Домашняя работа 3

Пасечник Даша

на 07.03.2019

Задача 1

Язык 2-COLOR состоит из кодировок всех графов, заданных матрицами смежности, вершины которых можно корректно окрасить в два цвета (никакие две смежные вершины не имеют один цвет). Верно ли, что язык 2-COLOR лежит в $\mathcal{P}$? В $\mathcal{NP}$? В $co-\mathcal{NP}$?

Граф подается в виде матрицы смежности, то есть описание графа имеет длину n^2, где n – количество вершин. Воспользуемся алгоритмом поиска в ширину, его сложность $O(V+E)$. Обозначим цвета 0 и 1.

1. Выберем произвольную нераскрашенную вершину и раскрасим ее в 1 б.о.о..

2. Переходим к смежным вершинам и, если они нераскрашены, красим в противоположный цвет. Затем запускаем пункт 2 от них. Если некоторая смежная вершина уже раскрашена в противоположный цвет (т.е. не противоречит корректности раскраски), не трогаем ее.

3. Повторяем пункт 2 пока для некоторой вершины не останется смежных нераскрашенных вершин, или не найдется смежная, раскрашенная в тот же цвет. Тогда в первом случае смотрим остались ли в графе нераскрашенные вершины: если да, значит в графе больше одной компоненты связности, тогда повторяем алгоритм с пункта 1; если нераскрашенных вершин не осталось, значит граф можно корректно окрасить в 2 цвета, что мы и сделали. Во втором случае корректность раскраски нарушилась - значит в графе существует цикл нечетной длины и граф нельзя окрасить в 2 цвета.

На каждом шаге происходит проверка цвета вершины - $O(1)$. Вершин n, рёбер не более n^2, поэтому сложность $O(n^2)$. Значит разрешающий алгоритм полиномиален, и верно, что язык 2-COLOR лежит в $\mathcal{P}$, $\mathcal{NP}$ и $co-\mathcal{NP}$.

Задача 2

Язык HP состоит из всех графов, имеющих гамильтонов путь (несамопересекающийся путь, проходящий через все вершины графа). Язык HC состоит из всех графов, имеющих гамильтонов цикл (цикл, проходящий через все вершины, в котором все вершины, кроме первой и последней, попарно различны). Постройте явные полиномиальные сводимости HC к HP и наоборот.

$HP \leqslant_p HC$

Построим функцию f такую, что $x \in HP \iff f(x) \in HC$.
Функция f будет создавать копию иходного графа G - G^* и проводить всевозможные ребра между G и G^*, затем убирать произвольное ребро из G^*, копировать G еще раз - G^{**} и проводить всевозможные ребра между G^{**} и вершинами, между которыми убрали ребро - назовем их A и B.
Пусть $G \in HP$ и C, D – начальная и конечная вершины пути соответственно. Покажем явно гамильтонов цикл в графе $f(G)$:

1. Пойдем по гамильтонову пути подграфа G графа $f(G)$ пока не окажемся в вершине D.

2. Из D перейдем в C^* – вершину подграфа G^* и пойдем по гамильтонову пути подграфа G^*.

3. Если удаленное ребро AB принадлежало гамильтонову пути, то оказавшись б.о.о. в A пройдем до B по гамильтонову пути подграфа G^{**}.

4. Оказавшись в D^* перейдем в вершину C, из которой начали путь.

Пусть теперь $G \notin HP$. Т.к. в G нет гамильтонова цикла, то его нет и в подграфах G^* и G^{**}. Тогда, если гамильтов цикл в $f(G)$ существует, то в подграф G^{**} он заходит через вершину A, а выходит через B (или наоборот), т.к. это единственные вершины связанные с данным подграфом. Но в самом подграфе G^{**} нет гамильтонова пути, значит нет и гамильтонова пути с начальной и конечной вершинами A и B в подграфе $G^{**} \cup \{A\} \cup \{B\}$, значит гамильтонова пути нет и в графе $f(G)$. Тогда в нем не может быть и гамильтонова цикла.
Очевидно, f работает полиномиально от величины вершин и ребер в исходном графе, т.к. создает $O(V)$ вершин и $O(V^2)$ ребер.
Другой способ (на всякий случай) заключается в следующем. Функция f создает новую вершину A и соединяет ее со всеми остальными вершинами. Если в графе G существовал гамильтонов путь (пусть из вершины C в D), то пройдем по нему до вершины D, из D перейдем в A, из A в C. Получили гамильтонов цикл. Пусть в G гамильтонова пути нет, тогда если в $f(G)$ есть гамильтонов цикл, то в $f(G)$ есть и гамильтонов путь с началом в A. Пусть из A он ведет в B. Тогда в изначальном графе также существовал гамильтонов путь, начинающийся в B, а далее совпадающий с гамильтоновым путем графа $f(G)$. Противоречие, значит в $f(G)$ нет гамильтонова цикла. f здесть так же очевидно полиномиальна, т.к. создает 1 вершину и проводит $O(V)$ ребер.

