Homework 9

Problem 1. Prove that TQBF restricted to formulas where the part following the quantifiers is in conjunctive normal form (CNF) is still PSPACE-complete.

Solution. Easy to verify that the TQCNF is in PSPACE.

For any boolean formula $C(x)$, we would create an CNF $\phi(x, y)$ such that $C(x) = \exists y \phi(x, y)$ holds for all x. With the reduction in hand, we can transform the TQBF formula $\forall x_1 \exists x_2 \ldots C(x)$ to a TQCNF instance $\forall x_1 \exists x_2 \ldots \exists y \phi(x, y)$. Viewing the boolean formula as a circuit, for each non input wire we introduce a variable y_i, and for each gate g, we can translate the boolean formula $y_{\text{out}} = g(y_{\text{in1}}, y_{\text{in2}})$ to a constant size CNF $\phi_g(y_{\text{out}}, y_{\text{in1}}, y_{\text{in2}})$. Our final CNF would be $\wedge_g \phi_g \wedge y_{\text{output}}$. Since we have polynomial size wire and gates, the reduction is in polynomial time.

Problem 2. Let $\text{SUM} = \{\langle x, y, z\rangle \mid x, y, z > 0$ are binary integers satisfying $x + y = z\}$. Show that $\text{SUM} \in \text{L}$.

Solution. Note that simultaneously add $x + y$ is not doable in logspace. The solution is simulating a column addition process, and maintain a log-size counter for the current bit we are working on.

Problem 3.

(a) An undirected graph is *bipartite* if its nodes may be divided into two sets so that all edges go from a node in one set to a node in the other set. Show that a graph is bipartite if and only if it doesn't contain a cycle that has an odd number of nodes.

(b) Let $\text{BIPARTITE} = \{\langle G\rangle \mid G$ is a bipartite graph$\}$. Prove that BIPARTITE is in NL.

Solution. (a) If the graph contains an odd cycle, we can see that any 2 coloring(partition) scheme will create an edge with both nodes of the same color, thus the graph is not bipartite.

If the graph doesn't contain an odd cycle, we now prove the graph is bipartite. WLOG, we can assume the graph is connected. Consider the following algorithm: we arbitrarily choose one vertex u in G, and assign it to

0. We iteratively perform the process: for a vertex assigned to x, we assign all its neighbors to $1-x$. If the assignment have a contradiction, for example, there is an edge (v, w) where v, w are both assigned to 0, we can find a odd cycle considering the path $u \to v \to w \to u$.

(b) Since NL = coNL, we only have to prove that $\overline{\text{BIPARTITE}} \in$ NL. By the proof in (a), we can construct an NL algorithm for the odd cycle problem, where the algorithm maintains a parity counter, nondeterministically choose a start point, and at each step nondeterministically choose a neighbor, update the parity counter, until the path returns to the starting point. The algorithm accepts if the parity counter is 1.

Problem 4. Let $S(n) \geq \log n$ be a space-constructible function. Show that NSPACE($S(n)$) = coNSPACE($S(n)$) is a consequence of NL = coNL.

Solution. Prove by padding. For any language $A \in$ NSPACE($S(n)$), consider the following language $A_{\text{pad}} = \{x\#1^{2^{S(|x|)}} \mid x \in A\}$. We can assume that the total space used by the NTM for A is bounded by $C \cdot S(n)$ for some constant C. We now show that $A_{\text{pad}} \in$ NL. For language A_{pad}, we construct the following NTM M in space $C \log n$.

The NTM first set a space bound on $C \log n$. If at any step the TM tries to reach space over the bound, it directly rejects.

It first reads the input, and counts the length of x as n'. It computes $S(n')$, and checks if the input is in the form $x\#1^{2^{S(|x|)}}$. We can see that for valid input, $S(n') \leq logn$, thus the computation can be done in space $C \log n$ since $S(n)$ is space constructible. If the input is not valid, it rejects.

The NTM simulates the NTM for A on x and aligns with its output, since A uses space $O(S(n')) = O(\log n)$, the step is doable in space $O(\log n)$. We can check that M decides A_{pad} in space $O(\log n)$.

By NL = coNL, there also exists a NTM A' that decides $\bar{A}_{\text{pad}}$, and we can construct a NTM B' that decides $\bar{A}$ in space $S(n)$ by first run the padding process for input x, and feed the padded result to A'.