

CS 341: Algorithms

Lecture 21: NP-completeness

Slides due to Éric Schost and based on lecture notes by many other CS341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2026

P and NP

Definition

P is the set of decision problems that can be solved in polynomial time

$$\text{PROB} \in \mathbf{P} := \exists A \quad \forall x \quad [A(x) = \text{“yes”} \iff x \in \text{PROB}]$$

poly time input

Cobham-Edmonds thesis

P is the class of *tractable* (efficiently solvable) problems

P and NP

Definition

NP is the set of decision problems where **yes**-instances can be **certified** in poly time.

$$\text{PROB} \in \mathbf{NP} := \exists B_{\text{poly time input}} \forall x \left[\left(\exists y_{\text{cert.}} B(x, y) = \text{"yes"} \right) \iff x \in \text{PROB} \right]$$

The rock





$NP \neq NP \cap coNP = P$

co-NP

Definition

co-NP is the set of decision problems whose **no**-instances can be certified in polynomial time.

Remark: most problems so far are thought to not be in **co-NP**

- certify that a formula not satisfiable?
- certify that a graph has no Hamiltonian path?
- but HASFACTOR is in **co-NP**
(certificate that M has no factor $\leq k$? all prime factors)

Exercise (after we see NP-completeness)

If a single NP-complete problem is in **co-NP**, **NP=co-NP**
(so doubtful that HASFACTOR is NP-complete)

NP \cap co-NP

These are the problems where we can certify **both** yes and no instances efficiently.

MaxFlowDecision:

- **input:** integer-weighted graph G , source s , sink t , K
- **output:** is there a flow of value **at least** K ?

MinCutDecision:

- **input:** integer-weighted graph G , source s , sink t , K
- **output:** is there a cut of capacity **at most** K ?

Claim: max flow = min cut \implies both problems in **NP \cap co-NP**

- MAXFLOWDECISION is **NP**

certificate that there is flow of value at least K : a flow of value at least K

- MAXFLOWDECISION is **co-NP**

certificate that there is no flow of value at least K : a cut of capacity at most $K - 1$

$\text{NP} \cap \text{co-NP} = \text{P}?$

Flow and cuts

- in P! (Edmonds-Karp)

Linear programming

- optimize a linear function while satisfying linear inequalities
- also have a $\max(\text{something}) = \min(\text{something else})$, so $\text{NP} \cap \text{co-NP}$
- in P!! (ellipsoid)

Primality

- certificates for non primes (easy) and for primes (not so easy), so $\text{NP} \cap \text{co-NP}$
- in P!!! (AKS)

Factoring

- HASFACOR is in $\text{NP} \cap \text{co-NP}$
- ?

Survey says

2.1 Does $P=NP$?

The following is a chart of the responses to the question **Does $P=NP$?** in my 2002, 2012, and 2019 polls. DK stands for Don't Know, DC stands for Don't Care, and Ind stands for Independent (which I assume means Independent of ZFC).

	$P \neq NP$	$P = NP$	Ind	DC	DK	DK and DC	other
2002	61 (61%)	9 (9%)	4 (4%)	1 (1%)	22 (22%)	0 (0%)	3 (3%)
2012	126 (83%)	12 (9%)	5 (3%)	5 (3%)	1 (0.66%)	1 (0.66%)	1 (0.66%)
2019	109 (88%)	15 (12%)	0	0	0	0	0

2.9 Does $P=NP \cap \text{co-NP}$?

98 responses.

- YES: 27 (28%)
- NO: 71 (72%)

NP-completeness

NP-complete problems

Definition

A decision problem **PROB** is NP-complete if

- **PROB** is in **NP**
- for any **PROB'** in **NP**, $\text{PROB}' \leq_P \text{PROB}$

polynomial time for **PROB** would give **P=NP** (so polynomial time for **SAT**, **INDEPENDENTSET**, **VERTEXCOVER**, **CLIQUE**, ...)

