



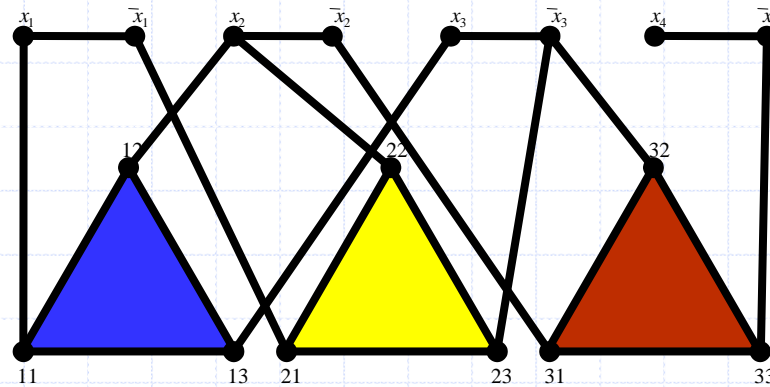
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ  
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

# Αλγόριθμοι και πολυπλοκότητα

## NP-Completeness

Ιωάννης Τόλλης  
Τμήμα Επιστήμης Υπολογιστών

# NP-Completeness



# Outline and Reading

## ◆ P and NP (§13.1)

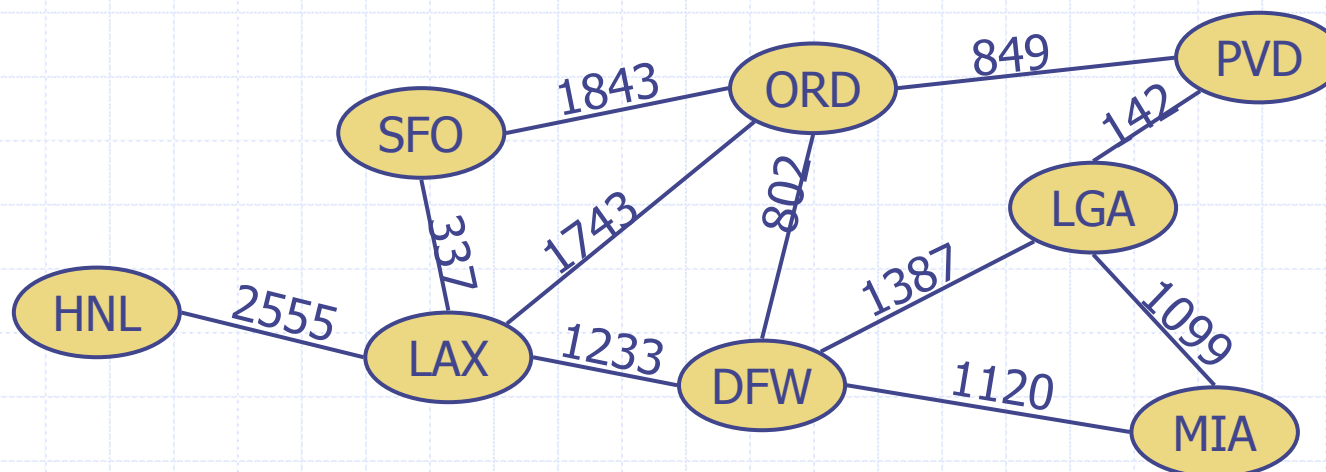
- Definition of P
- Definition of NP
- Alternate definition of NP

## ◆ NP-completeness (§13.2)

- Definition of NP-hard and NP-complete
- The Cook-Levin Theorem

# Running Time Revisited

- ◆ Input size,  $n$ 
  - To be exact, let  $n$  denote the number of **bits** in a nonunary encoding of the input
- ◆ All the polynomial-time algorithms studied so far in this course run in polynomial time using this definition of input size.
  - Exception: any pseudo-polynomial time algorithm



