18.404/6.840 Lecture 15

Last time:

- NTIME, NP
- P vs NP problem
- Dynamic Programming, P
- Polynomial-time reducibility

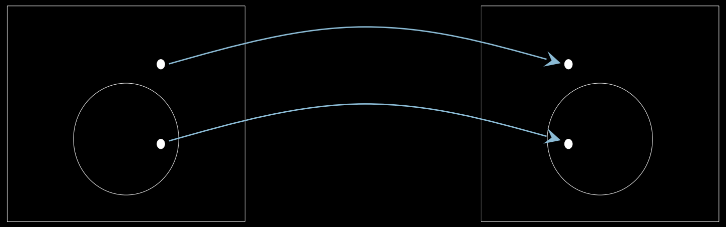
Today:

- NP-completeness

Quick Review

Defn: is <u>polynomial time reducible</u> to () if
by a reduction function that is computable in polynomial time.

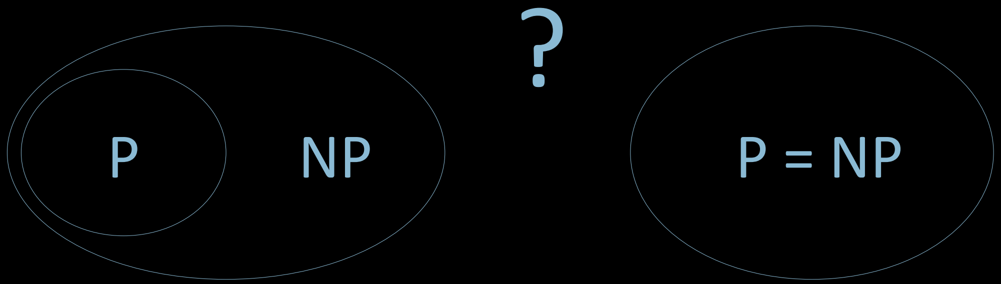
Theorem: If and P then P.



is computable in polynomial time

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

P versus NP question: Does P = NP?

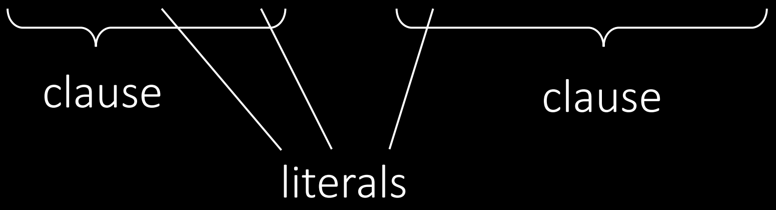


is a satisfiable Boolean formula}

Cook-Levin Theorem: P P = NP
Proof plan: Show that every is polynomial time reducible to .

Example: and

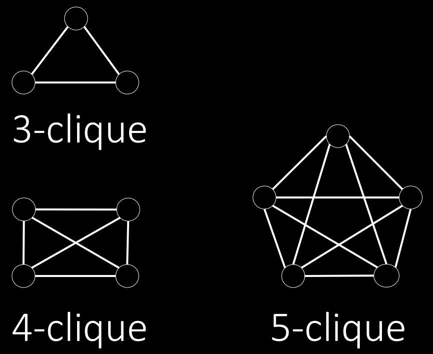
Defn: A Boolean formula is in Conjunctive Normal Form (CNF) if it has the form



Literal: a variable or a negated variable
Clause: an OR () of literals.
CNF: an AND () of clauses.
3CNF: a CNF with exactly 3 literals in each clause.
is a satisfiable 3CNF formula}

Defn: A -clique in a graph is a subset of k nodes all directly connected by edges.
graph contains a-clique}

