CSI - 3105 Design & Analysis of Algorithms
Course 17

Jean-Lou De Carufel

Fall 2019

For each of the 4 previous problems,

- Not known if it can be solved in polynomial time.

- If the answer to the question is YES, then
  - There is a "short" proof for this.

    Here, "short" means the length of the proof is "polynomial in the length of the input".

  - If someone gives us such a short proof, then we can "easily" verify this proof.

    Here, "easily" means "in polynomial time".

Complexity Class NP

A decision problem A is in NP if

- If for a given input I, the answer to the question $A(I)$ is YES, then there exists a proof/solution/certificate C such that
  - C is short (polynomial size in the length of I)
  - In polynomial time, we can verify that C is a correct proof for the fact that $A(I) = YES$.

Complexity Class *NP*

A decision problem A is in NP if

- If for a given input I, the answer to the question $A(I)$ is YES, then there exists a proof/solution/certificate C such that
  - C is short (polynomial size in the length of I)
  - In polynomial time, we can verify that C is a correct proof for the fact that $A(I) = YES$.

NP stands for *Nondeterministic Polynomial*.

Complexity Class *NP*

A decision problem A is in NP if

- If for a given input I, the answer to the question $A(I)$ is YES, then there exists a proof/solution/certificate C such that
  - C is short (polynomial size in the length of I)
  - In polynomial time, we can verify that C is a correct proof for the fact that $A(I) = YES$.

NP stands for *Nondeterministic Polynomial*.

The following problems are in NP:

HAM-CYCLE, TSP, SUBSET-SUM, CLIQUE

§6.2 A More Formal Approach Using Languages

Definition (Language of a Decision Problem)

The *language* of a decision problem is the set of all inputs (encoded as finite strings) for which the answer is YES.

§6.2 A More Formal Approach Using Languages

Definition (Language of a Decision Problem)

The *language* of a decision problem is the set of all inputs (encoded as finite strings) for which the answer is YES.

$$HAM-CYCLE = \{G \mid G \text{ is a graph that contains a Hamiltonian cycle}\}$$

§6.2 A More Formal Approach Using Languages

Definition (Language of a Decision Problem)

The *language* of a decision problem is the set of all inputs (encoded as finite strings) for which the answer is YES.

$$HAM-CYCLE = \{G \mid G \text{ is a graph that contains a Hamiltonian cycle}\}$$

$$\begin{aligned} TSP = \{(G,K) \mid\ & G \text{ is a complete directed graph } G=(V,E), \\ & \text{where each edge } (u,v) \in E \\ & \text{has a weight } wt(u,v) > 0, \\ & K \text{ is an integer} \\ & \text{and } G \text{ contains a Hamiltonian cycle} \\ & \text{with total weight at most } K.\} \end{aligned}$$

$$SUBSET-SUM = \{(S,t) \mid S \text{ is a set of integers, } t \text{ is an integer and } \exists S' \subseteq S \text{ such that } \sum_{x \in S'} x = t.\}$$

$$SUBSET-SUM = \{(S,t) \mid S \text{ is a set of integers, } t \text{ is an integer and } \exists S' \subseteq S \text{ such that } \sum_{x \in S'} x = t.\}$$

$$CLIQUE = \{(G,K) \mid G \text{ is an undirected graph, } K \text{ is an integer and } G \text{ contains a clique of size } K.\}$$

Definition (Complexity Class P)

The language L (of a decision problem) is in P if the following is true. There exists an algorithm A and a constant $c \geq 1$ such that for any input x,

- If $x \in L$, then $A(x)$ returns YES.
- If $x \notin L$, then $A(x)$ returns NO.
- The running time of $A(x)$ is $O(n^c)$, where n is the length of x.

Definition (Complexity Class P)

The language L (of a decision problem) is in P if the following is true. There exists an algorithm A and a constant $c \geq 1$ such that for any input x,

- $x \in L \iff A(x)$ returns YES.
- The running time of $A(x)$ is $O(n^c)$, where n is the length of x.

Definition (Complexity Class NP)

The language L (of a decision problem) is in NP if the following is true. There exists an algorithm V and a constant $c \geq 1$ such that for any input x,

$x \in L \iff$ there exists a certificate y such that

- $|y| = O(|x|^c)$,
- $V(x, y)$ returns YES
- and the running time of $V(x, y)$ is polynomial in the length of x.

