Computational Complexity

Lecture 4
in which Diagonalization takes on itself,
and we enter Space Complexity
(But first Ladner's Theorem)
Ladner’s Theorem
Ladner’s Theorem
Ladner’s Theorem

If $P \neq NP$, then are all non-$P$ NP languages equally hard? (Are all NP-complete?)
If $P \neq NP$, then are all non-$P$ NP languages equally hard? (Are all NP-complete?)

No!
Ladner’s Theorem

If P≠NP, then are all non-P NP languages equally hard? (Are all NP-complete?)

No!

Can show an NP language which is neither in P, nor NP complete (unless P = NP)
Ladner’s Theorem: Proof
Ladner's Theorem: Proof

\[ \text{SAT}_H = \{ (x, \text{pad}) \mid x \in \text{SAT} \text{ and } |\text{pad}| = |x|^{H(|x|)} \} \]
Ladner’s Theorem: Proof

- $\text{SAT}_H = \{ (x, \text{pad}) | x \in \text{SAT} \text{ and } |\text{pad}| = |x|^{H(|x|)} \}$

- $H(|x|)$ will be computable in poly($|x|$) time. $\text{SAT}_H$ in $\text{NP}$. 
Ladner’s Theorem: Proof

\[ \text{SAT}_H = \{ (x, \text{pad}) \mid x \in \text{SAT} \text{ and } |	ext{pad}| = |x|^{H(|x|)} \} \]

- \( H(|x|) \) will be computable in poly(|x|) time. \( \text{SAT}_H \) in NP.
- If \( \text{SAT}_H \) in P and \( H(|x|) \) bounded by const. then SAT in P!
Ladner’s Theorem: Proof

- \( \text{SAT}_H = \{ (x, \text{pad}) \mid x \in \text{SAT} \text{ and } |\text{pad}| = |x|^{H(|x|)} \} \)

- \( H(|x|) \) will be computable in poly(|x|) time. \( \text{SAT}_H \) in NP.

- If \( \text{SAT}_H \) in \( P \) and \( H(|x|) \) bounded by const. then \( \text{SAT} \) in \( P \)!

- \(|\text{pad}| < |x|^{i^*}\) implies \( \text{SAT} \leq_p \text{SAT}_H \)
Ladner's Theorem: Proof

SAT_H = \{ (x,\text{pad}) \mid x \in \text{SAT} \text{ and } |\text{pad}| = |x|^{H(|x|)} \}

H(|x|) will be computable in poly(|x|) time. SAT_H in NP.

If SAT_H in P and H(|x|) bounded by const. then SAT in P!

|\text{pad}| < |x|^{i*} implies SAT \leq_p SAT_H

If SAT_H is NPC (⇒ SAT_H not in P) and H(|x|) goes to infinity, then SAT in P!
Ladner’s Theorem: Proof

- $\text{SAT}_H = \{ (x, \text{pad}) \mid x \in \text{SAT} \text{ and } |\text{pad}| = |x|^{H(|x|)} \}$

- $H(|x|)$ will be computable in $\text{poly}(|x|)$ time. $\text{SAT}_H$ is in NP.

- If $\text{SAT}_H$ is in $P$ and $H(|x|)$ bounded by const. then SAT is in P!

- $|\text{pad}| < |x|^{i^*}$ implies SAT $\leq_P \text{SAT}_H$

- If $\text{SAT}_H$ is NPC ($\Rightarrow \text{SAT}_H$ not in P) and $H(|x|)$ goes to infinity, then SAT is in P!

- Suppose $f(x) = (x', \text{pad})$, $|(x', \text{pad})| \leq c|x|^c$. If $|x'| > |x|/2$, then $|\text{pad}| = |x'|^{H(|x'|)} > c|x|^c$ (for long enough x). So $|x'|$ is at most $|x|/2$. Repeat to solve SAT...
Ladner’s Theorem: Proof

- SAT_H = \{ (x,\text{pad}) \mid x \in \text{SAT} \text{ and } |\text{pad}| = |x|^{H(|x|)} \}

- H(|x|) will be computable in poly(|x|) time. SAT_H in NP.

- If SAT_H in P and H(|x|) bounded by const. then SAT in P!

