Nondeterministic polynomial time

Lecture 22

Thursday, November 26, 2020

LaTeXed: October 22, 2020 14:05

22.1
Review

22.1.1
Review: Polynomial reductions

Polynomial-time Reduction

Definition 22.1.

$X \leq_P Y$: **polynomial time reduction** from a decision problem X to a decision problem Y is an algorithm $\mathcal{A}$ such that:

1. Given an instance I_X of X, $\mathcal{A}$ produces an instance I_Y of Y.
2. $\mathcal{A}$ runs in time polynomial in $|I_X|$. $\quad$ ($|I_Y|$ = size of I_Y).
3. Answer to I_X YES $\iff$ answer to I_Y is YES.

Polynomial-time Reduction

Definition 22.1.

$X \leq_P Y$: **polynomial time reduction** from a decision problem X to a decision problem Y is an algorithm $\mathcal{A}$ such that:

1. Given an instance I_X of X, $\mathcal{A}$ produces an instance I_Y of Y.
2. $\mathcal{A}$ runs in time polynomial in $|I_X|$. $(|I_Y| = \text{size of } I_Y)$.
3. Answer to I_X YES $\iff$ answer to I_Y is YES.

Proposition 22.2.

If $X \leq_P Y$ then a polynomial time algorithm for Y implies a polynomial time algorithm for X.

Polynomial-time Reduction

Definition 22.1.

$X \leq_P Y$: **polynomial time reduction** from a decision problem X to a decision problem Y is an algorithm $\mathcal{A}$ such that:

1. Given an instance I_X of X, $\mathcal{A}$ produces an instance I_Y of Y.
2. $\mathcal{A}$ runs in time polynomial in $|I_X|$. ($|I_Y|$ = size of I_Y).
3. Answer to I_X YES $\iff$ answer to I_Y is YES.

Proposition 22.2.

If $X \leq_P Y$ then a polynomial time algorithm for Y implies a polynomial time algorithm for X.

This is a **Karp reduction**.

Composing polynomials...

A quick reminder

1. f and g monotone increasing. Assume that:
    1. $f(n) \leq a * n^b$ (i.e., $f(n) = O(n^b)$)
    2. $g(n) \leq c * n^d$ (i.e., $g(n) = O(n^d)$)

    a, b, c, d: constants.
2. $g\Big(f(n)\Big) \leq g\big(a * n^b\big) \leq c * \big(a * n^b\big)^d \leq c \cdot a^d * n^{bd}$
3. $\implies g(f(n)) = O\Big(n^{bd}\Big)$ is a polynomial.
4. **Conclusion:** Composition of two polynomials, is a polynomial.

Composing polynomials...

A quick reminder

1. f and g monotone increasing. Assume that:
    1. $f(n) \le a * n^b$ (i.e., $f(n) = O(n^b)$)
    2. $g(n) \le c * n^d$ (i.e., $g(n) = O(n^d)$)

    a, b, c, d: constants.
2. $g\Big(f(n)\Big) \le g\left(a * n^b\right) \le c * \left(a * n^b\right)^d \le c \cdot a^d * n^{bd}$
3. $\implies g(f(n)) = O\left(n^{bd}\right)$ is a polynomial.
4. **Conclusion:** Composition of two polynomials, is a polynomial.

Composing polynomials...

A quick reminder

1. f and g monotone increasing. Assume that:
    1. $f(n) \leq a * n^b$ (i.e., $f(n) = O(n^b)$)
    2. $g(n) \leq c * n^d$ (i.e., $g(n) = O(n^d)$)

   a, b, c, d: constants.
2. $g\Big(f(n)\Big) \leq g\big(a * n^b\big) \leq c * \big(a * n^b\big)^d \leq c \cdot a^d * n^{bd}$
3. $\implies g(f(n)) = O\Big(n^{bd}\Big)$ is a polynomial.
4. **Conclusion:** Composition of two polynomials, is a polynomial.

Composing polynomials...

A quick reminder

1. f and g monotone increasing. Assume that:
   1. $f(n) \leq a * n^b$ (i.e., $f(n) = O(n^b)$)
   2. $g(n) \leq c * n^d$ (i.e., $g(n) = O(n^d)$)

   a, b, c, d: constants.
2. $g\Big(f(n)\Big) \leq g\big(a * n^b\big) \leq c * \big(a * n^b\big)^d \leq c \cdot a^d * n^{bd}$
3. $\implies g(f(n)) = O\Big(n^{bd}\Big)$ is a polynomial.
4. **Conclusion:** Composition of two polynomials, is a polynomial.

