

- Примеры алгоритмов: проверка простоты, факторизация чисел; одновременное вычисление максимального и минимального элементов в массиве; быстрое умножение чисел и матриц (алгоритмы Карацубы и Штрассена); аддитивные цепочки. Модели вычислений. Формальное определение алгоритма. Различные определения трудоемкости алгоритма. Теоремы иерархии (без доказательства).
- Асимптотические обозначения (O , Ω , θ , o , ω) и их свойства. Алгоритмы типа «разделяй-и-властвуй». Основная теорема о рекуррентных оценках (нахождение асимптотики рекуррентности вида: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$). Дерево рекурсии. Линейный алгоритм нахождения медианы массива. Линейные рекуррентные последовательности.
- Детерминированные и недетерминированные МТ. Класс P . Примеры языков из P : принадлежность слова регулярному языку; принадлежность слова КС-языку; системы линейных уравнений (полиномиальная реализация метода Гаусса). Классы NP и со- NP . Примеры языков из NP : выполнимость, простые числа; пересечение регулярных языков, заданных конечными автоматами; непланарные графы;
- Полиномиальная сводимость. Сводимость по Карпу и по Куку (по Тьюрингу). Теорема Кука-Левина. Примеры полиномиально полных языков: выполнимость; пропыкающее множество; 3-сочетание; максимальное 2-сочетание; вершинное покрытие; клика; хроматическое число; гамильтонов цикл; рюкзак; разбиение; максимальный разрез. Класс PSPACE. Полных языки: истинность булевской формулы с кванторами; эквивалентность конечных автоматов; непустота пересечения последовательности ДКА.
- Потоки и разрезы в сети. Теорема о максимальном потоке и минимальном разрезе. Понятие остаточного графа и увеличивающего пути. Алгоритм Форда-Фалкерсона для вычисления максимального потока и минимального разреза. Полиномиальная реализация алгоритма максимального потока. Задача о максимальном потоке минимальной стоимости. Обобщения потоковой сети (пропускные способности узлов и пр.). Приложение потоковых алгоритмов: цепное разложение порядков (лемма Дильторта), задача о максимальном паросочетании в двудольном графе, задача о назначениях, расписание с прерываниями на идентичных процессорах.
- Задача линейного программирования (ЛП). Основные понятия. Выпуклые многогранники. Теорема двойственности. Использование ЛП для точного и приближенного решения переборных задач комбинаторной оптимизации: задача о назначении; паросочетание; вершинное покрытие; коммивояжер.
- Алгоритмы сортировки: пузырек; быстрая сортировка (quicksort); сортировка с помощью кучи; слияние; цифровая сортировка. Анализ трудоемкости алгоритма quicksort по наихудшему случаю и в среднем. Устойчивость алгоритма сортировки. Нижние оценки сортировки. Разрешающие деревья. Порядковые статистики.
- Обобщенный алгоритм Евклида. Модульная арифметика. Китайская теорема об остатках. Функция Эйлера. Первообразные корни. Кольца Z_n , в которых существуют первообразные корни. Индексы (дискретные логарифмы). Кодирование с открытым ключом. Квадратичные вычеты. Схема RSA. Протокол Диффи-Хелмана. Рациональные приближения. Элементарные свойства цепных дробей. Хеш-таблицы. Разрешение коллизий с помощью цепочек. Хеш-функции (деление с остатком, умножение). Универсальные и k -универсальные хеш-функции.
- Дискретное преобразование Фурье, алгоритм быстрого преобразования Фурье (БПФ), перемножение многочленов с помощью БПФ. Использование БПФ для распознавания образцов.
- Алгоритмы на графах и порядках: поиск в ширину; поиск в глубину; определение двусвязных и/или сильно связных компонент; кратчайшие пути: алгоритмы Дейкстры, Флойда, Беллмана-Форда; транзитивное замыкание; топологическая сортировка; остовные леса (алгоритмы Прима и Краскала); паросочетания.
- Методы решения переборных задач: динамическое программирование, шкалирование, ветви и границы, приближенные алгоритмы. ε -оптимальная процедура решения задачи о рюкзаке.
- Классы RP , BPP , ZPP . Лемма Шварца. Вероятностные алгоритмы: проверка простоты; вычисление медианы массива; проверка полиномиальных тождеств; поиск паросочетаний в графах; алгоритм Каргера поиска минимального разреза.

