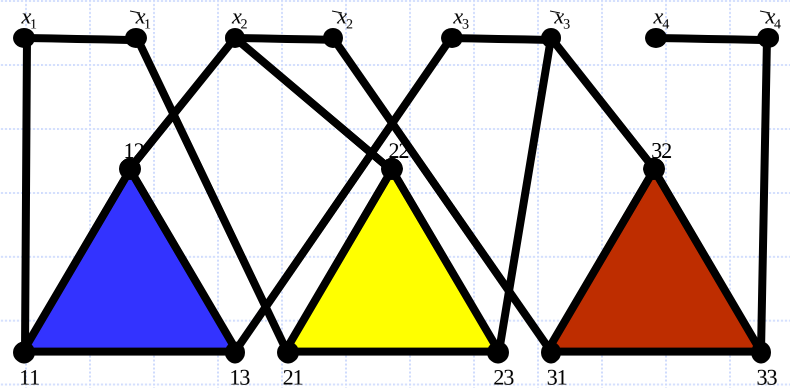
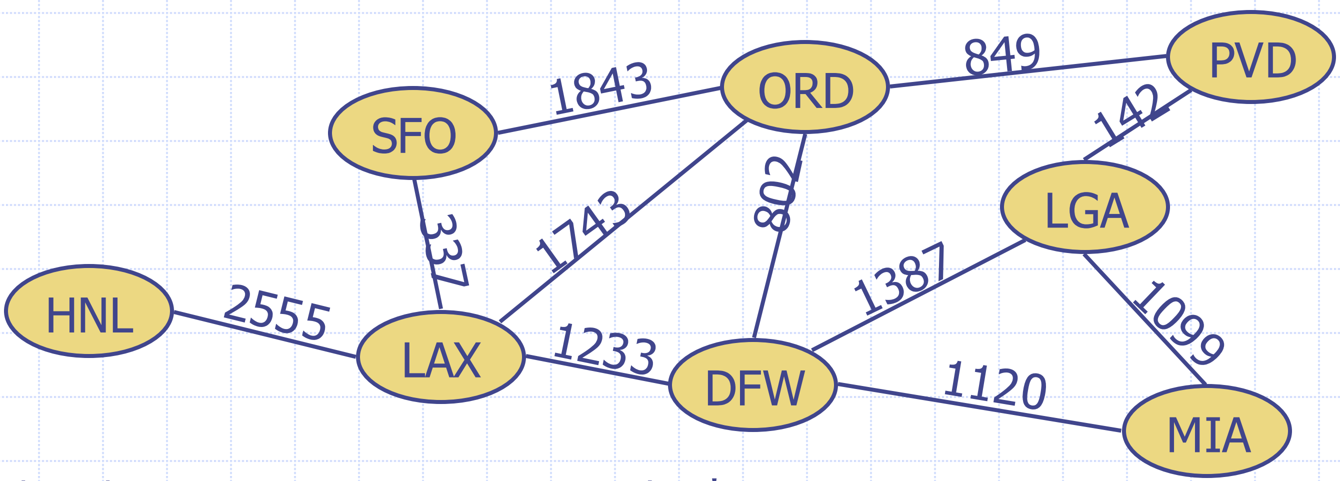
Παρουσίαση για χρήση με το σύγγραμμα, Αλγόριθμοι Σχεδίαση και Εφαρμογές, των M. T. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

