L and NL

a On the homework, we used "R2TM"s (2-tape TMs where the input tape is read-only) to create a new notion of space complexity WSPACE which can more meaningfully talk about sublinear complexity. Sipser introduces a version of the model with a few additional tweaks, which I'll call the S2TM: "We provide a way for the machine to detect when the head is at the left-hand and right-hand ends of the input. The input head must remain on the portion of the tape containing the input" (p349). Brainstorm why S2TMs might be more useful/convenient for discussing sublinear space complexity than R2TMs. (e.g. in what situations might they actually change which languages can be decided in some amount of space? in what ways might they better capture what we "really mean" by the space complexity of a problem? I'm just looking for some speculation; no exact answers are required.) Briefly consider how the new input-edge-detection feature could be reflected in the formal definition of a S2TM.

b We modify our definition of space complexity to now use S2TMs instead of TMs. (As in the HW, it'll be the same thing for linear space and above, and will help us discuss sublinear.) Define $\mathbf{L} = SPACE(\log n)$. For example, $\{0^n1^n \mid n \geq 0\} \in \mathbf{L}$, since we can decide it by checking that our input is of the form 0^*1^* and then counting the number of 0s and 1s to check for equality; the two counters take log space each. Show that $A_{DFA} \in \mathbf{L}$. *(Hint: an S2TM can store a pointer to a tape cell of its input in log space.)*

c Define $\mathbf{NL} = NSPACE(\log n)$. Show that $PATH = \{\langle G, s, t\rangle \mid G$ is a directed graph that has a (directed) path from s to $t\}$ is in $\mathbf{NL}$. *(Note that you can not do things like store the whole path or "mark nodes we've visited", since there are linearly-many nodes and only log space to work with.)*

d It is unknown whether **L**=**NL**. For P vs NP, we studied the issue using poly-time reductions ($\leq_P$) defined in terms of poly-time-computable functions, which we used to then define NP-completeness. Similarly, we will want log-space reductions ($\leq_L$) defined in terms of log-space-computable functions in order to define **NL**-completeness. Try to define "log-space-computable functions", keeping in mind that in logarithmic space you do not have space to store the input *or the output*. (You should create a new type of TM, in the same way that we defined S2TMs to handle the case where we can't store the input.) Sipser's definition appears on the next page; when you are done thinking here, compare your solution to that one and briefly think about whether any differences between your definitions seem important or just cosmetic.

e (Sipser Definition 8.21) A *log space transducer* is a TM with a read-only input tape, a write-only output tape, and a read/write work tape. The head on the output tape cannot move leftward, so it cannot read what it has written. The work tape may contain $O(\log n)$ symbols. A log space transducer M computes a function $f : \Sigma^* \to \Sigma^*$, where $f(w)$ is the string remaining on the output tape after M halts when it is started with w on its input tape. We call f a log space computable function. Language A is log space reducible to language B, written $A \leq_L B$, if A is mapping reducible to B by means of a log space computable function f.

Prove that if $A \leq_L B$ and $B \in L$, then $A \in L$. *(Note that given a log-space-computable function f, you can't actually just write "compute $f(w)$", since you don't have space to store the output. Hint: you are allowed to be* very *time-inefficient.)*

f (if you have extra time) Recall that a directed graph is *strongly connected* if every two nodes are connected by a (directed) path in each direction. Let STRONGLY-CONNECTED $= \{\langle G \rangle \mid G$ is a strongly connected graph$\}$. Show that STRONGLY-CONNECTED is **NL**-complete. (You may use without proof that PATH is **NL**-complete.)