Computational Complexity

Lecture 4
in which Diagonalization takes on itself,
and we enter Space Complexity
Meta-Questions
Meta-Questions
Meta-Questions

“Real” Questions
Meta-Questions

“Real” Questions  “Meta” Questions
Meta-Questions

“Real” Questions

SAT in DTIME(n^2)?

“Meta” Questions
Meta-Questions

“Real” Questions

SAT in DTIME(n^2)?

Is my problem NP-complete?

“Meta” Questions
Meta-Questions

“Real” Questions

SAT in DTIME(n^2)?

Is my problem
   NP-complete?

Results non-specialists
   would care about

“Meta” Questions
Meta-Questions

“Real” Questions

SAT in DTIME(n^2)?

Is my problem NP-complete?

Results non-specialists would care about

“Meta” Questions

What can we do with an oracle for SAT?
Meta-Questions

“Real” Questions

SAT in DTIME(n^2)?

Is my problem NP-complete?

Results non-specialists would care about

“Meta” Questions

What can we do with an oracle for SAT?

Will this proof technique work?
Meta-Questions

"Real" Questions

SAT in DTIME(n^2)?

Is my problem NP-complete?

Results non-specialists would care about

"Meta" Questions

What can we do with an oracle for SAT?

Will this proof technique work?

Tools & Techniques, intermediate results
Meta-Questions

“Real” Questions

SAT in \text{DTIME}(n^2)?

Is my problem \text{NP}-complete?

Results non-specialists would care about

“Meta” Questions

What can we do with an oracle for SAT?

Will this proof technique work?

Tools & Techniques, intermediate results

Under-the-hood stuff
Oracles
Oracles

What if we had an oracle for language A
Oracles

What if we had an oracle for language $A$

- **Class** $P_A$: $L \in P_A$ if
What if we had an oracle for language $A$

Class $P^A$: $L \in P^A$ if

$L$ decided by a TM $M^A$, in poly time
Oracles

What if we had an oracle for language $A$

- **Class $P^A$:** $L \in P^A$ if
  - $L$ decided by a TM $M^A$, in poly time

- Turing reduction: $L \leq_T A$
Oracles

What if we had an oracle for language $A$

- **Class $P^A$:** $L \in P^A$ if
  - $L$ decided by a TM $M^A$, in poly time
- Turing reduction: $L \leq_T A$

- **Class $NP^A$:** $L \in NP^A$ if
Oracles

What if we had an oracle for language $A$

**Class $P^A$:** $L \in P^A$ if
- $L$ decided by a TM $M^A$, in poly time

Turing reduction: $L \leq_T A$

**Class $NP^A$:** $L \in NP^A$ if
- $L$ decided by an NTM $M^A$, in poly time
Oracles

What if we had an oracle for language $A$

**Class $P^A$:** $L \in P^A$ if
- $L$ decided by a TM $M^A$, in poly time

Turing reduction: $L \leq_T A$

**Class $NP^A$:** $L \in NP^A$ if
- $L$ decided by an NTM $M^A$, in poly time

Equivalently, $L = \{x | \exists w, |w| < \text{poly}(|x|) \text{ s.t. } (x, w) \in L' \}$, where $L'$ is in $P^A$
Oracles

What if we had an oracle for language $A$

- **Class $P^A$:** $L \in P^A$ if
  - $L$ decided by a TM $M^A$, in poly time

- Turing reduction: $L \leq_T A$

- **Class $NP^A$:** $L \in NP^A$ if
  - $L$ decided by an NTM $M^A$, in poly time

Equivalently, $L = \{x| \exists w, |w| < \text{poly}(|x|) \text{ s.t. } (x,w) \in L'\}$, where $L'$ is in $P^A$

Equivalence carries over!
Proofs that Relativize
Proofs that Relativize

Often entire theorems/proofs carry over, with the oracle tagging along
Proofs that Relativize

Often entire theorems/proofs carry over, with the oracle tagging along

- e.g. Time hierarchy theorems (and proofs!) hold for machines with access to any given oracle A
Proofs that Relativize

Often entire theorems/proofs carry over, with the oracle tagging along.

- e.g. Time hierarchy theorems (and proofs!) hold for machines with access to any given oracle \( A \).