Remark: NP-hard problems = the second part of the definition

- decision problem **PROB** such that for any **PROB'** in **NP**, $\text{PROB}' \leq_P \text{PROB}$
- definitions of NP-hard not all consistent (sometimes defined not only for decision problems)

The Cook-Levin theorem

Claim

CIRCUITSAT is NP-complete

Remark 1: we already know it is in NP

Remark 2:

- we proved $\text{CIRCUITSAT} \leq_P \text{3SAT}$
- so 3SAT is NP-complete (it is in NP)
- we won't use CIRCUITSAT too much after that

World map

$$A \leftarrow B \equiv A \leq_P B$$

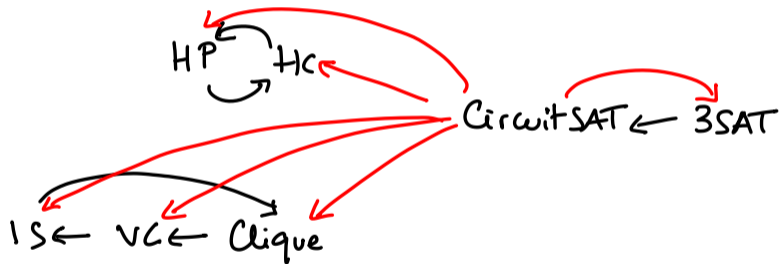


$$\text{CircuitSAT} \leftarrow 3\text{SAT}$$



World map

$$A \leftarrow B \equiv A \leq_p B$$



Sketch of proof

take PROB in **NP** (so there is a certifying algorithm B), want $\text{PROB} \leq \text{CIRCUITSAT}$
 \leadsto must transform an instance x of PROB into a circuit

Idea

- given x , verification algorithm $B(x, y)$ can be turned into a circuit with y as input
- we call CIRCUITSAT to find y

Example

- **problem** PROB: INDEPENDENTSET
- **instance** x : complete graph with 3 vertices (aka a triangle), $K = 2$
- **certificate** y : 3 bits y_1, y_2, y_3 (yes/no for each vertex)
- circuit for $B(x, y)$ computes the “formula”

$$(y_1 + y_2 + y_3 \geq 2) \wedge \overline{y_1 \wedge y_2} \wedge \overline{y_1 \wedge y_3} \wedge \overline{y_2 \wedge y_3}$$

Sketch of proof (bonus)

Turing machines

- RAM model too complicated, use Turing machines instead
- have a **pointer** to memory and a **state** (\simeq line in the source code)
- each step, pointer can write a new symbol, move left / right and change state

From machine to circuit

- on input bit vector x of size n , introduce a large table T of size $n^k \times n^k$ (k =exponent in runtime of B)
- cell (i, j) records contents of j th memory cell at time i , whether the pointer was there, and the machine state
- cells at row $i + 1$ are given by a boolean circuit taking row i as input (big, but polynomial size)
- output of the circuit = output of the Turing machine at the last time step

Some NP-complete problems

- CircuitSAT
- 3SAT, SAT
- independent set, vertex cover, clique
- (directed) Hamiltonian cycle, Hamiltonian path
- traveling salesman
- subset sum, 0/1 knapsack

(2SAT is polynomial time)

IndependentSet, VertexCover, Clique are NP-complete

We already know they are in **NP**

Claim

$$3\text{SAT} \leq_P \text{INDEPENDENTSET}$$

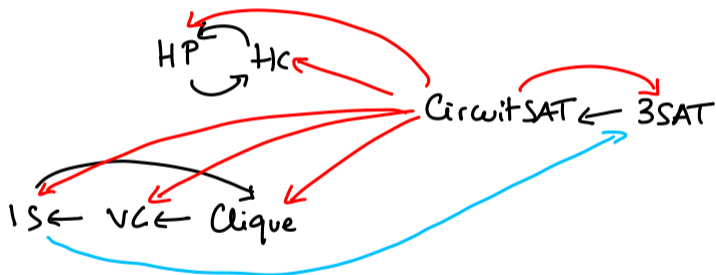
Reduction (transform an instance F of 3SAT with s clauses into an independent set instance)

- build a graph G with one vertex per literal
- connect all literals in any given clause
- connect all pairs x_i, \bar{x}_i

