Николай Комаров ДЗ №3

Задача 1

Докажите, что из L, M ∈ P следуют $\overline{\mathrm{L}}$, L ∩ M, L ∪ M ∈ P (здесь $\overline{\mathrm{L}} = \Sigma^* \setminus \mathrm{L}$), а также L · M ∈ P, где L · M = {uv | u ∈ L, v ∈ M} есть множество всевозможных конкатенаций двух слов из L и M соответственно.

Доказательство

1. $\overline{\mathbf{L}} \in \mathbf{P}$

□ Если $L \in P$, значит существует 0-1 poly м.т. K: $\begin{cases} K(\vec{x})=1, \vec{x} \in L \\ K(\vec{x})=0, \vec{x} \notin L \end{cases}$

Значит, так же можно построить такую poly м.т. K', которая будет выдавать противополоные K результаты (можно взять машину K и добавить к ней операцию инвертирования ответа, сложность которой $O(1)$, значит сама машина останется poly).

Тогда имеем K': $\begin{cases} K'(\vec{x})=0, \vec{x} \in L \Longleftrightarrow \vec{x} \notin \overline{L} \\ K'(\vec{x})=1, \vec{x} \notin L \Longleftrightarrow \vec{x} \in \overline{L} \end{cases} \Rightarrow \begin{cases} K'(\vec{x})=1, \vec{x} \in \overline{L} \\ K'(\vec{x})=0, \vec{x} \notin \overline{L} \end{cases}$

Таким образом имеем полиномиальную м.т. K', распознающую язык $\overline{L}$, значит $\overline{L} \in P$.

■

2. L ∩ M ∈ P

□ Если $L, M \in P$, значит существует 0-1 poly м.т. K: $\begin{cases} K(\vec{x})=1, \vec{x} \in L \\ K(\vec{x})=0, \vec{x} \notin L \end{cases}$ и Y: $\begin{cases} Y(\vec{x})=1, \vec{x} \in M \\ Y(\vec{x})=0, \vec{x} \notin M \end{cases}$

Соответственно, если слово $\vec{y} \in L \cap M$, то оно может быть распознано любой из приведенных машин, работающих за полиномиальное время, то есть существует poly м.т. (например K), распознающая все слова из $L \cap M$ за полином $\Rightarrow L \cap M \in P$.

■

3. L ∪ M ∈ P

□ Если $L, M \in P$, значит существует 0-1 poly м.т. K: $\begin{cases} K(\vec{x})=1, \vec{x} \in L \\ K(\vec{x})=0, \vec{x} \notin L \end{cases}$ и Y: $\begin{cases} Y(\vec{x})=1, \vec{x} \in M \\ Y(\vec{x})=0, \vec{x} \notin M \end{cases}$

Соответственно, для распознания слова $\vec{y} \in L \cup M$, можно сконструировать машину Z, которая будем представлять собой последовательно соединенные машины K и Y через логическое сложение: сначала отрабатывает программа машины K, записывает результат в выходную ленту, если результат 1, то завершаемся, если 0, то затем отрабатывает программа машины Y, и прибавляет свой результат к уже записанному результату алгоритма K, обе машины работают за полином, операция прибавления однобитовых чисел в данном случае $O(1)$, так как есть всего два варианта 0 + 0 и 0 + 1, для обоих нужно одно элементарное действие, получаем, что итоговая сложность составной машины Z тоже полином.

Таким образом, построили 0-1 poly м.т. Z: $\begin{cases} Z(\vec{x})=1, \vec{x} \in L \cup M \\ Z(\vec{x})=0, \vec{x} \notin L \cup M \end{cases} \Rightarrow L \cup M \in P$

■

4. L · M ∈ P, где L · M = {uv | u ∈ L, v ∈ M}

□ Если $L, M \in P$, значит существует 0-1 poly м.т. K: $\begin{cases} K(\vec{x})=1, \vec{x} \in L \\ K(\vec{x})=0, \vec{x} \notin L \end{cases}$ и Y: $\begin{cases} Y(\vec{x})=1, \vec{x} \in M \\ Y(\vec{x})=0, \vec{x} \notin M \end{cases}$

Для распознавания конкатенации слов из L и M можно воспользоваться следующим алгоритмом: мы знаем, что первая часть слова у нас из L, но не знаем какой длины, тогда мы можем сделать следующее - идти по слову и на каждом шаге запускать алгоритм машины Y, распознающий слово из M, если не распозналось, то сдвигаем головку вперед и повторяем, если слово распозналось, значит мы нашли "стык" двух слов, далее можно заменить уже распознанную часть слова на $\#$ и запустить алгоритм машины K, по результату распознавания последнего уже можно судить о нахождении слова в $L \cdot M$. Условие выхода при неудачном распозновании следующее, если в ходе прохода алгоритмом Y по слову мы дошли до конца, так ничего и не распознав, значит нет суффикса слова $\in M$, значит слово точно $\notin L \cdot M$.

В описанном алгоритме мы в худшем случае запускаем poly алгоритм машины Y n раз, где n – длина входа, таким образом увеличиваем стпень полинома на 1, но он все еще полином, далее, в случае упеха алгоритма Y, отработает еще poly аглоритм K, итого в конечном счете имеем все еще полином.

Таком образом, мы описали 0-1 м.т. распознающую слова из $L \cdot M$ за полиномиальное время, значит $L \cdot M \in P$.

■

Задача 2

Докажите, что задача распознавания наличия треугольника (т. е. подграфа, изоморфного K_3) в графе лежит в P. Граф задан матрицей смежности.

Доказательство

□ Обозначим матрицу смежности за A. Тогда чтобы понять есть ли треугольник (цикл длины 3) в графе, достаточно посмотреть на след матрицы A^3. Перемножение матриц занимает полиномиальное время (например, наивный алгоритм - $O(n^3)$), умножение нужно выполнить 2 раза, подсчет следа итоговой матрицы - $O(n)$, итого получаем, что общая сложность алгоритма проверки наличия треугольника - $O(n^3)$.

По тезису Чёрча-Тьюринга описанный выше алгоритм может быть выполнен на м.т. не более чем с полиномиальным замедлением относительно сложности программы $O(n^3)$ в РЯП. Полином от n^3 тоже полином $\Rightarrow$ значит существует работающая за полином м.т. M, распознающая наличие треугольника в графе $\Rightarrow$ задача $\in P$.

■