- Said to “relativize”
P vs. NP with oracles
P vs. NP with oracles

How does P vs. NP fare relative to different oracles?
P vs. NP with oracles

- How does P vs. NP fare relative to different oracles?
- Does their relation (equality or not) relativize?
P vs. NP with oracles

- How does P vs. NP fare relative to different oracles?
- Does their relation (equality or not) relativize?
- No! Different in different worlds!
P vs. NP with oracles

- How does P vs. NP fare relative to different oracles?
- Does their relation (equality or not) relativize?
- No! Different in different worlds!

- There exist languages A, B such that 
  \( P^A = NP^A \), but \( P^B \neq NP^B \)!
A s.t. $P^A = NP^A$
A s.t. $P^A = NP^A$

If $A$ is EXP-complete (w.r.t $\leq_{\text{Cook}}$ or $\leq_{P}$), $P^A = NP^A = EXP$
A s.t. $P^A = NP^A$

- If $A$ is EXP-complete (w.r.t $\leq_{\text{Cook}}$ or $\leq_P$), $P^A = NP^A = EXP$
- A EXP-hard $\Rightarrow EXP \subseteq P^A \subseteq NP^A$
A s.t. $P^A = NP^A$

- If $A$ is EXP-complete (w.r.t $\leq_{\text{Cook}}$ or $\leq_P$), $P^A = NP^A = EXP$
- $A$ EXP-hard $\Rightarrow$ $EXP \subseteq P^A \subseteq NP^A$
- $A$ in EXP $\Rightarrow$ $NP^A \subseteq EXP$ (note: to decide a language in $NP^A$ can try all possible witnesses, and carry out $P^A$ computation in exponential time)
A s.t. \( P^A = NP^A \)

- If A is EXP-complete (w.r.t \( \leq_{\text{Cook}} \) or \( \leq_{P} \)), \( P^A = NP^A = \text{EXP} \)

- A EXP-hard \( \Rightarrow \) \( \text{EXP} \subseteq P^A \subseteq NP^A \)

- A in EXP \( \Rightarrow \) \( NP^A \subseteq \text{EXP} \) (note: to decide a language in \( NP^A \) can try all possible witnesses, and carry out \( P^A \) computation in exponential time)

- A simple EXP-complete language:
A s.t. $P^A = NP^A$

- If $A$ is EXP-complete (w.r.t $\leq_{\text{Cook}}$ or $\leq_P$), $P^A = NP^A = EXP$

- $A$ EXP-hard $\Rightarrow$ EXP $\subseteq P^A \subseteq NP^A$

- $A$ in EXP $\Rightarrow NP^A \subseteq EXP$ (note: to decide a language in $NP^A$ can try all possible witnesses, and carry out $P^A$ computation in exponential time)

- A simple EXP-complete language:

  $EXPTM = \{ (M,x,1^n) \mid \text{TM represented by } M \text{ accepts } x \text{ within time } 2^n \}$
B s.t. $P^B \neq NP^B$
B s.t. $P^B \neq NP^B$

Building B and L, s.t. L in $NP^B \setminus P^B$
$B \text{ s.t. } P^B \neq NP^B$

Building $B$ and $L$, s.t. $L$ in $NP^B \setminus P^B$

$L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$
Building $B$ and $L$, s.t. $L$ in $\text{NP}^B \setminus \text{P}^B$.

$L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$
B s.t. \( P_B \neq NP_B \)

Building B and L, s.t. L in \( NP_B \setminus P_B \)

\( L = \{1^n| \exists w, |w|=n \text{ and } w \in B\} \)
B s.t. \( P^B \neq NP^B \)

Building B and L, s.t. L in \( NP^B \setminus P^B \)

\[ L = \{1^n | \exists w, |w| = n \text{ and } w \in B \} \]
B s.t. $P^B \neq NP^B$

Building B and L, s.t. L in $NP^B \setminus P^B$

\[ L = \{1^n | \exists w, |w| = n \text{ and } w \in B\} \]

L in $NP^B$. To do: L not in $P^B$
Building $B$ and $L$, s.t. $L$ in $\text{NP}^B \setminus \text{P}^B$

