АМВ. ДЗ на неделю 2.

Прохоров Юрий, 771

Задача 1

Докажите следующие свойства полиномиальной сводимости:

(i) Рефлексивность: $A \leq_p A$; транзитивность: если $A \leq_p B$ и $B \leq_p C$, то $A \leq_p C$;
(ii) Если $B \in \mathcal{P}$ и $A \leq_p B$, то $A \in \mathcal{P}$;
(iii) Если $B \in \mathcal{NP}$ и $A \leq_p B$, то $A \in \mathcal{NP}$.

Решение:

(i) Докажем рефлексивность. Запишем определение полиномиальной сводимости:

$$A \leq_p B \qquad \Longleftrightarrow \qquad \exists f(x): \ \Big[x \in A \iff f(x) \in B\Big],$$

где f — вычислимая (на МТ) за полиномиальное время функция.

При $A = B$ можно положить $f(x) = x$:

$$A \leq_p A \qquad \Longleftrightarrow \qquad \exists f(x) = x: \ \Big[x \in A \iff f(x) = x \in A\Big],$$

Докажем транзитивность. По определению:

$$A \leq_p B \qquad \Longleftrightarrow \qquad \exists f(x): \ \Big[x \in A \iff f(x) \in B\Big],$$

$$B \leq_p C \qquad \Longleftrightarrow \qquad \exists g(x): \ \Big[x \in B \iff g(x) \in C\Big],$$

Положим h :

$$h(x) = g(f(x)) \qquad \forall x \in A$$

Тогда видно, что

$$x \in A \qquad \Longleftrightarrow \qquad h(x) \in C$$

Поэтому выполняется определение полиномиальной сводимости $A \leq_p C$, если в качестве вычислимой функции взять функцию h. Заметим, что h тоже вычисляется за полиномиальное время, потому что суперпозиция полиномов есть полином.

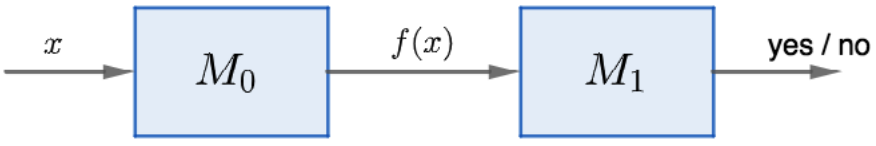
(ii) Запишем определение принадлежности языка классу $\mathcal{P}$:

$$A \in \mathcal{P} \iff \exists \text{ детерм. МТ, распознающая язык } A \text{ за полиномиальное время}$$

Построим такую машину Тюринга.

$A \leq_p B \ \Rightarrow$ существует детерминированная МТ M_0, вычисляющая сводящую функцию f за полиномиальное время.

$B \in \mathcal{P} \ \Rightarrow$ существует детерминированная МТ M_1, решающая задачу распознавания $z \in B$ за полиномиальное время.



Тогда машина Тюринга, которая получается последовательным соединением описанных выше МТ, будет распознавать язык A. Так как обе МТ работают за полиномиальное время, а сумма полиномов есть полином, то и новая МТ работает за полиномиальное время и будет детерминированной.

(iii) Запишем определение принадлежности языка классу $\mathcal{NP}$:

$$A \in \mathcal{NP} \iff \exists poly(|x|),\ \exists V(x,s):\ \Big[x \in A \iff \exists s : V(x,s) = 1\Big],$$

где верификатор (предикат) V вычисляется на детерминированной МТ за время $\leq poly(|x|)$, s — сертификат.

Строим абсолютно аналогичную конструкцию, как в предыдущем пункте. Язык $B \in \mathcal{NP} \Rightarrow$

$$z \in B \qquad \iff \qquad \exists s:\ M_1(z,s) = 1$$

Пусть МТ M — последовательное соединение M_0 и M_1. Она является детерминированной. Пусть на вход подается $x \in A$. По определению полиномиальной сводимости

$$x \in A \qquad \iff \qquad f(x) \in B \qquad \iff \qquad \exists s:\ M_1(f(x),s) = 1 \qquad \iff \qquad \exists s:\ M(x,s) = 1$$

Это и есть определение принадлежности $A \in \mathcal{NP}$.

