Complexité avancée
TD 1

Cristina Sirangelo - LSV, ENS-Cachan

September 24, 2014

Exercise 1 Let G be a directed graph (V, E), with $V = \{0, 1, ..., N-1\}$. The vertices are just integers, which are assumed to be written in binary. We can represent G by its adjacency matrix MG, which is an $N \times N$ matrix, where each element m_{ij} is the bit 1 if $(i, j) \in E$, the bit 0 otherwise. More precisely, G is represented by the word starting with N written in binary, followed by the special symbol #, and by the list of bits $m_{00}, m_{01}, \ldots, m_{0(N-1)}, m_{10}, m_{11}, \ldots, m_{1(N-1)}, \ldots, m_{(N-1)0}, m_{(N-1)1}, \ldots, m_{(N-1)(N-1)}$.

We can also represent G by its adjacency lists. The adjacency list of vertex i is the set of vertices j such that $(i, j) \in E$, sorted in ascendent order, say $j_1, \ldots, j_m$. We will represent it as the word $j_1; j_2; \ldots; j_m;$, where the j_i are written in binary, over the alphabet $\{0, 1\}$, and where ; is a third symbol. We represent then G as the string $L_0 \bullet L_1 \bullet \ldots L_{N-1} \bullet$, where each L_i is the adjacency list of vertex i, and where $\bullet$ is a fourth symbol.

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.

Exercise 2 Show that, conversely, we can also define 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.

Definition (Proper) *A function $f : \mathbb{N} \to \mathbb{N}$ is said to be* proper *if f is non-decreasing, and there exists a deterministic Turing machine $\mathcal{M}_f$ (with a fixed number of tapes) such that for each input x of size n, the machine $\mathcal{M}_f$ halts with exactly $f(n)$ blank symbols written on the output tape, and $\mathcal{M}_f$ runs in time $O(n+f(n))$ and space $O(f(n))$.*

Exercise 3 Show that we end up defining the same class $SPACE(f(n))$ if we do not require that the machine M halts on every input. More precisely, let $SPACE'(f(n))$ be the class of languages L such that there exists a deterministic Turing machine M using at most space $f(n)$ (n being the size of the input x), and accepting x if and only if $x \in L$. (Notice that we do not require that M halts when $x \notin L$). Show that if f is a proper complexity function and $f(n) = \Omega(log\ n)$, then $SPACE'(f(n)) = SPACE(f(n))$.

Exercise 4 Let $SPACE''(f(n))$ be the class of languages L such that there exists a deterministic Turing machine M such that M accepts x iff $x \in L$, and M accepts using space $f(n)$ on each input $x \in L$ of size n (however M might use more space, and might also not halt if $x \notin L$). Show that if f is a proper complexity function and $f(n) = \Omega(log\ n)$, then $SPACE''(f(n)) = SPACE(f(n))$.

Exercise 5 Show that $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(O(c^{f(n)+log\ n}))$ (assume that f is proper).

Exercise 6 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 (you can assume that SAT is).

Exercise 7 Let 2-SAT be the restriction of SAT to clauses consisting of at most two literals (called 2-clauses). In other words, the input is a finite set S of 2-clauses, and the question is whether S is satisfiable. Show that :

- 2-SAT is in $\mathbf{P}$.
- the complement of 2-SAT (i.e, the unsatisfiability of a set of 2-clauses) is NL-complete.

Exercise 8 Using Exercise 7, show that 2-SAT is NL-complete.

Exercise 9

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