Technische Universität München (I7)
Dr. J. Kreiker / Dr. M. Luttenberger

Summer term 2010
30. April 2010, 16:22

Complexity Theory – Homework 2

Discussed on 05.05.2010.

Exercise 2.1

Argue that the following theorem on the linear speedup of Turing machine holds:

> Let $L \subseteq \{0,1\}^*$ be a language decided by a Turing machine M in time $T(n)$. Then, for any $c > 0$ there is Turing machine M' which decides L in time $T'(n) := cT(n) + n + C$ (with C some constant independent of L or c, e.g., $C \leq 10$ should work).

Remark: Fix any constant $m \in \mathbb{N}$. Then M' first compresses the input from size n to size $\lceil \frac{n}{m} \rceil$ on some auxiliary work tape. Then M' simulates m steps of M within at most 10 steps. (In fact, 6 steps should be sufficient.) Finally, choose the constant m in such a way that M' simulates M in time $T'(n)$.

Exercise 2.2

Let M be a Turing machine which computes a function $f : \{0,1\}^* \to \{0,1\}^*$. As mentioned in the lecture, we are basically interested in two resources, time and space, needed by M for computing $f(x)$ from the input x. Measuring time is straightforward, we simply count the number of steps M does on input x. In the case of space, one is usually not interested in the space required for storing the input or the output, but only in the space required for computing the output from the input. One therefore defines:

> A function $f : \{0,1\}^* \to \{0,1\}^*$ is computable in space $S(n)$ if there is a Turing machine M_f such that
>
> (i) M_f computes f.
>
> (ii) M_f does not write any blanks ($\Box$).
>
> (iii) M_f never moves the head of the output tape to the left.
>
> (iv) For every input x of length $n = |x|$ the total number of non-blank symbols on all *work* tapes is bounded from above by $S(n)$ in every step of the computation.

Similar to the definition of **DTIME**, we write $f \in \mathbf{DSPACE}(S)$ if there is a Turing machine which computes f in space $S'(n)$ for some $S' \in \mathcal{O}(S)$. Finally, a language $L \subseteq \{0,1\}^*$ is decided in $\mathbf{SPACE}(S)$ if its characteristic function f_L is computable in $\mathbf{SPACE}(S)$ (with $f_L(x) := 1$ if $x \in L$, and $f_L(x) := 0$ if $x \notin L$).

(a) Show that the function inc : $\{0,1\}^* \to \{0,1\}^*$ which increases x by one (interpreting x as a natural number via the lsbf-encoding) is computable in constant space $\mathcal{O}(1)$.

(b) How much space is needed to decide the language of palindromes?

(c) Show or disprove that we may strengthen condition (iii) to "M_f never moves the head of the output tape to the left and never overwrites a non-blank symbol on the output tape".

(d) Argue that if a function f is computable in space $S(n)$, then it is also computable in space $cS(n)+C$ for any $c \in (0, \infty)$ (with C some constant independent of f or c, e.g., $C \leq 10$ should work).

*(e) For those who know two-way finite automata:

Argue that every Turing machine using bounded space is basically a finite automaton with output.

Exercise 2.3

In the lecture, you have seen the definition of "polynomial-time reducible" $\leq_p$:

> For two languages $A, B \subseteq \{0,1\}^*$ we write $A \leq_p B$ if there is a function $f : \{0,1\}^* \to \{0,1\}^*$ computable in polynomial time such that $x \in A \Leftrightarrow f(x) \in B$ for all $x \in \{0,1\}^*$.

Similarly, the notion of "log-space reducible" $\leq_{\log}$ is defined but this time the function f has to be computable by a Turing machine using at most $\mathcal{O}(\log n)$ space.

(a) Show that $A \leq_{\log} B$ implies $A \leq_p B$.

(b) Show that for any two languages A, B in **P** *with* $B \neq \emptyset, \{0,1\}^*$ we have $A \leq_p B$.

Remark: Using $\leq_{\log}$ one can also define **P**-complete problems in a meaningful way.

(c) Argue that $\leq_{\log}$ is also transitive, i.e., if $A \leq_{\log} B \leq_{\log} C$, then also $A \leq_{\log} C$.

Hint: Let M_f, resp. M_g be $\mathcal{O}(\log n)$-space Turing machines computing the functions f, resp. g underlying $A \leq_{\log} B$, resp. $B \leq_{\log} C$. Note that you cannot simply first calculate $f(x)$ using M_f and then $g(f(x))$ by intializing M_g on input $f(x)$ as the resulting composed Turing machine requires polynomial space for storing the intermediate result $f(x)$.