Observe that V is a verification algorithm. It has 2 input parameters.

We show that

$$\text{HAMCYCLE} = \{G : G \text{ is a graph that has a Hamiltonian cycle}\}$$

is in NP:

Verification algorithm V takes as input

- graph $G = (V, E)$, where $n = |V|$
- certificate $v_1, v_2, \ldots, v_k$

Step 1: check if $k = n$?

Step 2: check if $v_1, v_2, \ldots, v_k$ are all different?

Step 3: check if $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$ are edges in G.

Step 4: if Steps 1-3 were successful, return YES, otherwise, return NO.

G is in HAMCYCLE

$\Longleftrightarrow$

$\exists$ permutation $v_1, v_2, \ldots, v_n$ of G's vertex set such that
$\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$ are edges in G.

$\Longleftrightarrow$

$\exists$ certificate $(v_1, v_2, \ldots, v_K)$ with $K = n$ such that $V(G, (v_1, v_2, \ldots, v_K))$ returns YES

the length of the certificate

$= \#$ of vertices in G

$= O(\text{size of } G)$

Running time of $V = O((\text{size of } G)^2)$

Theorem

$$P \subseteq NP$$

PROOF:

Theorem

$$P \subseteq NP$$

PROOF: Let L be an arbitrary language (of a decision problem) in P.

Theorem

$$P \subseteq NP$$

PROOF: Let L be an arbitrary language (of a decision problem) in P. By definition, there is an algorithm A such that for any input x,

- $x \in L \Longleftrightarrow A(x)$ returns YES.
- The running time of $A(x)$ is polynomial in the length of x.

Theorem

$$P \subseteq NP$$

PROOF: Let L be an arbitrary language (of a decision problem) in P. By definition, there is an algorithm A such that for any input x,

- $x \in L \Longleftrightarrow A(x)$ returns YES.
- The running time of $A(x)$ is polynomial in the length of x.

We have to show that L is in NP.

Theorem

$$P \subseteq NP$$

PROOF: Let L be an arbitrary language (of a decision problem) in P. By definition, there is an algorithm A such that for any input x,

- $x \in L \iff A(x)$ returns YES.
- The running time of $A(x)$ is polynomial in the length of x.

We have to show that L is in NP. That is, we have to show that L satisfies the definition of NP.

Theorem

$$P \subseteq NP$$

PROOF: Let L be an arbitrary language (of a decision problem) in P. By definition, there is an algorithm A such that for any input x,

- $x \in L \iff A(x)$ returns YES.
- The running time of $A(x)$ is polynomial in the length of x.

We have to show that L is in NP. That is, we have to show that L satisfies the definition of NP.

Therefore, we need a verification algorithm V. We define it in the following way:

Theorem

$$P \subseteq NP$$

PROOF: Let L be an arbitrary language (of a decision problem) in P. By definition, there is an algorithm A such that for any input x,

- $x \in L \iff A(x)$ returns YES.
- The running time of $A(x)$ is polynomial in the length of x.

We have to show that L is in NP. That is, we have to show that L satisfies the definition of NP.

Therefore, we need a verification algorithm V. We define it in the following way: V takes as input

- the input x for A
- and a certificate y

Theorem

$$P \subseteq NP$$

PROOF: Let L be an arbitrary language (of a decision problem) in P. By definition, there is an algorithm A such that for any input x,

- $x \in L \Longleftrightarrow A(x)$ returns YES.
- The running time of $A(x)$ is polynomial in the length of x.

We have to show that L is in NP. That is, we have to show that L satisfies the definition of NP.

Therefore, we need a verification algorithm V. We define it in the following way: V takes as input

- the input x for A
- and a certificate y

$V(x, y)$ does the following: it runs $A(x)$ and that's it! (It ignores y.)