NP-Πληρότητα



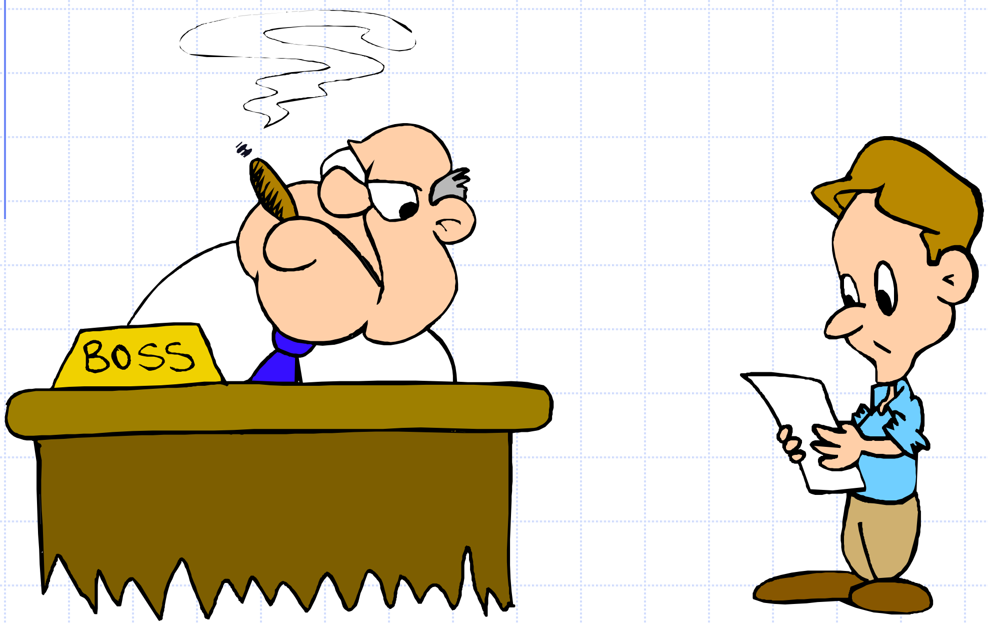
Επανεξέταση του χρόνου εκτέλεσης

- Μέγεθος εισόδου, ***n***
  - Για την ακρίβεια, έστω ***n*** ο αριθμός των **bits** μη μοναδιαίας δυαδικής κωδικοποίησης της εισόδου
- Όλοι οι αλγόριθμοι πολυωνυμικού χρόνου που έχουν μελετηθεί μέχρι στιγμής είναι πολυωνυμικού χρόνου βάση αυτής της διατύπωσης του μεγέθους εισόδου.
  - Εξαίρεση: κάθε αλγόριθμος ψεύδο-πολυωνυμικού χρόνου



Αντιμετωπίζοντας δύσκολα προβλήματα

- Τι κάνουμε όταν βρίσκουμε ένα πρόβλημα που φαίνεται δύσκολο...

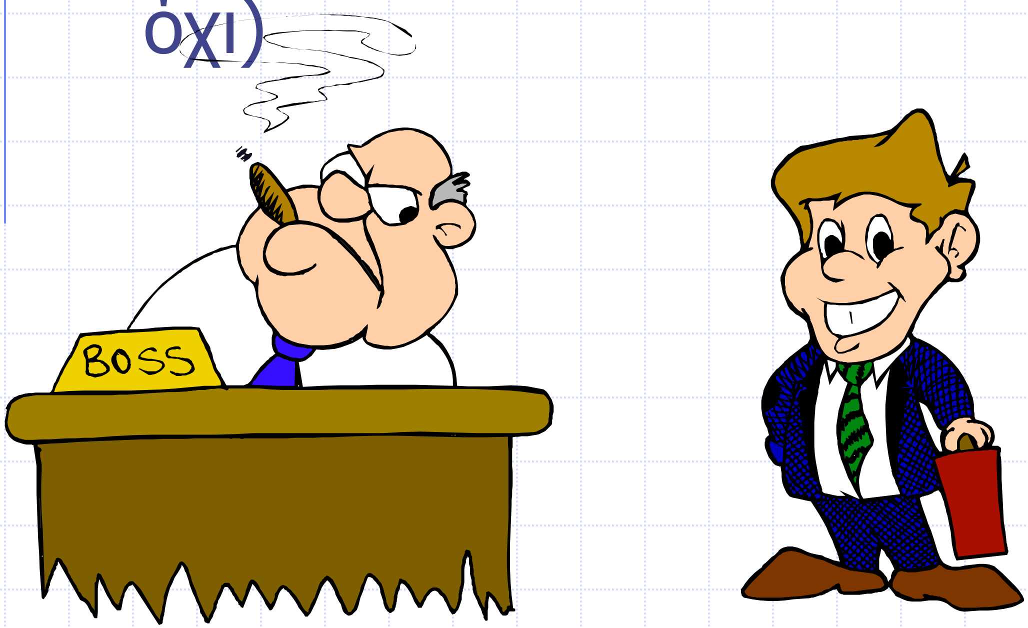


I couldn’t find a polynomial-time algorithm;
I guess I’m too dumb.

