Lecture 23

Oracle TMs and Limits of Diagonalization

Oracle Turing Machines

Idea:

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition:

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$.

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M.

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query},

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query}, then M moves to q_{yes} if $q \in O$

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query}, then M moves to q_{yes} if $q \in O$ and q_{no} if $q \notin O$,

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query}, then M moves to q_{yes} if $q \in O$ and q_{no} if $q \notin O$, where q denotes

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query}, then M moves to q_{yes} if $q \in O$ and q_{no} if $q \notin O$, where q denotes the contents of the special oracle tape.

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query}, then M moves to q_{yes} if $q \in O$ and q_{no} if $q \notin O$, where q denotes the contents of the special oracle tape. Output of oracle TM M with oracle access to O on

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query}, then M moves to q_{yes} if $q \in O$ and q_{no} if $q \notin O$, where q denotes the contents of the special oracle tape. Output of oracle TM M with oracle access to O on input x is denoted by $M^O(x)$.

Oracle Turing Machines

Idea: Oracle TMs have a way to magically solve some decision problem, say $O \subseteq \{0,1\}^*$.

- An oracle TM can write a string q on the special oracle tape and get the answer to "Is q in O?" in one step.
- If O is a difficult language, then it gives extra power to the TM.

Definition: An oracle TM is a TM M that has a special oracle tape and three special states $q_{query}, q_{yes}, q_{no}$. A language O is specified that is used as oracle for M. During the run, if M enters the state q_{query}, then M moves to q_{yes} if $q \in O$ and q_{no} if $q \notin O$, where q denotes the contents of the special oracle tape. Output of oracle TM M with oracle access to O on input x is denoted by $M^O(x)$.

Note: Nondeterministic oracle TMs are defined similarly.

Oracle Turing Machines

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, $\mathbf{P}^O$ and $\mathbf{NP}^O$ are the set of languages that can be

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$ and then output the opposite of it.

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$ and then output the opposite of it.

(2) Let $O \in \mathsf{P}$. Then, $\mathsf{P} = \mathsf{P}^O$.

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$ and then output the opposite of it.

(2) Let $O \in \mathsf{P}$. Then, $\mathsf{P} = \mathsf{P}^O$.

$\mathsf{P} \subseteq \mathsf{P}^O$ is trivial.

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$ and then output the opposite of it.

(2) Let $O \in \mathsf{P}$. Then, $\mathsf{P} = \mathsf{P}^O$.

$\mathsf{P} \subseteq \mathsf{P}^O$ is trivial. $\mathsf{P}^O \subseteq \mathsf{P}$ is true because any polytime oracle machine M with oracle O

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$ and then output the opposite of it.

(2) Let $O \in \mathsf{P}$. Then, $\mathsf{P} = \mathsf{P}^O$.

$\mathsf{P} \subseteq \mathsf{P}^O$ is trivial. $\mathsf{P}^O \subseteq \mathsf{P}$ is true because any polytime oracle machine M with oracle O can be converted into a polytime machine M'

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$ and then output the opposite of it.

(2) Let $O \in \mathsf{P}$. Then, $\mathsf{P} = \mathsf{P}^O$.

$\mathsf{P} \subseteq \mathsf{P}^O$ is trivial. $\mathsf{P}^O \subseteq \mathsf{P}$ is true because any polytime oracle machine M with oracle O can be converted into a polytime machine M' where instead of using O it simply

Oracle Turing Machines

Definition: For every $O \in \{0,1\}^*$, P^O and NP^O are the set of languages that can be decided by a polytime deterministic TMs and non-deterministic TMs with oracle access to O, respectively.

Examples:

(1) $\overline{SAT} \in \mathsf{P}^{SAT}$.

Polytime TM M will put input ϕ on oracle tape and ask its oracle whether $\phi \in SAT$ and then output the opposite of it.

(2) Let $O \in \mathsf{P}$. Then, $\mathsf{P} = \mathsf{P}^O$.

$\mathsf{P} \subseteq \mathsf{P}^O$ is trivial. $\mathsf{P}^O \subseteq \mathsf{P}$ is true because any polytime oracle machine M with oracle O can be converted into a polytime machine M' where instead of using O it simply computes whether $q \in O$ in polytime.

