Lecture 21:
Space Complexity
Measuring Space Complexity

We measure space complexity by finding the largest tape index reached during the computation.
Let M be a deterministic TM.

Definition: The space complexity of M is the function $S : \mathbb{N} \rightarrow \mathbb{N}$, where $S(n)$ is the largest tape index reached by M on any input of length $n$.

Definition: $\text{SPACE}(S(n)) =$

$$\{ L \mid L \text{ is decided by a Turing machine with } O(S(n)) \text{ space complexity} \}$$
Theorem: $3SAT \in SPACE(n)$

Proof Idea: Try all possible assignments to the (at most $n$) variables in a formula of length $n$. This can be done in $O(n)$ space.

Theorem: $NTIME(t(n))$ is in $SPACE(t(n))$

Proof Idea: Try all possible computation paths of $t(n)$ steps for an NTM on length-$n$ input. This can be done in $O(t(n))$ space.
Space Hierarchy Theorem

Intuition: If you have more space to work with, then you can solve strictly more problems!

Theorem: For functions $s, S : \mathbb{N} \rightarrow \mathbb{N}$ where $s(n)/S(n) \rightarrow 0$

\[ \text{SPACE}(s(n)) \subsetneq \text{SPACE}(S(n)) \]

Idea: Diagonalization

Make a machine $M$ that uses $S(n)$ space and “does the opposite” of all $O(s(n))$ space machines on at least one input

So $L(M)$ is in $\text{SPACE}(S(n))$ but not $\text{SPACE}(s(n))$
PSPACE = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)

Since for every $k$, NTIME($n^k$) is in SPACE($n^k$), we have:

\[ \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \]
The class PSPACE formalizes the set of problems solvable by computers with \textit{bounded memory}.

**Fundamental (Unanswered) Question:** How does time relate to space, in computing?

SPACE($n^2$) problems could potentially take much longer than $n^c$ time to solve, for \textit{any} $c$!

\textit{Intuition: You can always re-use space, but how can you re-use time?}

Is P = PSPACE?
Time Complexity of $\text{SPACE}[S(n)]$

Let $M$ be a halting TM that on input $x$, uses $S(|x|)$ space

How many time steps can $M(x)$ possibly take? Is there an upper bound?

The number of time steps is at most the total number of possible configurations of $M$.

(If a configuration repeats, the machine is in an infinite loop!)

A configuration of $M$ on $x$ specifies a head position, a state, and $S$ cells of tape content.
So the total number of configurations is at most:

$$S(|x|) \cdot |Q| \cdot |\Gamma|^S(|x|) \leq 2^{O(S(|x|))}$$
Theorem:
For every space-\(S(n)\) TM, there is a TM running in \(2^{O(S(n))}\) time that decides the same language.

\[
\text{SPACE}(s(n)) \subseteq \bigcup_{c \in \mathbb{N}} \text{TIME}(2^c \cdot s(n))
\]

Proof Idea: For each \(s(n)\)-space bounded TM \(M\) there is a \(c > 0\) so that on all inputs \(x\), if \(M\) runs for more than \(2^c s(|x|)\) time steps on \(x\), then \(M\) must have repeated a configuration, so \(M\) will never halt.
PSPACE = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)

\text{EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{TIME}(2^{n^k})

\text{PSPACE} \subseteq \text{EXPTIME}
\[ \text{P } \subseteq \text{NP } \subseteq \text{PSPACE} \]

Is \( \text{NP}^{\text{NP}} \) \( \subseteq \) PSPACE?  

YES

And \( \text{coNP}^{\text{NP}} \) \( \subseteq \) PSPACE
Example: MIN-FORMULA is in PSPACE

MIN-FORMULA = \{ \phi \mid \phi \text{ is minimal} \}

Recall the coNP^{NP} algorithm for MIN-FORMULA:

Given a formula \( \phi \),

Try all formulas \( \psi \) such that \( \psi \) is smaller than \( \phi \).

If \( ((\phi, \psi) \in \text{NEQUIV}) \) then accept else reject

Can store a formula \( \psi \) in space \( O(|\phi|) \)

Can check \( (\phi, \psi) \in \text{NEQUIV} \) by trying all assignments to the variables of \( \phi \) and \( \psi \)

Can store a variable assignment in space \( O(|\phi|) \)

Evaluating \( \psi \) or \( \phi \) on an assignment uses \( O(|\phi|) \) space
$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$

Theorem: $P \neq EXPTIME$

Why? The Time Hierarchy Theorem!

