту* T_k *на нашем входе* 1^{k^2} *и делаем ею* 2^{k^2} *шагов. Если она за это время не остановится, то отвергаем вход (выводим 0), а если остановится — то выводим ответ, противоположный ответу машины* T_k *на входе* 1^{k^2}*. Принимаемый этим алгоритмом язык обозначим* L*.*

   Пусть $f(x) = x^{|x|}$, то есть $f(x)$ — это слово x, повторённое столько раз, какова длина слова x. Тогда для любого x $x \in H$ тогда и только тогда, когда $f(x) \in L$. Ясно также, что $L \in E$. А вот сам язык H в E не лежит. Действительно, если H распознаётся некоторым алгоритмом T_k, лежащим в E и работающим время $O(2^{ck})$ на входах длины k, то при больших k этот алгоритм работает менее чем 2^{k^2} шагов, и по построению, ответ на таких входах противоположен ответу алгоритма T_k (а наш алгоритм встречается среди T_k для сколь угодно больших k).

3. C16. Существует язык, для которого любой алгоритм, работающий $O(n^2)$ решает его правильно на менее чем половине входов какой-то длины, но распознающийся некоторым алгоритмом, работающим $O(n^3)$.

   Нам снова понадобится эффективно занумеровать машины Тьюринга так, что каждому k может не соответствовать никакая машина Тьюринга или соответствовать некоторая машина Тьюринга, но каждая машина Тьюринга встречается в этой нумерации бесконечно много раз, и по номеру можно получить соответствующую машину Тьюринга за полиномиальное (от длины двоичной записи номера) время.

   Теперь построим алгоритм таким образом. Если $T_{|x|}$ не соответствует никакая машина Тьюринга, то выводим 0. Иначе на входе x алгоритм запускает машину $T_{|x|}$ и делает ею $|x|^3$ шагов. Если $T_{|x|}$ остановилась за это время, то выводим ответ, противоположный ответу $T_{|x|}$ на входе x, иначе 0.

   Искомый язык задаётся этим алгоритмом.

   Ясно, что этот алгоритм работает за $O(n^3)$, и если некоторый алгоритм T_k работает время $O(n^2)$, то при больших k он работает меньше чем k^3 на входах длины k, и при некотором достаточно большом k его выход отличается от выхода нашего алгоритма на всех словах длины k.

Замечание 1.3. *Мы неявно использовали существование "универсальной машины Тьюринга", которая может по любой машине Тьюринга моделировать её работу. Также мы использовали, что если машина Тьюринга работает на входах длины n время $f(n)$, то универсальная машина Тьюринга сможет смоделировать её работу за время $O(f(n))$.*