We then have the following:

$$x \in L \iff$$

We then have the following:

$$x \in L \Longleftrightarrow A(x) \text{ returns YES}$$

We then have the following:

$$\begin{aligned} x \in L &\iff A(x) \text{ returns YES} \\ &\iff V(x, \text{empty string } y) \text{ returns YES} \end{aligned}$$

We then have the following:

$$\begin{aligned}
x \in L &\iff A(x) \text{ returns YES} \\
&\iff V(x, \text{empty string } y) \text{ returns YES} \\
&\iff \text{there exists a certificate } y \text{ such that} \\
&\qquad\qquad \cdot\ |y| \text{ is polynomial in the length of } x, \\
&\qquad\qquad \cdot\ V(x, y) \text{ returns YES} \\
&\qquad\qquad \cdot \text{ and the running time of } V(x, y) \text{ is polynomial} \\
&\qquad\qquad\ \ \text{ in the length of } x.
\end{aligned}$$

We then have the following:

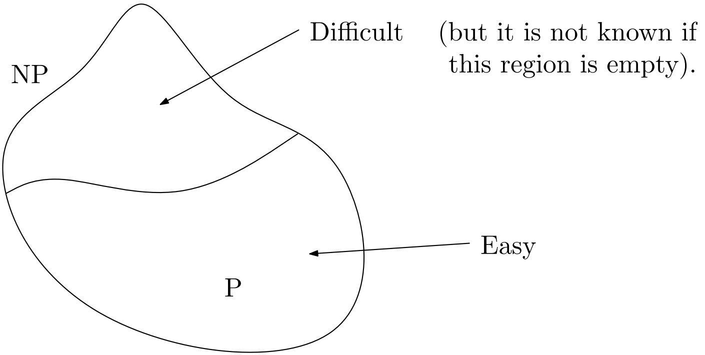
$$
\begin{aligned}
x \in L &\iff A(x) \text{ returns YES} \\
&\iff V(x, \text{empty string } y) \text{ returns YES} \\
&\iff \text{there exists a certificate } y \text{ such that} \\
&\qquad\qquad \cdot\ |y| \text{ is polynomial in the length of } x, \\
&\qquad\qquad \cdot\ V(x, y) \text{ returns YES} \\
&\qquad\qquad \cdot\ \text{and the running time of } V(x, y) \text{ is polynomial} \\
&\qquad\qquad\ \ \ \text{in the length of } x.
\end{aligned}
$$

Therefore L is in NP. $\square$

Big Question

Is $P = NP$ or $P \neq NP$?

Most people believe that $P \neq NP$.



If we want to prove that $P \neq NP$, then we have to show that there exists a language L (of a decision problem) such that

- $L \in NP$
- $L \notin P$.

If we want to prove that $P \neq NP$, then we have to show that there exists a language L (of a decision problem) such that

- $L \in NP$
- $L \notin P$.

Such an L must be "difficult".

If we want to prove that $P \neq NP$, then we have to show that there exists a language L (of a decision problem) such that

- $L \in NP$
- $L \notin P$.

Such an L must be "difficult".

So we should look at the "most difficult" problems.

If we want to prove that $P \neq NP$, then we have to show that there exists a language L (of a decision problem) such that

- $L \in NP$
- $L \notin P$.

Such an L must be "difficult".

So we should look at the "most difficult" problems.

But what does this mean?! How can we measure how difficult a problem is?!

§6.3 Reductions

Definition (Polynomial-Time Reduction)

Let L and L' be two languages. We say that L is *polynomial-time reducible to* L' if the following is true: There exists a function f which satisfies the following *famous* 3 *properties*:

1. f maps inputs for L to inputs for L'.
2. for every input x for L,

$$x \in L \Longleftrightarrow f(x) \in L'$$

3. for every input x for L, $f(x)$ can be computed in time that is polynomial in the length of x.

Notation: $L \leq_P L'$

What Does This Mean?

If we have a program A' that solves L', then we can use A' to solve L:

- Compute $x' = f(x)$
- Run A' on input x'.

Thus, we only have to write a program for the function f.

Example of a Reduction

$$CLIQUE = \{(G, K) \mid \text{graph } G \text{ has a clique with } K \text{ vertices.}\}$$
$$INDEP - SET = \{(G, K) \mid \text{graph } G \text{ has an independent set of } K \text{ vertices.}\}$$

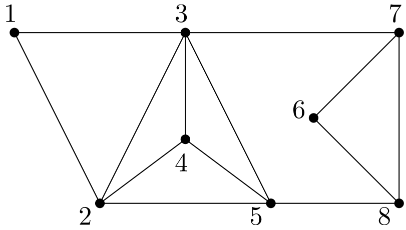
Clique: each pair of vertices is connected by an edge.

Independent set: no pair of vertices is connected by an edge.

Example of a Reduction

$$CLIQUE = \{(G, K) \mid \text{graph } G \text{ has a clique with } K \text{ vertices.}\}$$
$$INDEP-SET = \{(G, K) \mid \text{graph } G \text{ has an independent set of } K \text{ vertices.}\}$$

Clique: each pair of vertices is connected by an edge.
Independent set: no pair of vertices is connected by an edge.