Composing polynomials...

A quick reminder

1. f and g monotone increasing. Assume that:
   1. $f(n) \leq a * n^b$ (i.e., $f(n) = O(n^b)$)
   2. $g(n) \leq c * n^d$ (i.e., $g(n) = O(n^d)$)

   a, b, c, d: constants.
2. $g\Big(f(n)\Big) \leq g\big(a * n^b\big) \leq c * \big(a * n^b\big)^d \leq c \cdot a^d * n^{bd}$
3. $\implies g(f(n)) = O\Big(n^{bd}\Big)$ is a polynomial.
4. **Conclusion:** Composition of two polynomials, is a polynomial.

Composing polynomials...

A quick reminder

1. f and g monotone increasing. Assume that:
    1. $f(n) \leq a * n^b$ (i.e., $f(n) = O(n^b)$)
    2. $g(n) \leq c * n^d$ (i.e., $g(n) = O(n^d)$)

    a, b, c, d: constants.
2. $g\Big(f(n)\Big) \leq g\left(a * n^b\right) \leq c * \left(a * n^b\right)^d \leq c \cdot a^d * n^{bd}$
3. $\implies g(f(n)) = O\Big(n^{bd}\Big)$ is a polynomial.
4. **Conclusion:** Composition of two polynomials, is a polynomial.

Composing polynomials...

A quick reminder

1. f and g monotone increasing. Assume that:
   1. $f(n) \leq a * n^b$ (i.e., $f(n) = O(n^b)$)
   2. $g(n) \leq c * n^d$ (i.e., $g(n) = O(n^d)$)

   a, b, c, d: constants.
2. $g\Big(f(n)\Big) \leq g\left(a * n^b\right) \leq c * \left(a * n^b\right)^d \leq c \cdot a^d * n^{bd}$
3. $\implies g(f(n)) = O\Big(n^{bd}\Big)$ is a polynomial.
4. **Conclusion:** Composition of two polynomials, is a polynomial.

Transitivity of Reductions

Proposition 22.3.

$X \leq_P Y$ and $Y \leq_P Z$ implies that $X \leq_P Z$.

1. Note: $X \leq_P Y$ does not imply that $Y \leq_P X$ and hence it is very important to know the FROM and TO in a reduction.
2. To prove $X \leq_P Y$ you need to show a reduction FROM X TO Y
3. ...show that an algorithm for Y implies an algorithm for X.

Transitivity of Reductions

Proposition 22.3.

$X \leq_P Y$ and $Y \leq_P Z$ implies that $X \leq_P Z$.

1. Note: $X \leq_P Y$ does not imply that $Y \leq_P X$ and hence it is very important to know the FROM and TO in a reduction.
2. To prove $X \leq_P Y$ you need to show a reduction FROM X TO Y
3. ...show that an algorithm for Y implies an algorithm for X.

Transitivity of Reductions

Proposition 22.3.

$X \leq_P Y$ and $Y \leq_P Z$ implies that $X \leq_P Z$.

1. Note: $X \leq_P Y$ does not imply that $Y \leq_P X$ and hence it is very important to know the FROM and TO in a reduction.
2. To prove $X \leq_P Y$ you need to show a reduction FROM X TO Y
3. ...show that an algorithm for Y implies an algorithm for X.

Polynomial time reduction...

Proving Correctness of Reductions

To prove that $X \leq_P Y$ you need to give an algorithm $\mathcal{A}$ that:

1. Transforms an instance I_X of X into an instance I_Y of Y.
2. Satisfies the property that answer to I_X is YES iff I_Y is YES.
    1. typical easy direction to prove: answer to I_Y is YES if answer to I_X is YES
    2. typical difficult direction to prove: answer to I_X is YES if answer to I_Y is YES (equivalently answer to I_X is NO if answer to I_Y is NO).
3. Runs in **polynomial** time.

Polynomial time reduction...

Proving Correctness of Reductions

To prove that $X \leq_P Y$ you need to give an algorithm $\mathcal{A}$ that:

1. Transforms an instance I_X of X into an instance I_Y of Y.
2. Satisfies the property that answer to I_X is YES iff I_Y is YES.
    1. typical easy direction to prove: answer to I_Y is YES if answer to I_X is YES
    2. typical difficult direction to prove: answer to I_X is YES if answer to I_Y is YES (equivalently answer to I_X is NO if answer to I_Y is NO).
3. Runs in **polynomial** time.

THE END

...

(for now)