- |\text{pad}| < |x|^{i^*} implies SAT \leq_p SAT_H

- If SAT_H is NPC (⇒ SAT_H not in P) and H(|x|) goes to infinity, then SAT in P!

- Suppose f(x) = (x',\text{pad}), |(x',\text{pad})| \leq c|x|^c. If |x'| > |x|/2, then |\text{pad}| = |x'|^{H(|x'|)} > c|x|^c (for long enough x). So |x'| is at most |x|/2. Repeat to solve SAT

- To define H s.t. H(n) bounded by const. iff SAT_H in P
Proof (ctd.)
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i \cdot t^i$)
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i\cdot t^i$)
- $M_i|T_i$ be $M_i$ restricted to $T_i$
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i.t^i$)
- $M_i|T_i$ be $M_i$ restricted to $T_i$
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i.t^i$)
- $M_i|T_i$ be $M_i$ restricted to $T_i$
- Put $\square$ at $(i,t)$ if $M_i|T_i$ agrees with SAT$_H$ on all $z$, $|z|=t$; else put $\times$
**Proof (ctd.)**

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i\cdot t^i$)
- $M_i|T_i$ be $M_i$ restricted to $T_i$
- Put ☑ at $(i,t)$ if $M_i|T_i$ agrees with $\text{SAT}_H$ on all $z$, $|z|=t$; else put ☒
- $H(n)$ be least $i < \log \log n$ s.t. $M_i|T_i$ correct for all $|z|<\log n$

| $|z|$ |   |   |   |   |   |   |   |   |
|-----|---|---|---|---|---|---|---|---|
| $M_i|T_i$ |   |   |   |   |   |   |   |   |
| ☑   | ☒ | ☒ | ☐ | ☒ | ☒ | ☒ | ☒ | ☑ |
| ☒   | ☑ | ☑ | ☑ | ☑ | ☑ | ☐ | ☑ | ☑ |
| ☑   | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☒ | ☒ |
| ☑   | ☑ | ☒ | ☑ | ☐ | ☑ | ☒ | ☑ | ☑ |
| ☑   | ☑ | ☒ | ☒ | ☐ | ☐ | ☐ | ☑ | ☒ |
| ☒   | ☒ | ☒ | ☒ | ☒ | ☐ | ☑ | ☐ | ☐ |
| ☑   | ☑ | ☒ | ☑ | ☐ | ☐ | ☑ | ☑ | ☑ |
| ☑   | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |


Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i.t^i$)

- $M_i|T_i$ be $M_i$ restricted to $T_i$

- Put $\checkmark$ at $(i,t)$ if $M_i|T_i$ agrees with $\text{SAT}_H$ on all $z$, $|z|=t$; else put $\times$

- $H(n)$ be least $i < \log \log n$ s.t. $M_i|T_i$ correct for all $|z|<\log n$

| $|z|$ | $M_i|T_i$ | $\log n$ |
|------|--------|--------|
|      | $\checkmark$ | $\times$ |
|      | $\times$ | $\times$ |
|      | $\checkmark$ | $\checkmark$ |
|      | $\checkmark$ | $\times$ |
|      | $\checkmark$ | $\times$ |
|      | $\times$ | $\times$ |
|      | $\checkmark$ | $\checkmark$ |
|      | $\times$ | $\times$ |
|      | $\times$ | $\times$ |
|      | $\times$ | $\times$ |

Note: The table is a visual representation of the proof steps, with $\checkmark$ indicating agreement and $\times$ indicating disagreement with $\text{SAT}_H$. The values in the table correspond to the conditions and agreements across different values of $|z|$ and $n$. The $H(n)$ function is defined to find the least $i$ that satisfies the conditions.
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i\cdot t^i$)
- $M_i|T_i$ be $M_i$ restricted to $T_i$
- Put ☑ at $(i,t)$ if $M_i|T_i$ agrees with SAT$_H$ on all $z$, $|z|=t$; else put ☒
- $H(n)$ be least $i < \log \log n$ s.t. $M_i|T_i$ correct for all $|z|<\log n$
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t) = i \cdot t^i$)

- $M_i|T_i$ be $M_i$ restricted to $T_i$

- Put $\checkmark$ at $(i, t)$ if $M_i|T_i$ agrees with $\text{SAT}_H$ on all $z$, $|z| = t$; else put $\times$