Литература

- [АХУ] Ахо А., Хопкрофт Д., Ульман Д. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
 - Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. Электронный вариант: <http://trpl.narod.ru/NPC-book.htm>
 - [Кормен 1] Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. М.: МЦНМО, 2002.
 - [Кормен 2] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. 2-е изд. М.: Вильямс, 2005.
 - Кузюрин Н., Фомин С. Эффективные алгоритмы и сложность вычислений. М.: МФТИ, 2007.
- Дополнительная литература**
- Верещагин Н., Шень А. Вычислимые функции. М.: МЦНМО, 1999. Электронный вариант: www.mccme.ru/free-books
 - Виноградов И. Основы теории чисел. М.-Л.: Гостехиздат, 1952
 - Вялый М., Журавлев Ю., Флеров Ю. Дискретный анализ. Основы высшей алгебры. М.: МЗ Пресс, 2007.
 - [К-Ш-В] Китаев А., Шень А., Вялый М. Классические и квантовые вычисления. М.: МНЦМО-ЧеРо, 1999
 - [Хинчин] Хинчин А. Цепные дроби. М.: Наука, 1979.

11. Шенъ А. Программирование. Теоремы и задачи. М.: МЦНМО, 2007. Электронный вариант: www.mccme.ru/free-books
12. Lovasz L. Computational Complexity. Электронный вариант www.cs.elte.hu/~lovasz/complexity.pdf
13. Кнут Д. Искусство программирования для ЭВМ. Т. 1. М.: Мир, 1976; т. 2. М.: Мир, 1977; т. 3. М.: Мир, 1978; т. 1. М.: Вильямс, 2006; т. 2. М.: Вильямс, 2007; т. 3. М.: Вильямс, 2007; т. 4. М.: Вильямс, 2013.

ЗАДАНИЕ

Задание состоит из списка недельных заданий (указаны даты обсуждения на семинарах и/или сдачи соответствующих задач). Каждое недельное задание состоит из четырех-пяти обязательных задач и одной-двух дополнительных задач (дополнительные задачи выделены буквой «Д»).

Решение дополнительных задач не обязательно, но может быть полезно студентам, претендующим на более высокую оценку.

Обозначения

$\lfloor a \rfloor$ – наибольшее целое, не превосходящее число a ;

$\lceil a \rceil$ – обозначает наименьшее целое, превосходящее a .

$\phi(n)$ – функция Эйлера, равная количеству натуральных чисел, меньших n и взаимно простых с ним. При этом полагают, что число 1 взаимно просто со всеми натуральными числами.

По определению, $\log^* M = \min\{k \mid \underbrace{\log \log \dots \log}_k M \leq 1\}$.

Пусть $f(n)$ и $g(n)$ – неотрицательные функции.

- $f(n) = O(g(n))$ означает, что порядок роста g при $n \rightarrow \infty$ не меньше порядка роста f т. е. $\exists c > 0$, такое что при $n > n_0$, $f(n) \leq c g(n)$;
- $f(n) = \Omega(g(n)) \Leftrightarrow g(n) = O(f(n))$ (порядок роста f не меньше порядка роста g);
- $f(n) = o(g(n))$, если $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$ (f имеет меньший порядок роста, чем g);
- $f(n) = \omega(g(n)) \Leftrightarrow g(n) = o(f(n))$ (f имеет меньший порядок роста, чем g);
- $f(n) = \theta(g(n))$, если $f(n) = O(g(n))$ и $f(n) = \Omega(g(n))$ (порядки роста f и g одинаковы).

Задание на 1-ю неделю (8.02 -14.02). Темы 1 и 2 программы.

Оценки

1. Покажите, что $\sum_{i=1}^n \frac{1}{i} = \Theta(\log n)$.

2. Найдите θ -асимптотику C_{2n}^n .

3. Это задача на применение «Основной теоремы о рекуррентных оценках» [Кормен 1, § 4.3]. Вы должны проверить, выполняются ли условия теоремы, получить оценки и сравнить их. Пока можно считать, что $n = 2^k$ или $n = 3^k$.

Укажите рекурсивную процедуру с наименьшей асимптотической сложностью:

- алгоритм, разбивающий задачу размера n на девять подзадач размера $\frac{n}{3}$ и использующий для этого $\theta(n^2 \log n)$ операций;
- алгоритм, разбивающий задачу размера n на шестнадцать подзадач размера $\frac{n}{4}$ и использующий для этого $\theta(n^2)$ операций;
- алгоритм, разбивающий задачу размера n на четыре подзадачи размера $\frac{n}{2}$ и использующий для этого $\theta(\frac{n^2 \sqrt{n}}{\log^2 n})$ операций.

