1. Show that 3COL is in PSPACE.

Solution. We know 3COL is in NP, and NP ⊆ PSPACE, so 3COL in PSPACE follows immediately.

Another way of showing this is by giving a polynomial space algorithm for 3COL. This is similar to the polynomial space algorithm for SAT:

On input $G$, with $G$ a graph with $n$ nodes:

(a) For each assignment of colours (red, green or blue) to the variables $x_1, \ldots, x_n$, check that each edge connects nodes with different colours

(b) If the check succeeds for one of the assignments then accept, otherwise reject.

As space can be reused when checking different assignments of colours to nodes, the space required by the above algorithm is linear in the size of the graph. In addition to storing the graph, all we need to store at any one time is a single colouring, plus a few variables to keep track of where we are.

2. Show that TQBF is in PSPACE.

Solution. We assume that the given fully quantified boolean formula is such that all the quantifiers appear at the beginning. Any quantified boolean formula can be put into this form (with an algorithm that uses linear space only!).

The following is a recursive PSPACE algorithm for TQBF: On input $\Phi$ (fully quantified boolean formula):

(a) If $\phi$ is of the form $\exists x.\psi$, recursively call the algorithm on $\psi[0/x]$ and $\psi[1/x]$; if one evaluates to true then accept, otherwise reject.

(b) If $\phi$ is of the form $\forall x.\psi$, recursively call the algorithm on $\psi[0/x]$ and $\psi[1/x]$; if both evaluate to true then accept, otherwise reject.

(c) If $\phi$ contains no quantifiers, then evaluate it; accept if true, otherwise reject.
The space required by this algorithm is $O(n + m)$ where $n$ is the number of variables and $m$ is the size of the formula - at each step in the recursion we only need to store the value of one more variable; and we need $O(m)$-space to evaluate a formula with no quantifiers.

3. Let $\phi$ be a fully quantified boolean formula of the form

$$\exists x_1 \forall x_2 \exists x_3 \ldots Q x_k \psi$$

where $Q$ can be either $\exists$ or $\forall$, and $\psi$ is a boolean formula in the variables $x_1, \ldots, x_k$. The formula game for $\phi$ is played by two players E and A. Player E selects values for the variables that are bound by the $\exists$ quantifier and player A selects values for the variables that are bound by the $\forall$ quantifier. The order of play is driven by the order of the quantifiers in the formula. Once all the values have been selected, the game is won by E if the resulting value of $\psi$ is true, and by A otherwise.

Let FORMULA\_GAME be the problem of deciding if player E has a winning strategy in the formula game associated with a given formula. Show that FORMULA\_GAME is PSPACE-complete.

Solution. FORMULA\_GAME is almost the same problem as TQBF; the only difference is that the two quantifiers ($\forall$ or $\exists$) alternate strictly in inputs to FORMULA\_GAME.

The PSPACE algorithm for TQBF also works for the more restricted type of formulas which are the inputs of FORMULA\_GAME. Hence, FORMULA\_GAME is in PSPACE.

To show that FORMULA\_GAME is PSPACE-hard, reduce TQBF to it in polynomial time: the reduction simply inserts dummy quantifiers to produce a formula of the restricted form required by FORMULA\_GAME. For example, $\forall x. \exists z. \forall y. x \land y$ is mapped to $\forall x. \exists z. \forall y. x \land y$.

4. Consider the two-player geography game:

- players take turns naming cities (there is a fixed set of cities they can choose from)
- the chosen city must begin with the same letter that ended the previous city
- player 1 starts with a designated city
- no repetitions are allowed

Show that deciding if the first player has a winning strategy in this game is in PSPACE.

Solution. The relationship between cities can be encoded using a graph $G$. Then, at each step, a player can select any node of the graph which was not previously chosen and which is connected to the current node.

The following recursive algorithm runs in PSPACE:
On input $\langle G, b, p \rangle$ (where $b$ denotes the current position and $p \in \{1, 2\}$ denotes the player):

(a) if $b$ has no outgoing edges, then reject (Player $p$ looses).
(b) remove $b$ and all edges connected to $b$ to get a new graph $G_1$.
(c) For each of the nodes $b_i$ in $G_1$ that $b$ originally pointed at, recursively call the algorithm on $\langle G_1, b_i, q \rangle$, with $\{q\} = \{1, 2\} \setminus \{p\}$.
(d) If all these accept (so player $q$ has a winning strategy in the original game), then reject, otherwise accept.

The above uses polynomial (linear) space since (i) for each recursive call, removing $b$ and its outgoing edges from the graph can be done with constant space (e.g. can use a flag to indicate if a node has been removed) and (ii) there are at most $n$ recursive calls (where $n$ is the number of nodes).

5. Show that if every NP-hard problem is also PSPACE-hard, then PSPACE = NP.

**Solution.** We know NP $\subseteq$ PSPACE. To show the other inclusion, let $X \in$ PSPACE. We want to show $X \in$ NP. We know SAT is NP-hard, so by the hypothesis SAT is also PSPACE-hard. So $X$ can be reduced to SAT in polynomial time. But SAT $\in$ NP, therefore also $X \in$ NP.

6. Show that if a decision problem $X$ has complexity $3^n$, then $X$ is in EXP-TIME.

**Solution.** We need to show that $3^n$ is $O(2^{nk})$ for some fixed $k$. We have:

$$3^n \leq 4^n \leq 2^{2^n} \leq 2^{n^2}$$

for $n \geq 2$, hence we can take $k = 2$ and we are done.