(cartoon inspired by [Garey-Johnson, 79])

Αντιμετωπίζοντας δύσκολα προβλήματα

- Κάποιες φορές μπορούμε να αποδείξουμε ένα αυστηρό κατώτατο όριο… (συνήθως όμως όχι)



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

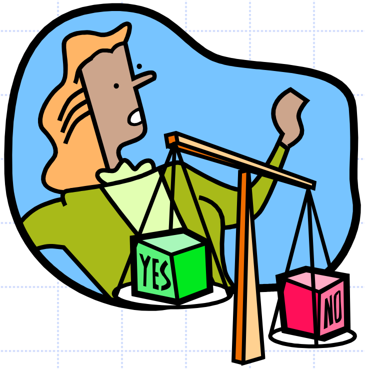
(cartoon inspired by [Garey-Johnson, 79])

Αντιμετωπίζοντας δύσκολα προβλήματα

- Η NP-Πληρότητα μας επιτρέπει να δείξουμε «συλλογικά» ότι ένα πρόβλημα είναι δύσκολο.

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

Προβλήματα απόφασης Πολυωνυμικού χρόνου



- Για να απλοποιήσουμε την έννοια της "δυσκολίας," θα επικεντρωθούμε στα ακόλουθα:
  - Ο πολυωνυμικός χρόνος ως το όριο για την αποδοτικότητα
  - Προβλήματα απόφασης: η έξοδος είναι 1 ή 0 ("ναι" ή " 'όχι")
    - Παραδείγματα:
    - Διαθέτει ο γράφος G διάσχιση Euler?
    - Περιέχει το κείμενο T το μοτίβο P?
    - Υπάρχει λύση για μία περίσταση 0/1 Knapsack με όφελος τουλάχιστον K?
    - Διαθέτει ο γράφος G Δέντρο επικάλυψης ελάχιστου κόστους (MST) με βάρος το πολύ K?

Προβλήματα και γλώσσες

- Μία **γλώσσα** L είναι ένα σύνολο συμβολοσειρών για κάποιο αλφάβητο Σ
- Κάθε αλγόριθμος απόφασης A ορίζει την γλώσσα L
  - το L είναι το σύνολο που αποτελείτε από κάθε συμβολοσειρά x έτσι ώστε ο A δίνει έξοδο "ναι" για είσοδο x.
  - Λέμε ότι "ο A **δέχεται το** x'' σε αυτήν την περίπτωση
    - Παράδειγμα:
    - Εάν ο A ορίζει ένα ένας γράφος G έχει μία διαδρομή Euler, τότε η γλώσσα L για τον A είναι όλοι οι γράφοι με διαδρομές Euler.

Η κλάση πολυπλοκότητας P

- Μία **κλάση πολυπλοκότητας** είναι μία συλλογή γλωσσών
- P είναι η κλάση πολυπλοκότητας που περιλαμβάνει όλες τις γλώσσες που γίνονται αποδεκτές από αλγόριθμους **πολυωνυμικού χρόνου**
- Για κάθε γλώσσα L στη P υπάρχει ένας αλγόριθμος απόφασης πολυωνυμικού χρόνου A για την L.
  - Εάν $n=|x|$, για x στο L, τότε ο A είναι χρόνου $p(n)$ για είσοδο x.
  - Η συνάρτηση $p(n)$ είναι πολυωνυμική