Задача 2

Докажите, что следующие языки принадлежат классу $\mathcal{P}$. Считайте, что графы заданы матрицами смежности.

(i) Язык двудольных графов, содержащих не менее 2018 треугольников (троек попарно смежных вершин);
(ii) Язык несвязных графов без циклов;
(iii) Язык квадратных $\{0;1\}$-матриц порядка $n \geq 3000$, в которых есть квадратная подматрица порядка $n - 2018$, заполненная одними единицами.

Решение:

(i) Покажем, что такой язык L пуст. Отсюда будет следовать, что $L \in \mathcal{P}$, так как МТ может выдавать константный ответ 0.

Допустим, в двудольном графе есть хотя бы один треугольник. Тогда, по принципу Дирихле, есть две вершины, попавшие в одну и ту же долю графа. Но тогда между ними не может быть ребра — противоречие.

(ii) Приведем детерминированный полиномиальный алгоритм распознавания такого языка.

Произведем обход графа с помощью DFS (поиска в глубину). Если мы нашли более одной компоненты связности, то граф будет несвязным. Если DFS нашел цикл в какой-либо компоненте, то граф будет циклическим. Таким образом, этих двух условий необходимо и достаточно, чтобы некоторый граф принадлежал языку. DFS работает за время, линейное по числу вершин и ребер графа.

(iii) Посчитаем количество подматриц нужного нам размера. Нам нужно из n столбцов выбрать $n - 2018$, и то же самое сделать со строками, поэтому всего подматриц:

$$\left(C_n^{n-2018}\right)^2 = \left(C_n^{2018}\right)^2 = \left(\frac{n(n-1)\dots(n-2017))}{2018!}\right)^2 = \left(\Theta(n^{2018})\right)^2 = \Theta(n^{4036})$$

В каждой подматрице надо проверить каждый из $(n-2018)^2 = \Theta(n^2)$ элементов на равенство единице. Поэтому итоговая асимптотика такого алгоритма будет $\Theta(n^{4038})$, что является полиномиальной асимптотикой.

Задача 3

Рассмотрим СЛУ $Ax = b$ с целыми коэффициентами. Пусть в этой системе m уравнений и n неизвестных, причем максимальный модуль элемента в матрице A и столбце b равен h.

(i) Оцените сверху числители и знаменатели чисел, которые могут возникнуть при непосредственном применении метода Гаусса. Приведите пример, в котором в процессе вычислений в промежуточных результатах длина возникающих чисел растёт быстрее, чем любой полином от длины записи системы в битовой арифметике.

(ii) Оказывается, что если на каждом шаге эмулировать рациональную арифметику и сокращать дроби с помощью алгоритма Евклида, модифицированный таким образом метод Гаусса окажется полиномиальным по входу (по поводу этого факта будет выложен доп. файл). Оцените трудоемкость такого модифицированного метода по параметрам m, n и $\log h$.

Решение:

(i) Пусть нам дана система

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Будем применять к расширенной матрице системы $\tilde{A}$ размеров $(m \times (n+1))$ метод Гаусса. Посмотрим, что будет в элементе a_{22} после одной его итерации. За итерацию алгоритма Гаусса будем считать обнуление всех элементов ниже i-го, находящихся в i-ом столбце. Также для простоты будем считать, что не возникает частных случаев, когда появляется деление на 0 (это только ускоряет алгоритма и не дает числам расти). Итак, после первой итерации:

$$\widehat{a_{22}} = a_{22} - a_{21}\frac{a_{12}}{a_{11}} = \frac{a_{22}a_{11} - a_{12}a_{21}}{a_{11}} = \frac{b_{22}}{c_{22}}$$

Видно, что числитель и знаменатель можно оценить сверху

$$|b_{22}| \le 2h^2 = b, \qquad |c_{22}| \le h = c$$

Такие же рассуждения верны и для всех остальных элементов матрицы, кроме первой строки.

