Lecture 3
in which we come across
Diagonalization and Time-hierarchies
(But first some more of NP-completeness)
NP-Complete Languages
NP-Complete Languages

A language \( L_1 \) is NP-complete if \( L_1 \) is in NP and any NP language \( L \) can be reduced to \( L_1 \) (Karp reduction: polynomial time many-one reduction)
NP-Complete Languages

A language $L_1$ is NP-complete if $L_1$ is in NP and any NP language $L$ can be reduced to $L_1$ (Karp reduction: polynomial time many-one reduction)

Trivial example: $L_1 = \text{TMSAT}$
NP-Complete Languages

A language $L_1$ is NP-complete if $L_1$ is in NP and any NP language $L$ can be reduced to $L_1$ (Karp reduction: polynomial time many-one reduction)

- Trivial example: $L_1 = \text{TMSAT}$
- First interesting example: $L_1 = \text{CKT-SAT}$
NP-Complete Languages

A language $L_1$ is NP-complete if $L_1$ is in NP and any NP language $L$ can be reduced to $L_1$ (Karp reduction: polynomial time many-one reduction)

- Trivial example: $L_1 = \text{TMSAT}$
- First interesting example: $L_1 = \text{CKT-SAT}$
  - Convert $x$ into a circuit $C$ s.t. $C$ is satisfiable iff $x$ in $L$
NP-Complete Languages

A language $L_1$ is NP-complete if $L_1$ is in NP and any NP language $L$ can be reduced to $L_1$ (Karp reduction: polynomial time many-one reduction)

- Trivial example: $L_1 = \text{TMSAT}$
- First interesting example: $L_1 = \text{CKT-SAT}$
  - Convert $x$ into a circuit $C$ s.t. $C$ is satisfiable iff $x$ in $L$
- More examples, bootstrapping from CKT-SAT
NP-Complete Languages

A language $L_1$ is NP-complete if $L_1$ is in NP and any NP language $L$ can be reduced to $L_1$ (Karp reduction: polynomial time many-one reduction)

- Trivial example: $L_1 = \text{TMSAT}$
- First interesting example: $L_1 = \text{CKT-SAT}$
  - Convert $x$ into a circuit $C$ s.t. $C$ is satisfiable iff $x$ in $L$
- More examples, bootstrapping from CKT-SAT

If $L \leq_P L_1$ and $L_1 \leq_P L_2$, then $L \leq_P L_2$
CKT-SAT $\leq_p$ SAT
CKT-SAT $\leq_p$ SAT

SAT: Are all given “clauses” simultaneously satisfiable? (Conjunctive Normal Form)
CKT-SAT $\leq_p$ SAT

SAT: Are all given “clauses” simultaneously satisfiable? (Conjunctive Normal Form)

Converting a circuit to a collection of clauses:
CKT-SAT $\leq_p$ SAT

SAT: Are all given “clauses” simultaneously satisfiable? (Conjunctive Normal Form)

Converting a circuit to a collection of clauses:

- For each wire (connected component), add a variable
CKT-SAT $\leq_p$ SAT

SAT: Are all given “clauses” simultaneously satisfiable? (Conjunctive Normal Form)

Converting a circuit to a collection of clauses:

- For each wire (connected component), add a variable
- Add output variable as a clause. And for each gate, add a clause involving variables for wires connected to the gate:
SAT: Are all given “clauses” simultaneously satisfiable? (Conjunctive Normal Form)

Converting a circuit to a collection of clauses:

For each wire (connected component), add a variable

Add output variable as a clause. And for each gate, add a clause involving variables for wires connected to the gate:

\[ e.g. \quad \begin{array}{c}
\text{AND} \\
\xrightarrow{\text{AND}} \quad & \text{z: (z} \Rightarrow x), (z} \Rightarrow y), (\neg z \Rightarrow \neg x \lor \neg y).
\end{array} \]

i.e., \((\neg z \lor x), (\neg z \lor y), (z \lor \neg x \lor \neg y)).\]
SAT: Are all given “clauses” simultaneously satisfiable? ( Conjunctive Normal Form)

Converting a circuit to a collection of clauses:

For each wire (connected component), add a variable

Add output variable as a clause. And for each gate, add a clause involving variables for wires connected to the gate:

e.g. \[ \text{AND} \quad x \quad y \quad \text{and} \quad z: (z \Rightarrow x), (z \Rightarrow y), (\neg z \Rightarrow \neg x \lor \neg y). \]

i.e., \[(\neg z \lor x), (\neg z \lor y), (z \lor \neg x \lor \neg y). \]

\[ \text{OR} \quad x \quad y \quad z: (z \Rightarrow x \lor y), (\neg z \Rightarrow \neg x), (\neg z \Rightarrow \neg y). \]
SAT $\leq_p$ 3SAT
SAT $\leq_p$ 3SAT

Previous reduction was to 3SAT, so 3SAT is NP-complete. And SAT is in NP. So SAT $\leq_p$ 3SAT.
SAT $\leq_p$ 3SAT

Previous reduction was to 3SAT, so 3SAT is NP-complete. And SAT is in NP. So SAT $\leq_p$ 3SAT.