Η κλάση πολυπλοκότητας NP

- Λέμε ότι ένας αλγόριθμος είναι μη ντετερμινιστικός εάν χρησιμοποιεί την ακόλουθη λειτουργία:
  - Choose(b): επιλογή ενός bit b
  - Μπορεί να χρησιμοποιηθεί ολόκληρη την συμβολοσειρά y (με |y| επιλογές)
- Λέμε ότι ένας μη ντετερμινιστικός αλγόριθμος A **δέχεται** μία συμβολοσειρά x εάν υπάρχει κάποια ακολουθία λειτουργιών choose που κάνουν τον A να έχει έξοδο "ναι" για είσοδο x.
- NP είναι η κλάση πολυπλοκότητας που περιλαμβάνει όλες τις γλώσσες που γίνονται αποδεκτές από αλγόριθμους **μη ντετερμινιστικούς πολυωνυμικού χρόνου**

NP παράδειγμα

- Πρόβλημα: Έχει κάποιος γράφος MST βάρους K;
- Αλγόριθμος:
  1. Μη ντετερμινιστική επιλογή ενός συνόλου T n-1 ακμών
  2. Έλεγχος εάν το T δημιουργεί δέντρο επικάλυψης
  3. Έλεγχος ότι το T έχει βάρος το πολύ K
- Ανάλυση: Ο έλεγχος είναι χρόνου O(n+m), οπότε αυτός ο αλγόριθμος είναι πολυωνυμικού χρόνου.

Η κλάση πολυπλοκότητας NP Εναλλακτικός ορισμός

- Λέμε ότι ο αλγόριθμος B **επικυρώνει** μία γλώσσα L μόνο και μόνο αν για κάθε x στο L, υπάρχει ένα πιστοποιητικό y έτσι ώστε το B έχει έξοδο “ναι” για την είσοδο (x,y).
- NP είναι η κλάση πολυπλοκότητας που περιλαμβάνει όλες τις γλώσσες που επικυρώνονται από αλγόριθμους **πολυωνυμικού χρόνου**
- Γνωρίζουμε ότι: το P είναι υποσύνολο του NP.
- Ανοιχτή ερώτηση: P=NP?
- Οι περισσότεροι ερευνητές πιστεύουν ότι το P και το NP είναι διαφορετικά.

NP παράδειγμα (2)

- Πρόβλημα: Έχει κάποιος γράφος MST βάρους K;

- Αλγόριθμος επικύρωσης:
  1. Χρήση ενός πιστοποιητικού, y, ενός συνόλου T n-1 ακμών
  2. Έλεγχος εάν το T δημιουργεί δέντρο επικάλυψης
  3. Έλεγχος ότι το T έχει βάρος το πολύ K

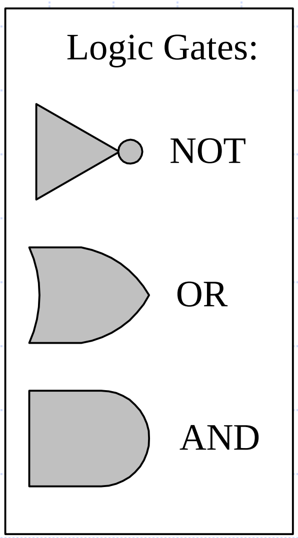
- Ανάλυση: Η επικύρωση είναι χρόνου O(n+m), οπότε αυτός ο αλγόριθμος είναι πολυωνυμικού χρόνου.