Посмотрим, что будет после следующей итерации. Теперь алгоритм Гаусса применяем к матрице, в которой стоят рациональные числа. Ниже в матрице вместо $n+1$ будем писать n, чтобы запись было более аккуратной.

$$\begin{pmatrix} 1 & \frac{a_{12}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} \\ 0 & \frac{b_{22}}{c_{22}} & \dots & \frac{b_{2n}}{c_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \frac{b_{m2}}{c_{m2}} & \dots & \frac{b_{mn}}{c_{mn}} \end{pmatrix}$$

Алгоритм Гаусса применяем к подматрице, не содержащей первой строки и первого столбца. Посмотрим, что будет в элементе a_{33}:

$$\widehat{a_{33}} = a_{33} - a_{32}\frac{a_{23}}{a_{22}} = \frac{b_{33}}{c_{33}} - \frac{b_{32}}{c_{32}}\frac{b_{23}}{c_{23}}\frac{c_{22}}{b_{22}} = \frac{b_{22}b_{33}c_{23}c_{32} - b_{23}b_{32}c_{22}c_{33}}{b_{22}c_{23}c_{32}c_{33}} = \frac{d_{33}}{e_{33}}$$

$$|d_{33}| \le 2 \cdot b^2c^2 = 2(2h^2)^2(h)^2 = 8h^6 = \Theta(h^6)$$

$$|e_{33}| \le bc^3 = 2h^2h^3 = 2h^5 = \Theta(h^5)$$

Таким образом, и числитель, и знаменатель как минимум возводятся в куб на каждой итерации. Также можно заметить, что верхняя оценка знаменателя всегда на одну степень h меньше, чем у числителя. В любом случае, длина записи этих чисел не является полиномом от $\log h$.

Алгоритм Гаусса работает, пока у него не закончатся строки или столбцы. Будем для определенности считать, что $m \geq n$. Тогда после $(n-1)$-ой итерации можно дать приближенные оценки:

$$\text{числители} \leq \Omega(h^{3^{n-1}}), \qquad \text{знаменатели} \leq \Omega(h^{3^{n-1}-1})$$

Приведем пример, когда длина чисел растет экспоненциально. При этом мы учитываем то, что дроби, получающиеся в процессе работы алгоритма, не сокращаются. Рассмотрим матрицу, в которой на диагонали стоят числа h, а во всех остальных клетках $r \ll h$.

$$\begin{pmatrix} h & r & \dots & r \\ r & h & \dots & r \\ \vdots & \vdots & \ddots & \vdots \\ r & r & \dots & h \end{pmatrix}$$

Диагональные элементы после одной итерации:

$$\widehat{a_{22}} = \frac{a_{22}a_{11} - a_{12}a_{21}}{a_{11}} = \frac{h^2 - r^2}{h}$$

Остальные элементы:

$$\widehat{a_{23}} = \frac{a_{23}a_{11} - a_{13}a_{21}}{a_{11}} = \frac{r(h-r)}{h}$$

Таким образом, на следующей итерации алгоритм будет применятся к матрице, порядок которой на 1 меньше, и

$$\hat{h} = \frac{h^2 - r^2}{h}, \qquad \hat{r} = \frac{r(h-r)}{h} \ll \hat{h}$$

Таким образом, числа на диагонали примерно каждый раз возводятся в квадрат, поэтому длина их двоичной записи увеличивается в 2 раза. То есть на k-ой итерации алгоритма, длина двоичной записи матрицы $\geq C2^k$, что растет быстрее, чем любой полином от k.

(ii) Будем после каждой итерации (см. выше, что считается итерацией) применять алгоритм Евклида, для сокращения рациональных дробей на наибольший общий делитель. Будем пользоваться утверждением доказанным в листке Сергей Тарасова: *все элементы матриц, возникающих в методе Гаусса, являются отношением каких-то миноров исходной расширенной матрицы системы.*

Если все элементы матрицы порядка k не превосходят h по модулю, то ее определитель и длину его записи можно оценить:

$$\Delta \leq k! \cdot h^k \leq n! \cdot h^n, \qquad \log \Delta = O(n \log n + n \log h)$$

Это будет доказано в пятой задаче.