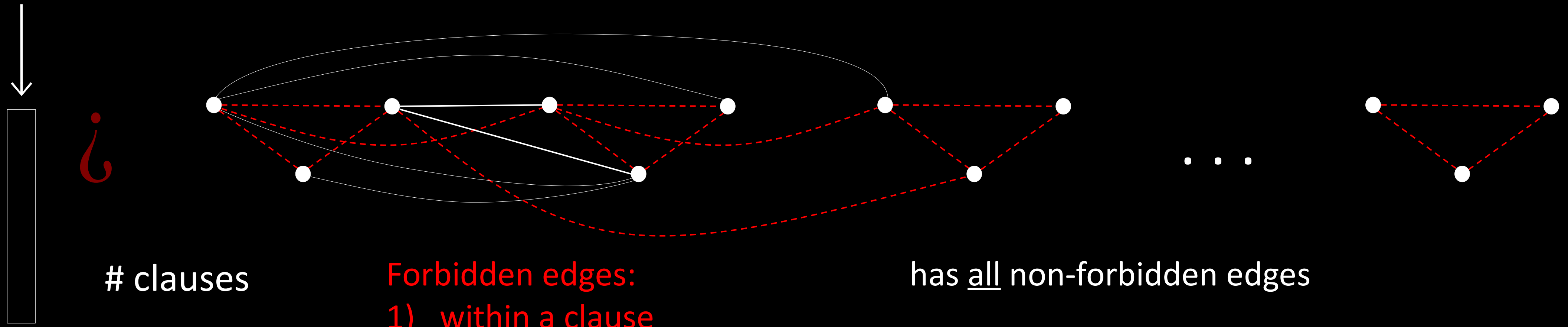
Will show:



Theorem:

Proof: Give polynomial-time reduction that maps to
where is satisfiable iff has a -clique.

A satisfying assignment to a CNF formula has 1 true literal in each clause.

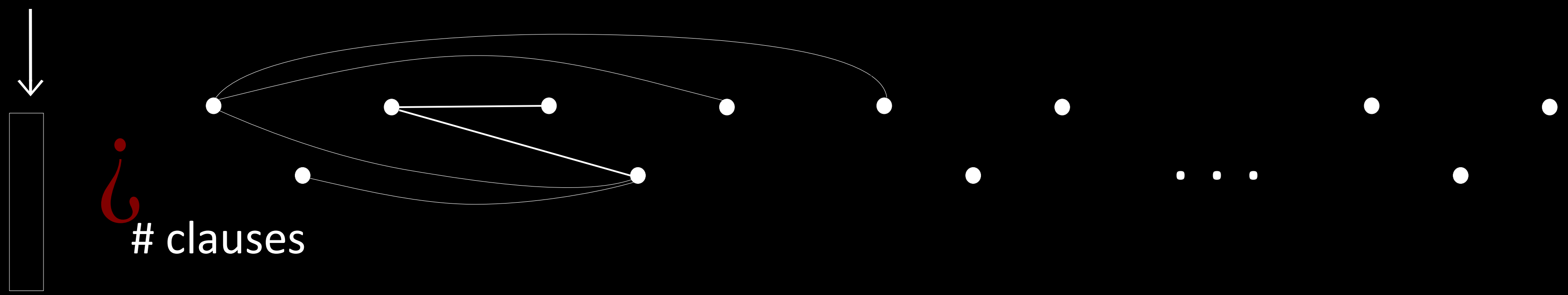


# clauses

Forbidden edges:
1) within a clause
2) inconsistent labels (and)

has all non-forbidden edges

conclusion



Claim: is satisfiable iff has a-clique

() **Take any satisfying assignment to . Pick 1 true literal in each clause.**

The corresponding nodes in G are a-clique because they don't have forbidden edges.

() **Take any -clique in . It must have 1 node in each clause.**

Set each corresponding literal TRUE. That gives a satisfying assignment to .

The reduction is computable in polynomial time.

Corollary:

Check-in 15.1

Does this proof require 3 literals per clause?

(a) Yes, to prove the claim.

(b) Yes, to show it is in poly time.

(c) No, it works for any size clauses.

