PSPACE

(from Sipser Definition 8.6) $PSPACE = \bigcup_k SPACE(n^k)$. $NPSPACE = \bigcup_k NSPACE(n^k)$.
Since $NSPACE(f(n)) \subseteq SPACE(f^2(n))$ (from last worksheet; "Savitch's Theorem"), $PSPACE = NPSPACE$.
(from Sipser Definition 8.8) A language B is PSPACE-hard if every A in PSPACE is polynomial time reducible to B. If B is also in PSPACE, it is PSPACE-complete.

a It might seem more natural to use polynomial-*space* reductions now that we're discussing space complexity. But prove that if we were to modify the above definition to use poly-space reductions, then *every* language in PSPACE (except two edge cases) would be PSPACE-complete. (Then fill in the analogous claim for time complexity, i.e. for what X would we say "Using poly-time reductions, every language in X (except two edge cases) is X-complete.")

b Show that PSPACE is closed under star. ($L^* = \{x_1 x_2 \cdots x_k \mid k \geq 0$ and each $x_i \in L\}$. Note that $|x_i|$ is not necessarily 1.)

c Show that NP $\subseteq$ PSPACE. Then show that any PSPACE-hard language is also NP-hard.

d We will now consider a harder version of SAT by adding quantifiers to our formulas. We will work with "fully quantified" formulas, meaning that there are no "free" variables, i.e. every variable must be in the scope of a corresponding quantifier (so e.g. $\forall x_1 \exists x_2, [x_1 \wedge x_2]$ is fine but $\forall x_1, [x_1 \wedge x_2]$ is not).

i For simplicity we will also only work with formulas in "prenex normal form", i.e. where all quantifiers appear at the beginning of the formula (and the whole formula is in their scope) - for example, $\forall x_1 \exists x_2, [x_1 \wedge x_2]$, but not $\forall x_1, [x_1 \wedge (\exists x_2, x_2)]$ or $\exists x_2, [(\forall x_1, x_1) \wedge x_2]$. Describe how to convert any fully quantified formula that is *not* in prenex normal form into one that is.

ii Define $TQBF = \{\langle\phi\rangle \mid \phi$ is a True fully quantified Boolean formula (in prenex normal form)$\}$. Prove that $TQBF \in PSPACE$.

e We will define a two-player game played using (prenex normal form) quantified boolean formulas. For each quantifier in order, if it is a $\forall$ then Player A chooses the value of the variable, and if it is a $\exists$ then Player E chooses the value of the variable. Once all variables have been assigned in this way, we see whether the unquantified part of the formula is true or false using this variable assignment. Player A wins on False, Player E on True. For example, if $\phi = \forall x_1 \exists x_2, [x_1 \wedge x_2]$, then Player A would go first and might choose $x_1 = F$, Player E would go next and might choose $x_2 = T$, and then Player A would win because $F \wedge T$ is False. Let $FORMULA\text{–}GAME = \{\langle\phi\rangle \mid$ Player E has a winning strategy in the formula game associated with $\phi\}$. (A player has a winning strategy if they will win so long as they play optimally.)

Prove that $TQBF \leq_P FORMULA\text{–}GAME$. (And then, though we won't prove this, $TQBF$ is PSPACE-complete so $FORMULA\text{–}GAME$ is too.)

f A *ladder* is a sequence of strings $s_1, s_2, \cdots, s_k$, wherein every string differs from the preceding one by exactly one character. For example, the following is a ladder of English words, starting with "head" and ending with "free": head, hear, near, fear, bear, beer, deer, deed, feed, feet, fret, free. Let $LADDER_{DFA} = \{\langle M, s, t\rangle \mid M$ is a DFA and there are strings in $L(M)$ which can be used to construct a ladder that starts with s and ends with $t\}$. Show that $LADDER_{DFA}$ is in PSPACE. *(Hint: PSPACE = NPSPACE)*

g Using our normal method of counting time and space complexity, there's not much point in talking about sub-linear complexity, since it takes n time and space just to read the whole input. Yet there are other models where we can talk about sub-linear complexity - for example, we normally consider binary search on real computers to take $O(log(n))$ time. What are two key features of that example which allow us to get sub-linear complexity? Design a new way of computing space complexity (which may involve using a slightly different model than a normal TM) which will allow us to meaningfully study sub-linear space complexity.