$L = \{1^n | \exists w, |w|=n \text{ and } w \in B\}$

$L$ in $\text{NP}^B$. To do: $L$ not in $\text{P}^B$

For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)
\[ B \text{ s.t. } P^B \neq NP^B \]

Building \( B \) and \( L \), s.t. \( L \) in \( NP^B \backslash P^B \)

- \( L = \{1^n| \exists w, |w|=n \text{ and } w \in B\} \)
- \( L \) in \( NP^B \). To do: \( L \) not in \( P^B \)
  - For each \( i \), ensure \( M_i^B \) in \( 2^{n-1} \) time gets \( L(1^n) \) wrong (for some new \( n \))
\( B \) s.t. \( P^B \neq NP^B \)

Building \( B \) and \( L \), s.t. \( L \) in \( NP^B \backslash P^B \)

- \( L = \{1^n| \exists w, |w|=n \text{ and } w \in B \} \)
- \( L \) in \( NP^B \). To do: \( L \) not in \( P^B \)
  - For each \( i \), ensure \( M_i^B \) in \( 2^{n-1} \) time gets \( L(1^n) \) wrong (for some new \( n \)
Building $B$ and $L$, s.t. $L$ in $\mathsf{NP}^B \setminus \mathsf{P}^B$

$L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$

$L$ is in $\mathsf{NP}^B$. To do: $L$ not in $\mathsf{P}^B$

For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)
B s.t. \( P^B \neq NP^B \)

Building \( B \) and \( L \), s.t. \( L \) in \( NP^B \setminus P^B \)

- \( L = \{ 1^n \mid \exists w, |w| = n \text{ and } w \in B \} \)
- \( L \) in \( NP^B \). To do: \( L \) not in \( P^B \)
  - For each \( i \), ensure \( M_i^B \) in \( 2^{n-1} \) time gets \( L(1^n) \) wrong (for some new \( n \))
  - Pick \( n \) s.t. \( B \) not yet set beyond \( 1^{n-1} \). Run \( M_i \) on \( 1^n \) for \( 2^{n-1} \) steps.
Building $B$ and $L$, s.t. $L$ in $\text{NP}^B \setminus \text{P}^B$

$L = \{1^n | \exists w, |w|=n \text{ and } w \in B\}$

$L$ in $\text{NP}^B$. To do: $L$ not in $\text{P}^B$

For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)

Pick $n$ s.t. $B$ not yet set beyond $1^{n-1}$. Run $M_i$ on $1^n$ for $2^{n-1}$ steps.
B s.t. $P^B \neq NP^B$

Building $B$ and $L$, s.t. $L$ in $NP^B \setminus P^B$

$L = \{1^n | \exists w, |w|=n \text{ and } w \in B\}$

$L$ in $NP^B$. To do: $L$ not in $P^B$

- For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)

- Pick $n$ s.t. $B$ not yet set beyond $1^{n-1}$. Run $M_i$ on $1^n$ for $2^{n-1}$ steps.
$B$ s.t. $P^B \neq NP^B$

Building $B$ and $L$, s.t. $L$ in $NP^B \setminus P^B$

- $L = \{1^n \mid \exists w, |w| = n \text{ and } w \in B\}$
- $L$ in $NP^B$. To do: $L$ not in $P^B$
  - For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)
  - Pick $n$ s.t. $B$ not yet set beyond $1^{n-1}$. Run $M_i$ on $1^n$ for $2^{n-1}$ steps.
Building $B$ and $L$, s.t. $L$ is in $\text{NP}^B \setminus \text{P}^B$

$L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$

$L$ is in $\text{NP}^B$. To do: $L$ is not in $\text{P}^B$.

For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new new $n$).

Pick $n$ s.t. $B$ is not yet set beyond $1^{n-1}$. Run $M_i$ on $1^n$ for $2^{n-1}$ steps.
B s.t. $P^B \neq NP^B$

Building $B$ and $L$, s.t. $L$ in $NP^B \setminus P^B$

- $L=\{1^n | \exists w, |w|=n \text{ and } w \in B\}$
- $L$ in $NP^B$. To do: $L$ not in $P^B$
  - For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)
  - Pick $n$ s.t. $B$ not yet set beyond $1^{n-1}$. Run $M_i$ on $1^n$ for $2^{n-1}$ steps.
B s.t. $P^B \neq NP^B$

Building $B$ and $L$, s.t. $L \in NP^B \backslash P^B$

$L = \{1^n | \exists w, |w|=n \text{ and } w \in B\}$

$L$ in $NP^B$. To do: $L$ not in $P^B$

- For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)

- Pick $n$ s.t. $B$ not yet set beyond $1^{n-1}$. Run $M_i$ on $1^n$ for $2^{n-1}$ steps.