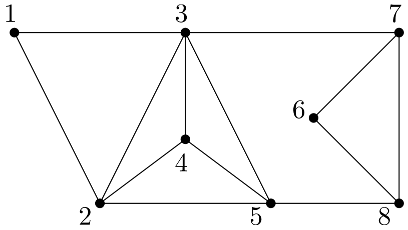
$\{2, 3, 4, 5\}$: clique of size 4

$\{1, 4, 6\}$: independent set of size 3

Example of a Reduction

$$CLIQUE = \{(G, K) \mid \text{graph } G \text{ has a clique with } K \text{ vertices.}\}$$
$$INDEP-SET = \{(G, K) \mid \text{graph } G \text{ has an independent set of } K \text{ vertices.}\}$$

Clique: each pair of vertices is connected by an edge.
Independent set: no pair of vertices is connected by an edge.



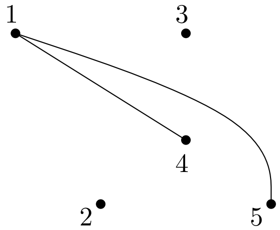
$\{2, 3, 4, 5\}$: clique of size 4

$\{1, 4, 6\}$: independent set of size 3

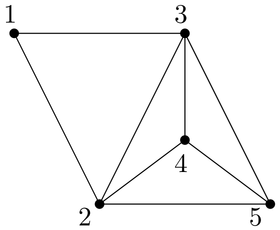
We want to show that

$$INDEP-SET \leq_P CLIQUE.$$

What is the connection between clique and independent set?

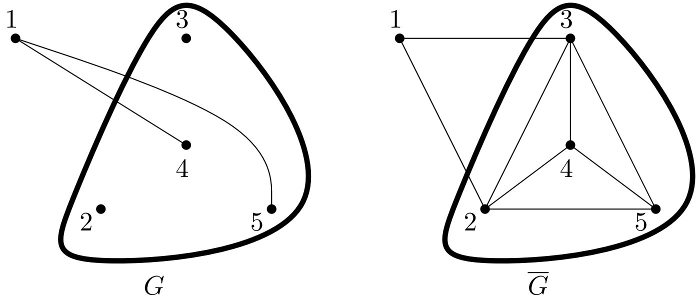


G

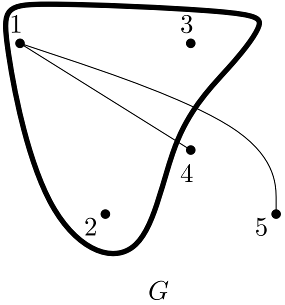


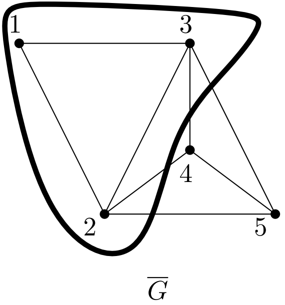
$\overline{G}$

What is the connection between clique and independent set?

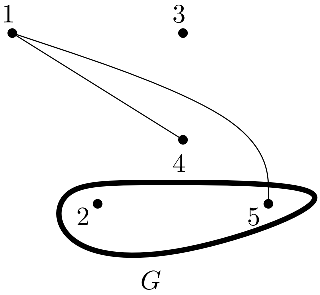


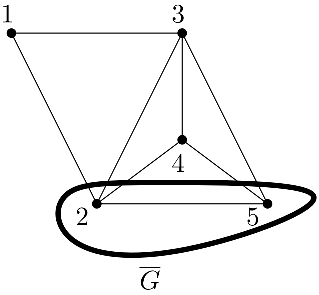
What is the connection between clique and independent set?



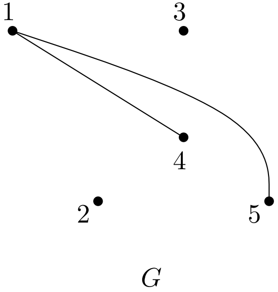


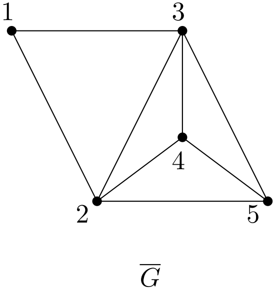
What is the connection between clique and independent set?