$HC \leqslant_p HP$

Построим функцию f такую, что $x \in HC \iff f(x) \in HP$.
Функция f выбирает произвольную вершину из G пусть A, дублирует ее - A^* и все ее ребра, затем создает 2 новые вершины C и C^* и соединяет их с A и A^* соответственно.
Пусть $G \in HC$. Покажем гамильтонов путь в $f(G)$.
Начнем из вершины C. Затем перейдем в A. Далее пройдем по гамильтонову циклу графа G, но вместо того чтобы вернуться в A на последнем шаге перейдем в $A*$, а оттуда в C^*. Итого, прошли по всем вершинам графа $f(G)$ один раз.
Пусть $G \notin HC$. Тогда если в $f(G)$ есть гамильтонов путь, то он начинается и заканчивается в вершинах C и C^*, т.к. они висячие. Тогда гамильтонов путь есть и в графе $f(G) \setminus \{C\} \setminus \{C^*\}$. Тогда в графе G есть гамильтонов цикл, т.к. последний переход в A^* можно заменить переходом в A. Противоречие. Значит в $f(G)$ гамильтонова пути нет.
Очевидно, f работает полиномиально от величины вершин и ребер в исходном графе, т.к. создает 3 вершины и $O(V)$ ребер.

Задача 4

Докажите следующие свойства полиномиальной сводимости:

(*i*) *Рефлексивность: $A \leq_p A$; транзитивность: $A \leq_p B, B \leq_p C \implies A \leq_p C$*
Рефлексивность: $A \leqslant_p A: \quad \exists f(x) = x : x \in A \iff f(x) \in A$.
Транзитивность: пусть f сводит A к B, а g сводит B к C. Тогда $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow g(f(x)) \in C$, т.е. $A \leqslant_p C$, т.к. $f \circ g$ вычисляется за полиномиальное время (функции g и f вычисляются за полиномиальное время).

(*ii*) *Если $B \in \mathcal{P}$ и $A \leq_p B$, то $A \in \mathcal{P}$*
Пусть χ_B - характеристическая функции B, f сводит A к B. Тогда рассмотрим $f \circ \chi_B$. Заметим, что это характеристическая функция для A: $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow \chi_B(f(x)) = 1$.
Т.к. χ_B полиномиальна и f полиномиальна, то их композиция - тоже. Значит $A \in \mathcal{P}$.

(*iii*) *Если $B \in \mathcal{NP}$ и $A \leq_p B$, то $A \in \mathcal{NP}$.*
χ_B вычисляется за полиномиальное время на недетерминированной МТ, f - на детерминированной МТ. Значит $\chi_A = \chi_B \circ f$ вычисляется на недетерминированной МТ за полиномиальное время. Значит $A \in \mathcal{NP}$.

Задача 5

Докажите, что классы $\mathcal{P}$ и $\mathcal{NP}$ замкнуты относительно операции $*$ — звезды Клини. Приведите также и сертификат принадлежности слова языку L^*, где $L \in \mathcal{NP}$.

$\mathcal{P}$

Пусть есть χ_L - характеристическая функция для L, построим алгоритм, вычисляющий χ_{L^*}.
На вход χ_{L^*} поступает слово ω, длиной n. Создадим множество END, куда будем класть индексы i такие, что

$\omega[j : i+1]$ для некоторого j – слово из L. Изначально $END = \{0\}$. Пойдем циклом по длине ω (i - индекс рассматриваемого элемента) и для каждого элемента j из END будем проверять принадлежность $\omega' = \omega[j+1 : i+1]$ языку L, запуская χ_L на ω'. Если $\chi_L(\omega') = 1$ и $i = n$, значит мы поделили все слово ω на подслова из L, а значит $\omega \in L^*$. Выводим 1. Если $i \neq n$, но $\chi_L(\omega') = 1$, то добавляем i в END и переходим на следующую итерацию. Если цикл пройден до конца слова, но на последней итерации $\omega[j+1 : n+1] \notin L$, то слово не разбивается на подслова из L, значит не принадлежит L^*. Выводим 0.

```
for i in range(n):
    for j in END:
        if (chi_L(w[j+1:i+1])):
            if (i == n):
                return true
            END.add(i)
    return false
```

В множестве END не может быть более n элементов, значит время работы χ_{L^*} оценивается $n^2 O(\chi_L)$ - т.е. полиномиально. Значит $L^* \in \mathcal{P}$.

$\mathcal{NP}$

Пусть $R_L(x, y)$ - предикат для L. Тогда построим R_{L^*} следующим образом. Пользуемся идеей, что слово принадлежит L^*, если может быть разбито на подслова из L. Для слова ω качестве посказки будем подавать разбиение слова на предполагаемые подслова из L (например, индексами начал слов) и подсказки для каждого из подслов. R_{L^*} будет запускать R_L на каждом подслове и переданной для него подсказке и выдавать 1, когда все вызовы R_L вернут 1. Длина подслов и их количество ограничено $O(n)$, значит подсказка полиномиальна от длины слова. R_L вычисляется полиномиально и запускается R_{L^*} не более n раз, значит R_{L^*} также полиномиальна. Т.о. предикат для L^* построен корректно и полиномиален, значит $L^* \in \mathcal{NP}$.