Complexity Classes and Polynomial Reductions

Algorithms and Complexity Theory

Matei Popovici[1]

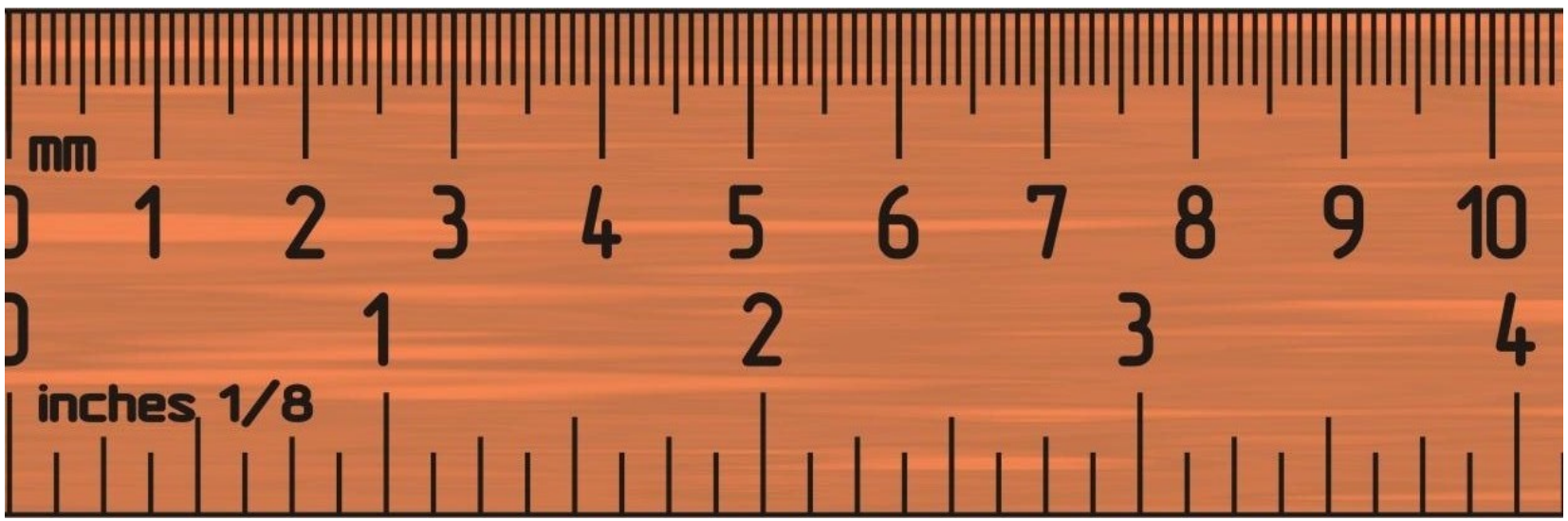
[1]POLITEHNICA University of Bucharest
Computer Science and Engineering Department, Bucharest, Romania