Remark: reduction takes polynomial time

World map

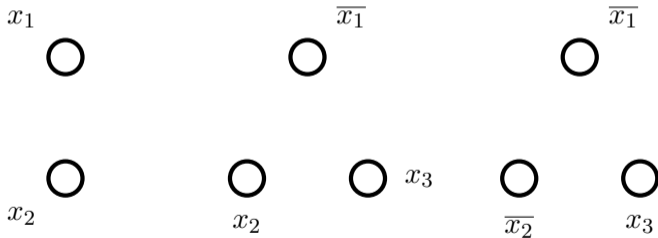
$$A \leftarrow B \equiv A \leq_p B$$



Example

A 3CNF formula with $s = 3$

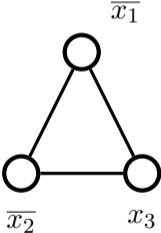
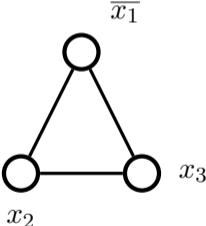
$$F = (x_1 \vee x_2) \wedge (x_2 \vee x_3 \vee \bar{x}_1) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_2).$$



Example

A 3CNF formula with $s = 3$

$$F = (x_1 \vee x_2) \wedge (x_2 \vee x_3 \vee \overline{x_1}) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_2}).$$



Example

A 3CNF formula with $s = 3$

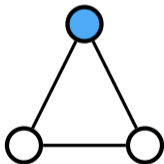
$$F = (x_1 \vee x_2) \wedge (x_2 \vee x_3 \vee \overline{x_1}) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_2}).$$

x_1



x_2

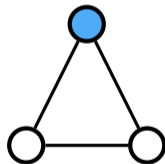
$\overline{x_1}$



x_2

x_3

$\overline{x_1}$



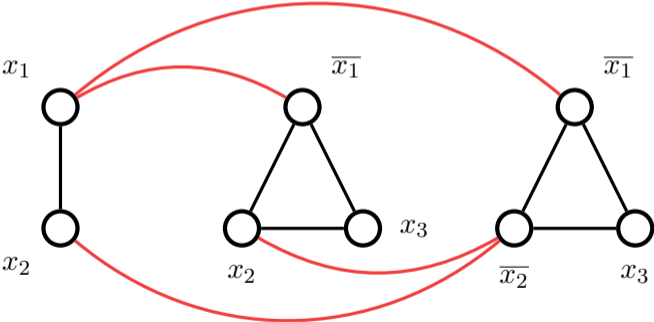
$\overline{x_2}$

x_3

Example

A 3CNF formula with $s = 3$

$$F = (x_1 \vee x_2) \wedge (x_2 \vee x_3 \vee \overline{x_1}) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_2}).$$



Proof

Claim

F satisfiable iff G has an independent set of size at least s

If F satisfiable

- pick **one** true literal in each clause as set S , so $|S| = s$
- no edge within clauses
- no edge $\{x_i, \bar{x}_i\}$ either

If G has an independent set S of size at least s

- S has (exactly) one vertex per clause
- make these literals true (for any variable we did not assign, arbitrary choice)
- no conflict, because any x_i, \bar{x}_i cannot be both in S

DirectedHamiltonianCycle, HamiltonianCycle, HamiltonianPath are NP-complete

Definition: DIRECTEDHAMILTONIANCYCLE

- **input:** directed graph G
- **output:** does G have a directed cycle that visits each vertex once?
- **NP**

Claim

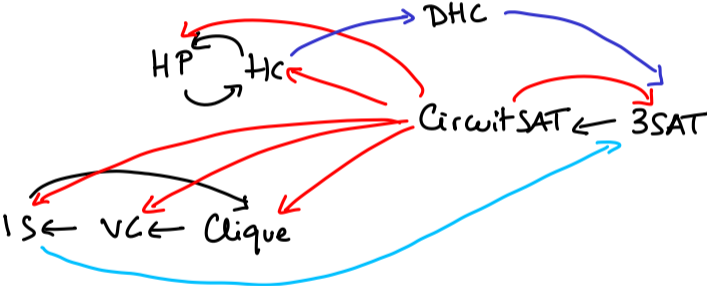
$3SAT \leq_P \text{DIRECTEDHAMILTONIANCYCLE} \leq_P \text{HAMILTONIANCYCLE}$

start with $3SAT \leq_P \text{DIRECTEDHAMILTONIANCYCLE}$, so we are given a formula in 3CNF