4. Найдите θ -асимптотику следующих рекуррентностей.

4.1. $T_1(n) = 2 \left(\frac{n}{2} \right) + n$.

4.2. $T_2(n) = 3T_2\left(\frac{n}{3}\right) + n^2$.

4.3. $T_3(n) = 4T_3\left(\frac{n}{2}\right) + \frac{n}{\log n}$.

Быстрое умножение чисел и матриц

5. Алгоритм быстрого умножения чисел (алгоритм Карацубы) подробно описан во многих источниках (Например, [АХУ]). Ниже приведен пример работы этого рекурсивного алгоритма (стрелка после произведения указывает на вспомогательные произведения, которые требуется вычислить; рекурсия останавливается на двузначных числах):

1. $216_{10} \times 139_{10} = 11011001 \times 10001011 \rightarrow$
2. $1101 \times 1000, 1001 \times 1011, (1101 + 1001) \times (1000 + 1011) = 10110 \times 10011;$
3. $1101 \times 1000 \rightarrow 11 \times 10 = 110; 01 \times 00 = 0; (11 + 01) \times (10 + 00) = 100 \times 10 = 1000;$
4. $1101 \times 1000 = 1100000 + (1000 - 110 - 0) \times 100 + 0 = 1101000;$
5. $1001 \times 1011 \rightarrow 10 \times 10 = 100, 01 \times 11 = 11, (10 + 01) \times (10 + 11) = 11 \times 101 = 1111;$
6. $1001 \times 1011 = 1000000 + (1111 - 100 - 11) \times 100 + 11 = 1100011;$
7. $010110 \times 010011 \rightarrow 010 \times 010 = 100, 110 \times 011, (010 + 110) \times (010 + 011) = 1000 \times 0101 = 101000;$
8. $0110 \times 0011 \rightarrow 01 \times 00 = 0, 10 \times 11 = 110, (1 + 10) \times (0 + 11) = 11 \times 11 = 1001;$
9. $0110 \times 0011 = 0 + (1001 - 0 - 110) \times 100 + 110 = 10010;$
10. $010110 \times 010011 = 100000000 + (101000 - 10010 - 100) \times 1000 + 10010 = 110100010;$
11. $11011001 \times 10001011 = 110100000000000 + (110100010 - 1101000 - 1100011) \times 10000 + 1100011 = 111010111010011 = 30163_{10}.$

Получим рекуррентность для оценки полного числа всех элементарных битовых арифметических операций алгоритма Карацубы (вы должны самостоятельно сформулировать понятие «элементарная битовая арифметическая операция»). Считаем, что $n = 2^k$. Обозначим $\text{Add}(m)$ — число элементарных битовых арифметических операций, требуемых для сложения двух m -битовых чисел. Обозначим $\text{Mult}(m)$ — число элементарных битовых арифметических операций, требуемых для умножения двух m -битовых чисел методом Карацубы. Запишем соотношения между $\text{Mult}(.)$ и $\text{Add}(.)$, вытекающие из алгоритма: $\text{Mult}(2m) \leq 3 \text{ Mult}(m+1) + O(\text{Add}(m))$. Обычное школьное сложение в столбик дает оценку для $\text{Add}(m) = O(m)$ и мы получаем рекуррентное соотношение на $\text{Mult}(.)$.

Приведите подробный асимптотический анализ рекуррентности.

Основная трудность заключается в оценке прибавления единицы к аргументу во втором члене: нужно привести формальные аргументы, показывающие, почему этой поправкой можно пренебречь.

Д1. В этой задаче нужно применить приемы вычисления рекуррентностей, изложенные в книге [Кормен 1] или [Кормен 2], а также провести анализ, связанный с использованием функций $\lfloor \cdot \rfloor$ в рекуррентности.

Оцените высоту дерева рекурсивных вызовов рекуррентности

$$T(n) = T(n - \lfloor \sqrt{n} \rfloor) + T(\lfloor \sqrt{n} \rfloor) + \theta(n) \text{ как можно точнее.}$$

Задание на 2-ю неделю (15.02 -21.02). Темы 2 и 3 программы.