More directly:
SAT $\leq_p$ 3SAT

Previous reduction was to 3SAT, so 3SAT is NP-complete. And SAT is in NP. So SAT $\leq_p$ 3SAT.

More directly:

$(a \lor b \lor c \lor d \lor e) \rightarrow (a \lor b \lor x), (\neg x \lor c \lor d \lor e)$

$\rightarrow (a \lor b \lor x), (\neg x \lor c \lor y), (\neg y \lor d \lor e)$
SAT \leq_p 3SAT

Previous reduction was to 3SAT, so 3SAT is NP-complete.
And SAT is in NP. So SAT \leq_p 3SAT.

More directly:

\((a \lor b \lor c \lor d \lor e) \rightarrow (a \lor b \lor x), (\neg x \lor c \lor d \lor e)\)
\rightarrow (a \lor b \lor x), (\neg x \lor c \lor y), (\neg y \lor d \lor e)\)

Reduction needs 3SAT
**SAT \leq_p 3SAT**

- Previous reduction was to 3SAT, so 3SAT is NP-complete. And SAT is in NP. So SAT \leq_p 3SAT.

- More directly:

  \[(a \lor b \lor c \lor d \lor e) \rightarrow (a \lor b \lor x), (\neg x \lor c \lor d \lor e)\]

  \[
  \rightarrow (a \lor b \lor x), (\neg x \lor c \lor y), (\neg y \lor d \lor e)
  \]

- Reduction needs 3SAT

- 2SAT is in fact in P! [Exercise]
SAT \leq_p 3SAT

Previous reduction was to 3SAT, so 3SAT is NP-complete. And SAT is in NP. So SAT \leq_p 3SAT.

More directly:

\[(a \lor b \lor c \lor d \lor e) \rightarrow (a \lor b \lor x), (\neg x \lor c \lor d \lor e)\]
\[(a \lor b \lor x), (\neg x \lor c \lor y), (\neg y \lor d \lor e)\]

Reduction needs 3SAT

2SAT is in fact in P! [Exercise]

Reduction not parsimonious (can you make it? [Exercise])
$3\text{SAT} \leq_p \text{CLIQUE}$
3SAT $\leq_p$ CLIQUE

CLIQUE: Does graph $G$ have a clique of size $m$?
3SAT \leq_p \text{CLIQUE}

- CLIQUE: Does graph G have a clique of size m?

\[(x \lor \neg y \lor \neg z)\]

\[(w \lor y)\]

\[(w \lor x \lor \neg z)\]
$3\text{SAT} \leq_p \text{CLIQUE}$

- **CLIQUE:** Does graph $G$ have a clique of size $m$?

- **Clauses $\rightarrow$ Graph**
  
  $$\begin{align*}
  & (x \lor \neg y \lor \neg z) \\
  & (w \lor y) \\
  & (w \lor x \lor \neg z)
  \end{align*}$$
3SAT \leq_p CLIQUE

- CLIQUE: Does graph G have a clique of size m?
- Clauses $\rightarrow$ Graph

$\begin{align*}
& (x \lor \neg y \lor \neg z) \\
& (w \lor y) \\
& (w \lor x \lor \neg z)
\end{align*}$
3SAT $\leq_p$ CLIQUE

- CLIQUE: Does graph $G$ have a clique of size $m$?
- Clauses $\rightarrow$ Graph

$(x \lor \neg y \lor \neg z)$

$(w \lor y)$

$(w \lor x \lor \neg z)$
3SAT \leq_p CLIQUE

- CLIQUE: Does graph G have a clique of size m?
- Clauses $\rightarrow$ Graph

$$(x \lor \neg y \lor \neg z)$$

$$(w \lor y)$$

$$(w \lor x \lor \neg z)$$
3SAT \leq_p CLIQUE

- CLIQUE: Does graph G have a clique of size m?
- Clauses $\rightarrow$ Graph

$(x \lor \neg y \lor \neg z)$

$(w \lor y)$

$(w \lor x \lor \neg z)$
3SAT $\leq_p$ CLIQUE

CLAQUE: Does graph G have a clique of size m?