December 14, 2012

*Is it possible to construct a **hierarchy** of problems, based on hardness ?*

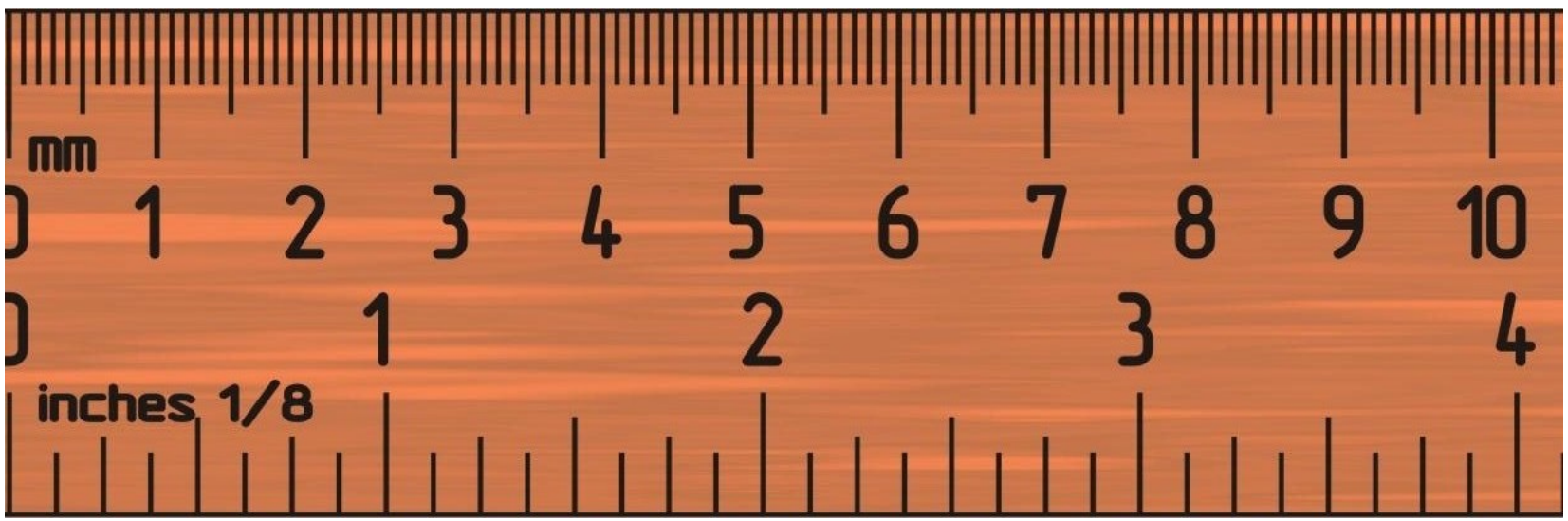
A hierarchy of complexity classes

P



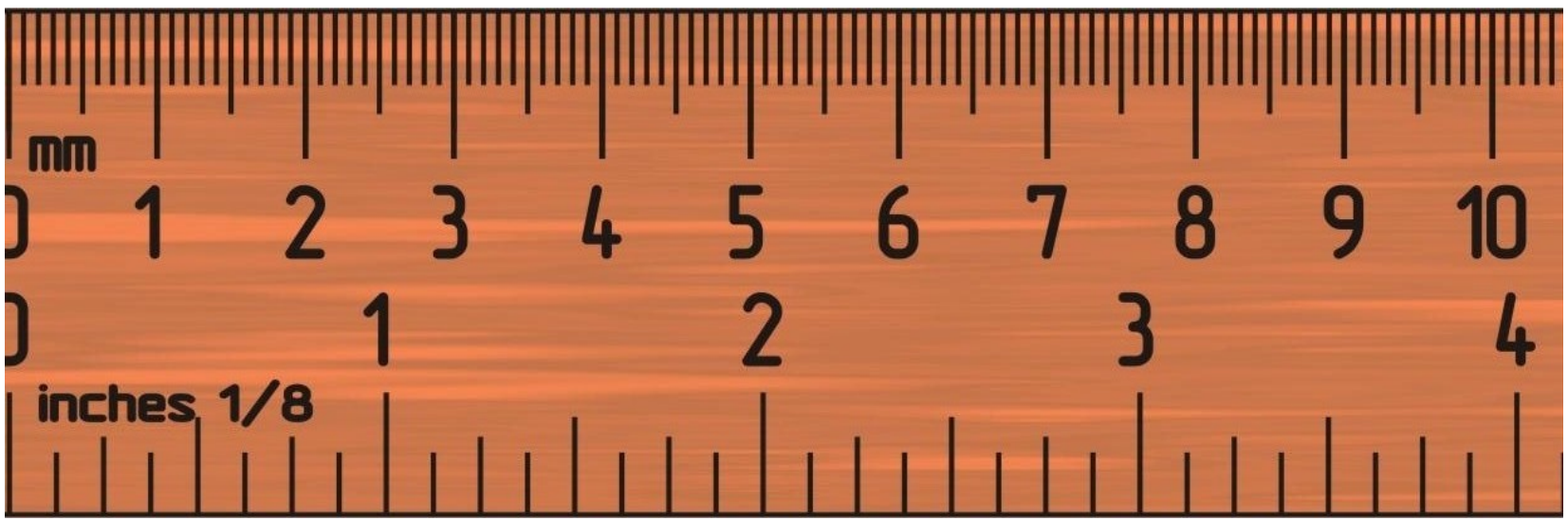
A hierarchy of complexity classes

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\}$$



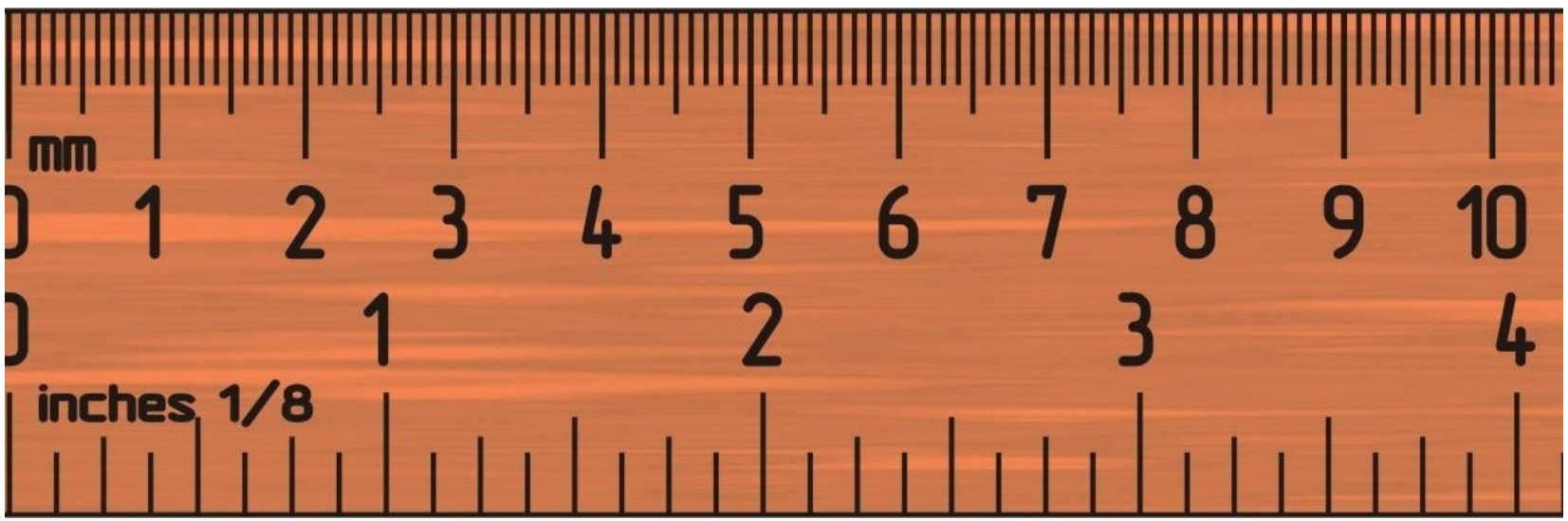
A hierarchy of complexity classes

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots$$



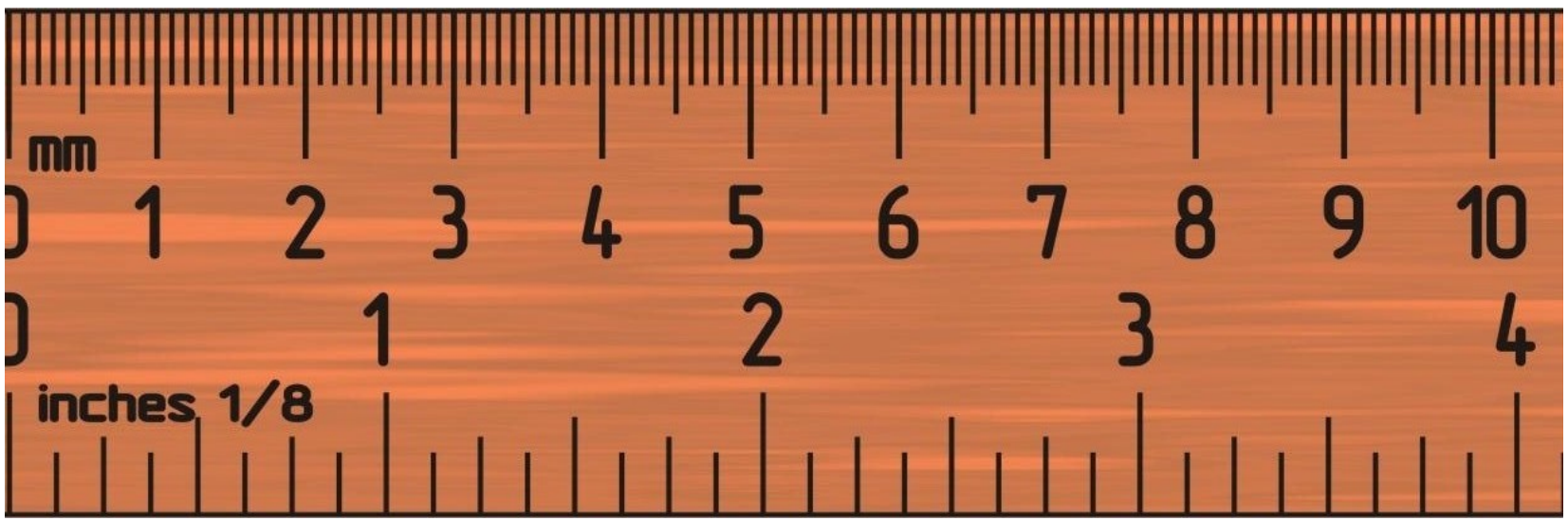
A hierarchy of complexity classes

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE$$



A hierarchy of complexity classes

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$



Complement

What is contained in *coNP* ?

Complement

What is contained in *coNP* ?

- The **complements** of problems from NP

Complement

What is contained in *coNP* ?

- The **complements** of problems from NP
- A problem in NP asks if there exists a path leading to `success`

Complement

What is contained in *coNP* ?

- The **complements** of problems from NP
- A problem in NP asks if there exists a path leading to `success`