6. Обычный способ перемножения 2×2 матриц требует 8 умножений и 4 сложения:

$$\begin{array}{cc|cc} a & b & e & f \\ c & d & g & h \end{array} \times \begin{array}{cc} e & f \\ g & h \end{array} = \begin{array}{ll} ae + bg & af + bh \\ ce + dg & cf + dh \end{array}$$

В 1969 году Ф.Штрассен (V.Strassen) открыл следующий алгоритм: пусть $p_1 = a(f-h); p_2 = (a+b)h; p_3 = (c+d)e; p_4 = d(g-e); p_5 = (a+d)(e+h); p_6 = (b-d)(g+h); p_7 = (a-c)(e+f)$. Тогда

$$\begin{aligned} ae + bg &= p_5 + p_4 - p_2 + p_6 & af + bh &= p_1 + p_2 \\ cf + dh &= p_5 + p_1 - p_3 - p_7 & ce + dg &= p_3 + p_4 \end{aligned}$$

Таким образом, нам требуется 7 умножений и 18 сложений. На первый взгляд, мы ничего не выиграли: умножений стало меньше, но сложений больше, но тут начинается маленькое чудо. Обратим внимание на то, формулы Штрассена справедливы и тогда, когда переменные некоммутативные, и поэтому можно рекурсивно запустить алгоритм (подробности можно прочитать в [Кормен 2, 28.2]), т. е. разбить матрицу порядка $n \times n$ на четыре блока порядка $\frac{n}{2} \times \frac{n}{2}$ и провести перемножения блоков по формулам, записанным выше.

Поскольку сложение матриц требует $O(n^2)$, то получится такая рекуррентная оценка трудоемкости алгоритма перемножения матриц: $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$.

Оцените трудоемкость алгоритма Штрассена, используя дерево рекурсии (можно считать, что $n = 2^k$).

7. Дано n точек плоскости. Предложите как можно более быструю процедуру нахождения круга минимального радиуса с центром в начале координат, содержащего не менее половины точек. (Считаем, что арифметические операции и сравнения выполняются за единицу времени.)

8. Рассмотрим рекурсивный алгоритм вычисления медианы массива, который отличается от стандартного ([Кормен 1, 10.3] или [Кормен 2, 9.3]) тем, что массив делится не на «пятерки», а на «девятки» элементов.

8.1. Получите рекуррентное соотношение для трудоемкости этого алгоритма.

8.2 Оцените трудоемкость алгоритма с помощью дерева рекурсий.

9. Пусть G — язык в алфавите $\{0,1,2\}$ состоит из всех слов, в которых соседние символы отличаются (как числа) не более, чем на 1, например, $210 \in G$, $201 \notin G$. Найдите рекуррентное соотношение, которому удовлетворяет последовательность $\{g_n, n=1,2,\dots\}$, где g_n равно числу слов длины n в G , и оцените g_{2014} .

Д2.

Д2.1. Сравните по величине 2014-е члены рекуррентных последовательностей: $a_0 = b_0 = 0$; $a_{n+1} = a_n^2 + 5$, $b_{n+1} = a_n^2 + 2^n$.

Д2.2. Найдите как можно более точную асимптотику последовательности a_n .

Задание на 3-ю неделю (22.02 -28.02). Темы 2 и 3 программы

Класс P. Полиномиальные алгоритмы

11. Язык $L_{\{m \times 2014-incons\}} = \{(A, b)\}$ состоит из кодировок всех несовместных систем линейных уравнений $Ax = b$ с целыми коэффициентами и 2014 неизвестными, в которых максимальный модуль целых коэффициентов матрицы A и вектора b («высота» системы) не превышает 10000.

Длиной записи системы считаем суммарную длину двоичных записей всех коэффициентов системы.

11.1 Укажите два слова, принадлежащие языку, и два слова, не принадлежащие языку.

11.2 Покажите, что $L_{\{m \times 2014-incons\}} \in P$.

Подсказка. Иначе говоря, нужно показать, что существует машина Тьюринга М со следующими свойствами. На вход М поступает слово, содержащее двоичные записи коэффициентов системы. М останавливается через полиномиальное по длине входа число шагов и печатает ответ 1, если система несовместна, или печатает 0, если система совместна.

11.3 Как изменится ваш ответ и/или аргументы, если убрать ограничение на число переменных и высоту системы.