Clauses $\rightarrow$ Graph

$(x \lor \neg y \lor \neg z)$

$(w \lor y)$

$(w \lor x \lor \neg z)$
3SAT $\leq_p$ CLIQUE

- **CLIQUE**: Does graph $G$ have a clique of size $m$?
- **Clauses $\rightarrow$ Graph**
  - Vertices: each clause's satisfying assignments (for its variables)

- (x $\lor$ $\neg$y $\lor$ $\neg$z)
- (w $\lor$ y)
- (w $\lor$ x $\lor$ $\neg$z)
3SAT $\leq_p$ CLIQUE

- **CLIQUE**: Does graph $G$ have a clique of size $m$?

- **Clauses $\rightarrow$ Graph**
  - Vertices: each clause's satisfying assignments (for its variables)

For example:
- $(x \lor \neg y \lor \neg z)$
- $(w \lor y)$
- $(w \lor x \lor \neg z)$

The diagram illustrates a graph with vertices labeled with binary strings representing possible assignments for clauses.
3SAT $\leq_P$ CLIQUE

- **CLIQUE**: Does graph G have a clique of size m?

- **Clauses $\rightarrow$ Graph**

- **Vertices**: each clause’s satisfying assignments (for its variables)

- $(x \lor \neg y \lor \neg z)$
- $(w \lor y)$
- $(w \lor x \lor \neg z)$
3SAT $\leq_p$ CLIQUE

CLIQUE: Does graph G have a clique of size m?

Clauses $\rightarrow$ Graph

vertices: each clause's satisfying assignments (for its variables)

$(x \lor \neg y \lor \neg z)$

$(w \lor y)$

$(w \lor x \lor \neg z)$
3SAT $\leq_p$ CLIQUE

- **CLIQUE**: Does graph $G$ have a clique of size $m$?

- **Clauses $\rightarrow$ Graph**

  - vertices: each clause's satisfying assignments (for its variables)
3SAT \leq_p CLIQUE

- CLIQUE: Does graph G have a clique of size m?
- Clauses $\rightarrow$ Graph
  - vertices: each clause's satisfying assignments (for its variables)
$3\text{SAT} \leq_p \text{CLIQUE}$

- **CLIQUE**: Does graph $G$ have a clique of size $m$?

- **Clauses $\rightarrow$ Graph**
  - Vertices: each clause's satisfying assignments (for its variables)
  - Edges between consistent assignments

Clauses:
- $\{x \lor \neg y \lor \neg z\}$
- $\{w \lor y\}$
- $\{w \lor x \lor \neg z\}$
**3SAT \( \leq_p \) CLIQUE**

- **CLIQUE:** Does graph \( G \) have a clique of size \( m \)?

- **Clauses \( \rightarrow \) Graph**

  - vertices: each clause's satisfying assignments (for its variables)
  - edges between consistent assignments
  - \( m \)-clique iff all \( m \) clauses satisfiable

\[
(x \lor \neg y \lor \neg z)
\]

\[
(w \lor y)
\]

\[
(w \lor x \lor \neg z)
\]
3SAT \leq_p CLIQUE

CLIQUE: Does graph G have a clique of size m?

Clauses \rightarrow Graph

vertices: each clause's satisfying assignments (for its variables)

edges between consistent assignments

m-clique iff all m clauses satisfiable

\[(x \lor \lnot y \lor \lnot z)\]

\[(w \lor y)\]

\[(w \lor x \lor \lnot z)\]
3SAT $\leq_p$ CLIQUE

- CLIQUE: Does graph $G$ have a clique of size $m$?

- Clauses $\rightarrow$ Graph
  - vertices: each clause's satisfying assignments (for its variables)
  - edges between consistent assignments
  - $m$-clique iff all $m$ clauses satisfiable

$$(x \lor \neg y \lor \neg z)$$

$$(w \lor y)$$

$$(w \lor x \lor \neg z)$$

3-Clique

sat assignment: 1110
INDEP-SET and VERTEX-COVER
INDEP-SET and VERTEX-COVER

\( \text{CLIQUE} \leq_p \text{INDEP-SET} \)
INDEP-SET and VERTEX-COVER

CLIQUE \leq_p INDEP-SET

G has an m-clique iff \( G^c \) has an m-independent-set
INDEP-SET and VERTEX-COVER

- CLIQUE \( \leq_p \) INDEP-SET
- G has an m-clique iff \( G^c \) has an m-independent-set
- INDEP-SET \( \leq_p \) VERTEX-COVER
INDEP-SET and VERTEX-COVER

- CLIQUE \(\leq_p\) INDEP-SET
  - \(G\) has an \(m\)-clique iff \(G^c\) has an \(m\)-independent-set

- INDEP-SET \(\leq_p\) VERTEX-COVER
  - \(G\) has an \(m\)-indep-set iff \(G\) has an \((n-m)\)-vertex-cover
NP, P, co-NP and NPC
NP, P, co-NP and NPC