Алгоритм Евклида работает за линейное по длине входа время, значит на его применение нужно число операций порядка $O(n \log n + n \log h)$.

После каждой итерации мы сокращаем $O(mn)$ дробей, значит, на это нужно время $O(mn^2 \log n + mn^2 \log h)$. Всего n итераций (считаем, что $n \leq m$ для определенности, так как алгоритм Гаусса работает, пока у него не кончатся столбцы или строки). Итак, итоговая асимптотика:

$$T(n, m, \log h) = O(mn^3(\log n + \log h))$$

Задача 4

Докажите, что классы $\mathcal{P}$ и $\mathcal{NP}$ замкнуты относительно операции $*$ — звезды Клини (была в ТРЯПе). Для языка из $\mathcal{NP}$ приведите также и сертификат принадлежности слова из Σ^* языку L^*, где $L \in \mathcal{NP}$.

Решение:

а)

$$L \in \mathcal{P} \qquad \Longrightarrow \qquad L^* \in \mathcal{P}$$

Требуется построить детерминированный полиномиальный алгоритм проверки $x \in L^*$, если у нас есть полиномиальный алгоритм проверки $x \in L$.

$$x \in L^* \qquad \Longleftrightarrow \qquad x = u_1 u_2 \dots u_m, \;\; u_i \in L \cup \{\varepsilon\}$$

Будем пытаться покрыть словами из $L \cup \{\varepsilon\}$ префикс слова x длины k. Если у нас получится это сделать при $k = n = |x|$, то это будет означать, что $x \in L^*$.

Пусть V — множество таких чисел k, что у нас получилось покрыть префикс x длины k словами из $L \cup \{\varepsilon\}$. Опишем тогда алгоритм: (считаем $x = x_1 x_2 \dots x_n$)

1. Положить $V = \{0\}$.
2. for $j = 1, 2, \dots, n$:
   for $i \in V$:
   if $x[i+1, j] \in L$:
   Добавить j в V.
3. if $n \in V$:
   return YES
   else:
   return NO

Напомним, что проверка принадлежности любого слова языку L осуществляется за полиномиальное время по условию. Количество таких проверок в алгоритме не превосходит n^2. Поэтому сам алгоритм будет тоже работать за полиномиальное время.

Докажем его корректность. Докажем следующее утверждение по индукции: *после k-ой итерации цикла по j во множестве V содержатся длины всех префиксов x длины не более k, которые можно покрыть подсловами из L.*

База: если $x = \varepsilon$, то алгоритм работает верно, так как $0 \in V$, если $|x| \geq 1$, то на первом шаге проверяется принадлежность $x_1 \in L$. Предположение: допустим утверждение верно до k-го шага включительно.

Переход: пусть можно покрыть префикс длины $k+1$ на $(k+1)$-ом шаге, тогда существует его разбиение на подслова $v_1, \dots, v_{s-1}, v_s \in L,\ v_s \neq \varepsilon$. Первые $s-1$ подслов покрывают префикс длины $l \leq k$ и, по предположению, $l \in V$. Тогда $v_s = x[l+1, k+1] \in L$, и алгоритм проверит это в цикле по i. Следовательно, $k+1$ будет добавлено в V.

Если же префикс длины $k+1$ покрыть нельзя, то не существует такого разбиения, и все проверки в цикле по i будут неуспешными. Таким образом, $k+1$ не будет добавлено в V.

b)

$$L \in \mathcal{NP} \qquad \Longrightarrow \qquad L^* \in \mathcal{NP}$$

$L \in \mathcal{NP} \Rightarrow x \in L$ тогда и только тогда, когда существует сертификат s такой, что проверка осуществляется за полиномиальное время.

Всего число подслов слова x длины n не превосходит n^2, поэтому в качестве сертификата можно дать n^2 сертификатов (подсказок) для проверки каждого из подслов и само разбиение x на нужные подслова. Длина сертификата, ясно, останется полиномиальной, и верификатор, проверяющий принадлежность подслов языку L, тоже будет работать за полиномиальное время.