# Dealing with Hard Problems

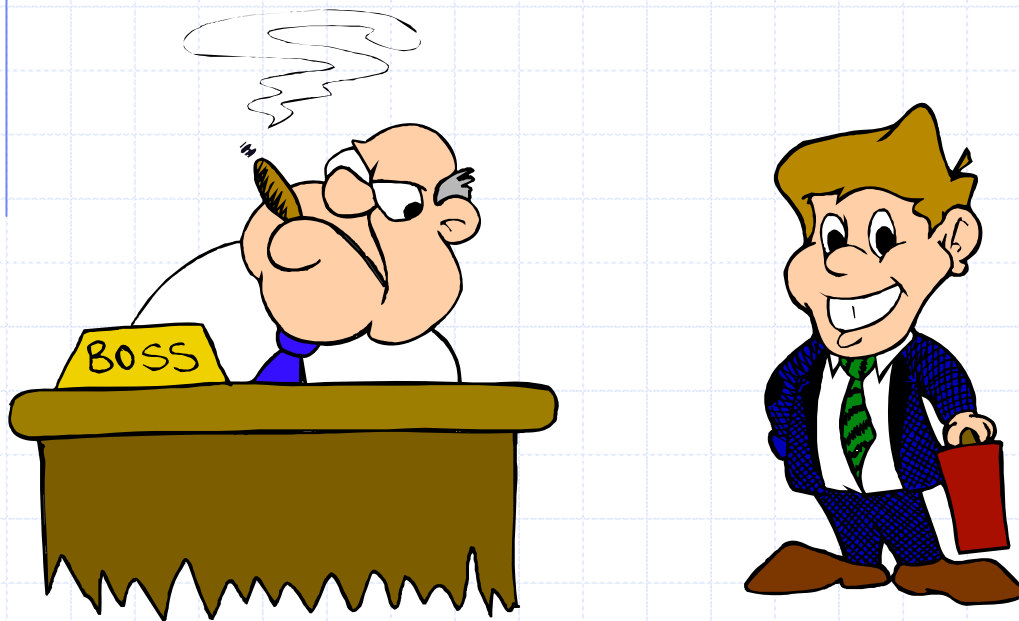
- ◆ What to do when we find a problem that looks hard...



I couldn't find a polynomial-time algorithm;  
I guess I'm too dumb.

# Dealing with Hard Problems

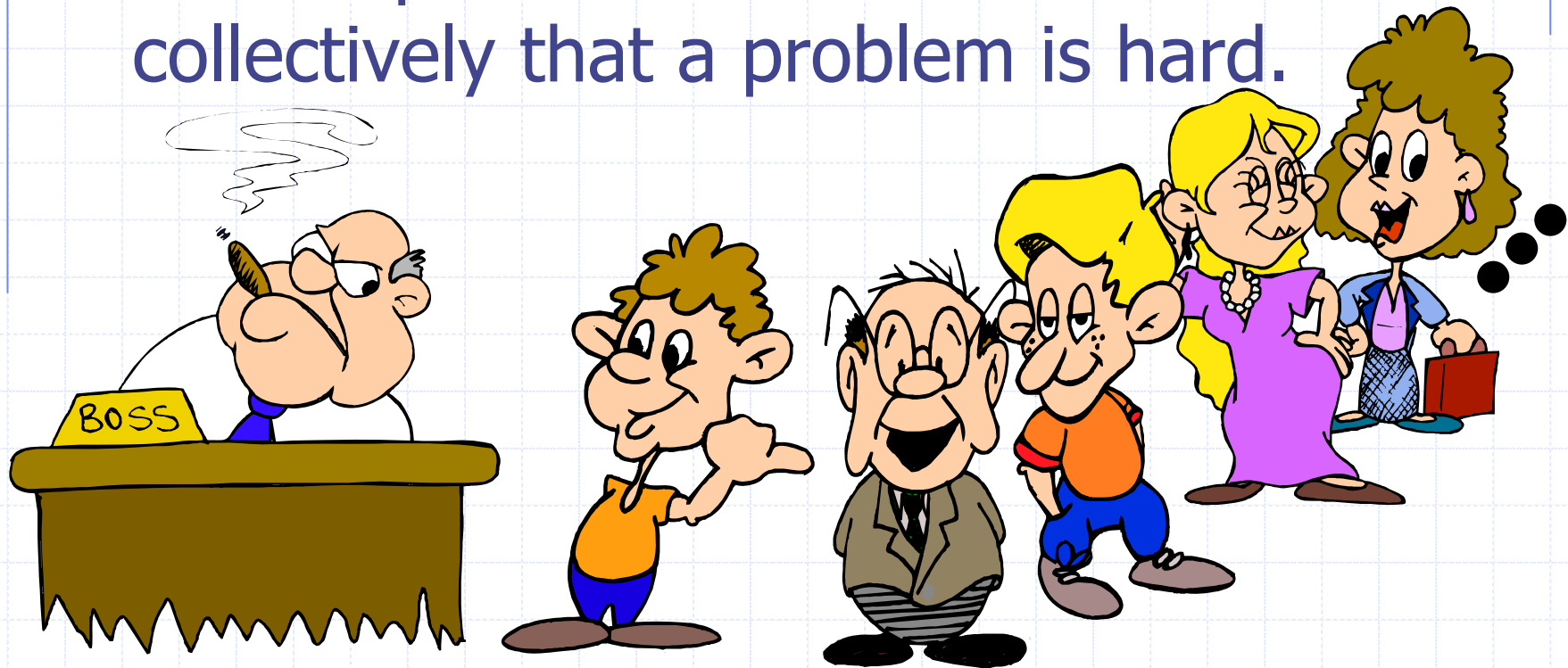
- ◆ Sometimes we can prove a strong lower bound... (but not usually)