We say class $X$ is “closed under polynomial reductions” if $(L_1 \leq_p L_2$ and $L_2$ in class $X$) implies $L_1$ in $X$
NP, P, co-NP and NPC

We say class X is “closed under polynomial reductions” if (\( L_1 \leq_p L_2 \) and \( L_2 \) in class X) implies \( L_1 \) in X

e.g. P, NP are closed under polynomial reductions
We say class X is “closed under polynomial reductions” if \((L_1 \leq_P L_2 \text{ and } L_2 \text{ in class } X)\) implies \(L_1 \text{ in } X\).

e.g. P, NP are closed under polynomial reductions

So is co-NP (If X is closed, so is co-X. Why?)
NP, P, co-NP and NPC

We say class $X$ is “closed under polynomial reductions” if $(L_1 \leq_p L_2$ and $L_2$ in class $X)$ implies $L_1$ in $X$

e.g. P, NP are closed under polynomial reductions

So is co-NP (If $X$ is closed, so is co-$X$. Why?)

If any NPC language is in P, then NP = P


**NP, P, co-NP and NPC**

We say class X is “closed under polynomial reductions” if $(L_1 \leq_p L_2$ and $L_2$ in class X) implies $L_1$ in X.

- e.g. P, NP are closed under polynomial reductions.
- So is co-NP (If X is closed, so is co-X. Why?)

- If any NPC language is in P, then NP = P.
- If any NPC language is in co-NP, then NP = co-NP.
NP, P, co-NP and NPC

We say class X is “closed under polynomial reductions” if \( L_1 \leq_p L_2 \) and \( L_2 \) in class X implies \( L_1 \) in X

- e.g. P, NP are closed under polynomial reductions
  
- So is co-NP (If X is closed, so is co-X. Why?)

If any NPC language is in P, then NP = P

If any NPC language is in co-NP, then NP = co-NP

Note: \( X \subseteq \text{co-X} \Rightarrow X = \text{co-X} \) (Why?)
NP, P, co-NP and NPC

- We say class X is “closed under polynomial reductions” if \((L_1 \leq_p L_2 \text{ and } L_2 \text{ in class } X) \implies L_1 \text{ in } X\)

  - e.g. P, NP are closed under polynomial reductions

  - So is co-NP (If X is closed, so is co-X. Why?)

- If any NPC language is in P, then NP = P

- If any NPC language is in co-NP, then NP = co-NP

  - Note: \(X \subseteq \text{co-}X \implies X = \text{co-}X\) (Why?)

- \(L\) is NP-complete iff \(L^c\) is co-NP-complete (Why?)
NP, P, co-NP and NPC

- We say class X is “closed under polynomial reductions” if $(L_1 \leq_p L_2$ and $L_2$ in class X) implies $L_1$ in X

  - e.g. P, NP are closed under polynomial reductions

  - So is co-NP (If X is closed, so is co-X. Why?)

- If any NPC language is in P, then NP = P

- If any NPC language is in co-NP, then NP = co-NP

  - Note: $X \subseteq \text{co-}X \Rightarrow X = \text{co-}X$ (Why?)

- L is NP-complete iff $L^c$ is co-NP-complete (Why?)

  - co-NP complete = co-(NP-complete)
Separating Classes
Separating Classes

How to prove a set $X$ strictly bigger than $Y$
Separating Classes

- How to prove a set $X$ strictly bigger than $Y$?
- Show an element not in $Y$, but in $X$? For us, not in $Y$ may often be difficult to prove for (familiar) elements.
Separating Classes

How to prove a set $X$ strictly bigger than $Y$

- Show an element not in $Y$, but in $X$? For us, not in $Y$ may often be difficult to prove for (familiar) elements

- Count? What if both infinite?!
Separating Classes

- How to prove a set $X$ strictly bigger than $Y$
  - Show an element not in $Y$, but in $X$? For us, not in $Y$ may often be difficult to prove for (familiar) elements
  - Count? What if both infinite?!
  - Comparing infinite sets: diagonalization!
Cantor's Diagonal Slash
Cantor’s Diagonal Slash

Are real numbers (say in the range \([0,1)\)) countable?
Cantor’s Diagonal Slash

Are real numbers (say in the range [0,1)) countable?

Suppose they were:
consider enumerating them along with their binary representations in a table
Are real numbers (say in the range [0,1)) countable?

Suppose they were: consider enumerating them along with their binary representations in a table:

<table>
<thead>
<tr>
<th>$R_i$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$R_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$R_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$R_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$R_5$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$R_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$R_7$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Cantor’s Diagonal Slash

Are real numbers (say in the range \([0,1)\)) countable?