$\mathsf{P}^A = \mathsf{NP}^A$

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof:

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let $EXPCOM = \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: $1)$ $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let $EXPCOM = \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

2) $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$:

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

2) $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$:

Suppose L has polytime oracle verifier N with access to *EXPCOM*.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n\rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

2) $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$:

Suppose L has polytime oracle verifier N with access to *EXPCOM*.

Then exponential time TM N' for L on input x:

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

2) $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$:

Suppose L has polytime oracle verifier N with access to *EXPCOM*.

Then exponential time TM N' for L on input x:

- Simulates N on all possible us

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

2) $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$:

Suppose L has polytime oracle verifier N with access to *EXPCOM*.

Then exponential time TM N' for L on input x:

- Simulates N on all possible us and replace every call to oracle on $\langle M, y, 1^n \rangle$

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

2) $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$:

Suppose L has polytime oracle verifier N with access to *EXPCOM*.

Then exponential time TM N' for L on input x:

- Simulates N on all possible us and replace every call to oracle on $\langle M, y, 1^n \rangle$ by simulating M on y for 2^n step.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM}$: Trivially true.

2) $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$:

Suppose L has polytime oracle verifier N with access to *EXPCOM*.

Then exponential time TM N' for L on input x:

- Simulates N on all possible us and replace every call to oracle on $\langle M, y, 1^n \rangle$ by simulating M on y for 2^n step.
- Outputs 1 if $\exists u$ such that $N(x, u) = 1$.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let $\mathit{EXPCOM} = \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{\mathit{EXPCOM}} = \mathsf{NP}^{\mathit{EXPCOM}} = \mathsf{EXP}$$

Proof: 1) $\mathsf{P}^{\mathit{EXPCOM}} \subseteq \mathsf{NP}^{\mathit{EXPCOM}}$: Trivially true.

2) $\mathsf{NP}^{\mathit{EXPCOM}} \subseteq \mathsf{EXP}$:

Suppose L has polytime oracle verifier N with access to *EXPCOM*.

Then exponential time TM N' for L on input x:

- Simulates N on all possible us and replace every call to oracle on $\langle M, y, 1^n \rangle$ by simulating M on y for 2^n step.
- Outputs 1 if $\exists u$ such that $N(x, u) = 1$.

…

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof:

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let $EXPCOM = \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 3) $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: $3)$ $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 3) $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

Construct a polytime oracle machine N' with access to *EXPCOM* that decides L.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 3) $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

Construct a polytime oracle machine N' with access to *EXPCOM* that decides L.

N' on input x:

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 3) $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

Construct a polytime oracle machine N' with access to *EXPCOM* that decides L.

N' on input x:

- Writes $\langle N, x, 1^{(n+1)^c} \rangle$

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 3) $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

Construct a polytime oracle machine N' with access to *EXPCOM* that decides L.

N' on input x:

- Writes $\langle N, x, 1^{(n+1)^c} \rangle$ and call *EXPCOM* and output its answer.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 3) $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

Construct a polytime oracle machine N' with access to *EXPCOM* that decides L.

N' on input x:

- If x is sufficiently small then solve it by brute-force.
- Writes $\langle N, x, 1^{(n+1)^c} \rangle$ and call *EXPCOM* and output its answer.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: $3)$ EXP $\subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

Construct a polytime oracle machine N' with access to *EXPCOM* that decides L.

N' on input x:

smaller than the point where $2^{(n+1)^c}$ exceeds $k2^{n^c}$

- If x is sufficiently small then solve it by brute-force.
- Writes $\langle N, x, 1^{(n+1)^c} \rangle$ and call *EXPCOM* and output its answer.

$\mathsf{P}^A = \mathsf{NP}^A$

Claim: Let *EXPCOM* $= \{\langle M, x, 1^n \rangle \mid M$ outputs 1 on x within 2^n steps$\}$. Then,

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP}$$

Proof: 3) $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$:

Suppose L has a decider N that runs in at most $k2^{n^c}$ steps.

Construct a polytime oracle machine N' with access to *EXPCOM* that decides L.