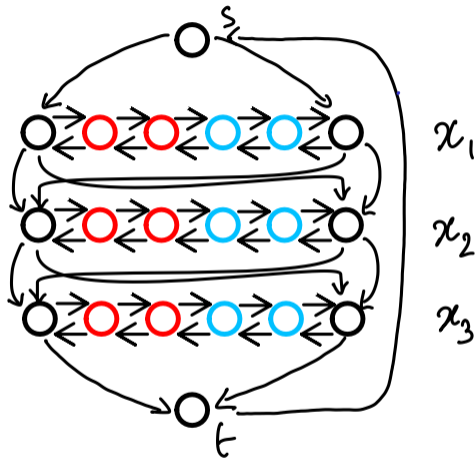
(**Remark:** almost the same construction works for DIRECTEDHAMILTONIANPATH)

World map

$$A \leftarrow B \equiv A \leq_p B$$



Starting the construction



Rules

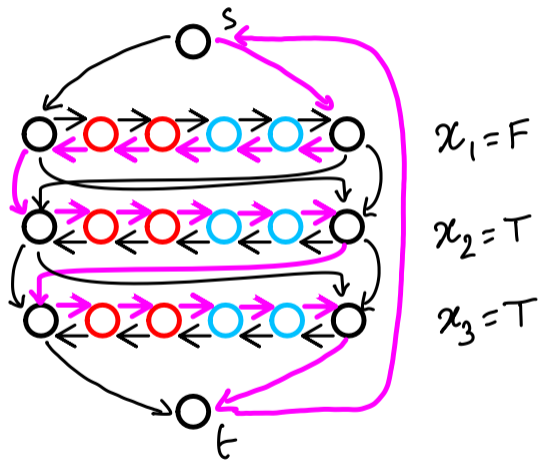
- source s , sink t
- **one row** of vertices per variable x_i
- on row i , **2** outside vertices (black) and **2** vertices $v_{i,j,1}, v_{i,j,2}$ per clause C_j
- example with $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2)$
(we're not done yet)

Remark:

- size of the graph and construction time polynomial in $n\ell$, ℓ = number of clauses

Hamiltonian cycles = variable assignments

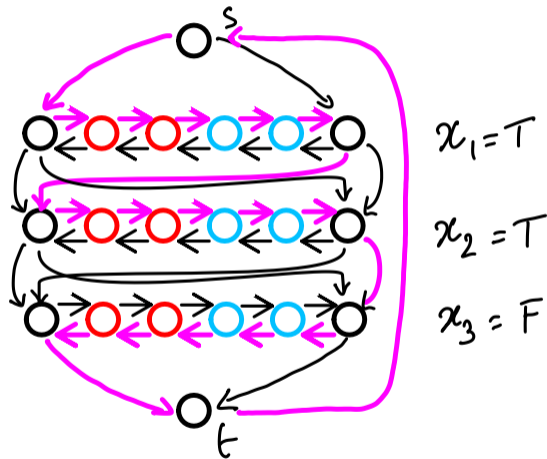
convention: T = left to right, F = right to left