I couldn't find a polynomial-time algorithm,  
because no such algorithm exists!

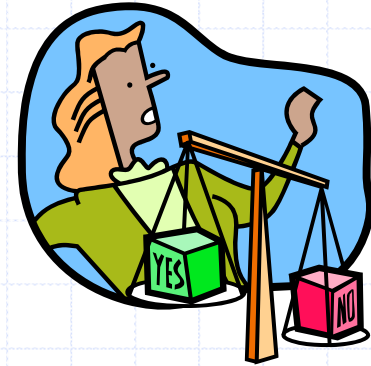
# Dealing with Hard Problems

- ◆ NP-completeness let's us show collectively that a problem is hard.



I couldn't find a polynomial-time algorithm,  
but neither could all these other smart people.

# Polynomial-Time Decision Problems



- ◆ To simplify the notion of “hardness,” we will focus on the following:
  - Polynomial-time as the cut-off for efficiency
  - Decision problems: output is 1 or 0 (“yes” or “no”)
    - ◆ Examples:
      - ◆ Does a given graph  $G$  have an Euler tour?
      - ◆ Does a text  $T$  contain a pattern  $P$ ?
      - ◆ Does an instance of 0/1 Knapsack have a solution with benefit at least  $K$ ?
      - ◆ Does a graph  $G$  have an MST with weight at most  $K$ ?

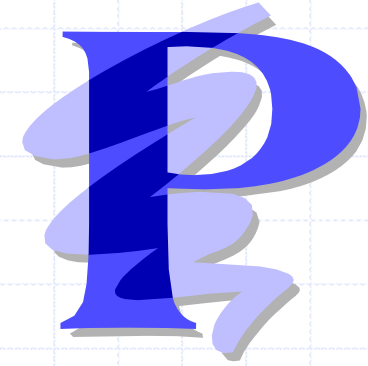


# Problems and Languages



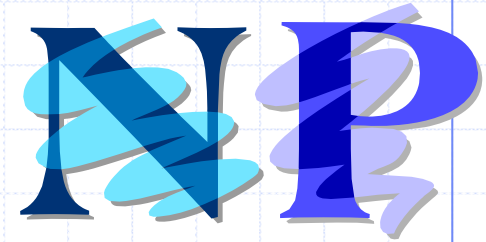
- ◆ A **language**  $L$  is a set of strings defined over some alphabet  $\Sigma$
- ◆ Every decision algorithm  $A$  defines a language  $L$ 
  - $L$  is the set consisting of every string  $x$  such that  $A$  outputs “yes” on input  $x$ .
  - We say “ $A$  **accepts**  $x$ ” in this case
    - ◆ Example:
    - ◆ If  $A$  determines whether or not a given graph  $G$  has an Euler tour, then the language  $L$  for  $A$  is all graphs with Euler tours.

