Space Complexity

Let $S : \mathbb{N} \rightarrow \mathbb{N}$. A deterministic Turing Machine $M$ is said to run in space $S(n)$ if on input $x \in \{0,1\}^*$, $M$ halts, having visited at most $c \cdot S(n)$ locations on its work tapes. A non-deterministic Turing Machine $M$ is said to run in space $S(n)$ if on input $x \in \{0,1\}^*$, $M$ halts, independent of its non-deterministic choices, and visits at most $c \cdot S(n)$ locations on its work tapes.

Note: The number of locations on the input tape visited by $M$ are not counted. So it makes sense to talk about a machine $M$ running in space $S(n)$.

$$\text{DSPACE}(S(n)) = \{ L \subseteq \{0,1\}^* | L \text{ is decided by a DTM that runs in space } S(n) \}$$

$$\text{NSPACE}(S(n)) = \{ L \subseteq \{0,1\}^* | L \text{ is decided by a NDTM that runs in space } S(n) \}$$

$$\text{PSPACE} = \bigcup_{c > 0} \text{DSPACE}(c^c)$$

$$\text{NPSPACE} = \bigcup_{c > 0} \text{NSPACE}(c^c)$$

$L = \text{DSPACE}(\log n)$

$NL = \text{NSPACE}(\log n)$

In Chapter 4 we explore the following questions:

1. What are relationships between time complexity & space complexity classes?
2. Can we prove space hierarchy theorems?

3. What are relationships between deterministic & non-deterministic space complexity classes?

4. Is there a notion of hardness for space complexity classes? (Related question: do notions of space bounded reductions make sense?)

Theorem: For every space constructible $S : \mathbb{N} \to \mathbb{N}$

\[
\begin{align*}
\text{DTIME}(S(n)) & \subseteq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n)) \subseteq \text{DTIME}(O(S(n)))
\end{align*}
\]

**Proof:**
(a) If $L \leq \text{DTIME}(S(n))$ then there is a DTM $M$ that runs in time $S(n)$ and decides $L$. Since $M$ runs in time $S(n)$, it also runs in space $S(n)$, so $L \leq \text{DSPACE}(S(n))$.

(b) If $L \leq \text{DSPACE}(S(n))$ then there is a DTM $M$ that runs in space $S(n)$ and decides $L$. $M$ is trivially an NDTM as well and therefore $L \leq \text{NSPACE}(S(n))$.

(c) To prove $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(O(S(n)))$ we introduce the notion of a configuration graph.

For a machine $M$ and input $x$, the configuration graph $G$ has vertex set equal to the configurations of $M$ $\langle M, x \rangle$ and directed edges $(C_1, C_2)$ connecting a config. $C_1$ to a config. $C_2$ if the machine $M$ takes config. $C_1$ to config. $C_2$ in one step.

What is a configuration of $M$? Suppose $M$ runs in space $S(n)$. Then for some const. $C$, the cells of work tapes that $M$ visits during its computation on input $x$ is represented by.
A configuration of $M$ on input $x$ is (i) $M$'s state, (ii) positions of all read & read/write heads, & (iii) contents of the cells shown above.

One of these configurations is the start configuration. Now assume that $M$ cleans up all its tapes when done with its computation. So when $M$ is done its work tapes look like:

Furthermore, assume that $M$'s read head (i.e., the head on the input tape) is also on the leftmost cell. This ensures that there is a unique accept configuration.

All of this discussion is independent of whether $M$ is deterministic or non-deterministic. If $M$ is det., then every node has out-degree 1 and if $M$ is non-det., then every node has out-degree 2.

Now suppose $L \in \text{NSPACE}(S(n))$. Then there is a non-det. TM $M$ that decides $L$ in space $S(n)$. We will now construct a det. TM $M'$ that decides $L$ in time $2O(S(n))$. On input $x$, $M'$ does a breadth-first search on the config. graph $G$, starting for the start configuration and searching $M,x$ for the accept configuration.
Note that the configuration graph has
\[ 1 \times (c \cdot S(1\times 1))^{k+1} \times M_{\text{nodes}} \]
state info. head positions contents of cells
\[ \leq 2 \quad \text{for some large enough } C. \]

Also, each node has out-degree \( \theta \leq 2 \). Hence, the number of edges is also
\[ \leq 2 \cdot C \cdot S(1\times 1) \quad \text{for some large enough const. } C'. \]

The size of the conf. graph \( G = M, x \) \( \frac{O(S(1\times 1))}{O(S(1\times 1))} \)

The number of vertices is \( 2 \cdot O(S(1\times 1)) \) BFS on a graph of this size runs in time \( 2^{O(S(1\times 1))} \) (since BFS runs in time \( O(m+n) \) on a graph with \( m \) edges and \( n \) vertices). Thus \( M' \) runs in time \( 2^{O(S(1\times 1))} \) and determines if \( x \in L \).
Hence, \( L \in \text{DTIME}(2^{O(S(1\times 1))}) \).

**Theorem:** \( \text{NP} \subseteq \text{PSPACE} \).

**Proof:** Think about how you would solve SAT in polynomial space! and a polynomial \( p \)

Consider \( L \in \text{NP} \), then there is a (det)TM \( M \) such that

\[ \forall x \in \{0,1\}^* : x \in L \iff \exists u \in \{0,1\}^p(1\times 1) : M(x,u) = 1 \]

and \( M \) runs in time \( q(n) \) for some polynomial \( q \).

Using \( M \) one can construct a TM \( M' \)
that runs in polynomial space and decides $L$.

\begin{algorithm}
\textbf{Algorithm for $M'$}
\begin{algorithmic}
\State \textbf{Input:} $x \in \{0, 1\}^*$
\For{each $u \in \{0, 1\}^*$}
\State \textbf{do}
\State \textbf{Accept if } $M(x, u) = 1$
\EndFor
\State \textbf{Reject}
\end{algorithmic}
\end{algorithm}

Each $u$ requires $p(1x1)$ space & this space is recycled. Imagine that each call to $M$ involves computation that uses an entirely separate set of tapes. Since $M$ runs in time $q(n)$, $M$ takes space $1x1 + p(1x1) + q(1x1)$, bounded above by a polynomial in $1x1$.

Thus the total space used by $M'$ is bounded above by a polynomial in $1x1$.

So $P \subseteq NP \subseteq \text{PSPACE}$

Of course, we don't yet know if this inclusion is strict.

In fact, we have no idea if $P = \text{PSPACE}$.