so far, 2^n Hamiltonian cycles

Hamiltonian cycles = variable assignments

convention: T = left to right, F = right to left

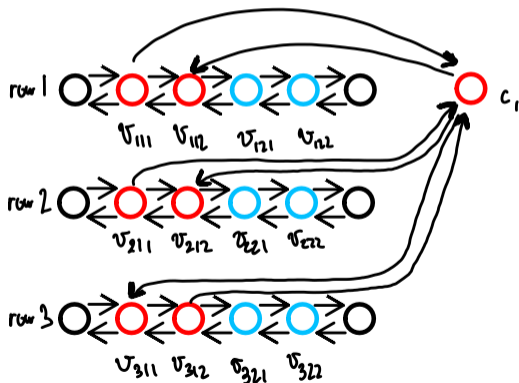


so far, 2^n Hamiltonian cycles

Using the clauses to finish the graph

For any clause C_j

- add a new vertex, also called c_j
- for any literal x_i in C_j , add edges $(v_{i,j,1}, c_j)$ and $(c_j, v_{i,j,2})$
- for any literal \bar{x}_i in C_j , add edges $(c_j, v_{i,j,1})$ and $(v_{i,j,2}, c_j)$



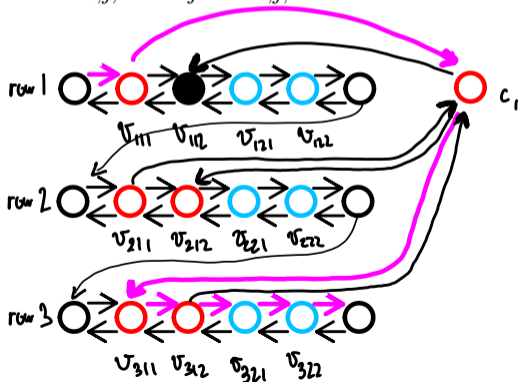
$$c_1 = (x_1 \vee x_2 \vee \bar{x}_3)$$

$3SAT \leq_P \text{DirectedHamiltonianCycle}$

Claim

if directed Hamiltonian cycle in G , formula satisfiable

Key Observation: if cycle goes from $v_{i,j,1}$ to c_j , must come back to $v_{i,j,2}$ (else, cannot put $v_{i,j,2}$ on the cycle), same with $v_{i,j,2} \rightarrow c_j \rightarrow v_{i,j,1}$



$3SAT \leq_P \text{DirectedHamiltonianCycle}$

Claim

if directed Hamiltonian cycle in G , formula satisfiable

Key Observation: if cycle goes from $v_{i,j,1}$ to c_j , must come back to $v_{i,j,2}$ (else, cannot put $v_{i,j,2}$ on the cycle), same with $v_{i,j,2} \rightarrow c_j \rightarrow v_{i,j,1}$

Consequences

- if we remove all vertices c_j from our cycle, we get a cycle on the clause-free graph
- so each row is visited LtoR or RtoL
- gives an assignment for x_1, \dots, x_n
- by design, it satisfies all clauses

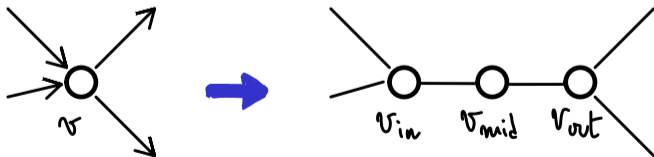
DirectedHamiltonianCycle \leq_P HamiltonianCycle

Reduction

- **given:** a directed graph G
- **build:** an undirected graph G'
- **ensure:** directed Hamiltonian cycle in $G \iff$ Hamiltonian cycle in G'

Gadget:

- replace each vertex v by v_{in}, v_{mid}, v_{out}
- make all edges undirected



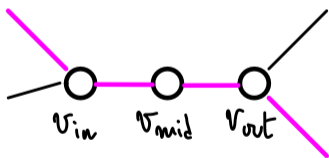
Directed Hamiltonian Cycle \leq_P Hamiltonian Cycle

Claim

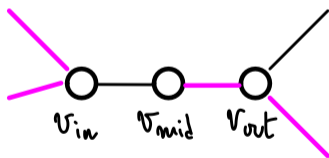
directed Hamiltonian cycle in $G \iff$ Hamiltonian cycle in G'

Proof

- if directed Hamiltonian cycle in G , Hamiltonian cycle in G' (follow the cycle)
- suppose Hamiltonian cycle in G' . Can only have



but not



(v_{mid} would be isolated)

gives a directed Hamiltonian cycle in G