Практика №5 по курсу «Теория алгоритмов» «Классы сложности»

Группы ФТ-301, ФТ-302

На прошлых практиках мы не задавились вопросом эффективности машин Тьюринга и смотрели на них с чисто теоретической точки как на модель, позволяющую формализовать понятие алгоритма. На этой практике задумаемся над вопросами эффективности машин Тьюринга.

1 Временная сложности

Для начала определим для машины Тьюринга M и входа x время работы M на x как $T(M,x)$ равное количеству итераций работы машины Тюьринга. Будем говорить, что M работает за время $T_M(n)$ на входах x таких, что $|x| = n$, если $T_M(n) = \max_{x\in\Sigma^n}\{T(M,x)\}$. Так как точное значение функции $T_M(n)$ имеет сложную структуру, то имеет смысл оценивать асимптотическое поведение этой функции.

Теперь определим класс $\mathtt{DTIME}(f(n))$ для некоторой функции $f : \mathcal{N} \mapsto \mathcal{R}^+$ состоящий из таких языков $\mathcal{L}$, что каждого из них существует машина Тьюринга M задающая данный язык и работающая за время $T_M(n) = O(f(n))$.

Важным классом языков является класс $\mathtt{P} = \cup_{c\geq 1}\mathtt{DTIME}(n^c)$ - класс задач, решающихся за полиномиальное время от длины входа.

2 Правильные функции

Для доказательства теоремы об иерархии времени времени нам понадобятся специальные функции f такие, что существует k-ленточная машина Тьюринга M, которая на входе вида 1^n выдает бинарное представление значения $f(n)$ за время $O(n + f(n))$. Мотивация рассматривать такие функции заключается в том, чтобы машина Тьюринга из класса $\mathtt{DTIME}(f(n))$ могла вычислить время работы машины Тьюринга из класса $\mathtt{DTIME}(g(n))$, где $g(n) \leq f(n)$. Если функция $g(n)$ не будет «правильной» то её вычисление может занять значительно больше времени и уже не уложится в класс $\mathtt{DTIME}(f(n))$.

2.1 Упражнение

Ясно, что сумма «правильных» функций является «правильной» функцией. Но можно ли такое же утверждать про разность? Более формально, пусть функции $sum(n) = f_1(n) + f_2(n)$ и $f_2(n)$ являются «правильными». При каких условиях можно утверждать, что $f_1(n)$ также является правильной?

Без дополнительных ограничений данное утверждение не будет верным, так как если $f_2(n) = \omega(f_1(n))$, то f_1 необязательно будет являться «правильной» функцией, не нарушая при этой «правильность» суммы.

Чтобы гарантировать правильность функции f_1 можно наложить следующее ограничение на функции: пусть существует некоторый $\varepsilon > 0$ такой, что $f_1(n) \geq \varepsilon f_2(n)$ для любого n. Действительно, в этом случае мы можем вычислить значение функции $f_1(n)$ путем вычисления суммы $sum(n)$, значения $f_2(n)$ и последующего линейного вычисления их разности. Весь данный процесс займет $O(f_1(n) + f_2(n) + n)$ времени, что эквивалентно $O(f_1(n) + \frac{1}{\varepsilon}f_1(n) + n) = O(f_1(n) + n)$, а значит f_1 — «правильная» функция.

3 Пространственная сложность

Для того, чтобы определить пространственную сложность (количество памяти, которую используем машина Тьюринга) будем рассматривать только МТ из не менее чем трёх лент, у которых входная лента доступна только для чтения, а выходная — только для записи. Все остальные ленты считаются рабочими и доступны как на запись, так и на чтение. Для таких машин Тьюринга определим пространственную сложность на входе x как суммарное количество ячеек всех рабочих лент, в которых машина Тьюринга записала в некоторый момент времени записала непустой символ. Обозначим пространственную сложность на конкретном входе как $S(M,x)$, а за $S_M(n)$ обозначим худший случай потребления памяти для всех входов длины n.

Аналогичным образом можно определить класс $DSPACE(f(n))$ как множество языков $\mathcal{L}$ таких, что для каждого существует машина Тьюринга M задающая данный язык и использующая $S_M(n) = O(f(n))$ памяти.

4 Задача проверки строки на палиндром

Существует несколько подходов для решения задачи о проверки строки на палиндром PAL:

1. Рассмотрим МТ из двух лент, которая сначала копирует и переворачивает входную строку на вторую лента, после чего двумя указателями сравнивает противоположные символы строки. Данная машина Тьюринга решает задачу за линейное время и линейную память, а значит $PAL \in DTIME(n) \cap DSPACE(n)$

2. Альтернативный подход к решению заключается в том, чтобы в $O(\log n)$ ячейках хранить бинарное представление пары указателей на текущие символы для сравнения и постоянно перемещать каретку МТ между данными позициями, чтобы производить сравнения. Данное решение использует $O(\log n)$ памяти и работает за $O(n^2)$, а значит $PAL \in DTIME(n^2) \cap DSPACE(\log n)$

Можно ли утверждать, что $PAL \in DTIME(n) \cap DSPACE(\log n)$? Нельзя, так как для любой МТ M решающей задачу проверки строки на палиндромность верно, что $T_M(n) \cdot S_M(n) = \Omega(n^2)$ (данный момент опустили на практике, для зацепок про доказательство можно посмотреть ссылку `http://bit.ly/pal-time-space`).

Однако если $T_M(n) \cdot S_M(n) = \Omega(n^2)$, значит ли это что мы можем ускорить второе решение и добиться результата: $PAL \in DTIME(\frac{n^2}{\log n}) \cap DSPACE(\log n)$?

Да, можем! Общая идея решения состоит в том, что так как мы уже используем $O(\log n)$ ячеек под хранение значение указателей на символы, то можно добавить ещё $\log n$ ячеек, в котором будет хранится подотрезок длины $\log n$ исходной строки, который сейчас нужно будет сравнить с подотрезком с противоположной стороны строки. В таком случае алгоритм будет совершать $O(\frac{n}{\log n})$ сравнений блоков, каждый из которых будет выполняться за $O(n)$ (все время будет затрачено на сдвиг указатель в другую сторону строки), а значит $PAL \in DTIME(\frac{n^2}{\log n}) \cap DSPACE(\log n)$.