Problem 1

Proof. First we show that $\text{SPACETM}$ is in $\text{PSPACE}$. For input $\langle M, w, 1^n \rangle$, we construct another TM $M'$ to simulate $M$. In addition, $M'$ has two counters. One counts the space used by $M$. The other one counts the number of steps running by $M$. Every time after $M'$ simulates one step of $M$, it updates the counters and checks whether the space used is larger than $n$ or the steps running by $M$ is larger than $2^{cn}$ for some constant $c$ which is related with $M$. If either is true, it rejects immediately. Note that the total number of possible configurations of $M$ is at most $2^{cn}$, thus if $M$ does not halt in this number of steps, then it must have gone into an infinite loop and $M'$ rejects. After simulating $M$ running on $w$, if $M$ accepts $w$ in space $n$, $M'$ accepts. Otherwise, $M'$ rejects. By definition of $\text{SPACETM}$, $M'$ decides $\text{SPACETM}$. Simulating $M$ only uses space $O(n)$. So $\text{SPACETM}$ is in $\text{PSPACE}$.

Next we prove that $\text{SPACETM}$ is $\text{PSPACE}$-hard. For every language $L$ in $\text{PSPACE}$, there is a DTM $M$ that decides $L$ using space $s(n)$ which is a polynomial of input length $n$. Consider the function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ such that $f(x) = \langle M, x, 1^{s(n)} \rangle$, where $m$ is a polynomial of $n$. The function $f$ can be computed in polynomial time and space. By definition of $\text{SPACETM}$, $x \in L$ if and only if $f(x)$ is in $\text{SPACETM}$. Thus $\text{SPACETM}$ is $\text{PSPACE}$-hard. So $\text{SPACETM}$ is $\text{PSPACE}$-complete.

Problem 2

Proof. By using binary strings to represent each player’s move, deciding which player wins at the end can be expressed as a Boolean formula $\phi$. Without loss of generality, let $\phi = 1$ indicate that player 2 wins. Thus player 2 has a winning strategy if and only if the following QBF is true.

$$\forall x_1 \exists x_2 \cdots \exists x_{2n} \phi(x_1, \ldots, x_{2n}).$$

If player 2 does not have a winning strategy, then the above QBF is false, and thus its negation must be true. That is,

$$-\forall x_1 \exists x_2 \cdots \exists x_{2n} \phi(x_1, \ldots, x_{2n}).$$

This means that

$$\exists x_1 \forall x_2 \cdots \forall x_{2n} \neg \phi(x_1, \ldots, x_{2n}).$$

Since $\phi(x_1, \ldots, x_{2n}) = 0$ means player 1 wins, this shows that there is a winning strategy for player 1. So one of the two players has a winning strategy.

Problem 3

Answer. (a) Because $\text{SC}$ can be decided by a TM running in both polynomial time and $\log^c n$ space for some constant $c$, $\text{PATH}$ is in $\text{NL} = \text{NSPACE}(\log n) \subseteq \text{SPACE}(\log^2 n)$ by Savitch’s Theorem. This means there is a DTM $M$ using $O(\log^2 n)$ space that can decide $\text{PATH}$. However, it does not tell us whether $M$ runs in polynomial time or not. So we do not know whether $\text{PATH}$ is in $\text{SC}$. 

1
(b) We do not know. Consider \text{polyL} \cap \text{P}. All the language in this set can be decided by a DTM \text{M}_1 using space \(O(\log^c n)\). It can also be decided by a DTM \text{M}_2 running in polynomial time. However, \text{M}_1 may not run in polynomial time and \text{M}_2 may use space \(\omega(\log^c n)\). This means that we do not know whether \(\text{SC} = \text{polyL} \cap \text{P}\).

\[\square\]

\textbf{Problem 4}

\textit{Proof.} (a) First consider the “if” part. Assume there is a directed cycle in \(G_\varphi\) in which both \(x_i\) and \(\overline{x_i}\) appear for some \(x_i\). No matter what value \(x_i\) is, one of \(x_i\) and \(\overline{x_i}\) is 0. Without loss of generality, assume \(\overline{x_i}\) is 0. Let the path from \(x_i\) to \(\overline{x_i}\) in the cycle be \(P = (x_i, v_1, \ldots, \overline{x_i})\). As \(\overline{x_i} = 0\), according to the definition of edge, in order to make \(\varphi = 1\), \(v_1\) must be 1. For the same reason, \(v_2\) must be 1. So in this way, if \(\varphi = 1\), it requires that \(v_1 = v_2 = \ldots = \overline{x_i} = 1\). However, \(\overline{x_i} = 0\). So there is no satisfying assignment for \(\varphi\).

Second consider the “only if” part. Suppose there is no directed cycle in \(G_\varphi\) in which both \(x_i\) and \(\overline{x_i}\) appear, for some variable \(x_i\). Consider the following algorithm.

