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

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

No!
Ladner’s Theorem

If $P \neq NP$, 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

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

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.

Padding maps problem to a lower complexity class.
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 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 \( \text{NP} \).
- If \( \text{SAT}_H \) in \( \text{P} \) and \( H(|x|) \) bounded by const. then \( \text{SAT} \) in \( \text{P} \)!
- \( |\text{pad}| < |x|^{i^*} \) implies \( \text{SAT} \preceq_\text{P} \text{SAT}_H \)
Ladner’s Theorem: Proof

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

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

|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',pad), |(x',pad)| \leq c|x|^c. If |x'| > |x|/2, then |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

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

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

| $|z|$ | $M_i|T_i$ |
|------|----------|
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
| ✓    | ✓        | ✓        | ✓        | ✓        |
| ✗    | ✗        | ✗        | ✗        | ✗        |
Proof (ctd.)

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

- \( M_i | T_i \) 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 \(|z| < \log n\)
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 $|z|<\log n$

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

<table>
<thead>
<tr>
<th></th>
<th>$z$</th>
<th>$\log n$</th>
<th>$\log \log n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_i</td>
<td>T_i$</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
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., \( T_i(t) = i \cdot t \))
- \( M_i|T_i \) 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(n²)?

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(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

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: $L \leq_T A$
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 $P^A$
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 $\leq_P$), $P^A = NP^A = \text{EXP}$
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 \( \leq_P \)), \( P^A = NP^A = EXP \)

- 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:
A s.t. $P^A = NP^A$

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

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. $P^B \neq N\!P^B$
B s.t. $P^B \neq NP^B$

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

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

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

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

1. $L=\{1^n| \exists w, |w|=n \text{ and } w \in B\}$
2. $L$ in $NP^B$. To do: $L$ not in $P^B$
3. 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 $NP^B \backslash P^B$

- $L = \{1^n | \exists w, |w| = n \text{ and } w \in B\}$
- $L$ 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 $n$)
$B$ 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.
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.
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 \( NP^B \backslash P^B \)

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

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

$L$ 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.

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

- $\text{DSPACE}(n)$ 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
- $\text{DSPACE}(n)$ 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

  - 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

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

- Non-deterministic M
- Input: x
- Makes non-det choices
- x ∈ 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

<table>
<thead>
<tr>
<th>Non-deterministic $M$</th>
<th>Deterministic $M'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>input: $x$</td>
<td></td>
</tr>
<tr>
<td>makes non-det choices</td>
<td></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 $w$
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

<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
$L \in \text{NSPACE}(S)$: 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

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

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

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

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

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

\[ \text{NL} = \bigcup_{a, b > 0} \text{NSPACE}(a \cdot \log n + b) \]
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 $\text{DSPACE}(T(n)) \subset \text{DSPACE}(T'(n))$

- 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 \( \text{DSPACE}(T(n)) \subset \text{DSPACE}(T'(n)) \)

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

- Same for NSPACE

  - Again, tighter than for 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) \subseteq DSPACE(S^2)
  Hence PSPACE = NPSPACE
Space, Today

DSPACE, NSPACE

Tight hierarchy.

Coming up:

Connections with DTIME/NTIME

Savitch's theorem: NSPACE(S) ⊆ DSPACE(S^2)

Hence PSPACE = NPSPACE

PSPACE-completeness and NL-completeness
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
  - $\text{NSPACE} = \text{co-NSPACE}$
Space, Today

DSPACE, NSPACE

Tight hierarchy.

Coming up:

Connections with DTIME/NTIME

Savitch’s theorem: NSPACE(S) \subseteq DSPACE(S^2)

Hence PSPACE = NPSPACE

PSPACE-completeness and NL-completeness

NSPACE = co-NSPACE