Space Complexity

- Time and space are two of the most important considerations when we seek practical solutions to computational problems.
- Space complexity shares many of the same features of time complexity
- Therefore, space complexity serves as a further way of classifying problems according to their computational difficulty.
- We will continue to use the TM as our model, but now we look at the tape space it consumes during its computation.

Space Complexity

Definition: Let M be a deterministic TM that halts on all inputs. We define the ***space complexity*** of M to be the function $f : \mathbb{N} \longrightarrow \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans on any input of length n. If the space complexity of M is $f(n)$, then we also say that M runs in space $f(n)$.

If M is a nondeterministic TM wherein ***all branches halt on all inputs*** we define its space complexity $f(n)$ to be the maximum number of tape cells that M scans on any branch of its computation on any input of length n.

Space Complexity

Definition: Let $f : \mathbb{N} \rightarrow \mathbb{R}^+$ be a function. The ***space complexity classes***, $SPACE(f(n))$ and $NSPACE(f(n))$, are defined as follows,

$$\begin{aligned} SPACE(f(n)) &= \{L \mid L \text{ is decided by an } O(f(n)) \text{ space TM}\} \\ NSPACE(f(n)) &= \{L \mid L \text{ is decided by an } O(f(n)) \text{ space NTM}\} \end{aligned}$$

$SAT \in SPACE(n)$

Theorem:

$$SAT \in SPACE(n)$$

Proof: By construction.

M = "On input $\langle \phi \rangle$, where ϕ is a Boolean formula:

```
1. For each truth assignment to variables $x_1, \ldots, x_m$ of $\phi$:
2.       Evaluate $\phi$ on that truth assignment.
3. If $\phi$ ever evaluates to $true$, accept; otherwise, reject."
```

This algorithm does not run in polynomial time, there are an exponential number of assignments to the variables,

$$\underbrace{2 \cdot 2 \cdot \ldots \cdot 2}_{m} = 2^m.$$

Therefore, as expected, this algorithms runs in $2^{O(n)}$ deterministic time where $n = |\phi|$. However, observe that it runs in $O(n)$ deterministic space. The insight is that we can reuse the space of the variables for the various assignments but we cannot reuse the time.

Savitch's Theorem

There is an interesting relationship between deterministic and nondeterministic space complexity in the sense that simulating a nondeterministic machine on a deterministic machine incurs at most a polynomial deterministic space penalty. Formally,

Theorem: (Savitch's Theorem) For any function $f : \mathbb{N} \rightarrow \mathbb{R}^+$, where $f(n) \geq n$,

$$NSPACE(f(n)) \subseteq SPACE(f^2(n)).$$

Proof Sketch: Naive, brute force simulation doesn't work, but if we are clever and partition the simulation then we can reuse space – *cif.* Quicksort. $\Box$[a]

[a] See website for a nice proof. Source: Wayne Goddard, An Introduction to the Theory of Computing, 2008.

$PSPACE$

Definition: $PSPACE$ is the class of languages that are decidable in polynomial space on a deterministic Turing machine. Formally,

$$PSPACE = \bigcup_k SPACE(n^k), \text{ with } k > 0.$$

Similarly, we define $NPSPACE = \bigcup_k NSPACE(n^k)$, for $k > 0$. But from Savitch's theorem it follows that

$$NPSPACE = PSPACE.$$

$NPSPACE = PSPACE$

Theorem: $NPSPACE = PSPACE$

Proof: We first show that $PSPACE \subseteq NPSPACE$. For any language $L \in PSPACE$ there exists a deterministic TM that decides that language in polynomial space. However, any deterministic TM can be seen as a special case of a non-deterministic TM. Therefore, $L \in NPSPACE$. It follows that $PSPACE \subseteq NPSPACE$.

We now show that the converse also holds. Let $L \in NPSPACE$ be a language that is decided by some non-deterministic TM in polynomial space, say $L \in NSPACE(n^j)$. Clearly, $L \in NPSPACE$. From Savitch's Theorem we know that $NSPACE(n^j) \subseteq SPACE(n^{j+2})$. Therefore, $L \in SPACE(n^{j+2})$. It follows that $L \in \bigcup_k SPACE(n^k)$ and therefore $L \in PSPACE$. This shows that $NPSPACE \subseteq PSPACE$.

Therefore, $NPSPACE = PSPACE$.$\square$

A Hierarchy of Complexity

We can now construct a hierarchy of complexity classes.

Observe that $P \subseteq PSPACE$. To see this let a machine execute in $t(n) \geq n$ polynomial time. But this implies that the machine can use at most $t(n)$ polynomial space (one cell per computation step).