# The Complexity Class P



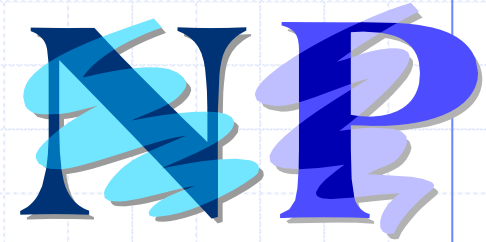
- ◆ A **complexity class** is a collection of languages
- ◆ P is the complexity class consisting of all languages that are accepted by **polynomial-time** algorithms
- ◆ For each language L in P there is a polynomial-time decision algorithm A for L.
  - If  $n=|x|$ , for  $x$  in L, then A runs in  $p(n)$  time on input  $x$ .
  - The function  $p(n)$  is some polynomial

# The Complexity Class NP



- ◆ We say that an algorithm is non-deterministic if it uses the following operation:
  - Choose(b): chooses a bit b
  - Can be used to choose an entire string  $y$  (with  $|y|$  choices)
- ◆ We say that a non-deterministic algorithm  $A$  **accepts** a string  $x$  if there exists some sequence of choose operations that causes  $A$  to output “yes” on input  $x$ .
- ◆ NP is the complexity class consisting of all languages accepted by **polynomial-time non-deterministic** algorithms.

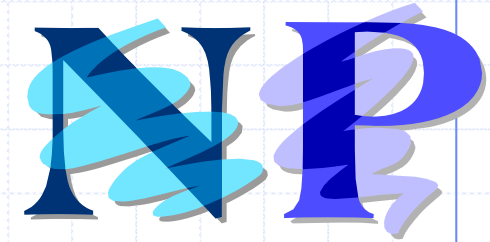
# NP example



- ◆ Problem: Decide if a graph has an MST of weight  $K$
- ◆ Algorithm:
  1. Non-deterministically choose a set  $T$  of  $n-1$  edges
  2. Test that  $T$  forms a spanning tree
  3. Test that  $T$  has weight at most  $K$
- ◆ Analysis: Testing takes  $O(n+m)$  time, so this algorithm runs in polynomial time.

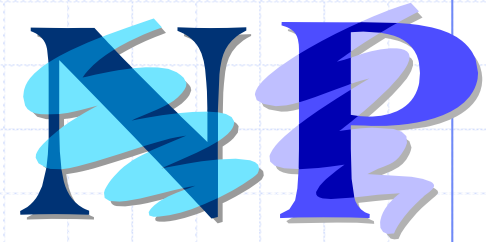
# The Complexity Class NP

## Alternate Definition



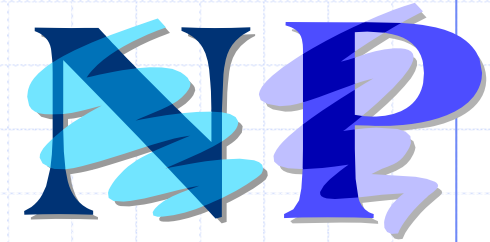
- ◆ We say that an algorithm B **verifies** the acceptance of a language L if and only if, for any  $x$  in L, there exists a certificate  $y$  such that B outputs “yes” on input  $(x,y)$ .
- ◆ NP is the complexity class consisting of all languages verified by **polynomial-time** algorithms.
- ◆ We know: P is a subset of NP.
- ◆ Major open question:  $P=NP$ ?
- ◆ Most researchers believe that P and NP are different.

