I will cover the material in Chapter 4 of the text.

1 Computational Model

1.1 Deterministic Model

The model is pictured in Figure 4.1 of the text. The model presented in class is almost identical, except it is closer to the original Turing machine model: a space bounded Turing machine has a separate input tape (which is read only), a single worktape, and an output tape that is write only and one way (so an output square, once written, cannot be changed.) It is $S(n)$ space bounded, if for all inputs of length $n$, it never uses more that $S(n)$ squares of its worktape. $DSPACE(S(n))$ is the set of problems decidable in $O(S(n))$ space.

Remark 1. 1. The model in the text makes explicit that the computer can ‘remember’ a character (it uses a Register to do so.) We have seen that this could be done by a Turing machine by enlarging the set of states $Q$ to a new set $Q \times \Sigma$ where $\Sigma$ is the set of characters.

2. We do not need more than one worktape. We saw that one could simulate a constant number $k$ of tapes by a single one. The simulation takes quadratic time, but that is of no concern, as we are measuring the space used.
3. We use a separate input tape because if we were allowed to write on the input tape, all space bounds would have to be at least linear. As we shall see, there are powerful sublinear algorithms.

1.2 Nondeterministic Model

We used ‘nondeterminism’ to denote acceptance by proof systems. We shall use the same idea here: we define nondeterministic acceptance by existential quantification. More precisely, a language $L$ is accepted in nondeterministic space $S(n)$ if there is an $S(n)$ space bounded Turing machine $M$ such that $L = \{x : \exists y M(x, y) = 1\}$.

A machine model of this process could be given by having the deterministic machine $M$ have a separate ‘proof’ or ‘guess’ tape, a 1-way read only extra input tape. Then $x$ is accepted iff there is a proof $z$ such that $M$ accepts $x$ when given the proof $z$. The worktape used on input $x$ is the minimum worktape used by $M$, for all $z$ that make $M$ accept $x$.

An alternative model (presented in the text ahead of the one we described) is to use ‘nondeterministic Turing machines’ that have a transition function $\delta$ that can associate different triples ($state$, $character$, $direction$) to an input to $\delta$. The machine then accepts iff there is a sequence of valid transitions of $M$ starting with $M$ in its initial configuration with input $x$, and ending in an accepting configuration.

It is not hard to see that the two models are equivalent: if $z$ is (encoding of) the choice of the transitions made by a nondeterministic machine (using the alternative definitions) then the first definition machine can be provided with the string $z$ as a valid proof, and conversely.

For reasons that will become clearer, unless otherwise specified we assume that all space bounds $S(n)$ satisfy $S(n) = \Omega(\log n)$.

1.3 Interesting Space Complexity Classes

- $L = DSPACE(\log n)$
- $NL = NSPACE(\log n)$
- $PSPACE = \bigcup_{c>0} DSPACE(n^c)$
- $NPSPACE = \bigcup_{c>0} NSPACE(n^c)$
2 Easy Theorems

There is a small additional difficulty when looking at space bounded computations (as opposed to time bounded ones): even a deterministic computation can have infinite loops. If a configuration repeats, the times at which this occur are distinct. With bounded space, it is possible to return to the same configuration, not using any additional worktape squares.

**Lemma 2.** An $f(n)$ space bounded Turing machine has only $2^{O(f(n))}$ different configurations.

**Proof.** We will use the fact that $f(n)\Omega(\log n)$.

Suppose the input has length $n$, the alphabet is $\Sigma$, and the set of states is $Q$. Then there are at most

- $|\Sigma|^{f(n)}$ different worktapes
- $f(n)$ places for the worktape head position
- $n$ places for the input head position
- $|Q|$ possible states.

Now $|\Sigma| = 2^a$ for some constant $a$, so the first term is $O(2^{O(f(n))})$. The same bound applies to all other terms (where we used that $f(n) = \Omega(\log n)$, and therefore $n = O(2^{O(f(n))})$). Multiplying together terms that are $2^{O(f(n))}$ yields a quantity that is $2^{O(f(n))}$. (we used that $O(2^{O(f(n))} = 2^{O(f(n))}$)

**Corollary 3.** For every $f(n)$ (deterministic) space bounded Turing machine $M$ there is a constant $c_M$ such that if for some input $x$ of length $n$ $M$ takes more than $2^{c_M f(n)}$ steps, it is in an infinite loop.

(Because if a configuration repeats, it will repeat again, and the machine is in an infinite loop.)

The similar corollary for nondeterministic machines is that if there is an accepting computation, there is one that takes no more than $2^{c_M f(n)}$ steps. (Prove!)

**Corollary 4.** If $f(n)$ is space constructible then for any $f(n)$ space bounded Turing machine $M$ there is an equivalent machine $M'$ that is $f(n)$ space bounded and halts for all inputs.
Proof. We can use a counter up to $2^{c_M f(n)}$. Such a counter requires $c_M f(n)$ bits. We can modify $M$ to compute $f(n)$ (which can be done in $O(f(n))$ space because $f(n)$ is space constructible). We increment the counter after simulating a move of $M$. If there is an overflow, we reject.

If $M$ accepts within the required time, we will not have to reject based on counter overflow.

Exercise: Provide the missing details for nondeterministic machines.

