Домашнее задание на третью неделю

Бельденова Камила, 675

28 февраля 2018

1	2	4	5	6	7	8	9	Σ

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) Рефлексивность довольно очевидна. Для этого нам достаточно рассмотреть функцию $f(x) = x$ (вычислимая за полиномиальное время функция). Видно, что $x \in A \Leftrightarrow f(x) = x \in A$, а значит согласно определению $A \leq_p A$.

Теперь докажем транзитивность. Если f сводит A к B, а g сводит B к C, то $f \circ g$ сводит A к C : $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow g(f(x)) \in C$. При этом если и f, и g вычисляются за полиномиальное время, то $f \circ g$ также вычисляется за полиномиальное время.

(ii) Характеристическая функция χ_A представляется как композиция $\chi_B \circ f$. Если $B \in P$, то χ_B вычисляется за полиномиальное время, а если $A \leq_p B$, то соответсвующая f также вычисляется за полиномиальное время. Это значит, что их композиция (т. е. χ_A) тоже вычисляется за полиномиальное время $\Leftrightarrow A \in \mathbf{P}$.

(iii) Если χ_B вычисляется недетерминированной машиной за полиномиальное время, а f (детерминированно) вычисляется за полиномиальное время, то $\chi_A = \chi_B \circ f$ тоже вычисляется недетерминированной машиной за полиномиальное время.

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.

(iv) Язык $L(a, m)$, зависящий от параметров $a, m \in \mathbb{N}$, определяемый следующим образом:

$$L = \{x_0, x_1, \ldots\}; x_0 = a (mod\ m); x_{i+1} = x_i^{2018} (mod\ m).$$

Решение

(i) Вспомним определение двудольного графа. Двудольный граф — граф, вершины которого можно разбить на два множества так, что каждое ребро соединяет вершины из разных множеств. Согласно определению в двудольных графах не может быть трех попарно смежных вершин, а значит множество графов, имеющих не менее 2018 треугольников, пусто $\Rightarrow$ язык принадлежит классу $\mathcal{P}$.

(ii) Проверим граф на связность с помощью поиска в глубину, сложность работы составляет $O(M)$, то есть линейна по количеству ребер. Произведём серию поисков в глубину в графе, т. е. из каждой вершины, в которую мы ещё ни разу не приходили, запустим поиск в глубину, который при входе в вершину будет красить её в серый цвет, а при выходе — в чёрный. Если после прохода алгоритма не все вершины окрашены в какой-то из цветов, то граф несвязный, значит можно продолжать его исследовать, иначе — он не принадлежит языку. Для каждого связного куска проверим граф на ацикличность, это также проверяется с помощью поиска в глубину. Если в течение поиска в глубину мы пытаемся пойти в серую вершину, то это означает, что мы нашли цикл (если граф неориентированный, то случаи, когда в течение поиска в глубину из какой-то вершины мы пытаемся пойти в предка, не считаются), и граф не принадлежит нашему языку. Язык принадлежит классу $\mathcal{P}$.

(iii) Будем проверять матрицу "лобовым"алгоритмом:

ЦИКЛ i,$j = 1..2018$
    start:
    count= 0;
    ЦИКЛ k,$l = 1..(n - 2018)$
    ЕСЛИ $(A[k + i,l + j] == 1)$
    count $+ = 1$
    ИНАЧЕ GOTO(start)
ЕСЛИ count == (n-2018)*(n-2018)
    ОТВЕТ(Принадлежит языку)
ИНАЧЕ
  ОТВЕТ(Не принадлежит языку)

Как видим, алгоритм содержит 2 цикла, работающих в худшем случае по $(n - 2018)$ раз, внутри которых происходят операции $O(1)$, значит, итоговая сложность алгоритма — $O(n^2)$.

Задача 4. Докажите, что классы $\mathcal{P}$ и $\mathcal{NP}$ замкнуты относительно операции $*$ — звезды Клини (была в ТРЯПе). Для языка NP приведите также и сертификат принадлежности слова из Σ^* языку L^*, где $L \in \mathcal{NP}$.

