19.04

На этой паре мы занимались тем, что ввели определения, обсудили связь между ними и попытались мотивировать изучение эвристик из класса $\mathrm{1DSPACE}'$.

Определение 1. *Машина Тьюринга называется онлайн машиной Тьюринга, если она может двигать (но не обязана) головку на входной ленте только в одном направлении.*

Определение 2. *Класс всех языков, распознаваемых онлайн машиной Тьюринга, использующей не более $f(n)$ памяти, будем обозначать $\mathrm{1DSPACE}(f)$.*

Определение 3. *Класс языков, распознаваемых онлайн машиной Тьюринга, которая может узнать длину входа и использует не более $f(n)$ памяти, будем обозначать $\mathrm{1DSPACE}'(f)$.*

Лемма 1. *Существует язык $L \in \mathrm{1DSPACE}'(\log n)$ такой, что $L \notin \mathrm{1DSPACE}(o(n))$.*

Здесь в качестве L подходит множество всех таких двоичных слов s, что двоичная запись $|s|$ является некоторым префиксом s.

Лемма 2. *Для любой функции f такой, что $f(n) < \frac{n}{2}$ при всех достаточно больших n, и f может быть вычислено с использованием $O(\log f)$ памяти, существует язык L такой, что $L \in \mathrm{DSPACE}(\log f) \cap \mathrm{1DSPACE}'(f)$ и $L \notin \mathrm{1DSPACE}'(o(f))$.*

Тут подходит множество всех f-периодичных строк.

В лемме 2 у нас возникла трудность с вычислением f и на семинаре я предложил просто взять $f = n/4$, но после мы выяснили, что в качестве f подходит любая достаточно разумная функция, просто нужно внимательно следить за определениями. Подробнее в разделе с замечаниями.

Основным результатом была такая теорема.

Теорема 1. *Для любой функции $f = \Omega(\log\log n)$ и языка L, распознаваемого оффлайн машиной Тьюринга M с использованием $f(n)$ памяти и рабочим алфавитом Γ, $L \in \mathrm{1DSPACE}'(f(n) \cdot |\Gamma|^{f(n)})$, если $f(n)$ может быть вычислено с использованием $O(f(n) \cdot |\Gamma|^{f(n)})$ памяти.*

Доказательство этой теоремы очень похоже на сведение двусторонних конечных автоматов к односторонним (обычным), с одной небольшой тонкостью, связанной с зацикливанием. Дело в том, что машина

Тьюринга, даже если она останавливается на любом входе, может зациклиться, если ее запустить из неправильной конфигурации. В автоматах такое тоже бывает, но там об этом можно не думать, поскольку все функции переходов можно вычислить заранее (они же конечные!). Я эту проблему обходил при помощи техники baby-step giant-step (то есть запуска двух симуляций с разными скоростями), но на самом деле ее можно решать как угодно, например, просто добавлением счетчика (но тогда надо внимательно следить за памятью).

Algorithm 1 DPLL

procedure $\mathrm{DPLL}_{A,B}(\varphi)$
    if φ is empty **then**
        return satisfiable
    if φ contain empty clause **then**
        return unsatisfiable
    $x \leftarrow A(\varphi)$
    $b \leftarrow B(\varphi, x)$
    if $\mathrm{DPLL}_{A,B}(\varphi[x = b]) =$ satisfiable **then**
        return satisfiable
    return $\mathrm{DPLL}_{A,B}(\varphi[x = \neg b])$

procedure $\mathrm{DPLL}_{H}(\varphi)$
    if φ is empty **then**
        return satisfiable
    if φ contain empty clause **then**
        return unsatisfiable
    $(x, b) \leftarrow H(\varphi)$
    if $\mathrm{DPLL}_{H}(\varphi[x = b]) =$ satisfiable **then**
        return satisfiable
    return $\mathrm{DPLL}_{H}(\varphi[x = \neg b])$

Еще мы определили два вида DPLL (классический с двумя эвристиками, а нужный для наших целей — с одной) и поняли, что они друг от друга в терминах сложности по памяти почти ничем не отличаются.