$TIME(2^n) \not\subseteq P$

Therefore $P \neq EXPTIME$

Corollary: At least one of the following is true: $P \neq NP, \ NP \neq PSPACE, \ or \ PSPACE \neq EXPTIME$

Proving any one of them would be major!
PSPACE
and Nondeterminism
Definition: SPACE(s(n)) =
{ L | L is decided by a Turing machine with O(s(n)) space complexity}

Definition: NSPACE(s(n)) =
{ L | L is decided by a non-deterministic Turing Machine with O(s(n)) space complexity}
Recall:
Space $S(n)$ computations can be simulated in at most $2^{O(S(n))}$ time steps

$$\text{SPACE}(s(n)) \subseteq \bigcup_{c \in \mathbb{N}} \text{TIME}(2^c \cdot s(n))$$

Idea: After $2^{O(s(n))}$ time steps, a $s(n)$-space bounded computation must have repeated a configuration, after which it will provably never halt.
Theorem:
NSPACE \( S(n) \) computations can also be simulated in at most \( 2^{O(S(n))} \) time steps

\[
\text{NSPACE}(s(n)) \subseteq \bigcup_{c \in \mathbb{N}} \text{TIME}(2^c \cdot s(n))
\]

Key Idea: Think of the problem of simulating NSPACE\((s(n))\) as a problem on graphs.
Def: The configuration graph of M on x has nodes $C$ for every configuration $C$ of M on x, and edges $(C, C')$ if and only if $C$ yields $C'$

$G_{M,x}$

M has space complexity $S(n)$
$\Rightarrow G_{M,x}$ has
$2^d\cdot S(|x|)$ nodes

M is deterministic
$\Rightarrow$ every node has outdegree $\leq 1$

M is nondeterministic
$\Rightarrow$ some nodes may have outdegree $> 1$

M accepts x $\iff$ there is a path in $G_{M,x}$ from the initial configuration node to a node in an accept state
Def: The configuration graph of $M$ on $x$ has nodes $C$ for every configuration $C$ of $M$ on $x$, and edges $(C, C')$ if and only if $C$ yields $C'$

$G_{M,x}$

M has space complexity $S(n)$
$\Rightarrow G_{M,x}$ has $2^{d \cdot S(|x|)}$ nodes

M is deterministic
$\Rightarrow$ every node has outdegree $\leq 1$

M is nondeterministic
$\Rightarrow$ some nodes may have outdegree $> 1$

To simulate a non-deterministic $M$ in $2^{O(S(|x|))}$ time: do BFS in $G_{M,x}$ from the initial configuration!
PSPACE = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)

NPSPACE = \bigcup_{k \in \mathbb{N}} \text{NSPACE}(n^k)
SPACE versus NSPACE

Is NTIME(n) ⊆ TIME(n^2)?

Is NTIME(n) ⊆ TIME(n^k) for some k > 1?

Nobody knows!

If the answer is yes, then P = NP...

What about the space-bounded setting?

Is NSPACE(s(n)) ⊆ SPACE(s(n)^k) for some k? Is PSPACE = NPSPACE?
Savitch’s Theorem

Theorem: For functions $s(n)$ where $s(n) \geq n$

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)$$

Proof Try:

Let $N$ be a non-deterministic TM with space complexity $s(n)$.

Construct a deterministic machine $M$ that tries every possible branch of $N$.

Since each branch of $N$ uses space at most $s(n)$, then $M$ uses space at most $s(n)^2$.

There are $2^{2^s(n)}$ branches to keep track of!
Given configurations $C_1$ and $C_2$ of a $s(n)$ space machine $N$, and a number $k$ (in binary), want to know if $N$ can get from $C_1$ to $C_2$ within $2^k$ steps.