Suppose they were: consider enumerating them along with their binary representations in a table.

Consider the real number corresponding to the “flipped diagonal.”

<table>
<thead>
<tr>
<th>(R_i)</th>
<th>(1)</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_1) =</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(R_2) =</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(R_3) =</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(R_4) =</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(R_5) =</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(R_6) =</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(R_7) =</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Cantor’s Diagonal Slash

Are real numbers (say in the range [0,1) ) countable?

Suppose they were:
consider enumerating them along with their binary representations in a table

Consider the real number corresponding to the “flipped diagonal”

<table>
<thead>
<tr>
<th>( R_i )</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( R_6 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( R_7 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Cantor’s Diagonal Slash

Are real numbers (say in the range \([0,1)\) ) countable?

Suppose they were: consider enumerating them along with their binary representations in a table.

Consider the real number corresponding to the “flipped diagonal”

Doesn’t appear in this table!

<table>
<thead>
<tr>
<th>(R_i)</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_1)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(R_2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(R_3)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(R_4)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(R_5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(R_6)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Are real numbers (say in the range \([0,1)\) ) countable?

Suppose they were: consider enumerating them along with their binary representations in a table.

Consider the real number corresponding to the “flipped diagonal”.

Doesn’t appear in this table!

<table>
<thead>
<tr>
<th>(R_i)</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_1)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(R_2)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(R_3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(R_4)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(R_5)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(R_6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(R_7)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Undecidable Languages

<table>
<thead>
<tr>
<th>$L_{M1}$</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{M2}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$L_{M3}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$L_{M4}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$L_{M5}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$L_{M6}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$L_{M7}$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Undecidable Languages

Languages, like real numbers, can be represented as infinite bit-vectors.

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{M1}$ =</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$L_{M2}$ =</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$L_{M3}$ =</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$L_{M4}$ =</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$L_{M5}$ =</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$L_{M6}$ =</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$L_{M7}$ =</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Undecidable Languages

- Languages, like real numbers, can be represented as infinite bit-vectors
- TMs can be enumerated!

|   | L_{M1} = 10010000011 | L_{M2} = 00101001111 | L_{M3} = 11111100000 | L_{M4} = 11010101111 | L_{M5} = 11000010000 | L_{M6} = 00000011000 | L_{M7} = 01010101011 |
Undecidable Languages

Languages, like real numbers, can be represented as infinite bit-vectors.

TMs can be enumerated!

Table of languages recognized by the TMs

<table>
<thead>
<tr>
<th></th>
<th>L_{M1}</th>
<th>L_{M2}</th>
<th>L_{M3}</th>
<th>L_{M4}</th>
<th>L_{M5}</th>
<th>L_{M6}</th>
<th>L_{M7}</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Undecidable Languages

 Languages, like real numbers, can be represented as infinite bit-vectors

 TMs can be enumerated!

 Table of languages recognized by the TMs

 $L = \text{“diagonal language”}$
Undecidable Languages

Languages, like real numbers, can be represented as infinite bit-vectors.

TMs can be enumerated!

Table of languages recognized by the TMs

$L = \text{“diagonal language”}$

$L^c$ does not appear as a row in this table. Hence not recognizable!
Undecidable Languages

Languages, like real numbers, can be represented as infinite bit-vectors.

TMs can be enumerated!

Table of languages recognized by the TMs

L = "diagonal language"

L^c does not appear as a row in this table. Hence not recognizable!
Diagonalization to Separate Classes

- Diagonalization can separate the class of decidable languages (from the class of all languages)

- Plan: Use similar techniques to separate complexity classes
DTIME Hierarchy
DTIME Hierarchy

- Fix a TM model (one-tape, binary alphabet)
DTIME Hierarchy

- Fix a TM model (one-tape, binary alphabet)

  \[ \text{DTIME}(T) = \text{class of languages that can be decided in } O(T(n)) \text{ time, by such a TM} \]
DTIME Hierarchy

- Fix a TM model (one-tape, binary alphabet)
  - \( \text{DTIME}(T) = \text{class of languages that can be decided in } O(T(n)) \text{ time, by such a TM} \)

- Theorem: \( \text{DTIME}(n^c) \subseteq \text{DTIME}(n^{c+1}) \) for all \( c \geq 1 \)
DTIME Hierarchy

- Fix a TM model (one-tape, binary alphabet)

  \[ \text{DTIME}(T) = \text{class of languages that can be decided in } O(T(n)) \text{ time, by such a TM} \]

- Theorem: \( \text{DTIME}(n^c) \subsetneq \text{DTIME}(n^{c+1}) \) for all \( c \geq 1 \)

