NL = CoNL

In this handout, we describe a nondeterministic machine that computes the number of nodes reachable from a source node in directed acyclic graph, using $O(\log n)$ space. It is easy to modify this machine to get a nondeterministic logspace deciding Non-Reachability (how?). Since Reachability is complete for NL, it proves that NL=coNL; that is for any $L \in$ NL, $\overline{L}$; the complement language of L, is also in NL. This result is considered to be one of the beautiful results in complexity theory.

First let us define the languages that we are looking at. A graph $G = (V, E)$ for us is a directed graph on vertices $V = \{1, \ldots, n\}$.

- Reachability=$\{\langle G\rangle | G$ is a directed graph and there is a directed path from 1 to $n\}$.
- Non-Reachability=$\{\langle G\rangle | G$ is a directed graph and there are no directed path from 1 to $n\}$ (complement of the above language)
- Reachability$_{\leq k}$ =$\{\langle G, u\rangle | G$ is a directed graph and there is a path of length $\leq k$ from 1 to $u\}$.

Notice that for $k = n - 1$, Reachability$_{\leq k}$ is essentially same as Reachability, since if there is a path between two vertices then there is a path with $\leq n - 1$ vertices.

Finally, we will consider the counting problem #Reachability where one needs to count the number of vertices reachable from vertex 1. It is easy to see that a machine for #Reachability can be very easily modified to get a machine for both Reachability and Non-Reachability. We will show how to solve this general problem using a nondeterministic machine using logarithmic space. Since we are dealing with a problem to compute (rather than a language to decide) we need to define what exactly we mean for a nondeterministic machine to compute a function.

Definition. We say that a nondeterministic machine computes a function $f : \mathbf{N} \rightarrow \mathbf{N}$, if for any input x; firstly all the computation either halts outputting a string or rejects. Additionally, on those computations that outputs, it should output the binary representation of $f(x)$. Finally, there should be at least one computation that outputs $f(x)$.

In other words, machine on any input x should *unambiguously* output the binary value of $f(x)$.

First we will give a nondeterministic log space machine for Reachability$_{\leq k}$. This machine will be used as a subroutine in the final counting machine.

Machine $M_{\leq k}$ on input $G = (V, E)$

1. $w_0 \leftarrow 1$ and $w_k \leftarrow u$
2. For $i = 1$ to k
3. Guess vertex w_i;/* Reusing space for each w_i */
4. If $(w_{i-1}, w_i) \in E$ or $w_{i-1} = w_i$ continue with the next i;
5. Else Reject;
6. Accept.

Note that at each iteration at most four w's need to be kept on the tape; w_1, w_k, w_{i-1} and w_i. Therefore the nondeterministic machine uses only $O(\log n)$ space. It is also clear that if there is a path from 1 to u of length $\leq k$, then one of the sequence of guessed vertices will lead to Accept and if there are no paths of length $\leq k$ from 1 to u, then none of the guesses will lead to accept. Therefore, the above machine decides REACHABILITY$_{\leq k}$.

Let us introduce some notations. Let S_k denote the set of all vertices that are reachable from 1 using a path of length $\leq k$. That is a vertex $u \in S_k$ iff $\langle G, u \rangle \in$REACHABILITY$_{\leq k}$. Since any vertex that is reachable from 1 is reachable using a path of length $\leq n-1$, what we need to compute is the cardinality $|S_{n-1}|$ ($|S_k|$ means the number of vertices in the set S_k).

Below is the description of a logspace nondeterministic machine for #REACHABILITY. You need to read the description below along with the explanations in the text book to get a clear picture of what is going on exactly.

<u>Machine $\#M$ on input $G = (V, E)$</u>

$|\mathbf{S_1}| \leftarrow 1$;
For $k = 1$ to $n-1$ /* This outerloop computes $|S_k|$ using $|S_{k-1}|$ */
    $|S_k| \leftarrow 0$;
    For each $v \in V$ /* This loop checkes whether $v \in S_k$ and if yes, increments $|S_k|$*/
    `Count` $\leftarrow 0$; /* `Count` is a log space counter */
    `Flag` $\leftarrow$ **F**;
        For each $u \in V$; /* This loop checkes whether $u \in S_{k-1}$ using machine $M_{\leq}$*/
            $w_0 \leftarrow 1$ and $w_k \leftarrow u$
            For $i = 1$ to k
                Guess vertex w_i;/* Reusing space for each w_i */
                If $(w_{i-1}, w_i) \in E$ or $w_{i-1} = w_i$ continue with the next i;
                Else Reject;
            `Count` $\leftarrow$ `Count` $+ 1$; /* At this point $u \in S_{k-1}$ */
            If $(u, v) \in E$ or $u = v$ then `Flag` $\leftarrow$ **T**;
        If `Count` $= |S_{k-1}|$&`Flag` $=$ **T** then $|S_k| \leftarrow |S_k| + 1$;
        Else Reject.
    Next k;
Accept and Output $|S_{n-1}|$.
