Exercise 1: Graph representation and why it does not matter

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

— By its adjacency matrix: \( m_{0,0} \bullet 0 \bullet \cdots \bullet m_{0,n-1} \bullet m_{n-1,0} \cdots m_{n-1,n-1} \), 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 adjacency list: \( k_0^0 \bullet \cdots \bullet k_0^{m_0} \bullet \cdots \bullet k_{m_1}^0 \cdots ^{m_1} \cdots \bullet k_{m_{n-1}}^0 \cdots ^{m_{n-1}} \), where for all \( i \), \([k_i^0, \ldots, k_i^{m_i}]\) 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 representation of \( G \).

2. Conversely, 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 \( G \).

Therefore, the complexity of the problem \textsc{Reach} seen in class does not depend on the representation of the graph.

Exercise 2: Inclusions of complexity classes

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

Show that for a space-constructible function,

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

Exercise 3: Restrictions in the definition of \textsc{Space}(f(n)), and why they do not matter

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 \textsc{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:

— \textsc{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.

— \textsc{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 \( f = \Omega(\log n) \), \textsc{Space}'(f(n)) = \textsc{Space}(f(n))

2. Show that for a space-constructible function \( f = \Omega(\log n) \), \textsc{Space}''(f(n)) = \textsc{Space}(f(n))
Exercise 4: Dyck’s language

Let $A$ be the language of balanced parentheses – that is the language generated by the grammar $S \rightarrow (S)|SS|\epsilon$. Show that $A \in L$.

What about the language $B$ of balanced parentheses of two types? That is the language generated by the grammar $S \rightarrow (S)||S||SS|\epsilon$.