CS 6810: Theory of Computing **Fall 2021**

Lecture 6: Space Complexity. September 14, 2021

Lecturer: Eshan Chattopadhyay *Scribe: Spencer Peters*

In this lecture, our model of computation is a multi-tape Turing machine. This model is appropriate for studying uniform space complexity. For studying non-uniform space complexity, there is a model called "branching programs" (analogous to Boolean circuits for non-uniform time complexity), but this will not be the focus of this lecture.

Definition 0.1. *A Turing machine M on input x uses space $S(|x|)$ if during its execution M uses at most $S(|x|)$ work tape cells.*

The point of excluding the input cells in this definition is to allow for sublinear-space computation.

Definition 0.2. *We say that a language L is in $DSPACE(S(n))$ if there is a TM M accepting L such that for all $n \geq 1$ and all x with $|x| = n$, M uses space $S(n)$.*

Unlike for deterministic time complexity, here $S(n) = o(n)$ is also interesting. However, we will usually assume $S(n) \geq \log(n)$; otherwise the machine cannot keep track of which bit of the input it is reading.

We can define $NSPACE(S(n))$ analogously to $NTIME(T(n))$. That is, $L \in NSPACE(S(n))$ iff there is a nondeterministic TM M accepting L such that for all n, all x with $|x| = n$, and all nondeterministic choices of the machine, the execution of M uses at most $S(n)$ tape cells. (Of course, this definition can be equivalently formulated in terms of deterministic machines and witnesses.)

Analogous to time complexity:

Definition 0.3.

$$PSPACE := \bigcup_{c \geq 1} DSPACE(n^c);$$

$$NPSPACE := \bigcup_{c \geq 1} NSPACE(n^c)$$

Notice that $P \subseteq PSPACE$ since the number of work tape cells used by a TM M is at most the number of steps in M's execution. However, this inclusion is not known to be strict! The statement $P \neq PSPACE$ is an unproven conjecture that is weaker than $P \neq NP$. Notice also that $NP \subseteq PSPACE$; given a problem in NP, we can decide it in polynomial space by enumerating all witnesses up to the polynomial upper bound for the problem, running the polynomial-time verifier on each witness (reusing space), and accepting if the verifier ever accepts.

Next, what can we say about the relationship between space and time complexity? To study this we introduce the notions of *configuration* and *configuration graph*.

A *configuration* of a TM M at a particular computation step consists of all the data required to continue running M from that step. That is, it consists of the symbols on the worktape, the input and work tape head locations, and the state of the TM. We do not include the output head or the symbols written on the output tape in the configuration. The main point is that if M repeats a configuration once, it will repeat the configuration infinitely many times and never halt. We can

use this to show that if a machine halts after t steps, it must use at least (roughly) $\log t$ space; otherwise by pigeonhole it must repeat a configuration.

More carefully: if a TM M uses $S(n)$ space, then M running on an input x such that $|x| = n$ reaches at most $|\Sigma|^{S(n)}| \cdot n \cdot S(n)^w \cdot Q$ configurations, where w is the number of worktapes, and Q the number of states in the machine's finite control. Since q, w, and $|\Sigma|$ are finite, and we assume $S(n) \geq \log(n)$, it follows that M reaches at most $2^{O(S(n))}$ configurations. Since M cannot repeat a configuration unless it does not halt, it follows

Claim 0.4. $DSPACE(S(n)) \subseteq DTIME(2^{O(S(n)})$.

An immediate corollary is that $PSPACE \subseteq EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$.

The *configuration graph* of M on input x, denoted $G_{M,x}$, is a directed graph whose nodes are all possible configurations of M on input x, and whose edges are all pairs of configurations (C, C') such that M can transition from C to C' in a single step.

We can efficiently check if an edge (C, C') is in the configuration graph, by, e.g., for each tape, checking for each triple of adjacent work-tape cells if (1) the head is at the middle cell of the triple (2) the symbols on the cells and the states recorded in C and C' match the transition function of M, and (3) the symbols elsewhere on the tape are the same in both C and C'. Formally, we have the following claim, which we will not prove rigorously (refer to the Barak textbook for details).

Claim 0.5. *There exists an $O(S(n))$-sized CNF formula $\phi_{M,x}$ such that $\phi_{M,x}(C, C') = 1$ iff (C, C') is an edge of $G_{M,x}$.*

Let C_{start} denote the starting configuration of M on x, and assume without loss of generality (e.g., by adding cleanup code) that when M accepts, it halts in the configuration C_{accept}. Then it is clear that M accepts x iff there is a path from C_{start} to C_{accept} in $G_{M,x}$.

We will spend the rest of the lecture using the configuration graph to prove the following important result, which is a more refined version of our earlier observation that $NP \subseteq PSPACE$.

Theorem 0.6 (Savitch's Theorem)**.** *For any $S(n) \in \Omega(\log n)$, we have*

$$NSPACE(S(n)) \subseteq DSPACE(S(n)^2).$$

In particular, $PSPACE = NPSPACE$.

Proof. Recall that if $L \in NSPACE(S(n))$, then there is a non-deterministic TM M (using $S(n)$ space) accepting L. Recall that M accepts x iff there is a path from C_{start} to C_{accept} in $G_{M,x}$. So abstractly, we have a large graph specified by an edge oracle (the CNF formula), and we want to determine st-connectivity. One natural approach is to start at C_{start} and follow directed edges until either C_{accept} is reached, or there are no new nodes to explore. But in the worst case this amounts to writing out $G_{M,x}$, which uses $2^{O(S(n))}$ space. We need a more space-efficient algorithm.

Let $N \leq 2^{O(S(n))}$ be the number of nodes in $G_{M,x}$. Define $Reach(u, v, i) = 1$ iff there is a path of length $l \leq 2^i$ from u to v in $G_{M,x}$. Notice that there is a path from u to v in $G_{M,x}$ iff $Reach(u, v, \log N) = 1$. So our approach is to design a deterministic TM M' that accepts iff $Reach(C_{start}, C_{accept}, \log N) = 1$. But

$$Reach(u, v, i) = 1 \Leftrightarrow \exists w : Reach(u, w, i-1) = 1 \wedge Reach(w, v, i-1) = 1.$$

So M' can calculate $Reach(u, v, i)$ by iterating over all nodes w, and for each w, making two recursive calls. Each iteration can reuse space; the only space shared by the iterations is a counter keeping track of the index $i \in N$ of the node w, which takes up $\log N$ space. Moreover, the

two recursive calls can be done sequentially, using the same space. Thus the space $s(i)$ used by M' satisfies $s(i) = \log N + s(i-1)$. Unrolling the recurrence (using our CNF formula to get $s(0) = S(n) = O(\log N)$), we get $s(i) = O(i \log N)$. In sum, M' decides the same language as M in space $s(\log N) = O(\log^2 N)$ space. Since $N = 2^{O(S(n))}$, $M' \in DSPACE(O(S(n)^2))$, as desired. □

As we just saw, questions about space complexity can often be reduced to combinatorial questions–in this case, we were able to reduce a question about the input TM M (accept/reject) to a question about its configuration graph (directed st-connectivity).