- $H(n)$ be least $i < \log \log n$ s.t. $M_i|T_i$ correct for all $|z| < \log n$

- $H$ is poly-time computable
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i.t^i$)
- $M_i|T_i$ be $M_i$ restricted to $T_i$
- Put $\checkmark$ at $(i,t)$ if $M_i|T_i$ agrees with $\text{SAT}_H$ on all $z$, $|z|=t$; else put $\times$
- $H(n)$ be least $i < \log \log n$ s.t. $M_i|T_i$ correct for all $|z|<\log n$
- $H$ is poly-time computable
- $\text{SAT}_H$ in P iff $H(n) < i^*$
Proof (ctd.)

- $M_i$ be $i^{th}$ TM. $T_i$ be $i^{th}$ polynomial (i.e., $T_i(t)=i.t^i$)
- $M_i|T_i$ be $M_i$ restricted to $T_i$
- Put $\checkmark$ at $(i,t)$ if $M_i|T_i$ agrees with $\text{SAT}_H$ on all $z$, $|z|=t$; else put $\times$
- $H(n)$ be least $i < \log \log n$ s.t. $M_i|T_i$ correct for all $|z|<\log n$
- $H$ is poly-time computable
- $\text{SAT}_H$ in P iff $H(n) < i^*$
- Both equivalent to having a row of all $\checkmark$
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 $\text{DTIME}(n^2)$?

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 DTIME(n²)?

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

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
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
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: $L \leq_T A$
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 $P^A$
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 $\Rightarrow$ $NP^A \subseteq EXP^A = EXP$ (note: $NP \subseteq EXP$, by trying all possible witnesses)
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 $\Rightarrow$ $NP^A \subseteq EXP^A = EXP$ (note: $NP \subseteq EXP$, by trying all possible witnesses)
- 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 \( \Rightarrow \) \( NP^A \subseteq EXP^A = EXP \) (note: \( NP \subseteq EXP \), by trying all possible witnesses)
- 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. \( P^B \neq N^P^B \)
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. $L \in NP^B \setminus P^B$

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

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

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

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

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

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

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

$L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$
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 $P^B$
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 $n$)
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 $n$)
**Building B and L, s.t. L in NP^B \not\in P^B**

- L={1^n| \exists w, |w|=n and w \in \mathbb{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 $NP^B \setminus P^B$

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

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

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

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

$L$ 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 $NP^B \setminus P^B$

- $L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$
- $L$ 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 $\mathbf{NP}^B \setminus \mathbf{P}^B$

- $L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$
- $L$ in $\mathbf{NP}^B$. To do: $L$ not in $\mathbf{P}^B$
  - 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 $B(X) = 0$
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.

- 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
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
 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
- $\text{DSPACE}(n)$ may already be inefficient in terms of time
- We shall stick to $\Omega(\log n)$
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 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
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 ∈ NSPACE(S):
Two Equivalent views

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

<table>
<thead>
<tr>
<th>Non-deterministic $M$</th>
<th>Deterministic $M'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>input: $x$</td>
<td>input: $x$ and read-once $w$</td>
</tr>
<tr>
<td>makes non-det choices</td>
<td>reads bits from $w$ (certificate)</td>
</tr>
<tr>
<td>$x \in L$ iff some thread of $M$ accepts</td>
<td></td>
</tr>
<tr>
<td>in at most $S(</td>
<td>x</td>
</tr>
</tbody>
</table>
\( L \in \text{NSPACE}(S): \) 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

<table>
<thead>
<tr>
<th>Non-deterministic ( M )</th>
<th>Deterministic ( M' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>input: ( x )</td>
<td>input: ( x ) and read-once ( w )</td>
</tr>
<tr>
<td>makes non-det choices</td>
<td>reads bits from ( w ) (certificate)</td>
</tr>
<tr>
<td>( x \in L ) iff some thread of ( M ) accepts</td>
<td>( x \in L ) iff for some cert. ( w ), ( M' ) accepts</td>
</tr>
<tr>
<td>in at most ( S(</td>
<td>x</td>
</tr>
</tbody>
</table>
$L \in \text{NSPACE}(S)$: 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 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