Check-in 15.1

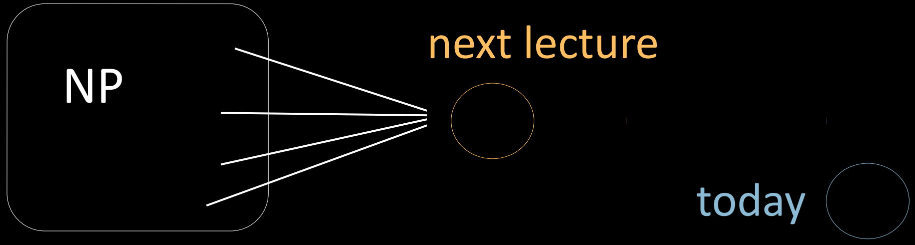
Coffee
Break

NP-completeness

Defn: is <u>NP-complete</u> if
1) NP
2) For all NP,

If is NP-complete and P then P NP.

Cook-Levin Theorem: is NP-complete
Proof: Next lecture; assume true



To show some language is NP-complete, show .

or some other previously shown NP-complete language

Check-in 15.2

What language that we've previously seen is most analogous to ?

(a)

(b)

(c)

Check-in 15.2

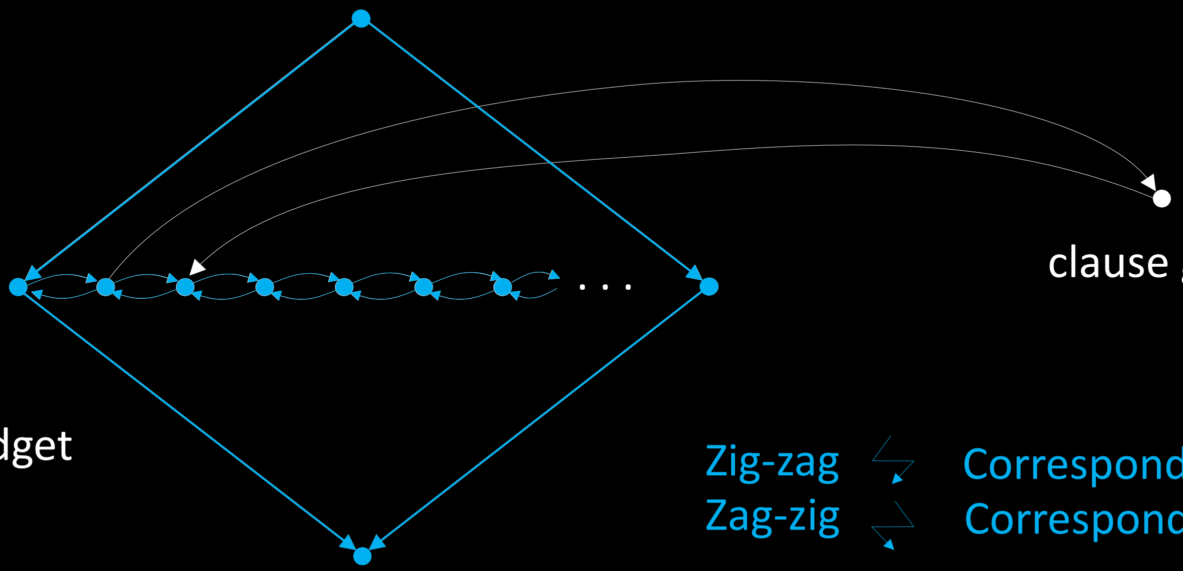
is NP-complete

Theorem: is NP-complete

Proof: Show (assumes is NP-complete)

Idea: “Simulate” variables and clauses with “gadgets”