Procedure SIM($C_1$, $C_2$, $k$):

- If $k = 0$ then accept iff $C_1 = C_2$ or $C_1$ yields $C_2$ within one step. [uses space $O(s(n))$]
- If $k > 0$, then for every config $C_m$ of $O(s(n))$ symbols,
  - if SIM($C_1$, $C_m$, $k-1$) and SIM($C_m$, $C_2$, $k-1$) accept then return accept
  - return reject if no such $C_m$ is found

SIM($C_1$, $C_2$, $k$) has $O(k)$ levels of recursion.
Each level of recursion uses $O(s(n))$ additional space.
Theorem: SIM($C_1$, $C_2$, $k$) uses only $O(k \cdot s(n))$ space.
Theorem: For functions $s(n)$ where $s(n) \geq n$

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)$$

Proof:
Let $N$ be a nondeterministic TM using $s(n)$ space
Let $d > 0$ be such that the number of configurations of $N(w)$ is at most $2^d s(|w|)$

Here’s a deterministic $O(s(n)^2)$ space algorithm for $N$:

$M(w)$: For all configurations $C_a$ of $N(w)$ in the accept state,
If $\text{SIM}(q_o, w, C_a, d s(|w|))$ accepts, then accept
else reject

Claim: $L(M) = L(N)$ and $M$ uses $O(s(n)^2)$ space
Theorem: For functions $s(n)$ where $s(n) \geq n$

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)$$

Proof:
Let $N$ be a nondeterministic TM using $s(n)$ space
Let $d > 0$ be such that the number of configurations of $N(w)$ is at most $2^d s(|w|)$

Here’s a deterministic $O(s(n)^2)$ space algorithm for $N$:

$M(w)$: For all configurations $C_a$ of $N(w)$ in the accept state,
If $\text{SIM}(q_0, w, C_a, d s(|w|))$ accepts, then accept
else reject

Why does it take only $s(n)^2$ space?
Theorem: For functions $s(n)$ where $s(n) \geq n$

$$NSPACE(s(n)) \subseteq SPACE(s(n)^2)$$

Proof:
Let $N$ be a nondeterministic TM using $s(n)$ space
Let $d > 0$ be such that the number of configurations of $N(w)$ is at most $2^d s(|w|)$

Here’s a deterministic $O(s(n)^2)$ space algorithm for $N$:

$M(w)$: For all configurations $C_a$ of $N(w)$ in the accept state,
If $SIM(q_0, w, C_a, d \cdot s(|w|))$ accepts, then accept
else reject

$SIM$ uses $O(k \cdot s(|w|))$ space to simulate $2^k$ steps of $N(w)$.
For $k = d \cdot s(|w|)$ we have $O(k \cdot s(|w|)) \leq O(s(|w|)^2)$ space
PSPACE = \bigcup_{k \in \mathbb{N}} SPACE(n^k)

NPSPACE = \bigcup_{k \in \mathbb{N}} NSPACE(n^k)

PSPACE = NPSPACE
PSPACE-complete problems
Definition: Language B is PSPACE-complete if:

1. \( \mathcal{B} \in \text{PSPACE} \)
2. Every \( \mathcal{A} \) in PSPACE is poly-time reducible to \( \mathcal{B} \) (i.e. \( \mathcal{B} \) is PSPACE-hard)

Why poly-time?

Theorem: If \( \mathcal{B} \) is PSPACE-complete and \( \mathcal{B} \) is in P then \( \text{P} = \text{PSPACE} \)

Theorem: If \( \mathcal{B} \) is PSPACE-complete and \( \mathcal{B} \) is in NP then \( \text{NP} = \text{PSPACE} \)
Definition:
A fully quantified Boolean formula is a Boolean formula where *every* variable in the formula is quantified (\( \exists \) or \( \forall \)) at the beginning the formula. These formulas are either true or false

\[
\exists x \exists y \left[ x \lor \neg y \right] \\
\forall x \left[ x \lor \neg x \right] \\
\forall x \left[ x \right] \\
\forall x \exists y \left[ (x \lor y) \land (\neg x \lor \neg y) \right]
\]
TQBF = \{ \phi \mid \phi \text{ is a true fully quantified Boolean formula} \}

- SAT is the special case where all quantifiers on all variables are $\exists$

- TAUTOLOGY is the special case where all quantifiers are $\forall$

So, SAT $\leq_P$ TQBF and TAUTOLOGY $\leq_P$ TQBF

Theorem (Meyer-Stockmeyer): TQBF is PSPACE-complete