- More generally \( \text{DTIME}(T) \subsetneq \text{DTIME}(T') \) if \( T, T' \) “nice” (and \( \geq n \)) and \( T(n) \log(T(n)) = o(T'(n)) \)
DTIME Hierarchy

- Fix a TM model (one-tape, binary alphabet)
  - $\text{DTIME}(T) =$ class of languages that can be decided in $O(T(n))$ time, by such a TM
- Theorem: $\text{DTIME}(n^c) \subseteq \text{DTIME}(n^{c+1})$ for all $c \geq 1$
- More generally $\text{DTIME}(T) \subseteq \text{DTIME}(T')$ if $T$, $T'$ “nice” (and $\geq n$) and $T(n)\log(T(n)) = o(T'(n))$
- Consequences, for e.g., $P \nsubseteq \text{EXP}$
DTIME Hierarchy

- Fix a TM model (one-tape, binary alphabet)
  - $\text{DTIME}(T) = \text{class of languages that can be decided in } O(T(n)) \text{ time, by such a TM}$

- **Theorem**: $\text{DTIME}(n^c) \subsetneq \text{DTIME}(n^{c+1})$ for all $c \geq 1$

- More generally $\text{DTIME}(T) \subsetneq \text{DTIME}(T')$ if $T$, $T'$ “nice” (and $\geq n$) and $T(n)\log(T(n)) = o(T'(n))$

- Consequences, for e.g., $P \subsetneq \text{EXP}$
  - $P \subseteq \text{DTIME}(2^n) \subsetneq \text{DTIME}(2^{2n}) \subseteq \text{EXP}$
DTIME Hierarchy: Proof
DTIME Hierarchy: Proof

Mi be an enumeration of TMs, each TM appearing infinitely often
DTIME Hierarchy: Proof

- $M_i$ be an enumeration of TMs, each TM appearing infinitely often
- Consider table$(i,j) = \text{UTM}_{T'}(M_i,j)$, where $T \log T = o(T')$
DTIME Hierarchy: Proof

Mi be an enumeration of TMs, each TM appearing infinitely often

Consider table(i,j) = UTM|T' (Mi,j), where T log T = o(T')

T' large and nice enough to allow simulation
**DTIME Hierarchy: Proof**

- $M_i$ be an enumeration of TMs, each TM appearing infinitely often.
- Consider $\text{table}(i,j) = UTM|_{T'}(M_i, j)$, where $T \log T = o(T')$.

$T'$ large and nice enough to allow simulation.

<table>
<thead>
<tr>
<th>$M_i$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 1 0 0 0 0 0</td>
<td>1 0 0 1 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0 1 0 0 1 1 1</td>
<td>0 0 1 0 1 0 0</td>
</tr>
<tr>
<td>1 1 1 1 1 1 1 1 0 0</td>
<td>0 0 1 0 1 0 0</td>
</tr>
<tr>
<td>1 1 0 1 0 1 0 1 1 1</td>
<td>1 1 0 1 0 1 0 1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0 0 0</td>
<td>1 1 0 1 0 1 0</td>
</tr>
<tr>
<td>Think DTIME(T) ⊆ rows</td>
<td>1 1 0 1 0 1 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>1 1 0 1 0 1 0</td>
</tr>
</tbody>
</table>
DTIME Hierarchy: Proof

- $M_i$ be an enumeration of TMs, each TM appearing infinitely often
- Consider table$(i,j) = UTM_{T'}(M_i,j)$, where $T \log T = o(T')$
**DTIME Hierarchy: Proof**

- $M_i$ be an enumeration of TMs, each TM appearing infinitely often
- Consider table\((i,j) = UTM|_{T'} (M_i,j)\), where $T \log T = o(T')$
- Let $L' = \text{inverted diagonal}$.  

\[
\begin{array}{cccccccccc}
M_i \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
\end{array}
\]

Think $\text{DTIME}(T) \subseteq \text{rows}$
DTIME Hierarchy: Proof

1. Let \( M_i \) be an enumeration of TMs, each TM appearing infinitely often.
2. Consider table \((i,j) = UTM_{T'}(M_i,j)\), where \( T \log T = o(T') \).
3. Let \( L' \) = inverted diagonal.
4. \( L' \) in DTIME\((T')\).

