Space Complexity

Let $S: \mathbb{N} \to \mathbb{N}$. 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 $S(n)$.

$$DSPACE(S(n)) = \left\{ L \subseteq \{0,1\}^* \;\middle|\; \begin{array}{l} L \text{ is decided by a DTM that} \\ \text{runs in space } S(n) \end{array} \right\}$$

$$NSPACE(S(n)) = \left\{ L \subseteq \{0,1\}^* \;\middle|\; \begin{array}{l} L \text{ is decided by a NDTM that} \\ \text{runs in space } S(n) \end{array} \right\}$$

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

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

$$L = DSPACE(\log n)$$

$$NL = NSPACE(\log n)$$

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? (<u>Related question</u>: do notions of space bounded reductions make sense?)

<u>Theorem</u>: For every space constructible $S: \mathbb{N} \to \mathbb{N}$

$$DTIME(S(n)) \overset{(a)}{\subseteq} DSPACE(S(n)) \overset{(b)}{\subseteq} NSPACE(S(n)) \overset{(c)}{\subseteq} DTIME(2^{O(S(n))})$$

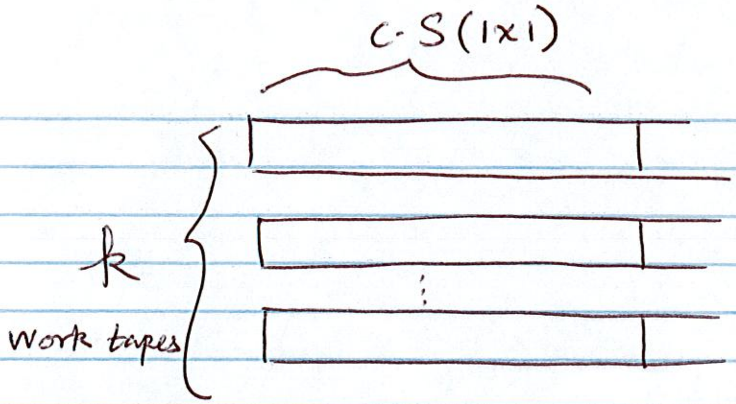
<u>PROOF</u>: (a) If $L \in DTIME(S(n))$ then there is a DTM M that runs in time $S(n)$ and decides L. Since M runs in time $S(n)$, it also runs in space $S(n)$. $\therefore$ $L \in DSPACE(S(n))$.

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

(c) To prove $NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$ we introduce the notion of a <u>configuration graph</u>.

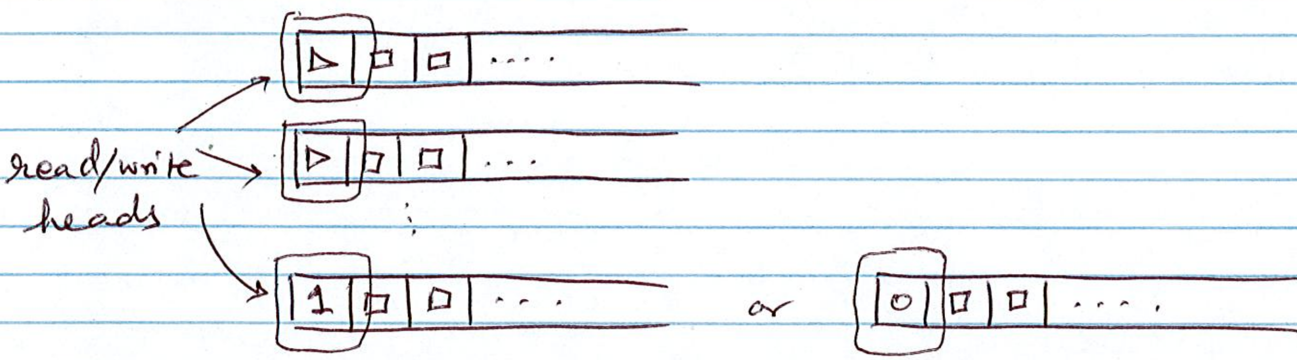
For a machine M and input x, the configuration graph $G_{M,x}$ has vertex set equal to the <u>configurations</u> of M and directed edges (C_1, C_2) connecting a config. C_1 to a config. C_2 if machine M takes config. C_1 to config. C_2 in one step.

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 <u>start configuration</u>. 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 left most cell. This ensures that there is a unique <u>accept configuration</u>.

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 $2^{O(S(n))}$. On input x, M' does a <u>breadth-first search</u> on the config. graph $G_{M,x}$ starting for the <u>start configuration</u> and searching for the <u>accept configuration</u>.

Note that the configuration graph has

$$\underbrace{|Q|}_{\text{State info.}} \times \underbrace{(S(|x|))^{k+1}}_{\text{head positions}} \times \underbrace{|\Gamma|^{k \cdot c \cdot S(|x|)}}_{\text{contents of cells}} \quad \text{nodes.}$$

~~nodes~~

$$\leq 2^{C \cdot S(|x|)}$$ (capital C) for some large enough const. C.

Also, each node has out-degree ≤ 2. Hence, the number of edges is also

$$\leq 2^{C' \cdot S(|x|)}$$ for some large enough const. C'.

The size of the conf. graph $G_{M,x}$, i.e., # of edges + # of vertices is $2^{O(S(|x|))}$. BFS on a graph of this size runs in time $2^{O(S(|x|))}$ (since BFS runs in time $O(m+n)$ on a graph with m edges and n vertices). Thus M' runs in time $2^{O(S|x|)}$ and determines if $x \in L$. Hence, $L \in DTIME(2^{O(S(n))})$. □

Theorem: $NP \subseteq PSPACE$.

PROOF: Think about how you would solve SAT in polynomial space!

Consider $L \in NP$. Then there is a (det) TM M and a polynomial p such that

$$\forall x \in \{0,1\}^* : x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)} : M(x,u) = 1$$

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.

Algo. for M'
INPUT: $x \in \{0,1\}^*$
for each $u \in \{0,1\}^{p(|x|)}$ do
    Accept if $M(x,u)=1$
Reject

Each u requires $p(|x|)$ 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 $|x| + p(|x|) + q(|x|)$.
Thus the total space used by M' is ~~[illegible]~~ bounded above by a polynomial in $|x|$. $\square$

So $P \subseteq NP \subseteq PSPACE$

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

In fact, we have no idea if $P = PSPACE$.