Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

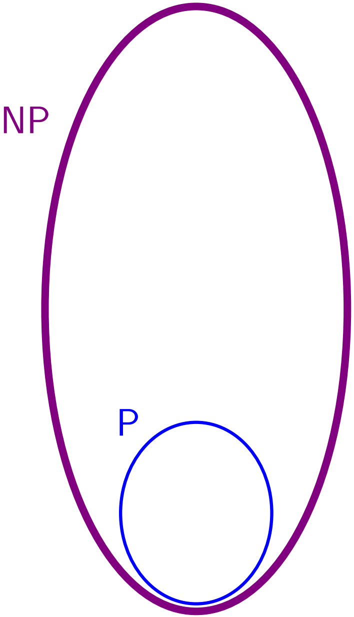
Lecture 27
NP-completeness

slides by Graham Farr

Overview

- Definition of NP-completeness
- Basic properties
- Existence of an NP-complete language
- Statement of the Cook-Levin Theorem

If $P \neq NP$:



In NP, not known to be in P:

SATISFIABILITY, 3-SAT,
HAMILTONIAN CIRCUIT,
3-COLOURABILITY,
VERTEX COVER,
INDEPENDENT SET, . . .

GRAPH ISOMORPHISM,

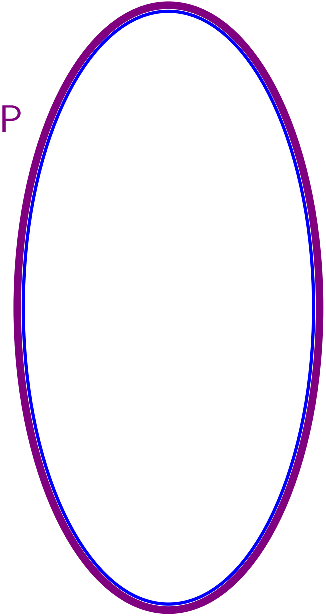
INTEGER FACTORISATION, . . .

In P:

2-SAT,
EULERIAN CIRCUIT,
2-COLOURABILITY,
CONNECTED GRAPHS,
SHORTEST PATH,
PRIMES,
Invertible matrices,
. . . ,
All Context-Free Languages,
All Regular Languages.

If P = NP:

If P = NP



SATISFIABILITY, 3-SAT,
HAMILTONIAN CIRCUIT,
3-COLOURABILITY,
VERTEX COVER,
INDEPENDENT SET, ...

GRAPH ISOMORPHISM,

INTEGER FACTORISATION, ...

2-SAT,
EULERIAN CIRCUIT,
2-COLOURABILITY,
CONNECTED GRAPHS,
SHORTEST PATH,
PRIMES,
Invertible matrices,
...,
All Context-Free Languages,
All Regular Languages.

P and NP

Languages in P are “easy” . . . or, at least, they have “efficient” (polynomial time) deciders.

Languages outside P are “hard”.

They don’t have “efficient” (polynomial time) deciders.

Languages in NP: membership is “easy” (polynomial time) to verify.

NP contains many languages of great practical importance.

What are the “hardest” languages in NP ?

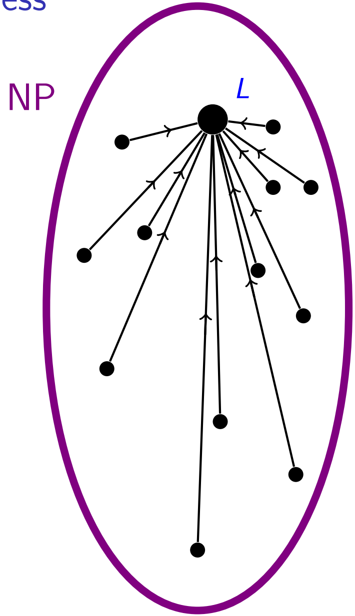
Use polynomial-time reduction . . .