# NP example (2)



- ◆ Problem: Decide if a graph has an MST of weight  $K$
- ◆ Verification Algorithm:
  1. Use as a certificate,  $y$ , a set  $T$  of  $n-1$  edges
  2. Test that  $T$  forms a spanning tree
  3. Test that  $T$  has weight at most  $K$
- ◆ Analysis: Verification takes  $O(n+m)$  time, so this algorithm runs in polynomial time.

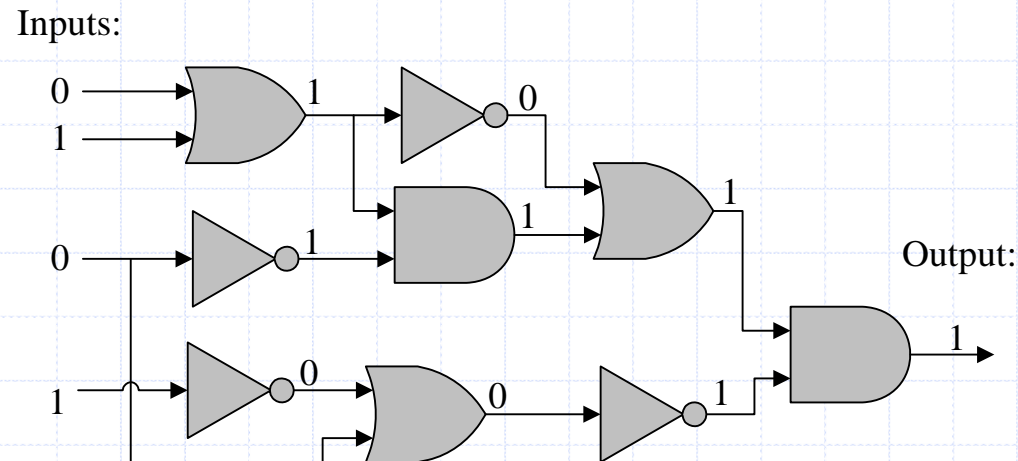
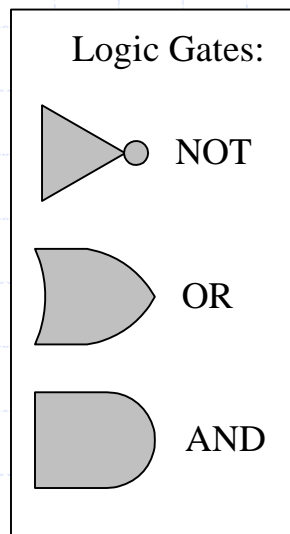
# Equivalence of the Two Definitions



- ◆ Suppose  $A$  is a non-deterministic algorithm
  - ◆ Let  $y$  be a certificate consisting of all the outcomes of the choose steps that  $A$  uses
  - ◆ We can create a verification algorithm that uses  $y$  instead of  $A$ 's choose steps
  - ◆ If  $A$  accepts on  $x$ , then there is a certificate  $y$  that allows us to verify this (namely, the choose steps  $A$  made)
  - ◆ If  $A$  runs in polynomial-time, so does this verification algorithm
- ◆ Suppose  $B$  is a verification algorithm
  - ◆ Non-deterministically choose a certificate  $y$
  - ◆ Run  $B$  on  $y$
  - ◆ If  $B$  runs in polynomial-time, so does this non-deterministic algorithm