Задача 5

Покажите, что классу $\mathcal{NP}$ принадлежит язык несовместных систем линейных уравнений с целыми коэффициентами от 2018 неизвестных, и постройте соответствующий сертификат y и проверочный предикат $R(x, y)$.

Решение:

По теореме Кронекера-Капелли, система линейные уравнений несовместна тогда и только тогда, когда ранг матрицы системы меньше ранга расширенной матрицы системы. Построим сертификат и верификатор.

В качестве сертификата возьмем следующие величины:

- Номера столбцов и строк матрицы системы (базисных строк), пересечение которых образует невырожденную подматрицу.
- Коэффициенты разложения остальных строк матрицы системы по базисным строкам.
- Номера столбцов и строк расширенной матрицы системы, пересечение которых образует невырожденную подматрицу.

Опишем действия верификатора (проверочного предиката):

1. Убедиться, что число переменных (столбцов матрицы системы) равно 2018.
2. Посчитать детерминант подматрицы (порядка m) матрицы системы. Так мы докажем, что ранг матрицы системы не меньше m.
3. Убедиться, что остальные строки раскладываются по базисным с коэффициентами из сертификата. Так мы докажем, что ранг матрицы системы не превосходит m.
4. Посчитать детерминант подматрицы (порядка $m+1$) расширенной матрицы системы. Так мы покажем, что ранг расширенной матрицы больше m. Это и будет означать, что система несовместна.

Осталось показать, что все действия занимают полиномиальное время и длина записи сертификата тоже полиномиальна.

Для подсчета детерминанта существуют полиномиальные алгоритмы (об этом Тарасов говорил на лекции) за, например, $O(n^4)$. Но числа в промежуточных вычислениях могут расти очень быстро, но можно показать, что их рост (а точнее длины их двоичной записи) не быстрее полиномиального. Оценим сверху значение детерминанта (при условии, что все числа матрицы не превосходят по модулю число h):

$$\Delta \leq n! \cdot h^n \qquad \Longrightarrow \qquad L \leq \log(n! h^n) = \Theta(n \log n + n \log h)$$

Длина двоичной записи полиномиальна по n и $\log h$, поэтому в промежуточных вычислениях присутствуют числа такого порядка.

Теперь покажем, что длина записи коэффициентов разложения строк матрицы системы по базисным полиномиальна. Коэффициенты этого разложения — это решение системы m уравнений с m неизвестными. Из метода Крамера следует, что решение системы есть отношение двух определителей. А так как мы показали, что длина записи определителей полиномиальна, то и длина записей коэффициентов тоже полиномиальна.

Задача 6

Покажите, что язык разложения на множители

$$L_{factor} = \{(N, M) \in \mathbb{Z}^2 \mid 1 < M < N \text{ и } N \text{ имеет делитель } d, 1 < d \leq M\}$$

лежит в пересечении $\mathcal{NP} \cap co - \mathcal{NP}$.

Решение:

а) $L_{factor} \in \mathcal{NP}$

Сертификат — делитель d числа N, который не превосходит M. Верификатор — проверка делимости $d|N$ и сравнение d с M.

b) $L_{factor} \in co-\mathcal{NP} \iff \overline{L_{factor}} \in \mathcal{NP}$

Сертификат — разложение числа N на простые множители. Его запись полиномиальна.

Верификатор:

1. Проверить, выполняются ли неравенства $1 < M < N$, и если нет, то сразу сказать, что вход не лежит в L_{factor}, то есть лежит в $\overline{L_{factor}}$.
2. Перемножить все простые числа из сертификата и убедиться, что это действительно разложение N на множители.
3. Проверить все числа из сертификата на простоту. На семинаре мы обсуждали, что это процедура полиномиальна.
4. Найти из всех простых делителей минимальный и сравнить его с M.

Задача 7

Язык HP состоит из описаний графов, имеющих гамильтонов путь. Язык HC состоит из описаний графов, имеющих гамильтонов цикл (проходящий через все вершины, причем все вершины в этом цикле, кроме первой и последней, попарно различны). Постройте явные полиномиальные сводимости HC к HP и HP к HC.