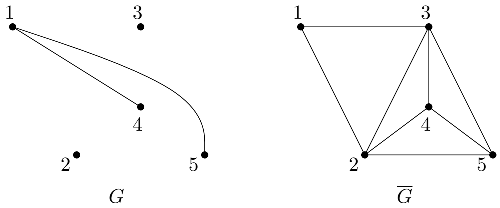


What is the connection between clique and independent set?

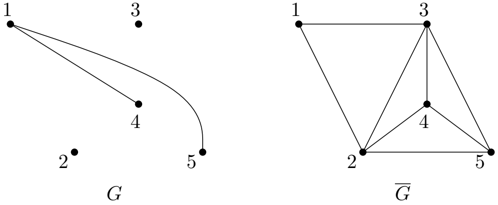




Is this a coincidence?

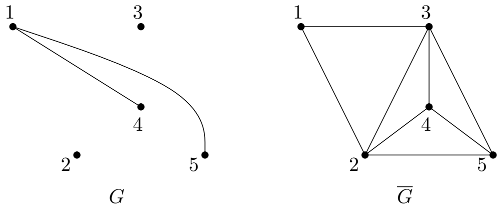


To prove the polynomial-time reduction $INDEP-SET \leq_P CLIQUE$, we need a function f which satisfies the *famous* 3 *properties*.



To prove the polynomial-time reduction $INDEP - SET \leq_P CLIQUE$, we need a function f which satisfies the *famous* 3 *properties*. We take

$$f(G, K) = \left(\overline{G}, K\right)$$

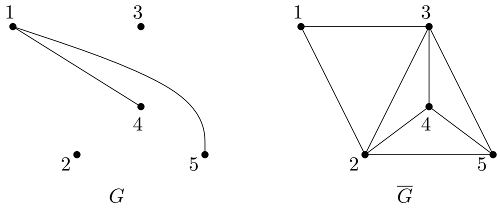


To prove the polynomial-time reduction $INDEP - SET \leq_P CLIQUE$, we need a function f which satisfies the *famous* 3 *properties*. We take

$$f(G, K) = \left(\overline{G}, K\right)$$

We have

1. f maps inputs for INDEP-SET to inputs for CLIQUE.



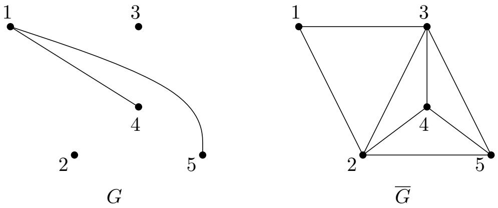
To prove the polynomial-time reduction $INDEP - SET \leq_P CLIQUE$, we need a function f which satisfies the *famous* 3 *properties*. We take

$$f(G, K) = \left(\overline{G}, K\right)$$

We have

1. f maps inputs for INDEP-SET to inputs for CLIQUE.

3. Time to construct $\left(\overline{G}, K\right)$, when given (G, K), is $O(|V| + |E|)$ which is polynomial in the size of (G, K).



To prove the polynomial-time reduction $INDEP - SET \leq_P CLIQUE$, we need a function f which satisfies the *famous* 3 *properties*. We take

$$f(G, K) = \left(\overline{G}, K\right)$$

We have

1. f maps inputs for INDEP-SET to inputs for CLIQUE.
2. Is it true that G has an independent set of size K if and only if $\overline{G}$ has a clique of size K?
3. Time to construct $\left(\overline{G}, K\right)$, when given (G, K), is $O(|V| + |E|)$ which is polynomial in the size of (G, K).

Let us prove Property 2.

Let us prove Property 2. We must prove that G has an independent set of size K if and only if $\overline{G}$ has a clique of size K.

Let us prove Property 2. We must prove that G has an independent set of size K if and only if $\overline{G}$ has a clique of size K.

Let $V' \subseteq V$ be an independent set of size K in G.

Let us prove Property 2. We must prove that G has an independent set of size K if and only if $\overline{G}$ has a clique of size K.

Let $V' \subseteq V$ be an independent set of size K in G.

V' is an independent set in $G \iff$

Let us prove Property 2. We must prove that G has an independent set of size K if and only if $\overline{G}$ has a clique of size K.

