Домашняя работа №5

Бредихин Александр

24 марта 2020 г.

Задача 1

Докажите, что язык палиндромов лежит в **L**.

Построим алгоритм, который определяет является ли слово палиндромом или нет (пусть на вход подаётся слово s):

1) Определяем длину s, n и записываем в ячейку памяти $\left[\frac{n}{2}\right]$.

2) Создаём в памяти ячейку с переменной i, которая будет принимать значения от 0 до $\left[\frac{n}{2}\right]$. Превоначально $i = 0$, а указатель стоит на центральном элементе слова: $\left[\frac{n}{2}\right]$. Также создаём две переменные: $char_1$ и $char_2$ и переменную $flag = 1$, которая будет обозначать является ли слово палиндромом или нет.

3) Записываем в созданные переменные: $char_1 = s[i]$ и $char_2 = s[n - i - 1]$. Сравниваем их, если они совпадают, то увеличиваем значение i на единицу, если нет, то $flag = 0$ и заканчиваем алгоритм.

4) Если $i = \left[\frac{n}{2}\right]$ и $flag = 1$, то завершаем алгоритм и выводим, что слово является палиндромом, если $flag = 0$ - не является. Иначе начинаем выполнять с 3го пункта.

Алгоритм работает корректно, так как проверяет по определению является ли слово палиндромом или нет. Используемая дополнительная память: для записи длины слова $\log(n)$ ещё создаём 4 переменные размер которых не превашает этого, поэтому требуемая память $\mathbf{O}(\log(n))$, следовательно, язык лежит в **L**.

Задача 2

Докажите, что язык правильных скобочных выражений из двух типов скобок лежит **L**.

Для того, чтобы проверить является ли последовательность из двух видов скобок правильной или нет, введём переменную S и идя циклом по входной строчке будем прибавлять к ней 1, когда встречаем открывающую скобку и отнимать 1, если встречаем закрывающуюся.
Если S всегда будет ≥ 0 и в конце будет равняться 0, то это правильная скобочная последовательность, так как количество открывающихся и закрывающихся скобок одинаковое число (в конце $S = 0$) и закрывающиеся скобки всегда идут после открывающихся (на протяжении всего цикла $S \geq 0$).
Получается нам потребовалась 1 ячейка памяти, которая занимает $\log(\log n)$, следовательно, по определению этот язык лежит в **L**.

Это решение не сработает для скобок двух видов (пример: ([)] - неправильная). Построим алгоритм, который сможет распознавать такие скобочные последовательности:
сначала, проверяем не обращая внимание на тип скобок её «скобочную сумму» как написано выше, если это не выполняется, то это последовательность точно неправильная.
Если это это выполнилось, то последовательность начинается с открывающейся скобки какого-то типа. Запоминаем в переменную $type$ номер её типа и считаем скобочную сумму только данного типа с этой позиции, запоминая индекс позиции (обозначаем i). Значение суммы, храним в переменной S. Считаем S до момента, когда она не обратится в 0 (если этого не произошло, то скобочная последовательность неправильная). Запоминаем индекс скобки, где происходит обнуление - j. Затем считаем скобочную сумму аналогичным образом (учитывая тип скобки), начиная с $i + 1$, до j. Если эта скобочная сумма не равна 0, то скобочная последовательность неправильная, если это выполнено, то увеличиваем i до первой открывающейся скобки и делаем весь алгоритм сначала.

Если i дошла до конца, то это правильная скобочная последовательность из скобок двух видов. Используем константное количество переменных, размер которых не превышает $\log(n)$, где n - длина входа. Следовательно по памяти алгоритм работает за $\mathcal{O}(\log n) \longrightarrow$ лежит в **L**.

Задача 3

Докажите, что композиция функций, вычислимых на логарифмической памяти, тоже функция, вычислимая на логарифмической памяти.

Хотим показать, что $g(f(x))$ вычисляется на логарифмической памяти, если известно, что f и g вычисляются на логарифмической памяти. Воспользуемся свойством, что если f вычислима на логарифмической памяти, то область её опредеделения и значений лежат в $\mathbf{L}$. Тогда запустим вычисление g на входе f.
Заведём два счётчика: для значения элемента и для индекса этого элемента. Если машина, вычисляющая g перемещается, то мы соответствующим образом смещаем счётчик индекса элемента и вычисляем новое значение (так как множества определения и значений разрешимы). Счётчик займёт логарифмическую память, так как длина f – полиномиальна. В итоге используем два дополнительных счётчика, в которые перезаписываем значения, следовательно, память остаётся логарифмической.