Решение:

a) $HP \leq_p HC$

Лемма 1. *Если в граф добавить новую вершину и соединить ее ребрами со всеми остальными, то в новом графе будет гамильтонов цикл тогда и только тогда, когда в старом графе был гамильтонов путь.*

Доказательство:
Пусть в графе был гамильтонов путь. Если u и v — начальная и конечная вершины этого гамильтонова пути, а s — новая вершина, то в новом графе гамильтонов путь

$$s \to u \to \ldots \to v \to s$$

будет также гамильтоновым циклом, так как его начальная и конечная вершины совпадают.

Допустим, в новом графе есть гамильтонов цикл, а в старом его не было. s — новая вершина, u, v — соседи s в гамильтоновом цикле. Ясно, что часть гамильтонова цикла, не содержащая вершину s, проходит через все остальные вершины графа (это определение гамильтонового цикла) и не проходит через s. Поэтому если удалить s и все исходящие из нее ребра, то в графе останется гамильтонов путь — противоречие. □

Из этой леммы ясно, как можно построить полиномиальную сводимость. Сводящая функция f будет добавлять в граф еще одну вершину и соединять ее со всеми остальными.

b) $HC \leq_p HP$

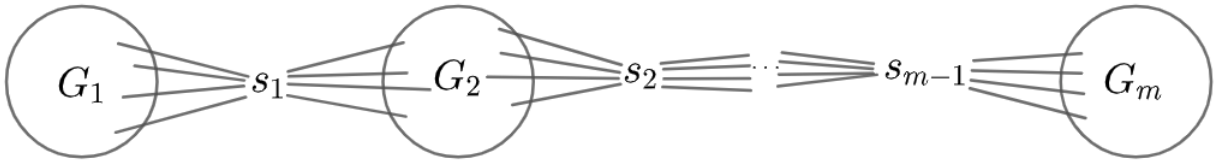
Лемма 2. *Пусть в графе G есть гамильтонов цикл. Тогда для любого ребра $e \in G$ выполняется, что в графе без этого ребра $G \backslash e$ есть гамильтонов путь.*

Доказательство:
Если ребро e не входит в гамильтонов цикл, то цикл останется, а, следовательно, останется и гамильтонов путь. Пусть e — часть гамильтонова цикла. Но тогда концы ребра e будут началом и концом гамильтонова пути в графе. □

Лемма 3. *Пусть в графе G есть гамильтонов путь, но нет гамильтонова цикла. Тогда существует ребро $e \in G$ такое, что в графе $G \backslash e$ нет гамильтонова пути.*

Доказательство:
Допустим противное: Если в графе есть гамильтонов путь, но не цикл, то какое-бы ребро мы ни удалили, в графе все равно останется гамильтонов путь. При удалении ребра гамильтонов цикл появится никак не может, поэтому условие предположения снова выполняется. Таким образом, мы можем удалить все ребра из графа, что является противоречием. □

Построим следующую полиномиальную сводимость. Пусть в графе G есть m ребер. Построим m новых графов $G_1, \ldots, G_m$, в каждом из которых отсутствует одно из ребер исходного графа. Из них составим один большой граф G'. Добавим $m-1$ новую вершину $s_1, \ldots s_{m-1}$. Добавим ребра так, что вершина s_k соединена со всеми вершинами графов G_k и G_{k+1}, $k = \overline{1, m-1}$. Это и будет граф G'.



Пусть в исходном графе G был гамильтонов цикл. Тогда, по лемме 2, в каждом из графов G_k будет гамильтонов путь с концами $u_k \to \ldots \to v_k$. Тогда в графе G' есть гамильтонов путь:

$$u_1 \to \ldots \to v_1 \to s_1 \to u_2 \to \ldots \to v_{k-1} \to s_{m-1} \to u_m \to \ldots \to v_m$$

Допустим, в G не было гамильтонова цикла. Даже если в нем был гамильтоном путь, то, по лемме 3, хотя бы в одном из графов G_k нет гамильтонова пути. Но тогда и в G' нет гамильтонова пути. Действительно, если бы но был, то чтобы попасть из вершины s_k в s_{k+1} надо пройти через все вершины графа G_k по одному разу. Аналогичные рассуждения применимы, если $k = 1$ или $k = m$, то есть такой граф с краю.