N' on input x:

Smaller than the point where $2^{(n+1)^c}$ exceeds $k2^{n^c}$

- If x is sufficiently small then solve it by brute-force.
- Writes $\langle N, x, 1^{(n+1)^c} \rangle$ and call *EXPCOM* and output its answer.

■

Limits of Diagonalization

Limits of Diagonalization

Recall that

Limits of Diagonalization

Recall that

Limits of Diagonalization

Recall that

Diagonalization is any technique that relies solely upon the following properties of TMs:

- The existence of an effective representation of Turing machines by strings.
- The ability of one TM to simulate any another without much overhead in running time or space.

Limits of Diagonalization

Recall that

Diagonalization is any technique that relies solely upon the following properties of TMs:

- The existence of an effective representation of Turing machines by strings.
- The ability of one TM to simulate any another without much overhead in running time or space.

- For any oracle O, oracle TMs with access to O satisfy the above two properties.

Limits of Diagonalization

Recall that

Diagonalization is any technique that relies solely upon the following properties of TMs:

- The existence of an effective representation of Turing machines by strings.
- The ability of one TM to simulate any another without much overhead in running time or space.

- For any oracle O, oracle TMs with access to O satisfy the above two properties.
- Any result on TMs or complexity classes that uses only these two properties holds

Limits of Diagonalization

Recall that

Diagonalization is any technique that relies solely upon the following properties of TMs:

- The existence of an effective representation of Turing machines by strings.
- The ability of one TM to simulate any another without much overhead in running time or space.

- For any oracle O, oracle TMs with access to O satisfy the above two properties.
- Any result on TMs or complexity classes that uses only these two properties holds w.r.t oracle TMs with access to O as well.

Limits of Diagonalization

Recall that

Diagonalization is any technique that relies solely upon the following properties of TMs:

- The existence of an effective representation of Turing machines by strings.
- The ability of one TM to simulate any another without much overhead in running time or space.

- For any oracle O, oracle TMs with access to O satisfy the above two properties.
- Any result on TMs or complexity classes that uses only these two properties holds w.r.t oracle TMs with access to O as well. (We will see DTH w.r.t oracles soon)

Limits of Diagonalization

Recall that

Diagonalization is any technique that relies solely upon the following properties of TMs:

- The existence of an effective representation of Turing machines by strings.
- The ability of one TM to simulate any another without much overhead in running time or space.

- For any oracle O, oracle TMs with access to O satisfy the above two properties.
- Any result on TMs or complexity classes that uses only these two properties holds w.r.t oracle TMs with access to O as well. (We will see DTH w.r.t oracles soon)

Theorem (BGS75): There exist oracles A and B such that $\mathsf{P}^A = \mathsf{NP}^A$ and $\mathsf{P}^B \neq \mathsf{NP}^B$.

Limits of Diagonalization

Recall that

Diagonalization is any technique that relies solely upon the following properties of TMs:

- The existence of an effective representation of Turing machines by strings.
- The ability of one TM to simulate any another without much overhead in running time or space.

- For any oracle O, oracle TMs with access to O satisfy the above two properties.
- Any result on TMs or complexity classes that uses only these two properties holds w.r.t oracle TMs with access to O as well. (We will see DTH w.r.t oracles soon)

Theorem (BGS75): There exist oracles A and B such that $\mathbf{P}^A = \mathbf{NP}^A$ and $\mathbf{P}^B \neq \mathbf{NP}^B$.

Thus, **P** vs **NP** question cannot be settled through diagonalization "alone".

Deterministic Time Hierarchy with Oracle

Theorem: If $f:\mathbb{N}\to\mathbb{N}, g:\mathbb{N}\to\mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \rightarrow \mathbb{N}, g : \mathbb{N} \rightarrow \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof:

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \rightarrow \mathbb{N}, g : \mathbb{N} \rightarrow \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f: \mathbb{N} \to \mathbb{N}, g: \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \rightarrow \mathbb{N}, g : \mathbb{N} \rightarrow \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time,

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathsf{DTIME}(f(n))^O \subset \mathsf{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Let $L(D^O)$ denote the language decided by D.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathsf{DTIME}(f(n))^O \subset \mathsf{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Let $L(D^O)$ denote the language decided by D.

