Complexité avancée - TD 1

Simon Halfon

September 14, 2016

Exercise 1: Graph representation and why it does not matter

Let $\Sigma = \{0, 1, [,], \bullet\}$, $n \in \mathbb{N}$ and $V = [0, n-1]$. We consider the following two representations of a directed graph $G = (V, E)$ by a word in Σ^*:

- By its adjency matrix: $[m_{0,0} \bullet m_{0,1} \cdots \bullet m_{0,n-1}] \dots [m_{n-1,0} \bullet \cdots \bullet m_{n-1,n-1}]$, where for all $i, j \in [0, n-1]$, $m_{i,j}$ is equal to 1 if $(i, j) \in E$, 0 otherwise.
- By its adjency list: $[k_0^0 \bullet \cdots \bullet k_{m_1}^0] \dots [k_0^{n-1} \bullet k_{m_{n-1}}^{n-1}]$, where for all i, $[k_1^i, \dots, k_{m_i}^i]$ is the list of neighbors of vertex i, written in binary, in increasing order.

1. Describe a logarithmic space bounded deterministic Turing machine which takes as input the graph G, represented by adjacency lists, and returns the adjacency matrix of G.
2. Conversely, describe a logarithmic space bounded deterministic Turing machine taking as input the adjacency matrix of a graph G, and computing the adjacency list representation of G.

Therefore, the complexity of the problem `REACH` seen in class does not depend on the representation of the graph.

Definition 1 *A function $f : \mathbb{N} \to \mathbb{N}$ is said to be* space-constructible *if there exists a deterministic Turing machine that computes $f(|x|)$ in $O(f(|x|))$ space given x as input.*

Exercise 2: restrictions in the definition of $\mathsf{SPACE}(f(n))$, and why they do not matter

In the course, we restriced our attention to Turing machines that always halt, and whose computations are space-bounded on every input. In particular, remember that $\mathsf{SPACE}(f(n))$ is defined as the class of languages L for which there exists some deterministic Turing machine M that always halts (i.e. on every input), whose computations are $f(n)$ space-bounded (on every input), such that M decides L.

Now, Consider the following two classes of languages:

- $\mathsf{SPACE}'(f(n))$ is the class of languages L such that there exists a deterministic Turing machine M using at most space bounded by f (on every input), and accepting x if and only if $x \in L$. (Notice that we do not require that M halts when $x \notin L$).
- $\mathsf{SPACE}''(f(n))$ is the class of languages L such that there exists a deterministic Turing machine M such that M accepts x using space $f(n)$ iff $x \in L$. Note that if $x \notin L$, M might use more space, or even not halt.

1. Show that for a space-constructible function $f = \Omega(logn)$, $\mathsf{SPACE}'(f(n)) = \mathsf{SPACE}(f(n))$
2. Show that for a space-constructible function $f = \Omega(logn)$, $\mathsf{SPACE}''(f(n)) = \mathsf{SPACE}(f(n))$

Exercise 3: One-minute-long exercise

Prove that any language $L \subset \{0,1\}^*$ that is neither empty nor $\{0,1\}^*$ is hard for NL for polynomial-time reductions.

Exercise 4: Dyck's language

- Let A be the language of balanced parentheses – that is the language generated by the grammar $S \to (S)|SS|\epsilon$. Show that $A \in \mathsf{L}$.
- What about the language B of balanced parentheses of two types? that is the language generated by the grammar $S \to (S)|[S]|SS|\epsilon$

Exercise 5: Inclusions of complexity classes

Show that for a space-constructible function,

$$\mathsf{NSPACE}(f(n)) \subseteq \mathsf{DTIME}(2^{O(f(n))})$$

Exercise 6: NL alternative definition

A Turing machine with *certificate tape* is a deterministic Turing machine with an extra read-only input tape called *the certificate tape*, which moreover is *read once* (*i.e.* the head on that tape can either remain on the same cell or move right, but never move left).

Define NL_{certif} to be the class of languages L such that there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a Turing machine with certificate tape M that runs in logarithmic space such that:

$$x \in L \text{ iff } \exists u, |u| \leq p(|x|) \text{ and } M \text{ accepts on input } (x, u)$$

1. Show that $\mathsf{NL}_{certif} = \mathsf{NL}$
2. What complexity class do you obtain if you remove the read-only constraint in the definition of a machine with certification tape ?

Exercise 7: restrictions of the `SAT` problem

1. Let 3-`SAT` be the restriction of `SAT` to clauses consisting of at most three literals (called 3-clauses). In other words, the input is a finite set S of 3-clauses, and the question is whether S is satisfiable. Show that 3-`SAT` is NP-complete for logspace reductions (assuming `SAT` is).
2. Let 2-`SAT` be the restriction of `SAT` to clauses consisting of at most two literals (called 2-clauses). Show that 2-`SAT` is in P, using proofs by resolution.
3. Show that the complement of 2-`SAT` (i.e, the unsatisfiability of a set of 2-clauses) is NL-complete.
4. Conclude that 2-`SAT` is NL-complete.