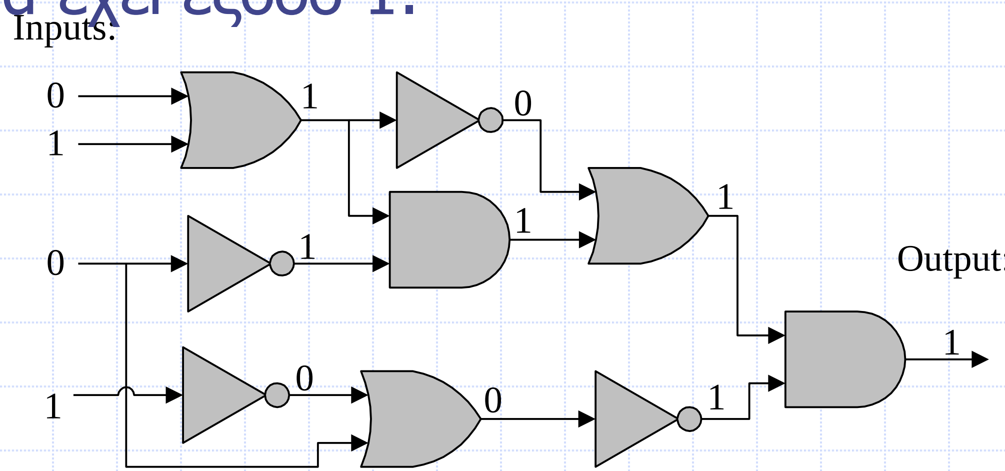
Ισοδυναμία των δύο ορισμών

- Υποθέστε ότι ο A είναι ένας μη ντετερμινιστικός αλγόριθμος
  - Έστω y το πιστοποιητικό που περιλαμβάνει όλα τα ενδεχόμενα για τα βήματα επιλογής που χρησιμοποιεί ο A
  - Μπορούμε να δημιουργήσουμε έναν αλγόριθμο επικύρωσης που χρησιμοποιεί το y αντί για τα βήματα επιλογής του A
  - Εάν ο A δεχτεί το x, τότε υπάρχει πιστοποιητικό y που μας επιτρέπει να το επικυρώσουμε (ονομαστικά, τα βήματα επιλογής που έκανε ο A)
  - Εάν ο A είναι πολυωνυμικού χρόνου το ίδιο είναι και ο αλγόριθμος επικύρωσης
- Υποθέστε ότι ο B είναι ένας αλγόριθμος επικύρωσης
  - Μη ντετερμινιστική επιλογή ενός πιστοποιητικού y
  - Εκτέλεση του B στο y
  - Εάν ο B είναι πολυωνυμικού χρόνου το ίδιο είναι και ο μη ντετερμινιστικός αλγόριθμος

Ένα ενδιαφέρον πρόβλημα

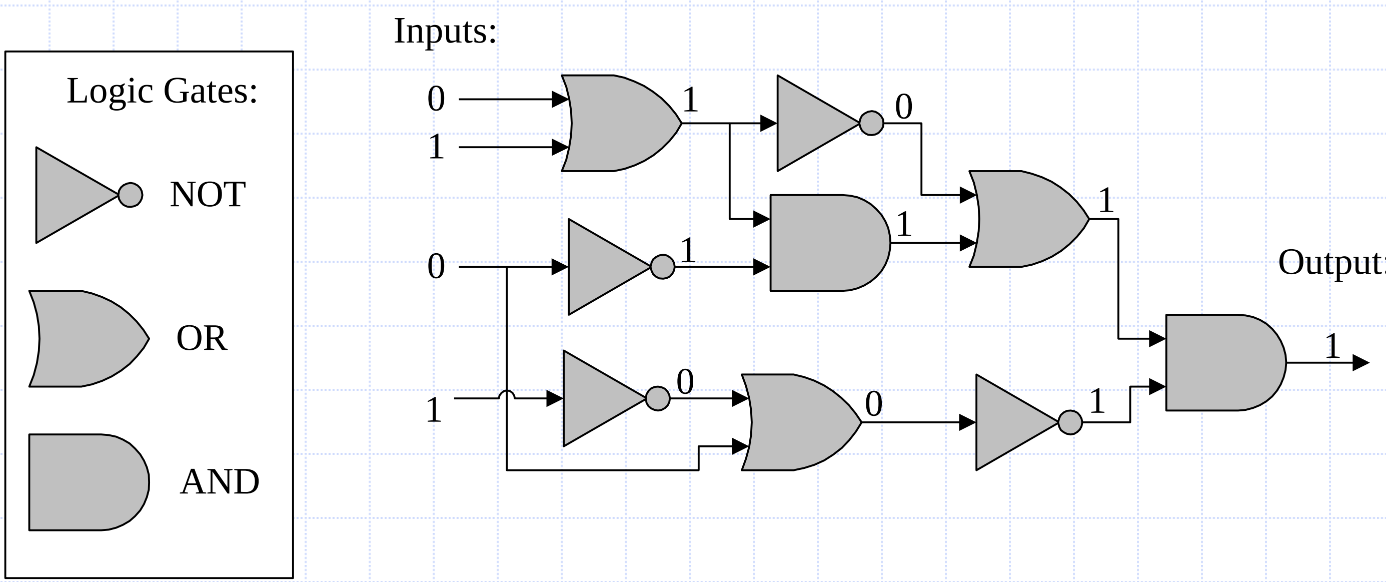
- Ένα Boolean κύκλωμα είναι ένα κύκλωμα από πύλες AND, OR και NOT; το πρόβλημα CIRCUIT-SAT αφορά στο να αποφασιστεί εάν υπάρχει κάποια εκχώρηση από 0 και 1 στις εισόδους του κυκλώματος έτσι ώστε το κύκλωμα να έχει έξοδο 1.