Think DTIME\((T)\) \( \subseteq \) rows.
DTIME Hierarchy: Proof

- $M_i$ be an enumeration of TMs, each TM appearing infinitely often
- Consider table$(i,j) = \text{UTM}_{T'}(M_i,j)$, where $T \log T = o(T')$
- Let $L'$ = inverted diagonal.
- $L'$ in DTIME($T'$)
- On input $i$, run $\text{UTM}_{T'}(M_i,i)$, modified to invert output

Think DTIME($T$) $\subseteq$ rows
DTIME Hierarchy: Proof

- Let $M_i$ be an enumeration of TMs, each TM appearing infinitely often.
- Consider table(i,j) = $\text{UTM}_{T'} (M_i,j)$, where $T \log T = o(T')$.
- Let $L'$ = inverted diagonal.
- $L'$ in DTIME($T'$).
- On input $i$, run $\text{UTM}_{T'} (M_i,i)$, modified to invert output.
- $L'$ not in DTIME($T$).

Think DTIME($T$) $\subseteq$ rows.
DTIME Hierarchy: Proof

- $M_i$ be an enumeration of TMs, each TM appearing infinitely often

- Consider $\text{table}(i,j) = UTM_{|T'}(M_i,j)$, where $T \log T = o(T')$

- Let $L'$ = inverted diagonal.

- $L'$ in $\text{DTIME}(T')$

- On input $i$, run $UTM_{|T'}(M_i,i)$, modified to invert output

- $L'$ not in $\text{DTIME}(T)$

- If $M$ accepts $L'$ in time $T$, then for sufficiently large $i$ s.t. $M_i=M$, UTM can finish simulating $M_i(i)$. Then $\text{table}(i,i)=L'(i)!$
NTIME Hierarchy
NTIME Hierarchy

- Finer hierarchy
NTIME Hierarchy

- Finer hierarchy

- $\text{NTIME}(T) \subsetneq \text{NTIME}(T')$ if $T(n) = o(T'(n))$, and $T$, $T'$ nice
NTIME Hierarchy

- Finer hierarchy

- \( \text{NTIME}(T) \subsetneq \text{NTIME}(T') \) if \( T(n) = o(T'(n)) \), and \( T, T' \) nice

- Because a more sophisticated Universal NTM has less overhead
NTIME Hierarchy

- Finer hierarchy
  
  \( \text{NTIME}(T) \subsetneq \text{NTIME}(T') \) if \( T(n) = o(T'(n)) \), and \( T, T' \) nice

- Because a more sophisticated Universal NTM has less overhead

- Diagonalization is more complicated
NTIME Hierarchy

- Finer hierarchy
  - $\text{NTIME}(T) \subsetneq \text{NTIME}(T')$ if $T(n) = o(T'(n))$, and $T, T'$ nice
  
  Because a more sophisticated Universal NTM has less overhead

- Diagonalization is more complicated
  
  Issue: $\text{NTIME}(T')$ enough to simulate $\text{NTIME}(T)$, but not to simulate $\text{co-NTIME}(T)$!
NTIME Hierarchy

- Finer hierarchy
  - \( \text{NTIME}(T) \nsubseteq \text{NTIME}(T') \) if \( T(n) = o(T'(n)) \), and \( T, T' \) nice

- Because a more sophisticated Universal NTM has less overhead

- Diagonalization is more complicated

- Issue: NTIME(T’) enough to simulate NTIME(T), but not to simulate co-NTIME(T)!

In fact, \( T(n+1) = o(T'(n)) \)
NTIME Hierarchy
NTIME Hierarchy

“Delayed flip” on a “rapidly thickening diagonal”
NTIME Hierarchy

“Delayed flip” on a “rapidly thickening diagonal”
**NTIME Hierarchy**

- “Delayed flip” on a “rapidly thickening diagonal”

Think $\text{NTIME}(T) \subseteq \text{rows}$
NTIME Hierarchy

"Delayed flip" on a "rapidly thickening diagonal"

\[ f(i+1) = \exp(f(i)) \]
NTIME Hierarchy

“Delayed flip” on a “rapidly thickening diagonal”

\[ f(i+1) = \exp(f(i)) \]
NTIME Hierarchy

- “Delayed flip” on a “rapidly thickening diagonal”
- $f(i+1) = \exp(f(i))$
- Let $L$ be the “diagonal” language

Think $\text{NTIME}(T) \subseteq \text{rows}$
NTIME Hierarchy

- “Delayed flip” on a “rapidly thickening diagonal”
- \( f(i+1) = \exp(f(i)) \)
- Let \( L \) be the “diagonal” language

Think \( \text{NTIME}(T) \subseteq \text{rows} \)
NTIME Hierarchy

“Delayed flip” on a “rapidly thickening diagonal”

\[ f(i+1) = \exp(f(i)) \]

Let \( L \) be the “diagonal” language

Think \( \text{NTIME}(T) \subseteq \text{rows} \)
"Delayed flip" on a "rapidly thickening diagonal"

\[ f(i+1) = \exp(f(i)) \]