Итак, мы построили такое преобразование f, что $G' = f(G)$, и в G' есть гамильтонов путь тогда и только тогда, когда в G есть гамильтонов цикл.

Задача 8

Регулярный язык L задан регулярным выражением. Постройте полиномиальный алгоритм проверки непринадлежности $w \notin L$. Вы должны определить, что вы понимаете под длиной входа, и выписать явную оценку трудоёмкости алгоритма.

Решение:

Вход алгоритма: регулярное выражение длины n (число символов, включая скобки, литералы, и т.п.) и слово w длины. Построим алгоритм проверки принадлежности, а в конце инвертируем ответ.

1. Строим по регулярному выражению КС-грамматику. Так как на каждую операцию $(\bullet, |, *)$ требуется константное число правил, то в ней будет $O(n)$ правил. Длина всех правых частей тоже $O(n)$, потому что в правые части записывается либо константное число символов, либо какая-то подцепочка литералов РВ, которая не встречается в других правых частях.

2. Приводим эту КС-грамматику к нормальной форме Хомского. Это было у нас в ТРЯПе. Сначала удаляем длинные правила за $O(n)$. Потом все ε-правила за $O(n)$. Затем удаляем все цепные правила за $O(n^2)$. Затем удаляем правила, в которых 2 терминала или 1 терминал и 1 нетерминал справа за $O(n)$.

3. Применяем алгоритм Кока-Янгера-Касами к грамматике в нормальной форме Хомского. Из ТРЯПа мы знаем, что он работает за $O(|w|^3 n)$.

Итоговая асимптотика:

$$T(n, |w|) = O(|w|^3 n + n^2)$$

Есть и более простое решение: построить НКА по РВ и ДКА, принимающий только слово w. Затем построить их пересечение и проверить достижимость финального состояния.

Задача 9

(i) Языки $L_1, L_2, \ldots, L_{2019}$ заданы регулярными выражениями. Постройте полиномиальный алгоритм, проверяющий, что их пересечение не пусто, т. е. $\bigcap\limits_{i=1}^{2019} L_i \neq \varnothing$.

(ii) Следует ли из решения предыдущей задачи, что проверка непустоты пересечения конечного семейства ДКА (в фиксированным алфавите, например, унарном) принадлежит классу $\mathcal{P}$?
Является ли эта задача разрешимой?

Решение:

(i) Считаем, что $K = 2019$ — фиксированная константа. Пусть n — максимальная длина РВ.

Приведем алгоритм:

1. Для каждого из языков L_k по регулярному выражению строим НКА. Это занимает линейное время $O(n)$. Такие НКА будут иметь $O(n)$ состояний и $O(n)$ переходов (используем алгоритм построения НКА по РВ).
2. Используем конструкцию пересечения для построения нового НКА. В нем будет $O(n^K)$ состояний и $O(n^K)$ переходов, то есть полиномиальное количество.
3. Чтобы пересечение языков было непусто, необходимо и достаточно, чтобы в новом НКА было достижимо какое-то финальное состояние. Для этого надо сделать обход графа, которым является НКА, например, с помощью обхода в глубину, который делается за время линейное по числу состояний и переходов.

Все действия полиномиальны при фиксированном K.

(ii) Нет, из решения предыдущей задачи этого не следует. Дело в том, что там число $K = 2019$ было фиксировано и не являлось частью входа. Здесь же, нам н вход подается k автоматов, длина описания каждого из которых не превосходит n. Тот же алгоритм, как в предыдущем пункте, дает асимптотику $O(n^k)$, что растет быстрее, чем любой полином от длины входа $n \cdot k$.

Тем не менее, алгоритм приводит к решению, хотя работает за время хуже полиномиального. Таким образом, задача разрешима.