Решение

(1) Сначала докажем замкнутость класса $\mathcal{P}$ относительно заданной операции:

Пусть p — разрешитель языка L, работающий за полиномиальное время. Построим разрешитель q для языка L^* :

$n = |w|$;
endPoses = $\{0\}$ — места, где могут заканчиваться слова из L
for(i=1..n)
    for ($j \in$ endPoses)
        if(p(w[j+1..i])) {

```
        if (i = n)
            return true;
        endPoses ∪ = {i}
    }
return false;
```

Худшая оценка времени работы разрешителя q равна $n^2O(p(w))$. Т. к. в множестве endPoses может быть максимум n элементов, значит, итерироваться по множеству можно за $O(n)$, если реализовать его на основе битового массива, например. При этом операция добавления элемента в множество будет работать за $O(1)$. Итого, разрешитель q работает за полиномиальное время (т. к. произведение полиномов есть полином). Значит, $L^* \in P$.

(2) Теперь докажем замкнутость класса $\mathcal{NP}$ относительно заданной операции:

Предъявим предикат $M(x,y)$, который проверяет правильность сертификата y для языка L^*.
Сертификатом y^* для слов из L^* будут:
(i) позиции, на которых находятся начала слов, принадлежащих L;
(ii) сертификаты y для каждого из этих слов.

Далее нужно проверить, что каждая часть входного слова принадлежит языку L, это делается предикатом $R(x_i,y_i)$, что является $\mathcal{NP}$ задачей (по условию); проход по всем подсловам — линейная операция от длинны входа $\Rightarrow$ полученный предикат — $\mathcal{NP}$.

Задача 5. Покажите, что классу NP принадлежит язык несовместных систем линейных уравнений с целыми коэффициентами от 2018 неизвестных, и постройте соответствующий сертификат y и проверочный предикат $R(x, y)$. Один из возможных способов заключается в применении теоремы Фредгольма.

Решение

Вспомним, что критерием совместности СЛАУ является равенство рангов расширенной и исходной матриц.
Сертификатом будет число линейно независимых (далее ЛНЗ) строк в обоих матрицах, их номера и разложения по ним остальных строк.
Предикатом $R(x,y)$ будет проверка на то, что данные в сертификате строки действительно являются ЛНЗ, приведение матрицы к верхнетреугольному виду и проверка на наличие "0"на главной диагонали. Если сертификат верен, то нужно сравнить количество ЛНЗ строк в исходной и расширенной матрицах. Если оно не одинаково, то система несовместна.

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

Покажем, что данный язык $\mathcal{L} \in \mathcal{P} \Rightarrow \mathcal{L} \in \mathcal{NP} \cap co - \mathcal{NP}$.
Алгоритм проверки принадлежности слова к L_{factor} такой:

```
for(i = 2; i < M; i++){
   if(N  mod i == 0){
      printf("Слово принадлежит L_factor");
```

```
        return OK;
    }
}
printf("Слово не принадлежит L_factor");
return BAD;
```

Задача 7. Язык ГП состоит из описаний графов, имеющих гамильтонов путь. Язык ГЦ 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 ГЦ к ГП и ГП к ГЦ.

Решение

Вспомним определение полиномиальной сводимости: $L_1 \leq_p L_2$, если существует всюду определенная функция $f : \{0,1\}^* \to \{0,1\}^*$, вычислимая за полиномиальное время, такая что $x \in L_1 \Leftrightarrow f(x) \in L_2$

Рассмотрим сводимость ГЦ в ГП:
Пусть нам задан граф x, в котором есть гамильтонов цикл, тогда найдем этот цикл и построим граф $f(x)$ отбрасыванием одного ребра, учавствующего в построении цикла; таким образом получим граф, в котором будет самонеперсекающийся путь, проходящий через все вершины. Алгоритм нахождения гамильтонова цикла принадлежит $\mathcal{NP}$ (сертификатом будет последовательность вершин, образующая гамильтонов цикл, проверка линейна).

В обратную сторону — сводимость ГП к ГЦ:
Аналогично, по заданному графу найдем в нем гамильтонов путь ($\mathcal{NP}$ задача), далее построим $f(x)$ добавлением к исходному графу одного ребра, соединяющего начальную вершину пути с конечной.
Т. к. сводимость ГП к ГЦ — обратное сведение от ГЦ к ГП, то приведенное преобразование удовлетворяет определению сводимости.