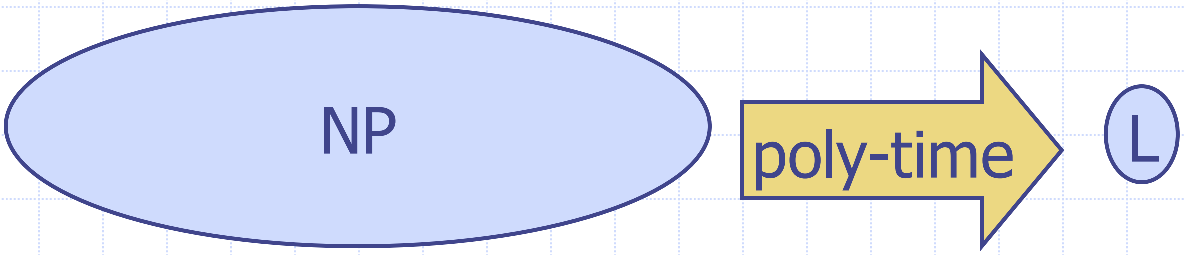
Το CIRCUIT-SAT είναι NP

- Μη ντετερμινιστική επιλογή ενός συνόλου εισόδων και τα εξόδου κάθε πύλης, έπειτα έλεγχος κάθε πύλης I/O.