- When $M_i$ queries $B$ on $x > 1^{n-1}$, set $B(X)=0$
B s.t. $P^B \neq NP^B$

Building $B$ and $L$, s.t. $L$ in $NP^B \setminus P^B$

$L = \{1^n | \exists w, |w|=n$ and $w \in B\}$
$L$ in $NP^B$. To do: $L$ not in $P^B$
- For each $i$, ensure $M_i^B$ in $2^{n-1}$ time gets $L(1^n)$ wrong (for some new $n$)
- Pick $n$ s.t. $B$ not yet set beyond $1^{n-1}$. Run $M_i$ on $1^n$ for $2^{n-1}$ steps.
- When $M_i$ queries $B$ on $x > 1^{n-1}$, set $B(X) = 0$
- After $M_i$ finished set $B$ up to $x=1^n$ s.t. $L(1^n) \neq M_i^B(1^n)$
Meta-Result of the Day
Meta-Result of the Day

- P vs. NP cannot be resolved using a relativizing proof
Meta-Result of the Day

- P vs. NP cannot be resolved using a relativizing proof
- “Diagonalization proofs” relativize
Meta-Result of the Day

- P vs. NP cannot be resolved using a relativizing proof
- “Diagonalization proofs” relativize
- Just need a way to enumerate/encode machines, and to simulate one without much overhead given its encoding
Meta-Result of the Day

- P vs. NP cannot be resolved using a relativizing proof
- “Diagonalization proofs” relativize
- Just need a way to enumerate/encode machines, and to simulate one without much overhead given its encoding
- Do not further depend on internals of computation
Meta-Result of the Day

- P vs. NP cannot be resolved using a relativizing proof
- “Diagonalization proofs” relativize
- Just need a way to enumerate/encode machines, and to simulate one without much overhead given its encoding
- Do not further depend on internals of computation
- e.g. of non-relativizing proof: that of Cook-Levin theorem
Space Complexity
Space Complexity
Space Complexity

- Natural complexity question
Space Complexity

- Natural complexity question
- How much memory is needed
Space Complexity

- Natural complexity question
  - How much memory is needed
  - More pressing than time:
Space Complexity

- Natural complexity question
  - How much memory is needed
  - More pressing than time:
    - Can’t generate memory on the fly
Space Complexity

- Natural complexity question
  - How much memory is needed
  - More pressing than time:
    - Can’t generate memory on the fly
  - Or maybe less pressing:
Space Complexity

- Natural complexity question
- How much memory is needed
- More pressing than time:
  - Can't generate memory on the fly
- Or maybe less pressing:
  - Turns out, often a little memory can go a long way (if we can spare the time)
DSPACE and NSPACE
DSPACE and NSPACE

Measure of working memory (work-tape) used by a TM/NTM: input kept in a read-only tape
DSPACE and NSPACE

- Measure of working memory (work-tape) used by a TM/NTM: input kept in a read-only tape
- Model allows o(n) memory usage
DSPACE and NSPACE

- Measure of working memory (work-tape) used by a TM/NTM: input kept in a read-only tape
- Model allows $o(n)$ memory usage
- DSPACE(n) may already be inefficient in terms of time
DSPACE and NSPACE

- Measure of working memory (work-tape) used by a TM/NTM: input kept in a read-only tape
- Model allows $o(n)$ memory usage
- DSPACE(n) may already be inefficient in terms of time
- We shall stick to $\Omega(\log n)$
DSPACE and NSPACE

- Measure of \textit{working} memory (work-tape) used by a TM/NTM: input kept in a read-only tape

- Model allows $o(n)$ memory usage

- $\text{DSPACE}(n)$ may already be inefficient in terms of time

- We shall stick to $\Omega(\log n)$

- Less than log is too little space to remember locations in the input
DSPACE and NSPACE