Комментарий. Не рекомендуется бодро ссылаться, скажем, на теорему Кронекера-Капелли, не указывая, как вычислять ранг матрицы за полиномиальное по длине записи время, поскольку в известном со школы методе исключения игнорируется длина записи промежуточных вычислений.

Покажем на примере этой простой задачи критическую важность выбора модели вычислений.

Допустим, что в нашем распоряжении имеется РАМ, т.е. равнодоступная адресная машина (random access machine — RAM)¹, в которой арифметические операции

над очень большими числами, например, имеющими экспоненциальное по входу число битов, выполняются за единицу времени. В этом случае можно, например, не учитывать длину записи промежуточных вычислений в методе Гаусса и его трудоемкость будет в точности такая, как об этом пишут учебники вычислительной математики.

Построим алгоритм, выполняющий полиномиальное по длине записи натурального n количество арифметических операций над экспоненциально длинными числами, который решает задачу факторизации, т.е. находит нетривиальные делители числа n . Подобная РАМ может в принципе вскрывать большинство современных шифров, в которых факторизация является одним из важнейших примитивов. Идея процедуры следующая². Рассмотрим функцию натурального аргумента $f_n(a) = \text{НОД}(a!, n)$. Следующие три утверждения непосредственно следуют из определения:

- 1) $f_n(1) = 1$, $f_n(n) = n$;
- 2) $f_n(a)$ неу бывает при $1 \leq a \leq n$;
- 3) число n составное тогда и только тогда, когда существует такое число $1 < a_0 < n$, что $f_n(a_0) > 1$.

Итак, нетривиальный делитель a_0 отвечает первому «скачку» $f_n(a)$, а для того, чтобы найти его, нужно воспользоваться обычным двоичным поиском. Осталось научиться вычислять $f_n(a)$, используя $\text{poly}(\log n)$ арифметических операций [над, возможно, очень большими числами]. Можно ограничиться $O(\log n)$ операциями, но мы рассмотрим более простой и более трудоемкий способ. Сначала решим следующую задачу.

12.

12.1. Покажите, что язык $L_{ind} = \{a, b, c, d \text{ — натуральные числа, такие что } a^b = c \bmod d\}$ принадлежит P

12.2 Укажите два слова, принадлежащие языку, и два слова, не принадлежащие языку.

Возвращаемся к задаче вычисления скачка $f_n(a)$. По аналогии с возведением в степень в предыдущей задаче факториал инкрементально пересчитывать, если воспользоваться следующими формулами, прямо следующими из определения: $(2i+1)! = (2i+1)(2i)!$

и $(2b)! = (b!)^2 C_{2b}^b$. Для вычисления C_{2b}^b применим еще один трюк. Пусть $2b < n$. Тогда можно, используя $\text{poly}(\log n)$ арифметических операций, получить

число C_{2b}^b , сначала вычисляя степень $A = (2^n + 1)^{2b}$, а потом выделяя нужные биты A . Так можно получить, алгоритм, требующий для факторизации $O(\log^3 n)$ операций, который, конечно, не будет полиномиальным].

13. Покажите, что в класс P входит язык $L_{powers} = \{1, 10^{2014}, \dots\}$, состоящий из двоичных записей 2014-х степеней натуральных чисел.

14. Покажите, что класс P как множество языков замкнут относительно итерации (операции Клини).

В изображенных ниже таблицах используются следующие обозначения:

- А — конечный автомат (КА),
НА — недетерминированный КА (НКА),
ДА — детерминированный КА (ДКА),
Р — регулярное выражение (РВ),
О — магазинный автомат (МП),
Н — магазинный автомат, принимающий по пустому стеку,
F — магазинный автомат, принимающий по финальному состоянию,

¹ Грубо говоря, РАМ-программа --- это программа на BASIC'e. Здесь полезно подумать, как формально определить РАМ, и чем она operationально отличается от МТ. Описание РАМ можно найти, например, в книге [АХУ] или посмотреть в сети.

² Возможно, этот пример рассказывался в прошлом году на дискретном анализе.

DP — детерминированный МП-автомат,
 G — контекстно-свободная грамматика (КСГ).
 M — машина Тьюринга.

$L(\bullet)$, где вместо «» можно подставить один из перечисленных выше объектов, обозначает класс языков, принимаемых (порождаемых) объектами указанного класса. Например, $L(G)$ означает класс всех КС-языков, причем описание каждого языка дается соответствующей КСГ. Будем считать, что объекты задаются своими стандартными описаниями, например, НКА кодируется его диаграммой.

