Задачи к Лекции 1

В этом листке Вы найдёте не только все задачи из Лекции 1, но и несколько новых задач — чтобы не было скучно. Решайте только то, что Вам нравится.

Exercise 1.e.1. В этой задаче мы работаем с неотрицательными числами, которые могут быть больше 2^{31}. Напомним, что на ленте машины Тьюринга такие числа представляются в виде $n_0, n_1, \ldots, n_k, \#$, где $\# = 2^{31}$ и $\overline{n} = n_k \ldots n_1 n_0$ в системе счисления с основанием 2^{31}. Первый символ в записи — это цифра младшего разряда.

(a) Постройте машину Тьюринга, вычисляющую функцию $n \mapsto n + 1$.

(b) Положим $z = 2^{31} - 1$; нарисуйте график работы (space-time diagram) машины Тьюринга из пункта (a) на входе $z, z, z, \#$.

(c) Постройте машину Тьюринга, вычисляющую функцию $m, n \mapsto m + n$.

(d) Постройте машину Тьюринга, вычисляющую функцию $m, n \mapsto 0$ если $m = n$ и 1 иначе.

(e) Постройте машину Тьюринга, вычисляющую функцию $m, n \mapsto 0$ если $m \leq n$ и 1 иначе.

(f) Постройте машину Тьюринга, вычисляющую произведение двух натуральных чисел.

(g) Постройте машины Тьюринга, производящую вычитание и деление натуральных чисел. *Подсказка:* используйте машины для сложения/умножения и алгоритм перебора.

Exercise 1.e.2.

(a) Напишите интерпретатор для машин Тьюринга на Python или любом другом языке программирования.

(b) Дополните Ваш интерпретатор так, чтобы он выводил график работы (space-time diagram) вычисления, которое он проводит.

Exercise 1.e.3. Постройте машину Тьюринга, которая принимает на вход последовательность целых чисел между 0 и $2^{31} - 1$ включительно (последний символ последовательности — #) и сортирует эти числа в неубывающем порядке.

Exercise 1.e.4.

(a) Докажите, что функция f вычислима с помощью обычной машины Тьюринга тогда и только тогда, когда она вычислима машиной Тьюринга с двумя лентами (a 2-memory Turing machine, cf. §1.6.6).

(b) Напишите определение машины Тьюринга с k лентами (a k-memory Turing machine) и докажите, что машины Тьюринга с k и ℓ лентами эквивалентны (как это определено в предыдущей задаче) для любых $k, \ell \geq 1$.

(c) Докажите, что множество вычислимых функций не меняется в зависимости от ленточного алфавита машины Тьюринга (memory alphabet) $\mathbb{I}, |\mathbb{I}| \geq 2$. Например, какое бы из следующих множеств мы не выбрали в качестве $\mathbb{I}$: $\{0, \ldots, 2^{32} - 1\}$, $\{0, 1, \ldots, 9\}$ или $\{0, 1\}$, множество вычислимых функций останется одним и тем же

(d) Докажите, что машина Тьюринга с бесконечной в обе стороны лентой (bi-infinite memory), то есть, функция $\mathbb{Z} \to \mathbb{I}$ вместо $\mathbb{N} \to \mathbb{I}$, вычислительно эквивалентна обычной машине Тьюринга.

Exercise 1.e.5. Найдите в интернете

(a) или в книжке неразрешимую проблему, отличную от пробемы Остановки (the Halting problem) и поймите, почему она неразрешима (выберите проблему с несложным доказательством).

(b) компилятор, который по коду (выберите любой язык программирования) строит машину Тьюринга. Попробуйте запустить его на простых программах и посмотрите на построенные машины.

(c) Попробуйте разобраться в коде найденного компилятора.

Exercise 1.e.6 ( ). Напишите класс (например, на Python), моделирующий арифметическое выражение: бинарное дерево, внутренние узлы которого содержат арифметические операции $(+, -, \times, \div, =, <)$, а в листьях содержатся целые числа. Затем напишите транслятор, преобразующий арифметические выражения в машины Тьюринга.

Exercise 1.e.7 ( ). В этой задаче Вы можете поменять $\mathbb{I}$, чтобы иметь больше дополнительных символов: например, $\mathbb{I} = \{2^{33} - 1\}$.

(a) Для любого i из $\mathbb{N}$, постройте машину Тьюринга, которая, работая на входе:

v_0	#	v_1	#	...	v_{k-1}	#	#	0	0	0	...

выписывает копию v_i после ##. Заметьте, что $v_0, v_1, \ldots, v_{k-1}$ — это *блоки*, состоящие из нескольких клеток. *Подсказка:* Вам понадобится временно изменить изначальное v_i, чтобы выписать копию, но вы сможете реконструировать их после этого.

(b) А теперь, для любого i из $\mathbb{N}$, постройте машину Тьюринга, которая, работая на входе:

v_0	#	v_1	#	...	v_{k-1}	#	#	v_k	#	0	0	0	...

стирает v_i и пишет вместо него копию v_k. Заметьте, что v_k может оказаться короче или длиннее v_i! Кроме того, может иметь место следующее: $i \geq k$; в этом случае, добавьте нужное количество блоков, содержащих 0, чтобы расширить последовательность до нужной длины.

Exercise 1.e.8 ( ). Предположим, нам даны три машины Тьюринга T_1, T_2, и T_3, выполняющие программы на Python'е P_1, P_2, и P_3 соответственно. Постройте машины Тьюринга, вычисляющие:

- `if`(P_1): P_2 `else`: P_3
- `while`(P_1): P_2

Exercise 1.e.9 ( ). Напишите компилятор, который строит машину Тьюринга по коду на Simple Python. (См. определение Simple Python в Claim 1.6.2). Используйте конструкцию, построенную в предыдущей задаче!

Contacts

- Daria Pchelina (dpchelina@clipper.ens.fr)
- Guilhem Gamard (guilhem.gamard@normale.fr)