- Measure of working memory (work-tape) used by a TM/NTM: input kept in a read-only tape

- Model allows $o(n)$ memory usage

  - DSPACE(n) may already be inefficient in terms of time

  - We shall stick to $\Omega(\log n)$

    - Less than log is too little space to remember locations in the input

- DSPACE/NSPACE more robust across models
DSPACE and NSPACE

- Measure of working memory (work-tape) used by a TM/NTM: input kept in a read-only tape

- Model allows $o(n)$ memory usage

- $\text{DSPACE}(n)$ may already be inefficient in terms of time

- We shall stick to $\Omega(\log n)$

- Less than $\log$ is too little space to remember locations in the input

- $\text{DSPACE}/\text{NSPACE}$ more robust across models

- Constant factor ($+O(\log n)$) simulation overhead
$L \in \text{NSPACE}(S)$: Two Equivalent views
$L \in \text{NSPACE}(S)$: Two Equivalent views

Non-deterministic $M$
$L \in \text{NSPACE}(S)$: Two Equivalent views

- Non-deterministic $M$
- input: $x$
\( L \in \text{NSPACE}(S) \): Two Equivalent views

- Non-deterministic \( M \)
- \text{input: } x
- makes non-det choices
$L \in \text{NSPACE}(S)$: Two Equivalent views

- Non-deterministic $M$
- input: $x$
- makes non-det choices
- $x \in L$ iff some thread of $M$ accepts
$L \in \text{NSPACE}(S)$: Two Equivalent views

- Non-deterministic $M$
- input: $x$
- makes non-det choices
- $x \in L$ iff some thread of $M$ accepts
- in at most $S(|x|)$ space
\( L \in \text{NSPACE}(S) \):
Two Equivalent views

- **Non-deterministic M**
  - input: \( x \)
  - makes non-det choices
  - \( x \in L \) iff some thread of \( M \) accepts
  - in at most \( S(|x|) \) space

- **Deterministic \( M' \)**
$L \in \text{NSPACE}(S)$: Two Equivalent views

- Non-deterministic $M$
  - input: $x$
  - makes non-det choices
  - $x \in L$ iff some thread of $M$ accepts
  - in at most $S(|x|)$ space

- Deterministic $M'$
  - input: $x$ and read-once $w$
L ∈ NSPACE(S):
Two Equivalent views

- Non-deterministic M
  - input: x
  - makes non-det choices
  - $x ∈ L$ iff some thread of $M$ accepts
  - in at most $S(|x|)$ space

- Deterministic $M'$
  - input: $x$ and read-once $w$
  - reads bits from $w$ (certificate)
\( L \in \text{NSPACE}(S): \) Two Equivalent views

- Non-deterministic \( M \)
  - input: \( x \)
  - makes non-det choices
  - \( x \in L \) iff some thread of \( M \) accepts
  - in at most \( S(|x|) \) space

- Deterministic \( M' \)
  - input: \( x \) and read-once \( w \)
  - reads bits from \( w \) (certificate)
  - \( x \in L \) iff for some cert. \( w \), \( M' \) accepts
\( L \in \text{NSPACE}(S): \) Two Equivalent views

- **Non-deterministic M**
  - input: \( x \)
  - makes non-det choices
  - \( x \in L \) iff some thread of \( M \) accepts
  - in at most \( S(|x|) \) space

- **Deterministic \( M' \)**
  - input: \( x \) and read-once \( w \)
  - reads bits from \( w \) (certificate)
  - \( x \in L \) iff for some cert. \( w \), \( M' \) accepts
  - in at most \( S(|x|) \) space
$L \in \text{NSPACE}(S)$: Two Equivalent views

- Non-deterministic $M$
  - input: $x$
  - makes non-det choices
  - $x \in L$ iff some thread of $M$ accepts
  - in at most $S(|x|)$ space

- Deterministic $M'$
  - input: $x$ and read-once $w$
  - reads bits from $w$ (certificate)
  - $x \in L$ iff for some cert. $w$, $M'$ accepts
  - in at most $S(|x|)$ space
L and NL
$L$ and $\text{NL}$