Если специально не оговорено, то предполагается, что все языки заданы над двоичным алфавитом $\{0,1\}$.

В таблицах ниже строки отвечают предикатам, а столбцы — классам языков, которые задаются указанными объектами. Рассматриваются следующие предикаты³

$L(\bullet) = \emptyset$? — язык пуст,
 $|L(\bullet)| = \infty$? — язык бесконечен,
 $w \in (\notin) L(\bullet)$? — слово «w» принадлежит (не принадлежит) языку,
 $L(\bullet_1) = L(\bullet_2)$? — языки, порождаемыми объектами (\bullet_1) и (\bullet_2) , эквивалентны.

Нашей целью является оценка сложности получаемых «задач», т. е. оценка сложности вычисления указанных предикатов на соответствующих классах языков. Например, ячейке (2,2) Таблицы 1 отвечает задача проверки непустоты языка, заданного ДКА, а ячейке (2,4) Таблицы 2 — задача проверки эквивалентности языков, заданных магазинными автоматами. Таблицы будут заполняться в несколько проходов, и в нулевом приближении полезно прикинуть, какие из указанных задач разрешимы, а какие — нет. Уточнением этой таблицы мы будем заниматься на протяжении курса.

Таблица 1.

	DA	NA	R	G	N	F
$L(\bullet) = \emptyset?$						
$ L(\bullet) = \infty?$						
$w \in L(\bullet)?$						
$w \notin L(\bullet)?$						

Таблица 2.

	DA и DA	DA и NA	DA и R	O и O
$L(\bullet_1) = L(\bullet_2)?$				
$L(\bullet_1) \neq L(\bullet_2)?$				

15. Покажите, в столбцах Таблицы 1, отвечающих, соответственно DA, NA, R, и G и в столбце Таблицы 2, отвечающему DA и DA, можно поставить знак «P», что означает, что соответствующие задачи разрешимы за полиномиальное по входу время.

Д3.

Д3.1. Постройте полиномиальный по входу алгоритм для задачи линейного программирования на плоскости:

$a_0x + b_0y \rightarrow \min$
$a_1x + b_1y = c_1$
...
$a_kx + b_ky = c_k$

здесь все коэффициенты a_i, b_i, c_i — целые числа, по абсолютной

величине меньшие h (высоты системы). Входом задачи можно считать $\max\{k, \log h\}$.

Д3.2. Будем теперь считать, что любая арифметическая операция над коэффициентами и операция сравнения занимают единицу времени. Постройте линейный $O(k)$ -алгоритм для рассмотренной задачи.

Д4. Покажите, в столбцах Таблицы 1, отвечающих, соответственно, N и F, можно поставить знак «P».

Комментарий. Итак, все задачи, указанные в Таблице 1, разрешимы за полиномиальное по входу время.

Задание на 4-ю неделю (1.03 -8.03). Темы 3 и 4 программы

Класс NP

16. Покажите, что следующие языки принадлежат NP (постройте соответствующие сертификаты “y” и проверочные предикаты $R(x, y)$).

16.1. Язык совместных систем линейных уравнений с целыми коэффициентами от 2014 неизвестных.

16.2. Язык совместных систем линейных неравенств с целыми коэффициентами от 2014 неизвестных.

³ Обратите, пожалуйста, внимание на то, что предикаты в некоторых строках дополнительны, т. е. являются отрицаниями друг друга. Это отражает одну из особенностей мира алгоритмов, в котором ответы «Да» и «Нет» принципиально несимметричны: Каждый понимает, что ситуация, когда программа остановилась или когда она еще работает, существенно отличаются, и поэтому, например, есть предикаты, имеющие алгоритм проверки выполнимости, и не имеющие алгоритма проверки невыполнимости и наоборот. Знаете ли вы такие предикаты?

16.3. Составные числа, не содержащие в двоичной записи под слова 10101.

Разберем теперь нетривиальный пример. Просто понять, что в NP лежит язык составных чисел: $A = \{1, 4, 6, 8, 10, \dots\}$ (сертификатом служат предъявляемые сомножители).

