Reduction

$$\Pi_1 \leq \Pi_2 \quad \text{and} \quad \Pi_2 \leq \Pi_1$$

$$\Rightarrow \quad \Pi_1 \equiv \Pi_2$$

$$\begin{array}{ccc} \text{HAM-CYCLE} & \equiv & \text{HAM-PATH} \\ ||| & & ||| \\ \text{DHAM-CYCLE} & \equiv & \text{DHAM-PATH} \end{array}$$

HAM-CYCLE $\leq$ DHAM-CYCLE

$$G \longmapsto G'$$

G has an undirected Hamiltonian cycle

$\Leftrightarrow$ G' has a directed Hamiltonian cycle

$$V' = V$$

for $(u, v) \in E$, add two directed edges (u, v) and (v, u) to E'.

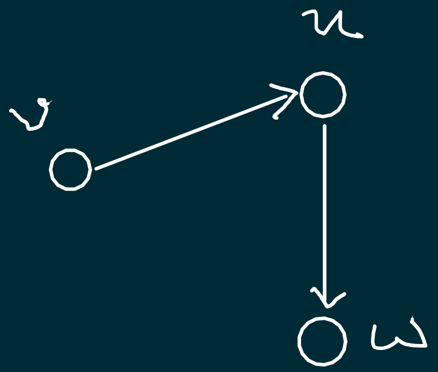
DHAM-CYCLE $\leq$ HAM-CYCLE

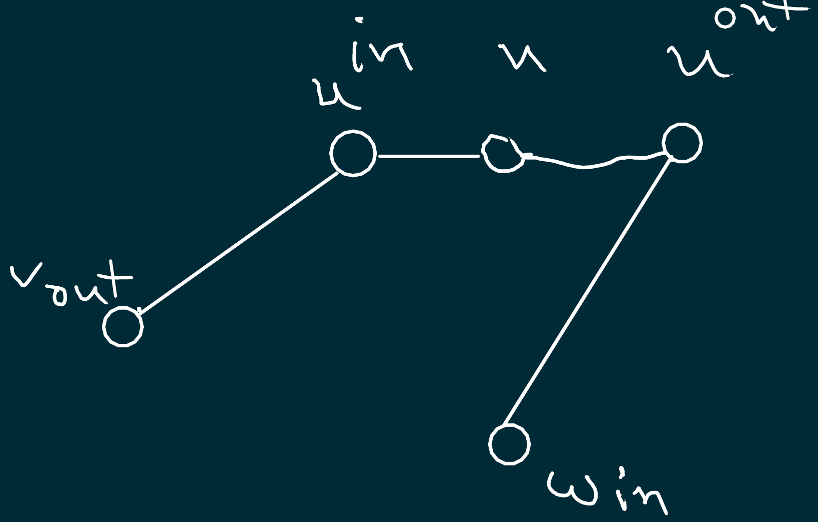
$$G \longmapsto G'$$

G has a directed Ham cycle $\iff$

G' has an undirected Ham cycle

$$u \in V \qquad u, u^{in}, u^{out}$$





Exercises

- correctness of the construction
- Construction does not work if u is not there.

Definition: A problem Π is called NP-hard if there is a poly-time reduction from every problem $\Pi' \in NP$ to Π.

$$\Pi' \leq \Pi$$

Definition: A problem Π is called NP-complete if

(1) $\Pi \in NP$ ← not automatic

(2) Π is NP-hard.

(An NP-complete problem is one of the most difficult problems in NP)

Theorem: If any NP-complete problem can be solved in poly time, then P = NP.

If some NP-complete can be proved to be unsolvable in poly time, then $P \neq NP$. → trivial

Proof: $P \subseteq NP$. Let $\Pi \in NP$ be NPC and let A be a poly-time algo for Π.

$\Pi' \in NP$

$$\Pi' \xrightarrow{R} \Pi \xrightarrow{A} \text{Solve}$$

poly-time

$\Pi' \in P$

Are there NP-Complete problems?

YES. There are <u>thousands</u> of known NP-Complete problems.

How to prove NP-Completeness?

- <u>Generic reduction</u> (NTM, poly-time)

- Problem-to-problem reduction

Theorem: $\Pi_1 \leq \Pi_2$ and $\Pi_2 \leq \Pi_3$, then $\Pi_1 \leq \Pi_3$.

Theorem: If $\Pi_1 \leq \Pi_2$ and Π_1 is NP-Complete, then Π_2 is also NP-complete.

Cook, Levin 1971

SAT, CNFSAT

HAMPATH

VERTEX COVER

CLIQUE

TSP

LONGEST PATH

INDEP_SET

SAT – Satisfiability of Boolean formulas

$x_1, x_2, \ldots, x_n$ Boolean variables

$\phi(x_1, x_2, \ldots, x_n)$ evaluates to True/False for each truth assignment of $x_1, x_2, \ldots, x_n$

AND	$\wedge$	products	conjunction
OR	$\vee$	$+$	disjunction
NOT	$\neg$	bar	negation

CNF – conjunctive normal form

product-of-sum expression

$$(x_1 + \bar{x}_2 + x_4)\, x_3\, (x_2 + \bar{x}_3 + x_5)\, \bar{x}_4\, x_6$$

$$\vee \qquad \vee \qquad \wedge \qquad \wedge \qquad \vee \qquad \vee \qquad \wedge \qquad \wedge$$

variables : $x_1, x_2, \ldots, x_n$

literal : $x_i, \bar{x}_i$

clause : An OR of literals

Boolean formula in CNF : And of clauses

ϕ be a CNF formula

If every clause contains exactly k literals, ϕ is said to be in the k-CNF. (k constant)

$$(x_1 + \bar{x}_2 + x_4)(\bar{x}_1 + x_3 + x_5)(x_6 + \bar{x}_7 + \bar{x}_8)$$

3-CNF formula

A Boolean formula ϕ is called satisfiable if it evaluates to True for at least one truth assignment of the variables.

unsatisfiable if ϕ evaluates to False for all truth assignments of the variables.

$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_4)$

$x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 1$

satifiable

$(x_1 \vee \bar{x}_2) \wedge \bar{x}_1 \wedge x_2$

not satisfiable

SAT : Given a Boolean formula, decide whether the formula is satisfiable.

CNFSAT : Give a Boolean formula in the CNF, decide whether it is satisfiable.

k-CNFSAT : Given a Boolean formula in the k-CNF, decide whether it is satisfiable.

If satisfiable, a satisfying truth assignment is a certificate.

Cook-Levin Theorem:

SAT is NP-hard.

Corollary:

CNFSAT is NP-hard.

Michael Sipser, Theory of Computation