NP-completeness

Definition
A language L is **NP-complete** if

(a) L is in NP, and
(b) every language in NP is polynomial-time reducible to L.
i.e.,
$$\forall K \in \mathrm{NP} : \; K \leq_P L.$$

NP-completeness



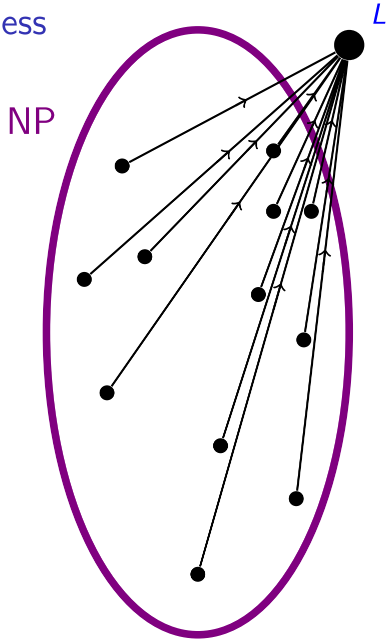
L is **NP-complete** because ...

L is in NP,

and

everything in NP ...
is polynomial-time reducible to L

NP-completeness

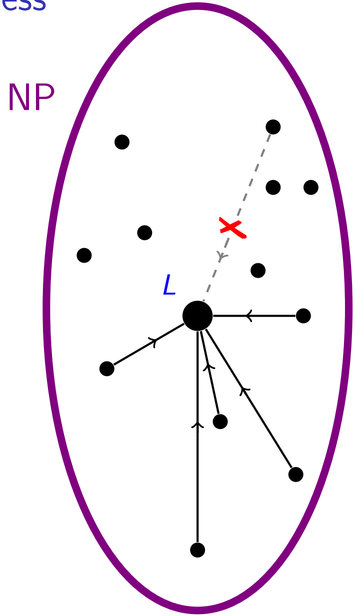


L is <u>not</u> NP-complete because ...

L is <u>not</u> in NP.

It doesn't matter if
everything in NP
is polynomial-time reducible to L

NP-completeness



L is not NP-complete because ...

not everything in NP
is polynomial-time reducible to L

NP-completeness

Theorem.
Let L be any NP-complete language.
There is a polynomial-time decider for L if and only if $\text{P} = \text{NP}$.

Proof.

$(\Longleftarrow)$

If $\text{P} = \text{NP}$, then every language in NP has a polynomial-time decider.

Since L is NP-complete, it must be in NP
(using the first part of the definition of NP-completeness).

Therefore L has a polynomial-time decider.

$(\Longrightarrow)$

We know P $\subseteq$ NP.
It remains to show that, under our assumptions, NP $\subseteq$ P.
Then we'll know that P $=$ NP.

Let K be any language in NP. We will show it is also in P.

Since L is NP-complete, any language in NP is polynomial-time reducible to L.

Therefore $K \leq_P L$.

But we know that, if $K \leq_P L$ and L is in P, then K is in P too. (See Lecture 26.)

So K is in P.

We have shown that NP $\subseteq$ P, which completes the proof. $\square$

Exercises

Prove:

Theorem.
Let L be any NP-complete language.
For every language K,
K is in NP if and only if $K \leq_P L$.

Theorem.
Let L be any NP-complete language.
For every language K,
K is in NP-complete if and only if $K \leq_P L$ and $L \leq_P K$.

Cook-Levin Theorem

Our first NP-complete language:

$$\text{SATISFIABILITY} \quad := \quad \{\, \text{satisfiable Boolean expressions in CNF} \,\}$$

Cook-Levin Theorem
SATISFIABILITY is NP-complete.

History: S. Cook (1971), L. Levin (1972)

Cook-Levin Theorem

To prove the Cook-Levin Theorem, we must show:

(a) SATISFIABILITY is in NP

- the easy part
- Given Boolean expression φ in CNF:
- Certificate $=$ truth assignnment to variables of φ.
- Verification: check that each clause is satisfied ...
- Prove verification works and takes polynomial time.

(b) For every L in NP, $L \leq_P$ SATISFIABILITY.

- much harder.

Reduction to SATISFIABILITY

Let's begin by looking at:

PARTITION INTO TRIANGLES:
the set of graphs G such that the vertex set of G can be partitioned into 3-sets (i.e., sets of size 3) so that each of these 3-sets induces a triangle in G.

How to do a polynomial-time reduction from PARTITION INTO TRIANGLES to SATISFIABILITY?

Revision

Things to think about:

- If P = NP, which languages would be NP-complete?
- How to do a polynomial-time reduction from PARTITION INTO TRIANGLES to SATISFIABILITY?

Reading:

- Sipser, section 7.4, pp. 299–304.
- M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., San Francisco, 1979: §2.5.