- A problem in coNP asks if **all** paths lead to `fail`

Complement

What is contained in *coNP* ?

- The **complements** of problems from NP
- A problem in NP asks if there exists a path leading to `success`
- A problem in coNP asks if **all** paths lead to `fail`
- Example: Given a graph $G = (V, E)$, there exists $S \subseteq V$ such that S is a $k - clique$?

Complement

What is contained in *coNP* ?

- The **complements** of problems from NP
- A problem in NP asks if there exists a path leading to `success`
- A problem in coNP asks if **all** paths lead to `fail`
- Example: Given a graph $G = (V, E)$, there exists $S \subseteq V$ such that S is a $k - clique$? NP !

Complement

What is contained in *coNP* ?

- The **complements** of problems from NP
- A problem in NP asks if there exists a path leading to `success`
- A problem in coNP asks if **all** paths lead to `fail`
- Example: Given a graph $G = (V, E)$, there exists $S \subseteq V$ such that S is a $k - clique$? NP !
- Given a graph $G = (V, E)$ all subsets $S \subseteq V$ of size k are **not** cliques ? coNP !

Complement

What is contained in *coNP* ?

- The **complements** of problems from NP
- A problem in NP asks if there exists a path leading to `success`
- A problem in coNP asks if **all** paths lead to `fail`
- Example: Given a graph $G = (V, E)$, there exists $S \subseteq V$ such that S is a $k - clique$? NP !
- Given a graph $G = (V, E)$ all subsets $S \subseteq V$ of size k are **not** cliques ? coNP !
- to answer *yes* to this problem means to answer *no* to its complement

Containment

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

Containment

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- It is not known whether the above containments are **strict** or not

Containment

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- It is not known whether the above containments are **strict** or not
- If any of the above complexity classes are shown to be equal, then *most* of the hierarchy **collapses**:

Containment

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- It is not known whether the above containments are **strict** or not
- If any of the above complexity classes are shown to be equal, then *most* of the hierarchy **collapses**:
  - Implications of $P = NP$:

Containment

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- It is not known whether the above containments are **strict** or not
- If any of the above complexity classes are shown to be equal, then *most* of the hierarchy **collapses**:
  - Implications of $P = NP$:
    - All cryptographic frameworks are useless

Containment

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- It is not known whether the above containments are **strict** or not
- If any of the above complexity classes are shown to be equal, then *most* of the hierarchy **collapses**:
  - Implications of $P = NP$:
    - All cryptographic frameworks are useless
    - Verifying partial correctness becomes easy. Any program can be verified

Containment

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- It is not known whether the above containments are **strict** or not
- If any of the above complexity classes are shown to be equal, then *most* of the hierarchy **collapses**:
  - Implications of $P = NP$:
    - All cryptographic frameworks are useless
    - Verifying partial correctness becomes easy. Any program can be verified
    - Planning problems in: HPC systems, air space control, etc. are feasible.

Classifying a problem

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

Classifying a problem

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- Given a problem **Q** and a complexity class X such that $\mathbf{Q} \in X$, then X is an upper bound on the difficulty of **Q**

Classifying a problem

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- Given a problem **Q** and a complexity class X such that $\mathbf{Q} \in X$, then X is an upper bound on the difficulty of **Q**
- How can we provide with lower bounds for **Q** ?

Hardness

When is a problem Q ***harder*** *than a problem Q′ ?*

Hardness

When is a problem Q ***harder*** *than a problem Q′ ?*

Definition (Reduction)

A problem $Q_1 : I_1 \to \{0, 1\}$ is **reducible** to a problem $Q_2 : I_2 \to \{0, 1\}$ (written $Q_1 \leq_p Q_2$) if there exists a transformation $F : I_1 \to I_2$ such that:

Hardness

When is a problem Q ***harder*** *than a problem Q′ ?*

Definition (Reduction)

A problem $Q_1 : I_1 \to \{0,1\}$ is **reducible** to a problem $Q_2 : I_2 \to \{0,1\}$ (written $Q_1 \leq_p Q_2$) if there exists a transformation $F : I_1 \to I_2$ such that:

- F can be computed in polynomial time by a DTM

Hardness

When is a problem Q ***harder*** *than a problem Q′ ?*

Definition (Reduction)

A problem $Q_1 : I_1 \to \{0,1\}$ is **reducible** to a problem $Q_2 : I_2 \to \{0,1\}$ (written $Q_1 \leq_p Q_2$) if there exists a transformation $F : I_1 \to I_2$ such that:

- F can be computed in polynomial time by a DTM
- $Q_1(w) = 1 \iff Q_2(F(w)) = 1$

Hardness

When is a problem Q ***harder*** *than a problem Q′ ?*

Definition (Reduction)

A problem $Q_1 : I_1 \to \{0,1\}$ is **reducible** to a problem $Q_2 : I_2 \to \{0,1\}$ (written $Q_1 \leq_p Q_2$) if there exists a transformation $F : I_1 \to I_2$ such that:

- F can be computed in polynomial time by a DTM
- $Q_1(w) = 1 \iff Q_2(F(w)) = 1$

Proposition

$Q_2 \in P$ and $Q_1 \leq_p Q_2$, then $Q_1 \in P$.

Hardness

*When is a problem Q **harder** than a problem Q' ?*

Definition (Reduction)

A problem $Q_1 : I_1 \to \{0,1\}$ is **reducible** to a problem $Q_2 : I_2 \to \{0,1\}$ (written $Q_1 \leq_p Q_2$) if there exists a transformation $F : I_1 \to I_2$ such that:

- F can be computed in polynomial time by a DTM
- $Q_1(w) = 1 \iff Q_2(F(w)) = 1$

Proposition

$Q_2 \in P$ and $Q_1 \leq_p Q_2$, then $Q_1 \in P$.

Proposition

$Q_1 \notin P$ and $Q_1 \leq_p Q_2$, then $Q_2 \notin P$.