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

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

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

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

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

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

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

"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 $\text{DSPACE}(T(n)) \subset \text{DSPACE}(T'(n))$

- 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 $\text{DSPACE}(T(n)) \subsetneq \text{DSPACE}(T'(n))$

- 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 \( \mathsf{DSPACE}(T(n)) \subsetneq \mathsf{DSPACE}(T'(n)) \)

- Same for \( \mathsf{NSPACE} \)
  - Again, tighter than for \( \mathsf{NTIME} \) (where in fact, we needed \( T(n+1) = o(T'(n)) \) )
  - No “delayed flip,” because, as we will see later, \( \mathsf{NSPACE}(O(S)) = \mathsf{co-NSPACE}(O(S)) \)!
SPACE and TIME
SPACE and TIME

In time $T(n)$, can use at most $T(n)$ space
SPACE and TIME

- In time $T(n)$, can use at most $T(n)$ space
- $\text{DTIME}(T) \subseteq \text{DSpace}(T)$
SPACE and TIME

- In time $T(n)$, can use at most $T(n)$ space
- $\text{DTIME}(T) \subseteq \text{DSpace}(T)$
- In fact, $\text{NTIME}(T) \subseteq \text{DSpace}(O(T))$ (simulate with all $T$-long certificates)
SPACE and TIME

- In time $T(n)$, can use at most $T(n)$ space

- $\text{DTIME}(T) \subseteq \text{DSPACE}(T)$

- In fact, $\text{NTIME}(T) \subseteq \text{DSPACE}(O(T))$ (simulate with all $T$-long certificates)

- With space $S(n)$, only $2^{O(S(n))}$ configurations (for $S(n) = \Omega(\log n)$). So can take at most $2^{O(S(n))}$ time (else gets into an infinite loop)
SPACE and TIME

In time $T(n)$, can use at most $T(n)$ space

$\text{DTIME}(T) \subseteq \text{DSPACE}(T)$

In fact, $\text{NTIME}(T) \subseteq \text{DSPACE}(O(T))$ (simulate with all $T$-long certificates)

With space $S(n)$, only $2^{O(S(n))}$ configurations (for $S(n) = \Omega(\log n)$). So can take at most $2^{O(S(n))}$ time (else gets into an infinite loop)

$\text{DSPACE}(S) \subseteq \text{DTIME}(2^{O(S)})$
SPACE and TIME

In time $T(n)$, can use at most $T(n)$ space

$\text{DTIME}(T) \subseteq \text{DSPACE}(T)$

In fact, $\text{NTIME}(T) \subseteq \text{DSPACE}(O(T))$ (simulate with all $T$-long certificates)

With space $S(n)$, only $2^{O(S(n))}$ configurations (for $S(n) = \Omega(\log n)$). So can take at most $2^{O(S(n))}$ time (else gets into an infinite loop)

$\text{DSPACE}(S) \subseteq \text{DTIME}(2^{O(S)})$

In fact, $\text{NSPACE}(S) \subseteq \text{DTIME}(2^{O(S)})$
NSPACE(S) ⊆ DTIME(2^{O(S)})
$\text{NSPACE}(s) \subseteq \text{DTIME}(2^{O(s)})$
$\text{NSPACE}(S) \subseteq \text{DTIME}(2^{O(S)})$

Configuration graph as a DAG is of size $2^{O(S)}$. 

$h = 2^{O(S)}$
\text{NSPACE}(S) \subseteq \text{DTIME}(2^{O(S)})

- Configuration graph as a DAG is of size $2^{O(S)}$
- Write down all configurations and edges
NSPACE(S) ⊆ DTIME(2^{O(S)})

- Configuration graph as a DAG is of size $2^{O(S)}$
  - Write down all configurations and edges
  - Can do it less explicitly if space were a concern (but it’s not, here)
**NSPACE(S) ⊆ DTIME(2^{O(S)})**

Configuration graph as a DAG is of size $2^{O(S)}$

- Write down all configurations and edges
- Can do it less explicitly if space were a concern (but it’s not, here)
- Run (in poly time) any reachability algorithm (say, breadth-first search) to see if there is a (directed) path from start config. to an accept config.
\textbf{NSPACE}(S) \subseteq \textbf{DTIME}(2^{O(S)})

- Configuration graph as a DAG is of size $2^{O(S)}$.