Let \( L \) be the "diagonal" language

\[ L'(j) = L(j+1) \]
NTIME Hierarchy

- “Delayed flip” on a “rapidly thickening diagonal”
- $f(i+1) = \exp(f(i))$
- Let $L$ be the “diagonal” language
- $L'(j) = L(j+1)$

Think $\text{NTIME}(T) \subseteq \text{rows}$
“Delayed flip” on a “rapidly thickening diagonal”

- \( f(i+1) = \exp(f(i)) \)
- Let \( L \) be the “diagonal” language
- \( L'(j) = L(j+1) \)
- except if \( j = f(i) \), then \( L'(j) = 1 - L(f(i-1)+1) \)

Think \( \text{NTIME}(T) \subseteq \text{rows} \)
“Delayed flip” on a “rapidly thickening diagonal”

\[ f(i+1) = \exp(f(i)) \]

Let \( L \) be the “diagonal” language

\[ L'(j) = L(j+1) \]

except if \( j = f(i) \), then

\[ L'(j) = 1 - L(f(i-1)+1) \]
"Delayed flip" on a "rapidly thickening diagonal"

\[ f(i+1) = \exp(f(i)) \]

Let \( L \) be the "diagonal" language

\[ L'(j) = L(j+1) \]

except if \( j = f(i) \), then

\[ L'(j) = 1 - L(f(i-1)+1) \]

\( L' \) not in \( \text{NTIME}(T) \), but is in \( \text{NTIME}(T') \)
“Delayed flip” on a “rapidly thickening diagonal”

- \( f(i+1) = \exp(f(i)) \)
- Let \( L \) be the “diagonal” language
- \( L'(j) = L(j+1) \)
- except if \( j = f(i) \), then \( L'(j) = 1 - L(f(i-1)+1) \)

\( L' \) not in \( \text{NTIME}(T) \), but is in \( \text{NTIME}(T') \)

Think \( \text{NTIME}(T) \subseteq \text{rows} \)
NTIME Hierarchy

- “Delayed flip” on a “rapidly thickening diagonal”

- $f(i+1) = \exp(f(i))$

- Let $L$ be the “diagonal” language

- $L'(j) = L(j+1)$

- except if $j = f(i)$, then $L'(j) = 1 - L(f(i-1)+1)$

- $L'$ not in NTIME(T), but is in NTIME(T’)

Think NTIME(T) ⊆ rows
“Delayed flip” on a “rapidly thickening diagonal”

- $f(i+1) = \exp(f(i))$
- Let $L$ be the “diagonal” language
- $L'(j) = L(j+1)$
- except if $j = f(i)$, then $L'(j) = 1 - L(f(i-1)+1)$

$L'$ not in $\text{NTIME}(T)$, but is in $\text{NTIME}(T')$
“Delayed flip” on a “rapidly thickening diagonal”

- \( f(i+1) = \exp(f(i)) \)
- Let \( L \) be the “diagonal” language
- \( L'(j) = L(j+1) \)
- except if \( j = f(i) \), then \( L'(j) = 1 - L(f(i-1)+1) \)

\( L' \) not in \( \text{NTIME}(T) \), but is in \( \text{NTIME}(T') \)

Think \( \text{NTIME}(T) \subseteq \text{rows} \)
Time Hierarchy
Time Hierarchy

Within DTIME and NTIME fine gradation
Time Hierarchy

- Within DTIME and NTIME fine gradation
- In particular $P \not\subset EXP$, $NP \not\subset NEXP$
Time Hierarchy

- Within DTIME and NTIME fine gradation
- In particular $P \subsetneq \text{EXP}$, $\text{NP} \subsetneq \text{NEXP}$
- Tells nothing across DTIME and NTIME
Time Hierarchy

- Within DTIME and NTIME fine gradation
  - In particular $P \subseteq \text{EXP}$, $NP \subseteq \text{NEXP}$
- Tells nothing across DTIME and NTIME
- $P$ and $NP$?
Time Hierarchy

- Within DTIME and NTIME fine gradation
  - In particular $P \subset EXP$, $NP \subset NEXP$
- Tells nothing across DTIME and NTIME
- $P$ and $NP$?
  - Just diagonalization won’t help (next lecture)
Today

- **DTIME Hierarchy**
  - \( \text{DTIME}(T) \subsetneq \text{DTIME}(T') \) if \( T \log T = o(T') \)

- **NTIME Hierarchy**
  - \( \text{NTIME}(T) \subsetneq \text{NTIME}(T') \) if \( T = o(T') \)

- Using diagonalization
Next Lecture

- Another application of diagonalization
  - **Ladner’s Theorem**: If \( P \neq NP \), NP language which is neither in \( P \) nor NP-complete

- Limits of Diagonalization

- Starting Space Complexity