Задача 4

Докажите, что язык $2SAT$ **NL**-полный относительно сводимости на логарифмической памяти.

Покажем, что $2SAT \in \mathbf{NL}$. Возьмём за сертефикат список значений переменных. Предикат будет проверять истинна ли КНФ форма или нет. Вычисления проводятся на логарифмической памяти, так как вычисляем дизъюнкты по очереди, подставляя нужные значения. И отслеживаем результат с помощью созданной переменной, которая равняется 1, если дизъюнкт истенен или 0, если ложный (тогда и вся КНФ ложна). Ещё храним текущие положение, на котором находимся. В итоге дополнительно используем константное количество переменных, следовательно, вычисляется на логарифмической памяти $\longrightarrow 2SAT \in \mathbf{NL}$.

Для доказательства полноты сведём $\overline{PATH}$ к $2SAT$:
Функция сводимости f по тройке (G, s, t) строит 2КНФ следующим образом:

1) вершина графа $\Rightarrow$ переменная в 2КНФ

2) если есть ребро $u \to v$, делаем дизъюнкт $(\overline{u} \vee v)$

3) добавляем дизъюнкты s и $\overline{t}$

Пусть $x = (G, s, t) \in \overline{PATH}$, то есть из s нет пути в t. Покажем, что $f(x)$ – 2КНФ невыполним. Рассмотрим набор, в котором все переменные, достижимые из s истины, а все недостижимые – ложны. Заметим, что ребра идут между двумя истинным переменными, либо между двумя ложными, либо от ложной к истинной, следовательно, в любом случае $(\overline{u} \vee v) = 1$. Понятно, что дизъюнкты s и $\overline{t}$ тоже истенены, следовательно $f(x)$ – выполнима и $f(x) \in 2SAT$.

Обратно, пусть формула y – выполнима, покажем, что $x = (G, s, t)$: $f(x) = y$ тогда $x \in \overline{PATH}$. Если формула выполнима, то любая вершина, достижимая из s - истина (так как по индукции по длине пути: Б.И. s - истина, Ш.И. пусть вершина u - истина и есть ребро $u \to v$, тогда дизъюнкт $(\overline{u} \vee v) = 1$ (так как 2КНФ выполнима), следовательно, $v = 1$). $\overline{t} = 1$, значит t не может быть достижимым, то есть нет пути из s в t.

Задача 5

Покажите, что любые два существенных языка (не пустой и не полный) из **NL** полиномиально сводятся друг к другу.

Известно, что $NL \subseteq P$. Покажем, что есть полиномиальная сводимость $\forall A \in P$ и $\forall B \in NL$: $A \leq_p B$.
Так как $A \in P$, то существует полиномиальный алгоритм, который проверяет, лежит ли элемент в A или нет. Если этот элемент лежит, то берём какой-нибудь элемент из B, а если нет, то из $\overline{B}$. Эти элементы можно подставить в алгоритм в явном виде.
Следовательно, доказали, что любые 2 существенных языка из **NL** полиномиально сводятся друг к другу.
P.s. полиномиальную сводимость тут использовать «странно», так как мы получили, что $P \subset NL$, но также мы знаем, что $NL \subset P$ и получаем, что эти 2 множества совпадают ...

Задача 7

Докажите, что язык XO выигрышных позиций в крестики-нолики на доске $n \times n$ лежит в $\mathcal{PSPACE}$.

Из семинара известно, что задача
$TQBF = \{\varphi(x_1, x_2, \ldots, x_k) | \forall x_1 \exists x_2 \ldots (\forall, \exists) x_k : \varphi(x_1, \ldots x_k) = 1\} \in \mathcal{PSPACE}$
Построим сводимость $TQBF \leq_p XO$.

За набор x возьмём ходы из выигрышных позиций, то есть из выигрышной позиции: существует ход x_1, что для любого хода x_2, существует ход x_3 и так далее. То есть набором x, подаваемой функции является дополнение множества клеток, которые задают выигрышное положение. Поэтому по определению множеств: если $x \in TQBF \iff f(x) = x \in XO$. Следовательно, $XO \in \mathcal{P}SPACE$