Hardness

Definition (Hardness)

Let X be a complexity class and Q be a problem. We say Q is X-**hard** if $Q' \leq_p Q$, for all $Q' \in X$

Hardness

Definition (Hardness)

Let X be a complexity class and Q be a problem. We say Q is X-**hard** if $Q' \leq_p Q$, for all $Q' \in X$

*We can use Q to solve **any** problem in X*

Hardness

Definition (Hardness)

Let X be a complexity class and Q be a problem. We say Q is X**-hard** if $Q' \leq_p Q$, for all $Q' \in X$

We can use Q to solve ***any*** *problem in X*

Definition (Completeness)

We say a problem Q is X-complete if it is X-hard and $Q \in X$

Hardness

Definition (Hardness)

Let X be a complexity class and Q be a problem. We say Q is X-**hard** if $Q' \leq_p Q$, for all $Q' \in X$

*We can use Q to solve **any** problem in X*

Definition (Completeness)

We say a problem Q is X-complete if it is X-hard and $Q \in X$

Proposition

$\leq_p$ *is **transitive**.*

Hardness

Definition (Hardness)

Let X be a complexity class and Q be a problem. We say Q is X-**hard** if $Q' \leq_p Q$, for all $Q' \in X$

*We can use Q to solve **any** problem in X*

Definition (Completeness)

We say a problem Q is X-complete if it is X-hard and $Q \in X$

Proposition

$\leq_p$ *is* ***transitive****.*

Proposition (Tool for proving hardness)

Let Q, Q' be two problems such that Q' is X-hard and $Q' \leq_p Q$. Then Q is X-hard.

Hardness

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- Let Q be an NP-hard problem.

Hardness

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- Let Q be an NP-hard problem. Then, Q is **at least as hard** as any problem in NP (any problem in NP can be solved using Q)

Hardness

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- Let Q be an NP-hard problem. Then, Q is **at least as hard** as any problem in NP (any problem in NP can be solved using Q)
- Let Q be an NP-complete problem.

Hardness

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- Let Q be an NP-hard problem. Then, Q is **at least as hard** as any problem in NP (any problem in NP can be solved using Q)
- Let Q be an NP-complete problem. Then Q is one of the **hardest** problems in NP. (All NP-complete problems are *equally hard*)

Hardness

$$P \subseteq \left\{ \begin{array}{c} NP \\ coNP \end{array} \right\} \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n \subseteq \ldots \subseteq PSPACE \subseteq EXPTIME$$

- Let Q be an NP-hard problem. Then, Q is **at least as hard** as any problem in NP (any problem in NP can be solved using Q)
- Let Q be an NP-complete problem. Then Q is one of the **hardest** problems in NP. (All NP-complete problems are *equally hard*)
- Finding lower bounds for a problem $Q \equiv$ proving **hardness** w.r.t. a complexity class X.

New implications

Proposition

Let Q be NP-hard and also $Q \in P$. Then $P = NP$.

New implications

Proposition

Let Q be NP-hard and also $Q \in P$. Then $P = NP$.

If one **efficiently** solves some *NP*-complete problem, then *most* of the complexity hierarchy collapses.

New implications

Proposition

Let Q be NP-hard and also $Q \in P$. Then $P = NP$.

If one **efficiently** solves some *NP*-complete problem, then *most* of the complexity hierarchy collapses.

I conjecture that there is no good algorithm for the travelling salesman problem. My reasons are the same as for any mathematical conjecture: (i) it is a legitimate mathematical possibility, and (ii) I do not know. (Jack Edmonds, 1966)

*There exist problems which **cannot be solved efficiently** ?*

- How do we formally express that a problem is **solvable** ?
- Is it the case that, *finding the correct answer* is harder than *checking if a **given answer is correct*** ?
- How do we express that a problem is **harder** than another problem ?
- Is it possible to construct a **hierarchy** of problems, based on hardness ?

Bibliography I


Sanjeev Arora and Boaz Barak.
Computational Complexity: A Modern Approach.
Cambridge University Press, New York, NY, USA, 1st edition, 2009.


Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson.
Introduction to Algorithms.
McGraw-Hill Higher Education, 2nd edition, 2001.


C.H. Papadimitriou.
Computational complexity.
Addison-Wesley, 1994.