- Write down all configurations and edges.

- Can do it less explicitly if space were a concern (but it’s not, here).

- Run (in poly time) any reachability algorithm (say, breadth-first search) to see if there is a (directed) path from start config. to an accept config.

- $\text{poly}(2^{O(S)}) = 2^{O(S)}$.
SPACE and TIME
SPACE and TIME

NTIME(F)

DTIME(F)
SPACE and TIME

\[ DTIME(F) \]

\[ NTIME(F) \]
SPACE and TIME

\[ \text{NTIME}(2^{O(F)}) \] \quad \text{DTIME}(2^{O(F)})

\[ \text{NTIME}(F) \] \quad \text{DTIME}(F) \]
SPACE and TIME

\[
\text{NTIME}(2^{O(F)}) \quad \text{DTIME}(2^{O(F)}) \quad \text{NTIME}(F) \quad \text{DTIME}(F)
\]
SPACE and TIME

NTIME(2^{O(F)})

DTIME(2^{O(F)})

NTIME(F)

DTIME(F)
SPACE and TIME

NTIME($2^{O(F)}$) → DTIME($2^{O(F)}$)

NTIME($F$) → DTIME($F$)

NTIME($2^{O(F)}$) → NTIME($F$)

NSPACE($F$)

DSPACE($F$)
SPACE and TIME

NTIME($2^O(F)$)

DTIME($2^O(F)$)

NTIME($F$)

DTIME($F$)

NSPACE($F$)

DSPACE($F$)

NTIME($2^O(F)$)
SPACE and TIME

NTIME\(2^{O(F)}\) → DTIME\(2^{O(F)}\) → NSPACE(F)

NTIME(F) → DTIME(F) → DSPACE(F)

NTIME\(2^{O(F)}\) → DTIME\(2^{O(F)}\) → NSPACE(F)

DSPACE(F) → NSPACE(F)
SPACE and TIME

NTIME\left(2^{O(F)}\right) \rightarrow \text{DTIME}\left(2^{O(F)}\right)

NTIME(F) \rightarrow \text{DTIME}(F)

NTIME(2^{O(F)}) \rightarrow \text{DSPACE}(F)

NSPACE(F)

DSPACE(F)
SPACE and TIME

$\text{NTIME}(2^{O(F)}) \rightarrow \text{DTIME}(2^{O(F)})$

$\text{NTIME}(F) \rightarrow \text{DTIME}(F)$

$\text{NTIME}(F) \rightarrow \text{DSPACE}(F)$

$\text{DSPACE}(F) \rightarrow \text{NSPACE}(F)$

$F = \Omega(n)$

$F = \Omega(\log n)$
Space, Today
Space, Today

DSpace, NSpace
Space, Today

- DSPACE, NSPACE
- Tight hierarchy.
Space, Today

- DS\textsc{pace}, NS\textsc{pace}
- Tight hierarchy.
- Connections with D\textsc{time}/N\textsc{time}
Space, Today

- DSPACE, NSPACE
- Tight hierarchy.
- Connections with DTIME/NTIME
- Next class
Space, Today

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

- DSPACE, NSPACE
- Tight hierarchy.
- Connections with DTIME/NTIME
- Next class

- Savitch’s theorem: NSPACE(S) ⊆ DSPACE(S^2)
- Hence PSPACE = NPSPACE
Space, Today

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

- DSPACE, NSPACE

- Tight hierarchy.

- Connections with DTIME/NTIME

- Next class

- Savitch’s theorem: NSPACE(S) ⊆ DSPACE(S^2)
  - Hence PSPACE = NPSPACE

- PSPACE-completeness and NL-completeness

- NSPACE = co-NSPACE
Space, Today

- **DSPACE, NSPACE**
- Tight hierarchy.
- Connections with DTIME/NTIME
- Next class
  - Savitch's theorem: NSPACE(S) \( \subseteq \) DSPACE\((S^2)\)
  - Hence PSPACE = NPSPACE
  - PSPACE-completeness and NL-completeness
  - NSPACE = co-NSPACE