Exercise 1 (Simple questions).

1. At which conditions is a language $A \subset \{0,1\}^*$ hard for $\text{NL}$ under polynomial-time reductions?

2. Give an example of a non proper monotone function.

Exercise 2 (Graph representation and why it does not matter). In the definition of $\text{PATH}$, we did not specify which graph representation is used. In fact, this does not matter as the two usual representations (by adjacency list or matrix) are log-reducible from one another.

Let $\Sigma = \{0, 1, /, \bullet, \#\}$ with $\#$ the end-of-word symbol. For a directed graph $G = (V, E)$ with $V = [0, n - 1]$ for some $n \in \mathbb{N}$ and $E \subseteq V \times V$, we consider the following two representations of $G$ by a word in $\Sigma^*$:

- By its adjacency matrix $m_G \in \Sigma^*$:
  
  $$m_G \overset{\text{def}}{=} m_{0,0} m_{0,1} \ldots m_{0,n-1} \bullet \cdots \bullet m_{n-1,0} \ldots m_{n-1,n-1} \#$$

  where for all $0 \leq i, j < n$, $m_{i,j}$ is 1 if $(i, j) \in E$, 0 otherwise.

- By its adjacency list $l_G \in \Sigma^*$:
  
  $$l_G \overset{\text{def}}{=} k_0^0 / \ldots / k_m^0 \bullet \cdots \bullet k_{m-1}^{n-1} / \ldots / k_{m-1}^{n-1} \#$$

  where for all $0 \leq i < n$, $k_i^0, \ldots, k_i^{m_i}$ are binary words listing the (codes of) right neighbors of vertex $i$.

1. Show that it is possible to check in logarithmic space that a word $w \in \Sigma^*$ is a well-formed description of a graph (for any of the two representations).

2. Describe a logarithmic space bounded deterministic Turing machine taking as input a graph $G$, represented by its adjacency matrix, and computing the adjacency list representation of $G$.

3. Show that is $f$ and $g$ are computable in logarithmic space, $f \circ g$ is computable in logarithmic space.

4. Let $L$ be a $\text{NL}$-complete language, let $f$ and $L'$ be such that $f$ is logarithmic space computable and $L' \in \text{NL}$ and for all $x$ we have $x \in L$ iff $f(x) \in L'$, show that $L'$ is $\text{NL}$ complete.
Exercise 3 (Restrictions in the definition of $\text{SPACE}(f(n))$). In the course, we restricted our attention to Turing machines that always halt, and whose computations are space-bounded on every input. In particular, remember that $\text{SPACE}(f(n))$ 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:

- $\text{SPACE}'(f(n))$ is the class of languages $L$ such that there exists a deterministic Turing machine $M$, running in space bounded by $f(n)$, such that $M$ accepts $x$ iff $x \in L$. Note that if $x \notin L$, $M$ may not terminate.

- $\text{SPACE}''(f(n))$ is the class of languages $L$ such that there exists a deterministic Turing machine $M$ such that $M$ accepts $x$ using space bounded by $f(n)$ iff $x \in L$ ($M$ may use more space and not even halt when $x \notin L$).

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

Exercise 4 (Inclusions of complexity classes). Show that for a space-constructible function $f$,

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

Exercise 5 (Dyck’s language). Let $A$ be the language of balanced parentheses (i.e. generated by the grammar $S \rightarrow (S) | SS | \varepsilon$). Show that $A \in L$. What about the language of balanced parentheses of two types (i.e. generated by $S \rightarrow (S) | [S] | SS | \varepsilon$)?

Exercise 6 (A few NL-complete problems). Show that the following problems are NL-complete.

1. Deciding if a non-deterministic automaton $A$ accepts a word $w$.
2. Deciding if a directed graph is strongly connected.
3. Deciding if a directed graph has a cycle.

Bonus (Space hierarchy theorem). Consider two space-constructible functions $f$ and $g$ such that $f = o(g)$. Prove that $\text{DSPACE}(f) \subseteq \text{DSPACE}(g)$. 

2