**Theorem 5.** For ‘nice’ $f(n)$

\begin{align*}
i & \text{DTIME}[f(n)] \subseteq \text{DSPACE}[f(n)] \\
ii & \text{NTIME}[f(n)] \subseteq \text{NSPACE}[f(n)] \\
iii & \text{DSPACE}[f(n)] \subseteq \text{DTIME}[2^{cf(n)}] \text{ for some constant } c > 0 \\
iv & \text{NSPACE}[f(n)] \subseteq \text{NTIME}[2^{cf(n)}] \text{ for some constant } c > 0
\end{align*}

Proof. The first two follow from the fact that in order to use a new square of the worktape, the Turing machine must use a step.

The second two follow from the arguments above.

3 Graph Reachability with Limited Space

We can view the following as a straightforward algorithmic problem (Graph Reachability): How much space is required to decide whether two designated vertices, $s$, and $t$, of directed graph $G=(V,E)$ are connected by a path?

We know that the reachability problem is easy without the restriction: linear time algorithms like depth first search or breadth first search solve it. Note that an $n$ vertex graph can be specified with $O(n^2)$ bits, and each vertex can be addressed with log $n$ bits.

**Lemma 6.** If for all vertices the outdegree is at most 1 the Reachability problem can be solved in $L$ ($=\text{DSPACE}(\log n)$).

Proof. (sketch) Follow the unique edge from $s$, and the unique edges from subsequent vertices until you either reach $t$ (success), or a vertex of outdegree 0 (failure) or enter a loop (failure). The latter can be detected by a counter, as above. 

\[ \]
Corollary 7. Graph Reachability is in NL (for arbitrary graphs).

Proof. (sketch) Choose the outgoing edge nondeterministically, then use the outdegree 1 algorithm above.

The result above used nondeterminism in an essential manner. It is harder to see what we could do deterministically, since the length of the path from \( s \) to \( t \) may be \( O(n) \) which is exponential in \( \log n \).

However, there is an elegant efficient space bounded algorithm (due to Savitch), that uses divide and conquer.

Theorem 8. Graph Accessibility in \( DSPACE[(\log n)^2] \)

Proof. (sketch) Consider the predicate

\[
\text{Reach}(a, b, w) = \exists \text{ a directed path of length at most } w \text{ from } a \text{ to } b
\]

Now \( \text{Reach}(a, b, 1) \) is easy. Moreover, suppose there is a path of length \( 2^k \) from \( a \) to \( b \). Then there is a vertex \( m \) (the ‘midpoint’) such that there is a path of length at most \( 2^{k-1} \) from \( a \) to \( m \), and a path of length at most \( 2^{k-1} \) from \( m \) to \( b \).

So,

\[
\text{Reach}(a, b, 2^k) \text{ if } \exists m \text{ Reach}(a, m, 2^{k-1}) \text{ AND Reach}(m, b, 2^{k-1})
\]

We want \( \text{Reach}(s, t, n) \)

In the algorithm we try all possible \( m \) and use recursion, with a twist--this is where we can take advantage of space being reusable: after we computed (recursively) \( \text{Reach}(a, m, 2^{k-1}) \), if the answer is successful, we only need to keep a 1-bit result, and can reuse the space to compute the second part, \( \text{Reach}(m, b, 2^{k-1}) \). So the space needed for a single recursive call is \( O(\log n) \), and the depth of the recursion is \( O(\log n) \), so the total space needed is \( O((\log n)^2) \).

\[ \Box \]

4 Complete Problems

The graph accessibility result is important for another reason.

Consider an \( S(n) \) tape bounded nondeterministic Turing machine, its input \( x \), and build the following directed graph \( G_{M,x} = (V, E) \). \( V \) is the set
of all configurations of $M$ on $x$. (We saw that this is a set of cardinality $2^{O(f(|x|))}$). The edges $E$ are the ordered pairs $a, b$ such that there is a valid move that leads from configuration $a$ to configuration $b$.

Let us also make the nondeterministic machine ‘clean up’: there is a unique accepting configuration $t$ (modify the machine so that before accepting it erases its worktape, brings its heads to the first square of the tape and THEN go to the unique accepting state $q_Y$).

Let $s$ be the initial configuration of $M$ with input $x$.

Now there is a path from $s$ to $t$ in $G_{M,x}$ iff $M$ accepts $x$.

The results of the previous section show that this can be decided in space $O(S(n)^2)$. Therefore we have the Theorem

**Theorem 9.** $\text{NSPACE}[S(n)] \subseteq \text{DSPACE}[S(n)^2]$

**Corollary 10.** $L \subseteq \text{DSPACE}[(\log n)^2]$

**Corollary 11.** $\text{NPSPACE} = \text{PSPACE}$

*Proof.* If $L \in \text{NPSPACE}$, there is some $c > 0$ such that $L \in \text{NSPACE}[n^c]$. By Savitch’s Theorem, $L \in \text{DSPACE}[n^{2c}]$, and so $L \in \text{PSPACE}$.

Next class we’ll show that given $m$ and $x$, the graph $G_{M,x}$ can be ‘built’ in $L$, and the Graph Accessibility problem is a complete problem for $\text{NL}$ under reductions in $L$ (and, I hope this sentence will make sense to you then.)