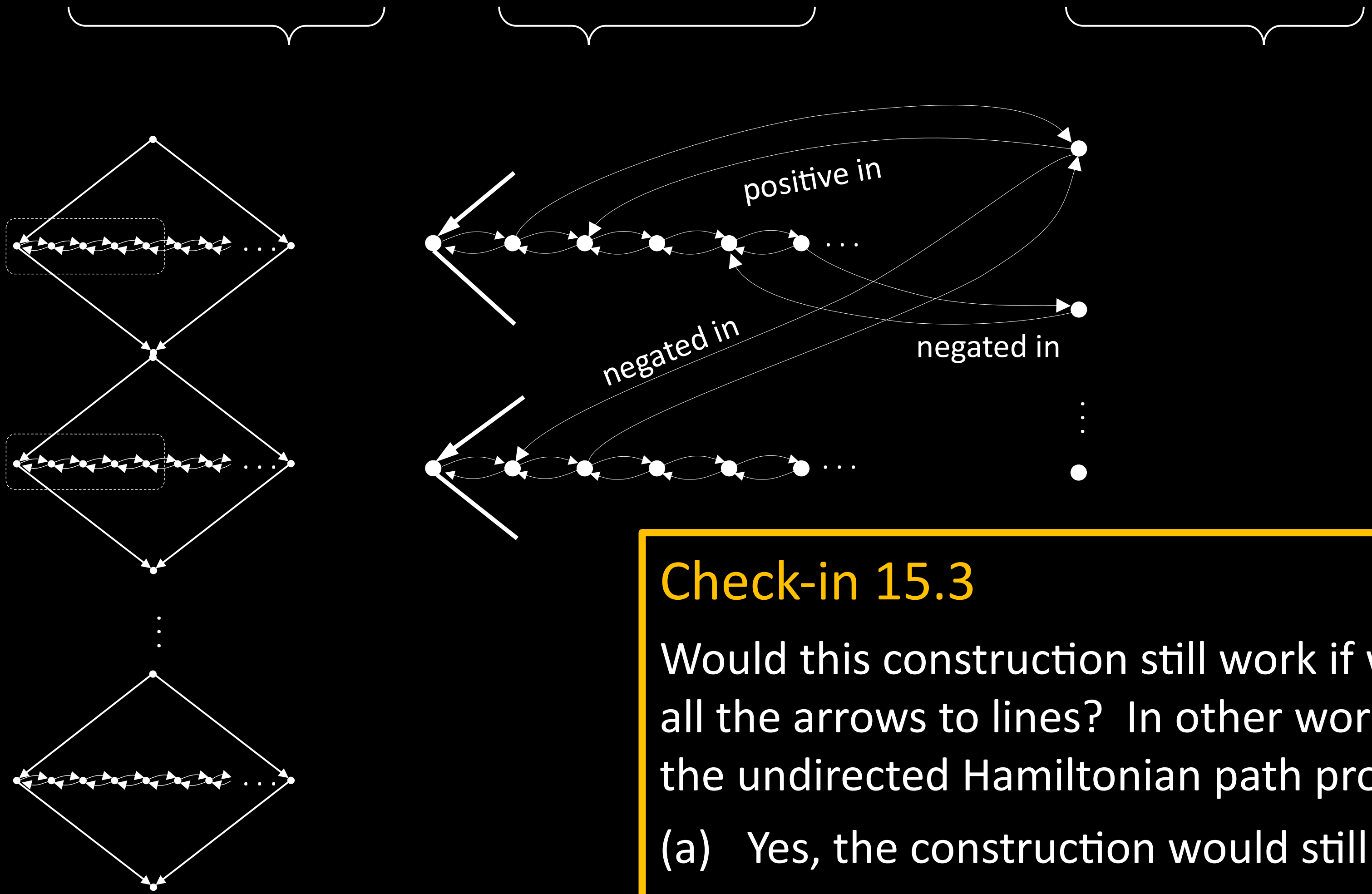
$\downarrow$



Zig-zag Corresponds to setting TRUE

Zag-zig Corresponds to setting FALSE

Construction of



variables
clauses

The reduction is computable
in polynomial time.

Check-in 15.3

Would this construction still work if we made undirected by changing all the arrows to lines? In other words, would this construction show that the undirected Hamiltonian path problem is NP-complete?

(a) Yes, the construction would still work.

(b) No, the construction depends on being directed.

Check-in 15.3

Quick review of today

1. NP-completeness
2. and
3.
4.
5. Strategy for proving NP-completeness: Reduce from by constructing gadgets that simulate variables and clauses.