In a similar argument we have $NP \subseteq NPSPACE$ and from the argument above $NP \subseteq PSPACE$.

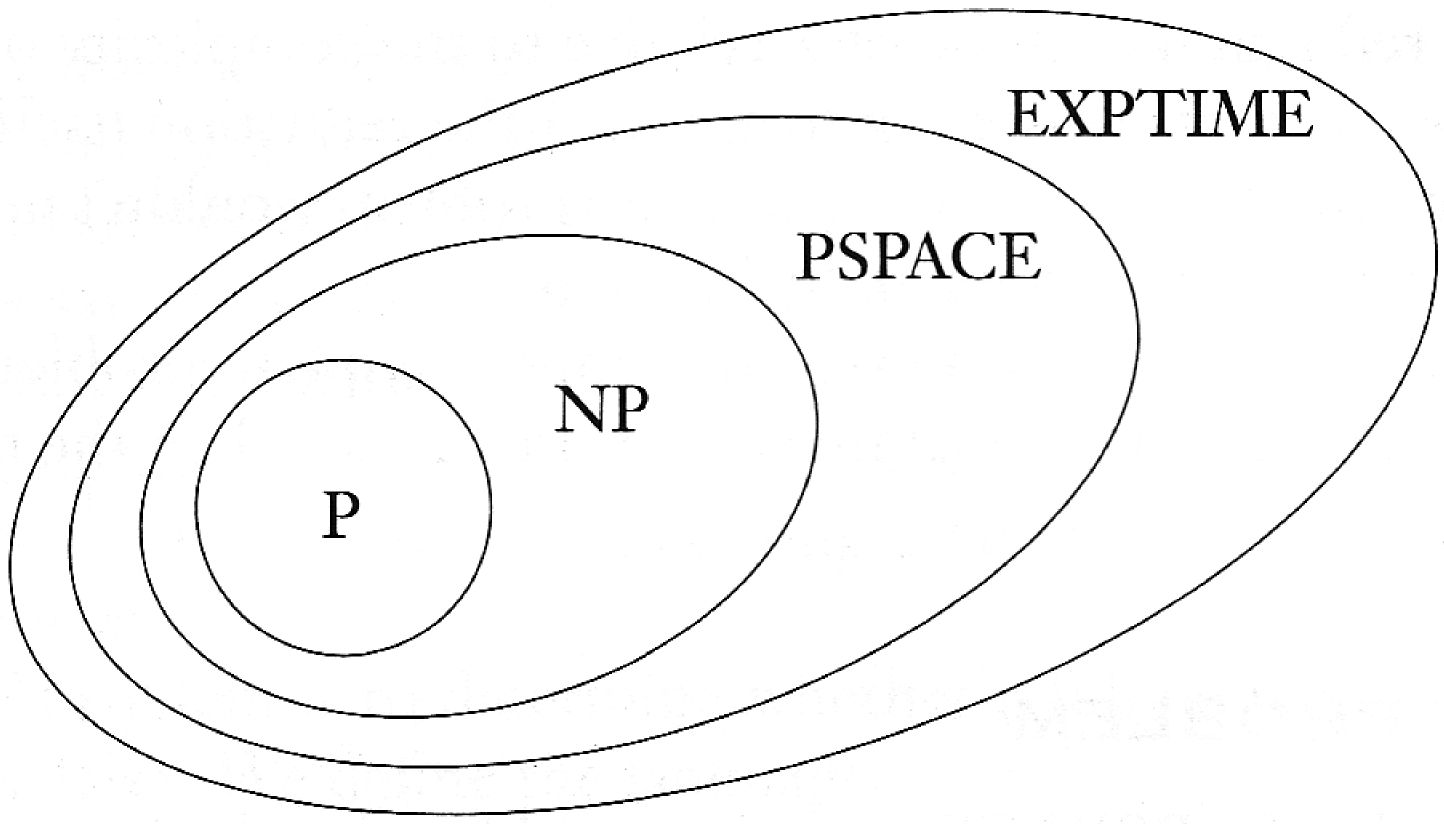
We already know that $P \subseteq NP$.

Finally, if we have a $f(n) \geq n$ polynomial space machine it can be shown that this machine runs in at most $2^{O(f(n))}$ exponential time. Thus, $PSPACE \subseteq EXPTIME$.

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME.$$[a]

[a] It is believed that all the set containments are proper.

A Hierarchy of Complexity


EXPTIME
PSPACE
NP
P