Claim 1**:** $L(D^O) \in \mathsf{DTIME}(g(n))^O$.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\mathrm{DTIME}(f(n))^O \subset \mathrm{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Let $L(D^O)$ denote the language decided by D.

Claim 1: $L(D^O) \in \mathrm{DTIME}(g(n))^O$.

Claim 2: $L(D^O) \notin \mathrm{DTIME}(f(n))^O$.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\text{DTIME}(f(n))^O \subset \text{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Let $L(D^O)$ denote the language decided by D.

Claim 1**:** $L(D^O) \in \text{DTIME}(g(n))^O$. **Proof:** By defn. … using time-constructibility…

Claim 2**:** $L(D^O) \notin \text{DTIME}(f(n))^O$.

…

Deterministic Time Hierarchy with Oracle

Theorem: If $f : \mathbb{N} \to \mathbb{N}, g : \mathbb{N} \to \mathbb{N}$ are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then for any $O \subseteq \{0,1\}^*$, $\text{DTIME}(f(n))^O \subset \text{DTIME}(g(n))^O$.

Proof: Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Let $L(D^O)$ denote the language decided by D.

Claim 1**:** $L(D^O) \in \text{DTIME}(g(n))^O$. **Proof:** By defn. … using time-constructibility…

Claim 2**:** $L(D^O) \notin \text{DTIME}(f(n))^O$. **Proof:** By contradiction …

…

Deterministic Time Hierarchy with Oracle

Claim 2: $L(D) \notin \mathsf{DTIME}(f(n))^{O}$.

Deterministic Time Hierarchy with Oracle

Claim 2: $L(D) \notin \mathsf{DTIME}(f(n))^O$.

Proof:

Deterministic Time Hierarchy with Oracle

Claim 2: $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

Deterministic Time Hierarchy with Oracle

Claim 2: $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.

Deterministic Time Hierarchy with Oracle

Claim 2: $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin$ DTIME$(f(n))^O$.

Proof: Suppose ∃ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose ∃ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Deterministic Time Hierarchy with Oracle

Claim 2: $L(D) \notin \text{DTIME}(f(n))^{O}$.

Proof: Suppose ∃ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \mathsf{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

Deterministic Time Hierarchy with Oracle

Claim 2: $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose ∃ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

Recall D:

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \mathsf{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

Recall D**:**

Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \mathsf{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

- M_x halts on x within $g(|x|)$ steps of U.

Recall D**:**

Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \mathrm{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

- M_x halts on x within $g(|x|)$ steps of U.
- If M_x accepts x,

Recall D**:**

Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin$ DTIME$(f(n))^O$.

Proof: Suppose ∃ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

- M_x halts on x within $g(|x|)$ steps of U.
- If M_x accepts x,
- If M_x rejects x,

Recall D**:**

Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose ∃ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

- M_x halts on x within $g(|x|)$ steps of U.
- If M_x accepts x, then D rejects x.
- If M_x rejects x,

Recall D**:**

Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose ∃ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

- M_x halts on x within $g(|x|)$ steps of U.
- If M_x accepts x, then D rejects x.
- If M_x rejects x, then D accepts x.

Recall D:

Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

Deterministic Time Hierarchy with Oracle

Claim 2**:** $L(D) \notin \text{DTIME}(f(n))^O$.

Proof: Suppose $\exists$ an oracle TM M with access to O and run time $O(f(n))$ that decides $L(D)$.

- M on any input x halts within $cf(|x|)$ steps, where c is a constant.
- UTM U with access to O can simulate M with access to O on input x within $c'f(|x|)\log f(|x|)$ steps, where c' is a constant…

Let x be a binary representation of M whose length is sufficiently large.

What happens when D with access to O gets x as input?

- M_x halts on x within $g(|x|)$ steps of U.
- If M_x accepts x, then D rejects x.
- If M_x rejects x, then D accepts x.

Recall D:

Consider an oracle TM D with access to O that on input x:

- Runs UTM U on (x, x) for $g(|x|)$ steps.
- If M_x with access to O halts on x and writes some bits on the output tape within this time, then D outputs the opposite of the first bit.
- Else, D outputs 0.

■