While there are vertices that has not been assigned, choose one of them, denoting as \(v\), such that there is no path from \(v\) to \(\overline{v}\). There must exists such a \(v\), because if not, there is a directed cycle in \(G_\varphi\) in which both \(v\) and \(\overline{v}\) appear, for some variable \(x_i\), contradicting the assumption. Let \(v = 1\) and set all vertices reachable from \(v\) to be 1. The negation of all these vertices are set to be 0.

Finally we will get a assignment for all the vertices. We claim the assignment is consistent. Suppose not. It happens that for some vertex \(v\) and its negation \(\overline{v}\), after \(v\) is set to be 1, \(\overline{v}\) also need to be set to 1. In this situation, as \(\overline{v}\) should be set to 1, there is an incoming edge \((u, \overline{v})\) and \(u\) is set to be 1. However, \((u, \overline{v})\) is an edge if and only if \(\overline{u} \lor \overline{v} = \overline{v} \lor \overline{u}\) is a clause of \(\varphi\) which means \((v, \overline{u})\) is also an edge of \(G_\varphi\). It concludes that \(\overline{u}\) has already been set to 1 as \(\overline{u}\) can be reached from \(v\). This contradicts that \(u = 1\). So the assignment is consistent.

This assignment shows that \(\varphi\) is satisfiable.

Thus we proved the claim.

(b) As we know \(\text{NL} = \text{coNL}\), we first prove that \(\text{2SAT} \in \text{NL}\). First, given any 2CNF \(\varphi\), the graph \(G_\varphi\) is implicitly logspace computable, since to determine any bit in the adjacency matrix it suffices to check if the corresponding clause is in \(\varphi\), which only requires memory to remember the index of the two variables. Technically, a clause in a 2CNF \(\varphi\) may only has one literal \(u\). In this case, we view it as \(u \lor \overline{u}\) and create an edge \((\overline{u}, u)\). Now according to (a), \(\varphi \in \text{2SAT}\) if and only if there is a directed cycle in \(G_\varphi\) in which both \(x_i\) and \(\overline{x_i}\) appear for some \(x_i\). We can decide this in \(\text{NL}\), by choosing a start node \(x\) in the graph \(G_\varphi\) non-deterministically and then doing a non-deterministic walk of length at most the number of vertices in \(G_\varphi\); we accept if the walk encounters \(\overline{x}\) and then \(x\).

Thus \(\text{2SAT} \in \text{NL}\), which means \(\text{2SAT} \in \text{coNL}\). As \(\text{NL} = \text{coNL}\), \(\text{2SAT} \in \text{NL}\).

(c) For any input \(\langle G, s, t \rangle\) for \text{PATH}, we will construct a function \(f\) that is implicitly logspace computable such that \(\langle G, s, t \rangle \in \text{PATH} \iff f(G) \in \text{2SAT}\).

The function \(f\) is constructed as follows. Function \(f\) will output a 2CNF. Each variable in this formula corresponds to a vertex in \(G\). For each edge \(e = (u, v)\) in \(G\), if \(u \neq t\), construct clause \(C_e = (\overline{u} \lor v) \land (u \lor \overline{v})\) (This is true iff \(u = v\)). Let \(f(G) = (s \lor \overline{s}) \land (\overline{t} \lor \overline{t}) \land (\bigwedge_m C_e)\). For each clause \(C_e\), its construction can be done using \(O(\log n)\) space because we only need to record the corresponding edge. The construction of \((s \lor \overline{s})\) and \((\overline{t} \lor \overline{t})\) can also be done in space \(O(\log n)\). The length of \(f(G)\) can be computed in \(O(\log n)\) space because we only need to count the number of edges \(m\) of \(G\) and then do a constant number of computations with \(m\) and some constants. This means that \(f\) is implicitly logspace computable.

Assume \(\langle G, s, t \rangle \in \text{PATH}\). In order to make \(f(G) = 1\), we have to let \(s = 1\) and \(t = 0\). If there is an edge from \(s\) to \(u\) then \(u\) should be set to 1 in order to make \(f(G) = 1\). As a result, all vertices reachable
from $s$ should be set to 1. Similarly, every vertex $u$, which has a path to $t$, should be set to 0. Other vertices can all be set to 1. Since there is no path from $s$ to $t$, these assignments are consistent. So in this way there is an assignment such that $f(G)$ is satisfied. So $f(G) \in \text{2SAT}$.

Assume $f(G) \in \text{2SAT}$. Suppose there is a path in $G$ from $s$ to $t$. From the above analysis, in order to make $f(G) = 1$, we must set $s = 1$. Also all the vertices that can be reached from $s$ should be set to 1. This means $t = 1$. However in $f(G)$, $t$ is a clause. Thus $f(G)$ is not satisfiable, contradicting the assumption $f(G) \in \text{2SAT}$. So there is no path in $G$ from $s$ to $t$. That is $\langle G, s, t \rangle \in \text{PATH}$.

This proves that $\text{PATH} \leq_l \text{2SAT}$. Also as $\text{2SAT} \in \text{NL}$, $\text{2SAT}$ is NL-complete.