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

\[ L = \text{DSPACE}(O(\log n)) \]

\[ L = \bigcup_{a,b > 0} \text{DSPACE}(a \log n + b) \]
L and NL

\[ L = \bigcup_{a,b > 0} \text{DSPACE}(a \log n + b) \]

\[ NL = \text{NSPACE}(O(\log n)) \]
L and NL

\[ L = \text{DSPACE}(O(\log n)) \]
\[ L = \bigcup_{a, b > 0} \text{DSPACE}(a \log n + b) \]
\[ NL = \text{NSPACE}(O(\log n)) \]
\[ NL = \bigcup_{a, b > 0} \text{NSPACE}(a \log n + b) \]
L and NL

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

L = \bigcup_{a, b > 0} \text{DSPACE}(a \cdot \log n + b)

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

NL = \bigcup_{a, b > 0} \text{NSPACE}(a \cdot \log n + b)

"L and NL are to space what P and NP are to time"
Space Hierarchy
Space Hierarchy

- UTM space-overhead is only a constant factor
**Space Hierarchy**

- UTM space-overhead is only a constant factor

- **Tight hierarchy**: if $T(n) = o(T'(n))$ (no log slack) then $\text{DSPACE}(T(n)) \subsetneq \text{DSPACE}(T'(n))$
Space Hierarchy

- UTM space-overhead is only a constant factor

  - Tight hierarchy: if $T(n) = o(T'(n))$ (no log slack) then $DSPACE(T(n)) \subsetneq DSPACE(T'(n))$

- Same for NSPACE
Space Hierarchy

- UTM space-overhead is only a constant factor

- **Tight hierarchy**: if $T(n) = o(T'(n))$ (no log slack) then $\text{DSPACE}(T(n)) \subseteq \text{DSPACE}(T'(n))$

- Same for NSPACE

- Again, tighter than for NTIME (where in fact, we needed $T(n+1) = o(T'(n))$)
Space Hierarchy

UTM space-overhead is only a constant factor

Tight hierarchy: if \( T(n) = o(T'(n)) \) (no log slack) then
\( \text{DSPACE}(T(n)) \subseteq \text{DSPACE}(T'(n)) \)

Same for \( \text{NSPACE} \)

Again, tighter than for \( \text{NTIME} \) (where in fact, we needed \( T(n+1) = o(T'(n)) \))

No “delayed flip,” because, as we will see later, \( \text{NSPACE}(O(S)) = \text{co-NSPACE}(O(S)) \)!
Space, Today
Space, Today

DSpace, NSpace
Space, Today

- DSPACE, NSPACE
- Tight hierarchy.
Space, Today

- DSPACE, NSPACE
- Tight hierarchy.
- Coming up:
Space, Today

- DSPACE, NSPACE
- Tight hierarchy.
- Coming up:
  - Connections with DTIME/NTIME
Space, Today

- DSPACE, NSPACE
- Tight hierarchy.
- Coming up:
  - Connections with DTIME/NTIME
  - Savitch’s theorem: $\text{NSPACE}(S) \subseteq \text{DSPACE}(S^2)$
Space, Today

- DSPACE, NSPACE
- Tight hierarchy.

Coming up:

- Connections with DTIME/NTIME
- Savitch’s theorem: NSPACE(S) ⊆ DSPACE(S²)
  - Hence PSPACE = NPSPACE
Space, Today

- **DSpace, NSpace**
- **Tight hierarchy.**

**Coming up:**

- **Connections with DTIME/NTIME**
- **Savitch’s theorem:** $\text{NSPACE}(S) \subseteq \text{DSpace}(S^2)$
  - Hence $\text{PSPACE} = \text{NPSPACE}$
- **PSPACE-completeness and NL-completeness**
Space, Today

DSPACES, NSPACE

Tight hierarchy.

Coming up:

Connections with DTIME/NTIME

Savitch’s theorem: NSPACE(S) ⊆ DSPACE(S²)

Hence PSPACE = NPSPACE

PSPACE-completeness and NL-completeness

NSPACE = co-NSPACE
DSPACE, NSPACE

Tight hierarchy.

Coming up:

Connections with DTIME/NTIME
Savitch's theorem: NSPACE(S) ⊆ DSPACE(S²)
Hence PSPACE = NPSPACE
PSPACE-completeness and NL-completeness
NSPACE = co-NSPACE