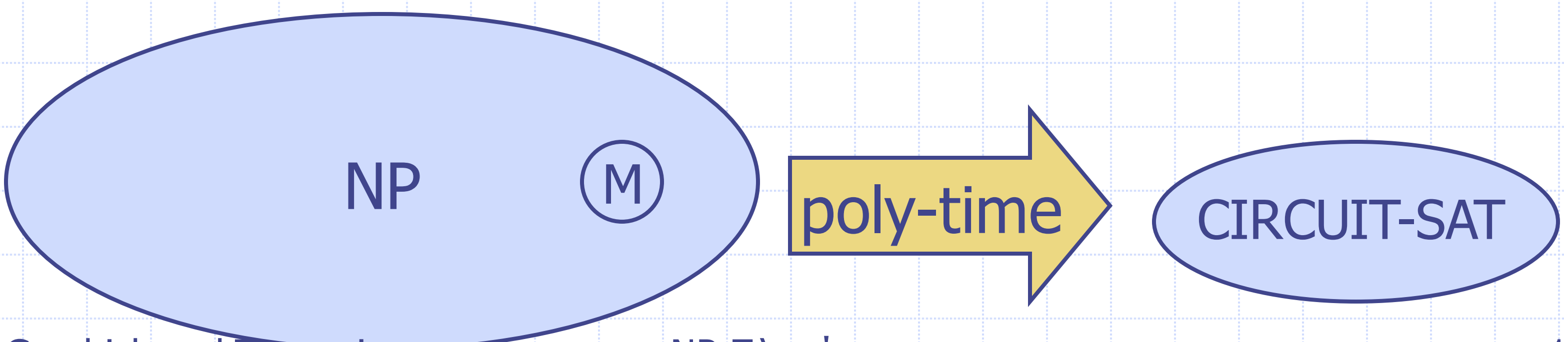
NP-Πληρότητα

- Ένα πρόβλημα (γλώσσα) L είναι **NP-hard** εάν κάθε πρόβλημα στο NP μπορεί να ελαχιστοποιηθεί στο L σε πολυωνυμικό χρόνο.
- Δηλαδή, για κάθε γλώσσα M στο NP, μπορούμε να πάρουμε μία είσοδο x για το M, **μετασχηματισμός** αυτού σε πολυωνυμικό χρόνο σε μία είσοδο x' για το L έτσι ώστε το x είναι στο M εάν και μόνο εάν το x' είναι στο L.
- Το L είναι **NP-πλήρες** εάν είναι NP και NP-hard.