Но оказывается, что в NP лежит и дополнительный язык $B = N \setminus A = \{2, 3, 5, \dots\}$ простых чисел⁴ Полиномиальный сертификат устроен хитро. Как мы знаем из курса дискретного анализа, $p \in B \Leftrightarrow \exists g: \{g^i \bmod p, i = 0, 1, \dots\} = \{1, 2, \dots, p - 1\}$ (указано равенство множеств). Поскольку длина записи числа p составляет $\log p$, то

длина NP -сертификата должна быть $\text{poly}(\log p)$. И если быстро возводить числа $\bmod p$ в степень мы еще умеем (каким образом?), то все равно массив $\{g^i \bmod p\}$ слишком длинный (экспоненциальный по длине записи). Но, как мы помним, вычет g с нужными свойствами существует тогда и только тогда,

когда выполнено: $g^{p_1} \neq 1 \bmod p, \dots, g^{p_k} \neq 1 \bmod p$, где p_1, \dots, p_k — это все простые делители числа $p - 1 = p_1^{l_1} \dots p_k^{l_k}$. Число проверок действительно уменьшилось и стало полиномиальным (их заведомо не больше $\log p$), но, кажется, что мы поменяли шило на мыло: нам ведь нужно решить ТУ ЖЕ ЗАДАЧУ ПОСТРОЕНИЯ СЕРТИФИКАТА ПРОСТОТЫ для всех p_1, \dots, p_k . Хитрость заключается в том, что нужно применить ту же идею рекурсивно, поскольку длина сертификатов для всех p_1 сильно уменьшилась! Фактически сертификатом будет дерево с нужными пометками в вершинах, и для завершения доказательства нужно показать, что суммарная длина всех участвующих в описании дерева компонентов останется полиномиальной по $\log p$ (это предлагается в Д

17. Постройте NP -сертификат простоты для числа $p = 3911$. Число «13» является первообразным корнем $\bmod p$. Простыми в рекурсивном построении считаются только числа 2, 3, 5 (они сами являются своими сертификатами).

18. Покажите, что класс NP замкнут операции итерации (операции Клини). Укажите, как построить для результирующего языка соответствующий сертификат «у» и проверочный предикат $R(x, y)$.

Решим подготовительную задачу, которая в следующем недельном задании позволит разобрать пример еще одного нетривиального языка из NP (смотри Д7).

Граф $G(V, E)$ называется *планарным*, если его можно вложить в плоскость, т. е. существует отображение $f: G \rightarrow R^2$, посылающее вершины V в различные точки плоскости, а ребра E — в кривые (которые можно считать ломанными), соединяющие соответствующие точки-вершины, при этом кривые-ребра могут пересекаться только по вершинам. Соответственно, граф $G(V, E)$ называется *торическим*, если его можно вложить в тор — поверхность бублика — $T = S^1 \times S^1$ (S^1 — это стандартное обозначение одномерной сферы — окружности). По определению, любой планарный граф является торическим, но не наоборот. Как известно, на плоскости нельзя нарисовать без пересечения ребер (вложить) ни граф K_5 (полный пяти вершинник), ни граф $K_{3,3}$ (три дома, три колодца), но их можно вложить в тор, поскольку на плоскости удается изобразить все ребра этих графов, кроме одного, которое можно пропустить по ручке (вдоль параллели тора). Но удастся ли такой трюк с графом $K_{3,6}$ (число домов осталось прежним, но добавлено еще три колодца)?

19. Докажите или опровергните, что граф $K_{3,6}$ является торическим. Тор удобно изображать прямоугольником с отождествленными противоположными сторонами.

Полиномиальная сводимость.

20. Язык ГЦ состоит из всех графов, имеющих гамильтонов цикл (несамопересекающийся путь, проходящий через все вершины графа). Язык ГП состоит из всех графов, имеющих гамильтонов путь. Постройте **явные** полиномиальные сводимости ГП к ГЦ и наоборот. Графы кодируются, например, их матрицей смежности.

Комментарий. Это очень хорошее упражнение для того, чтобы понять определение полиномиальной сводимости.

Д5. Покажите, что NP принадлежит язык из пункта 16.2 при дополнительном предположении, что переменные $\{x_i\}$ целочисленные. (Если эта задача представляет трудности, то разберите случай, когда неизвестных две.)

Д6. Покажите, что длина NP -сертификата простоты, описанного в задаче 17, равна $O(\log p)$.

Д7. Покажите, что класс торических графов принадлежит NP .

⁴ В 2002 году появилась сенсационная работа, показывающая, что $B \in P$. Если вы будете ссылаться на ее результаты, то ОБЯЗАНЫ привести доказательство