Задача 8. Рассмотрим n точек плоскости, заданных своими парами декартовых координат (x, y). Требуется найти их выпуклую оболочку, т. е. наименьшее по включению выпуклое множество, содержащее все эти n точек. Выпуклой оболочкой будет некоторый многоугольник, причем все его вершины — некоторые из этих точек, а остальные лежат внутри. Вывести на экран нужно вершины этого многоугольника по порядку обхода периметра (начиная с любой из них).

Рассмотрим модель вычисления, в которой за 1 такт можно делать одну из трёх операций: сравнивать два числа, складывать числа и возводить число в квадрат. Покажите, что в этой модели вычислений задача сортировки массива сводится к задаче построения выпуклой оболочки n точек плоскости за линейное время.

Решение

Мы рассмотрим метод Грэхема с улучшениями Эндрю. С его помощью можно построить выпуклую оболочку за время $O(n \log n)$ с использованием только операций сравнения, сложения и умножения. Алгоритм является асимптотически оптимальным (доказано, что не существует алгоритма с лучшей асимптотикой).

Сам алгоритм:
Найдём A и B — самую левую и самую правую точки (если таких точек несколько, то возьмём самую нижнюю среди левых, и самую верхнюю среди правых). Понятно, что и A, и B обязательно попадут в выпуклую оболочку. Дальше, проведём через эти точки прямую AB,тем самым разделив множество всех точек на верхнее и нижнее подмножества S_1 и S_2 (точки, лежащие на прямой, можно отнести к любому

из подмножеств — они всё равно не войдут в оболочку). Точки A и B отнесём к обоим множествам. Теперь построим для S_1 верхнюю оболочку, а для S_2 — нижнюю, и объединив их, получим ответ. Чтобы получить, например, верхнюю оболочку, нужно отсортировать все точки по абсциссе, потом пройтись по всем точкам, рассматривая на каждом шаге кроме самой точки две предыдущие, вошедшие в оболочку. Если текущая тройка точек образует не правый поворот (это легко проверить с помощью понятия "ориентированной площади"), то ближайшего соседа нужно удалить из оболочки. В итоге, останутся только точки, входящие в выпуклую оболочку.
Итак, алгоритм состоит в **сортировке всех точек** по абсциссе и двух (в худшем случае) обходах всех точек, т. е. требуемая асимптотика $O(n \log n)$ достигнута.

Но (по крайней мере, я так поняла), так как все операции, на которые опирается наш алгоритм (например, поиск самой левой и самой правой точек в начале представленного алгоритма), по условию занимают один такт для каждого элемента, то всего получаем $O(n)$ в нашей модели вычисления. Получили,что в этой модели вычислений задача сортировки массива свелась к задаче построения выпуклой оболочки n точек плоскости за линейное время.

Задача 9. Пусть $A \in \mathcal{NP} - complete$. Пусть машина имеет дополнительную функцию (оракул) за 1 такт получать ответ, лежит ли слово x в языке A. Тогда существует полиномиальный алгоритм, решающий задачу поиска для A. Докажите это утверждение.

Решение

Есть такая лемма:
Пусть языки A и B лежат в $\mathcal{NP}$ и задаются верификаторами W_A и W_B соответсвенно. Пусть также A сводится по Карпу к B. Тогда существуют верификаторы V_A и V_B, для которых также имеет место сводимость по Левину.

Теперь приступим к доказательству утверждения.

В силу указанной леммы можно считать, что и A сводится к $3SAT$, и $3SAT$ сводится к A в смысле Левина. В большинстве случаев (в том числе в теореме Кука–Левина) это получается и без дополнительных конструкций. Поэтому достаточно свести A к $3SAT$, решить задачу поиска для $3SAT$ и свести результат обратно.
Пусть дана формула φ, зависящая от переменных $p_1, \dots p_n$. Вначале можно проверить выполнимость φ (при помощи оракула), и если формула невыполнима, такой ответ и вернуть. Если же формула выполнима, рассмотрим формулы φ_1' и φ_1'', полученные из φ фиксацией значений $p_1 = 0$ и $p_1 = 1$ соответственно. Хотя бы одна из них будет выполнима, какая именно, можно найти при помощи двух запросов к оракулу (или даже одного). Обозначим выполнимую формулу через φ_1 и применим к ней ту же процедуру рекурсивно. Так будут определяться значения всех переменных, пока не останется только одна. Это будет база рекурсии: значение последней переменной можно найти непосредственной подстановкой обоих вариантов.