# An Interesting Problem

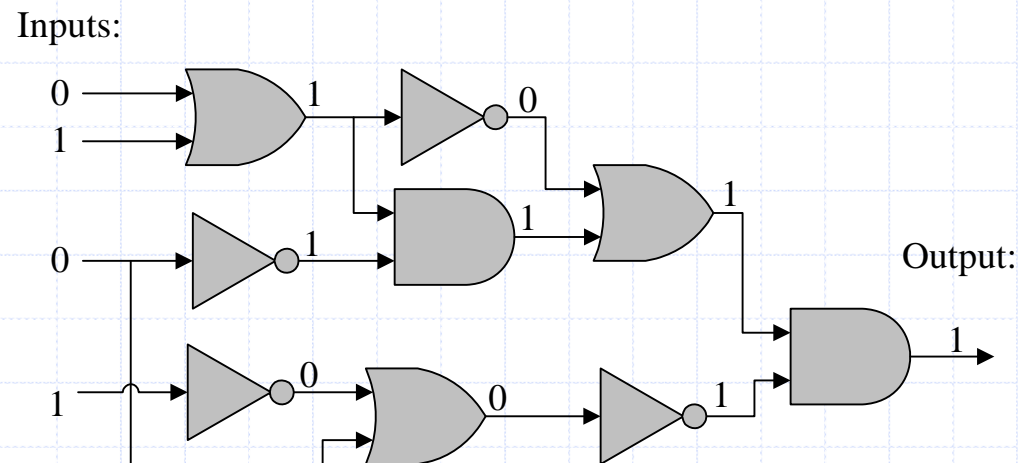
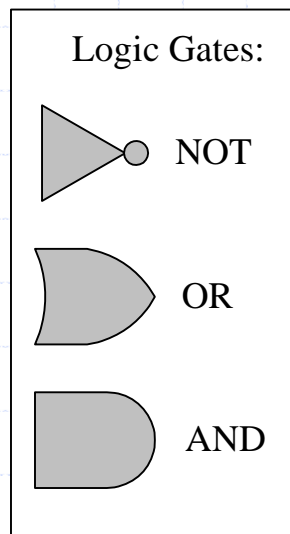
- ◆ A Boolean circuit is a circuit of AND, OR, and NOT gates; the CIRCUIT-SAT problem is to determine if there is an assignment of 0's and 1's to a circuit's inputs so that the circuit outputs 1.





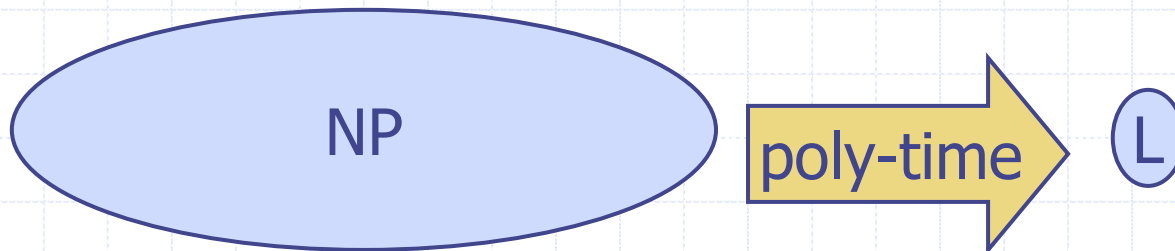
# CIRCUIT-SAT is in NP

- ◆ Non-deterministically choose a set of inputs and the outcome of every gate, then test each gate's I/O.