26.04

Мы всю пару доказывали экспоненциальную нижнюю оценку на DPLL_H с $H \in 1\mathrm{DSPACE}'(o(\frac{n}{\log n}))$ (здесь небольшая неточность из-за того, что нам нужно выводить $O(\log n)$ битов, а DSPACE — класс языков распознавания, но у меня таких неточностей будет полно, вы же не хотите читать 10 страниц текста?).

TL;DR: Доказательство сложное, ниже будет краткий пересказ его упрощенного варианты, но даже это будет сложно и длинно.

Для ее доказательства нам на самом деле достаточно добиться того, чтобы H ошиблась и попала в сложную невыполнимую формулу. Просто сложную невыполнимую формулу мы уже знаем, можно взять матрицу-экспандер и формулу $A\vec{x} = \vec{q}$ для $\vec{q} \notin Im(A)$. Мы будем генерировать A таким образом, чтобы у нее в каждой строчке было по три единички, а в каждом столбце $O(\log n)$ единиц, где n — это размер матрицы. Еще мы попросим, чтобы A была полного ранга, конечно, это запретит такое q, чтобы формула была невыполнима, но нам это и не нужно, нам хочется, чтобы она стала невыполнимой после означивания одной переменной.

Формула, которую мы будем строить имеет следующий вид $\Phi_{q,w} := (A\vec{x} = \vec{q} \vee u) \wedge (A\vec{x} = \vec{w} \vee \neg u) \wedge \psi(x)$, где $\vec{x}, u$ — переменные, а ψ — некоторая короткая формула. Мы считали, что все переменные имеют одинаковый размер при записи, так что параметры формулы можно свободно менять, не заботясь о том, что H может начать работать как-то иначе до того, как она эти изменения прочитает. Формула, которая здесь написана, имеет упрощенный вид относительно формулы на паре, но зато не решает одной проблемы, которая у нас возникнет по дороге (вы же не хотите читать 10 страниц?).

H у нас имеет $o(\frac{n}{\log n})$ памяти, но формула Φ для своей записи в КНФ требует $n \log n$ битов, поэтому на самом деле у H памяти $o(n)$, где n — размерность $\vec{x}$. Теперь остается заметить, что с таким количеством памяти после прочтения формулы обязательно найдутся такие неразличимые для H множества Q и W решений уравнений из первой и второй скобок соответственно, что $|Q|, |W| \geq 2^{n-o(n)}$ и $Q \oplus \vec{1} = W$ (здесь используются три единички в каждой строчке). Мы также попросим, чтобы $\vec{0} \in Q$ (так сделать, конечно, нельзя, но это обходится усложнением формулы). Теперь нам нужно так выбрать ψ, чтобы были такие $q_0 \in Q$, $w_0 \in W$, что формулы Φ_{0,w_0} и $\Phi_{q_0,1}$ одновыполнимы, выполняются в скобках, соответствующих $Ax = \vec{0}$ и $Ax = \vec{1}$ (следовательно имеют противоположные выполняющие наборы) и при этом ψ не портит параметры расширения A. Так сделать можно, если сделать ψ ксором двух переменных и воспользоваться тем, что в каждом столбце A по $O(\log n)$ единиц. Полученные формулы неразличимы для H, значит, на одной из них H ошибается и попадает в сложную невыполнимую подформулу. Итоговая оценка на количество запусков $DPLL_H$ получится как $2^{\Omega(\frac{n}{\log^c n})}$.

Общие замечания

В лемме 2 f правильно воспринимать как функцию $1^n \to f(n)$ (это следует из определения DSPACE), которая вычислима с нужной нам памятью. Проблема с функцией вида $n \to f(n)$ в том, что на хранение n нам нужно $\log n$, а это есть не во всех классах из условия.

Теорема 1 вообще говоря, сформулирована и доказана для задач распознавания, а чтобы применить ее к оценке со второй пары нам нужно научиться сводить алгоритмы, которые выводят $\log n$ битов. Для этого нужно сделать две вещи. Во-первых, будем считать, что вывод алгоритма write-only. А во-вторых, будем рассматривать одну такую функцию как $\log n$ задач распознавания. Здесь возникает дополнительная трудность с тем, что нам нужно как-то узнать какой именно бит из вывода нас интересует, но в интересующих нас классах памяти достаточно, чтобы оценки не ухудшились (это можно честно проверить руками).