1 Strongly Connected Graph

STRONGLY-CONNECTED $\in$ NL-Complete

We define the language STRONGLY-CONNECTED = \{⟨G⟩|G is a directed graph and there exists a path from every node in G to every other node in G\}

We first show that strongly connected is in NL.

TM $M_1 =$ "On input ⟨G⟩:
- Check that ⟨G⟩ is a valid encoding of a directed graph, else reject.
- Loop through all pairs of nodes (u,v) that are in G and run the nondeterministic procedure used in the PATH language NTM decider to determine if there exists a path starting from u and ending at v. On the branches that have not found a path and reject, $M_1$ rejects. If some branch finds a path, instead of accepting it will continue with the next pair of nodes.
- If we have looped through all the pairs of nodes and have not rejected (meaning we have found paths between all of them), then ACCEPT."

Now, we show that PATH $\leq_L$ STRONGLY-CONNECTED. Reminder: PATH = \{⟨G,s,t⟩|G is a directed graph and there exists a path from s to t\}.

Given the graph G from the PATH language input, we create a new graph $G'$ that is identical to G with some additional edges:
- Add an edge from node t to every node in the graph.
- Add an edge from every node in the graph to node s.

Now, if there exists a path from s to t in G, $G'$ is strongly connected, as we can get from any node u in $G'$ to any other node v in $G'$ by taking the edge from u to s, then taking the path from s to t and finally taking the edge from t to v. Conversely, if $G'$ is strongly connected, then there exists a path from s
to $t$ in $G$, because the edges we added to construct $G'$ do not help us on our journey from $s$ to $t$, they only help us in getting from $t$ or anywhere else back to $s$. Hence, the reduction works. It uses log-space because when outputting the new graph $G'$, we only have to make sure to add the extra edges in $G'$’s description which is very straightforward.

$2 - 2\overline{SAT}$

We show that $\text{PATH} \leq_L 2\overline{SAT}$.

For every node $u$ in the graph $G$ from PATH, we assign a variable $x_u$ in the boolean formula $\phi$ we are constructing. For every edge between $u$ and $v$ in $G$, we add the clause $(x_u \rightarrow x_v) \equiv (\overline{\pi_v} \lor \overline{x_v})$. We also make sure that $x_s$ is true by adding the clause $(x_s \lor \overline{x_s})$ and we add the clause $(x_t \rightarrow \overline{x_t}) \equiv (\overline{x_t} \lor \overline{x_s})$.

Now, if we have a path from $s$ to $t$, $x_s$ is true and because every node connected to $s$ is true, $x_t$ ends up being true. But we also have the clause $(x_t \rightarrow \overline{x_s})$, implying that because $x_s$ is true, then $x_s$ must be false. This cannot happen, so $\phi$ is unsatisfiable. Conversely, if $\phi$ is unsatisfiable, this means that there must be a path from $s$ to $t$ in $G$. If there was no such path, then we could set $x_s$ along with all the variables corresponding to nodes reachable from $s$ as true and all other nodes including $t$ false and would have a satisfying assignment.