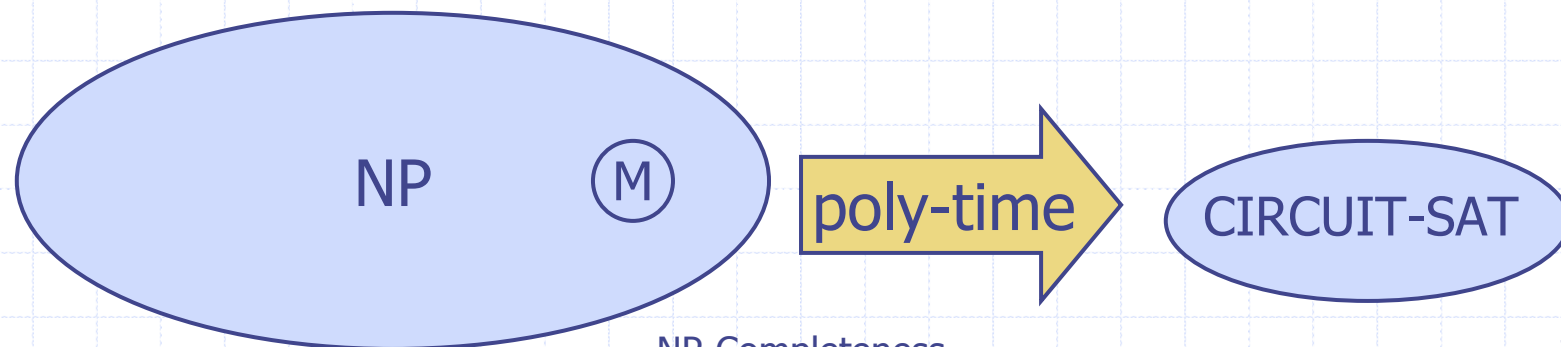
# NP-Completeness

- ◆ A problem (language)  $L$  is **NP-hard** if every problem in NP can be reduced to  $L$  in polynomial time.
- ◆ That is, for each language  $M$  in NP, we can take an input  $x$  for  $M$ , **transform** it in polynomial time to an input  $x'$  for  $L$  such that  $x$  is in  $M$  if and only if  $x'$  is in  $L$ .
- ◆  $L$  is **NP-complete** if it's in NP and is NP-hard.



# Cook-Levin Theorem

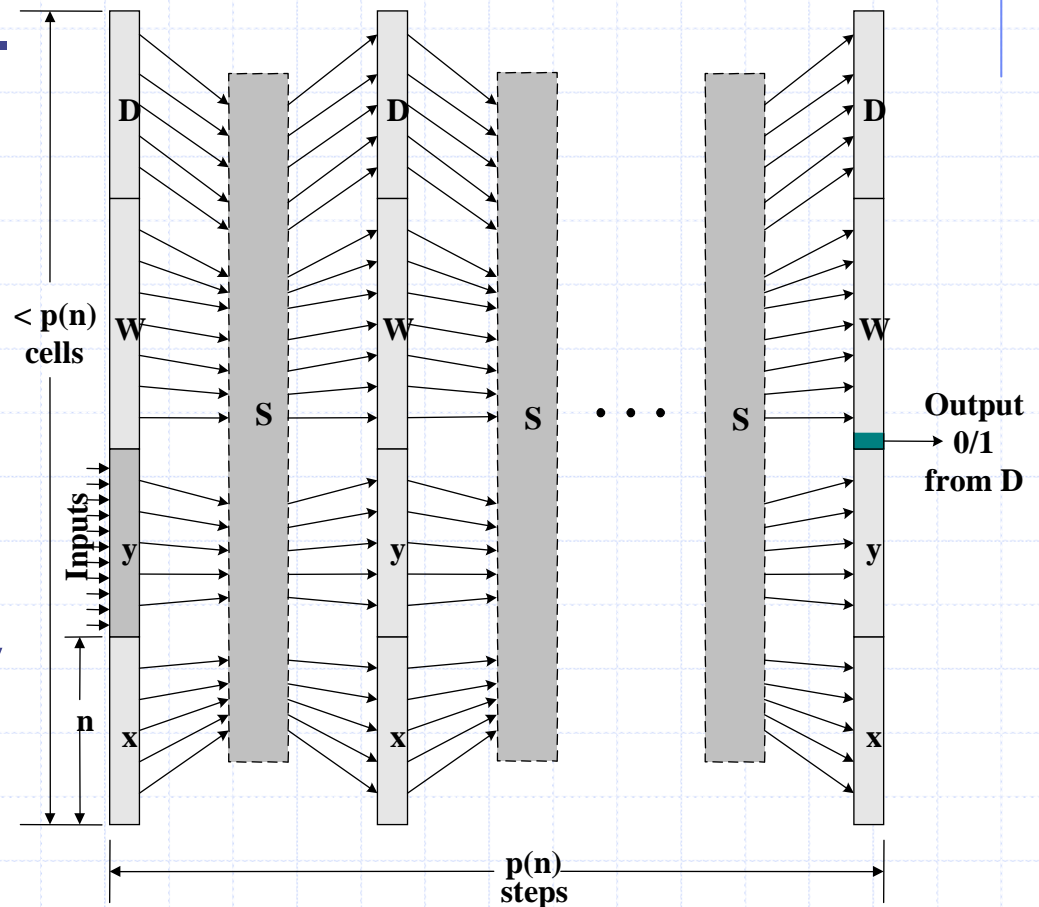
- ◆ CIRCUIIT-SAT is NP-complete.
  - We already showed it is in NP.
- ◆ To prove it is NP-hard, we have to show that every language in NP can be reduced to it.
  - Let  $M$  be in NP, and let  $x$  be an input for  $M$ .
  - Let  $y$  be a certificate that allows us to verify membership in  $M$  in polynomial time,  $p(n)$ , by some algorithm  $D$ .
  - Let  $S$  be a circuit of size at most  $O(p(n)^2)$  that simulates a computer (details omitted...)