Let $V' \subseteq V$ be an independent set of size K in G.

$$V' \text{ is an independent set in } G \iff \text{for all vertices } u, v \in V' \text{ with } u \neq v, \{u, v\} \text{ is not an edge of } G.$$

Let us prove Property 2. We must prove that G has an independent set of size K if and only if $\overline{G}$ has a clique of size K.

Let $V' \subseteq V$ be an independent set of size K in G.

$$\begin{aligned} V' \text{ is an independent set in } G &\iff \text{for all vertices } u, v \in V' \text{ with } u \neq v, \\ &\qquad \{u, v\} \text{ is not an edge of } G. \\ &\iff \text{for all vertices } u, v \in V' \text{ with } u \neq v, \\ &\qquad \{u, v\} \text{ is an edge of } \overline{G}. \end{aligned}$$

Let us prove Property 2. We must prove that G has an independent set of size K if and only if $\overline{G}$ has a clique of size K.

Let $V' \subseteq V$ be an independent set of size K in G.

$$\begin{aligned}
V' \text{ is an independent set in } G &\iff \text{for all vertices } u, v \in V' \text{ with } u \neq v, \\
&\qquad\qquad \{u, v\} \text{ is not an edge of } G. \\
&\iff \text{for all vertices } u, v \in V' \text{ with } u \neq v, \\
&\qquad\qquad \{u, v\} \text{ is an edge of } \overline{G}. \\
&\iff V' \text{ is a clique in } \overline{G}
\end{aligned}$$

$\square$

Theorem

If $L \leq_P L'$ and $L' \in P$, then $L \in P$.

PROOF:

Theorem

If $L \leq_P L'$ and $L' \in P$, then $L \in P$.

Intuition:

- $L' \in P$ means "L' is easy".
- $L \leq_P L'$ means "L is easier than L'".

So L is easy. So $L \in P$.

PROOF:

Theorem

If $L \leq_P L'$ and $L' \in P$, then $L \in P$.

Intuition:

- $L' \in P$ means "L' is easy".
- $L \leq_P L'$ means "L is easier than L'".

So L is easy. So $L \in P$.

PROOF: Since $L' \in P$, there is a polynomial-time algorithm A' such that for all inputs x' for L'

$$x' \in L' \iff A'(x') \text{ returns YES.}$$

Theorem

If $L \leq_P L'$ and $L' \in P$, then $L \in P$.

Intuition:

- $L' \in P$ means "L' is easy".
- $L \leq_P L'$ means "L is easier than L'".

So L is easy. So $L \in P$.

PROOF: Since $L' \in P$, there is a polynomial-time algorithm A' such that for all inputs x' for L'

$$x' \in L' \iff A'(x') \text{ returns YES.}$$

Since $L \leq_P L'$, there is a function f satisfying the *famous* 3 *conditions*.

Consider the following algorithm A:

Consider the following algorithm A: on input x,

- Compute $f(x)$
- Run $A'(f(x))$

Consider the following algorithm A: on input x,

- Compute $f(x)$
- Run $A'(f(x))$

We have

$$x \in L \iff$$

Consider the following algorithm A: on input x,

- Compute $f(x)$
- Run $A'(f(x))$

We have

$$x \in L \Longleftrightarrow f(x) \in L' \qquad \text{by definition of reduction}$$

Consider the following algorithm A: on input x,

- Compute $f(x)$
- Run $A'(f(x))$

We have

$$\begin{aligned} x \in L &\iff f(x) \in L' && \text{by definition of reduction} \\ &\iff A'(f(x)) \text{ returns YES} && \text{by definition of } A' \end{aligned}$$

Consider the following algorithm A: on input x,

- Compute $f(x)$
- Run $A'(f(x))$

We have

$$\begin{aligned} x \in L &\iff f(x) \in L' && \text{by definition of reduction} \\ &\iff A'(f(x)) \text{ returns YES} && \text{by definition of } A' \\ &\iff A(x) \text{ returns YES} && \text{by definition of } A \end{aligned}$$

Consider the following algorithm A: on input x,

- Compute $f(x)$
- Run $A'(f(x))$

We have

$$\begin{aligned} x \in L \iff & f(x) \in L' && \text{by definition of reduction} \\ \iff & A'(f(x)) \text{ returns YES} && \text{by definition of } A' \\ \iff & A(x) \text{ returns YES} && \text{by definition of } A \end{aligned}$$

The running time of A is polynomial in the length of x. So $L \in P$. □