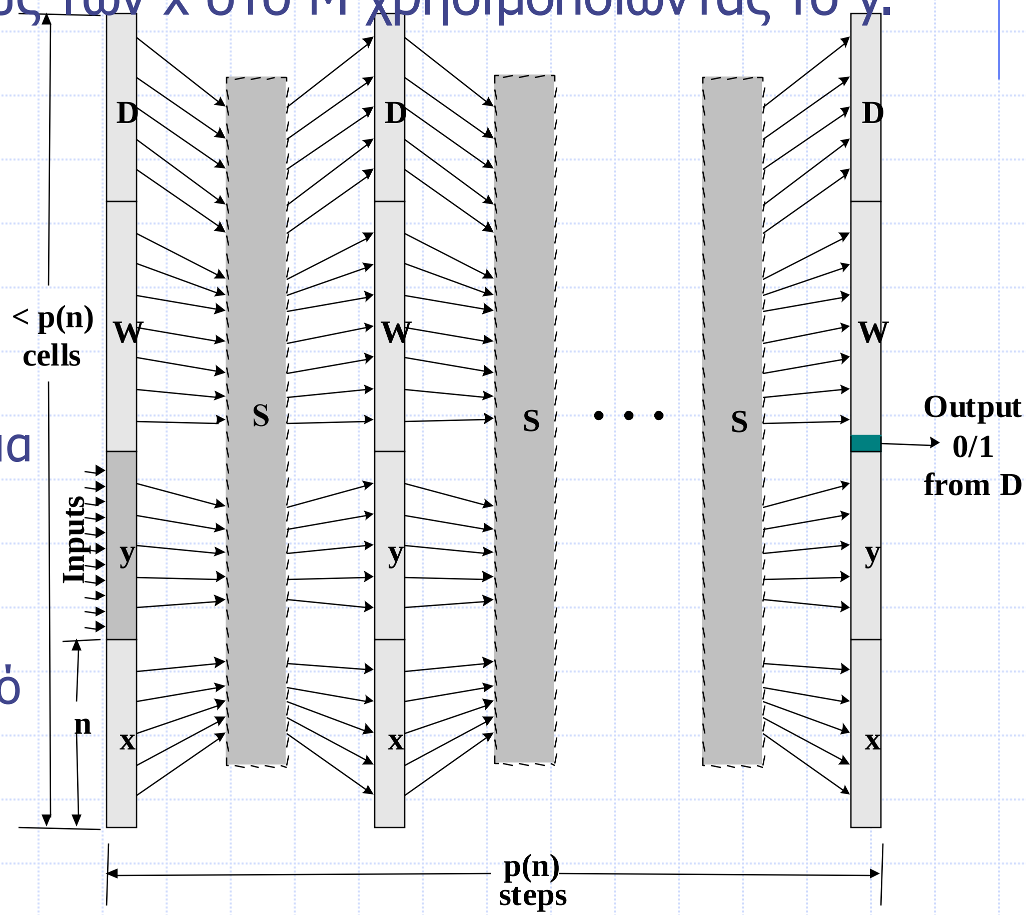
Το θεώρημα Cook-Levin

- Το CIRCUIT-SAT είναι NP-πλήρες.
  - Το αποδείξαμε στο NP.
- Για να αποδείξουμε ότι είναι NP-hard, πρέπει να αποδείξουμε ότι κάθε γλώσσα στο NP μπορεί να ελαχιστοποιηθεί σε αυτό.
  - Έστω ότι το M είναι NP, και έστω ότι το x είναι μία είσοδος για το M.
  - Έστω y ένα πιστοποιητικό που μας επιτρέπει να επικυρώσουμε το M σε πολυωνυμικό χρόνο, p(n), από κάποιον αλγόριθμο D.
  - Έστω S κύκλωμα μεγέθους το πολύ $O(p(n)^2)$ που εξομοιώνει έναν υπολογιστή (παραλείπονται οι λεπτομέρειες…)

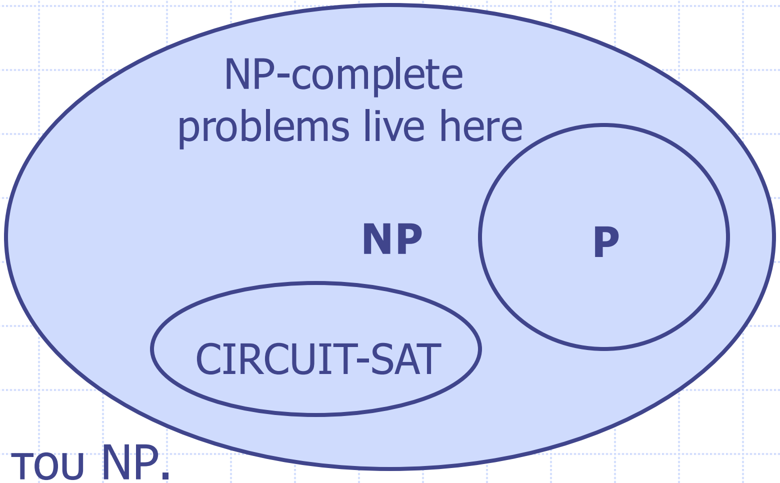


Cook-Levin Απόδειξη

- Μπορούμε να δημιουργήσουμε ένα κύκλωμα που εξομοιώνει την επικύρωση της ιδιότητας μέλους των x στο M χρησιμοποιώντας το y.
  - Έστω W ο χώρος εργασίας του D (περιλαμβάνοντας καταχωρητές, όπως ο απαριθμητής προγράμματος); Έστω ότι το D δίνεται σε “μηχανή κώδικα.”
  - Εξομοίωση p(n) βημάτων του D αναπαριστώντας το κύκλωμα S για κάθε βήμα του D. Μοναδική είσοδος: y.
  - Το κύκλωμα ικανοποιείτε εάν και μόνο αν το x γίνεται αποδεκτό από το D με κάποιο πιστοποιητικό y
  - Το συνολικό μέγεθος παραμένει πουλωνυμικό: $O(p(n)^3)$.



Κάποιες σκέψεις για P και NP



- Υπόθεση: Το P είναι γνήσιο υποσύνολο του NP.
- Επίπτωση : τα NP-πλήρη προβλήματα είναι τα δυσκολότερα στο NP.
- Γιατί: Επειδή ένα μπορούσαμε να λύσουμε κάποιο NP-πλήρες πρόβλημα σε πολυωνυμικό χρόνο, θα μπορούσαμε να λύσουμε κάθε πρόβλημα στο NP σε πολυωνυμικό χρόνο.
- Οπότε, εάν κάποιο NP-πλήρες πρόβλημα λύνεται σε πολυωνυμικό χρόνο, τότε P=NP.
- Από την στιγμή που τόσοι άνθρωποι έχουν προσπαθήσει χωρίς επιτυχία να βρουν λύσεις πολυωνυμικού χρόνου σε NP-πλήρη προβλήματα, δείχνοντας ΄τοι το πρόβλημα σου είναι NP-πλήρες ισοδυναμεί με το να αποδείκνυες ότι πολλοί έξυπνοι άνθρωποι έχουν εργαστεί πάνω στο πρόβλημα σου και δεν κατάφεραν να βρουν λύση πολυωνυμικού χρόνου.