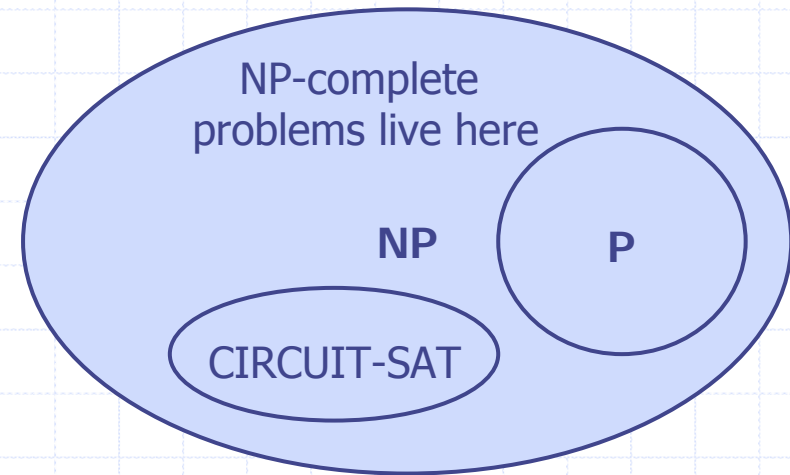
# Cook-Levin Proof

◆ We can build a circuit that simulates the verification of  $x$ 's membership in  $M$  using  $y$ .

- Let  $W$  be the working storage for  $D$  (including registers, such as program counter); let  $D$  be given in RAM "machine code."
- Simulate  $p(n)$  steps of  $D$  by replicating circuit  $S$  for each step of  $D$ . Only input:  $y$ .
- Circuit is satisfiable if and only if  $x$  is accepted by  $D$  with some certificate  $y$
- Total size is still polynomial:  $O(p(n)^3)$ .



# Some Thoughts about P and NP



- ◆ Belief: P is a proper subset of NP.
- ◆ Implication: the NP-complete problems are the hardest in NP.
- ◆ Why: Because if we could solve an NP-complete problem in polynomial time, we could solve every problem in NP in polynomial time.
- ◆ That is, if an NP-complete problem is solvable in polynomial time, then  $P=NP$ .
- ◆ Since so many people have attempted without success to find polynomial-time solutions to NP-complete problems, showing your problem is NP-complete is equivalent to showing that a lot of smart people have worked on your problem and found no polynomial-time algorithm.

# Τέλος Ενότητας



Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

# Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Σημειώματα



# Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] <http://creativecommons.org/licenses/by-nc-nd/4.0/>

- Ως Μη Εμπορική ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο
- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

# Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Ιωάννης Τόλλης 2015. «Αλγόριθμοι και πολυπλοκότητα. NP-Completeness». Έκδοση: 1.0. Ηράκλειο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:

<https://opencourses.uoc.gr/courses/course/view.php?id=368>

# Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.