CS 341 W22 Lecture 20 NP-Complete Problems

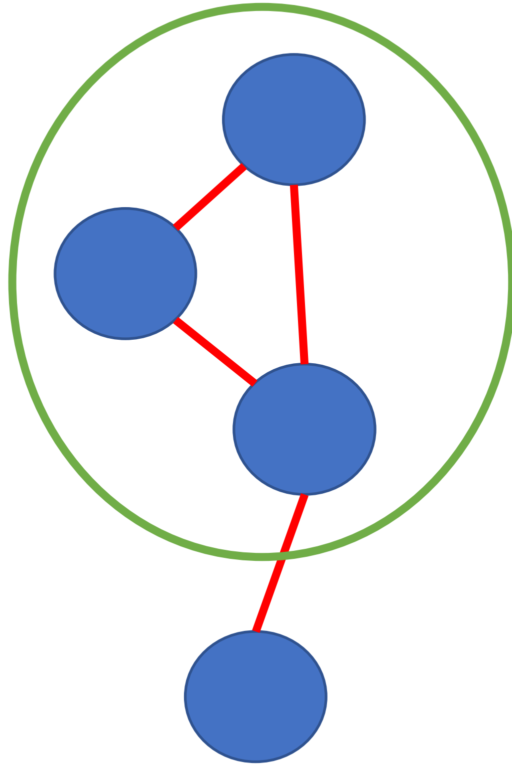
R. Peng, University of Waterloo

From Last Time

- To show a decision problem Z is NP-complete:
  - Prove Z is in NP
  - Prove $X \leq_P Z$ for a known NP-complete problem X
- A many-one reduction $X \leq Y$ uses the algorithm for Y once and outputs its answer
- NP-Complete Problems: 3-SAT, Independent Set
- Today: show NP-completeness of
  - Clique
  - Vertex Cover
  - Hamiltonian Cycle / TSP

Clique

- **Input**: undirected graph $G = (V, E)$, number k
- **Output**: does G have a clique (set of vertices that's pairwise joined by edges) of size $\geq k$



Observe: C is a clique if and only if C is an independent set in G^C (complement of G, edges <-> non-edge)

Theorem: Clique is NP-Complete

1. Clique is in NP: certificate is the set C, verification is to check $|C| \geq k$, and all pairs in C are in E, $O(n^2)$ time
2. Will show Independent Set $\leq_P$ Clique

Independent Set $\leq_P$ Clique

- Assume we have a polynomial time algorithm for Clique
- Invoke it to give a polynomial time algorithm for independent set

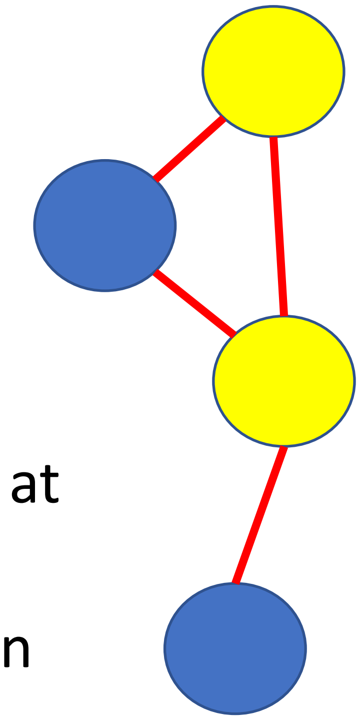
Independent Set

- **Input**: undirected graph G = (V, E), number k
- **Output**: does G have an independent set (set of pairwise disjoint vertices) of size ≥ k

Reduction: set G' = G^c, and k' = k

- Runtime: $O(n^2)$
- Correctness: G has an clique of size ≥ k if and only if G^c has an independent set of size ≥ k. S⊆V is a clique in G if and only if S (the same set) is an independent set in G^c

Vertex Cover



- **Input**: undirected graph G = (V, E), number k
- **Output**: does G have a vertex cover of size ≤ k

Vertex cover: S⊆V such that every edge (u, v) has at least one of u or v (or both) in S

Observe: S is a vertex cover if and only if V \ S is an independent set

Theorem: Vertex Cover is NP-complete

- Vertex Cover is in NP: certificate = S
- Will show Independent Set $\leq_P$ Vertex Cover

Independent Set $\leq_P$ Vertex Cover

- Assume we have a polynomial time algorithm for Vertex Cover
- Given inputs G and k, construct G' and k' such that G' has a vertex cover of size at most k' if and only if G has an independent set of size at least k.
- Return the result of the Vertex Cover algorithm on (G', k')

  Reduction: set G' = G, and k' = |V| - k

  - Runtime: exercise.
  - Correctness:
    - ➔ Let S be the vertex cover of size ≤ k' = |V| - k in G', then V \ S is an independent set in G, and it has size |V| - |S| ≥ |V| - k' = |V| - (|V| - k) = k
    - 🡐 Let I be an independent set of size ≥ k in G, then V \ I is a vertex cover of size at most |V| - k = k'.

Roadmap to NP-Completeness

Clique

$\leq_P$

Ind. Set $\leq_P$ Vertex Cover $\leq_P$ Set Cover

$\leq_P$

Circuit SAT $\leq_P$ 3-SAT $\leq_P$ Ham. Cycle $\leq_P$ TSP

$\leq_P$

Subset Sum

History of NP Completeness

- 3-SAT is NP Completeness:
  - Cook 1971
  - Independently Levin
- [Karp `72] “Reducibility Among Combinatorial Problems”: first 21 NP-Complete problems: https://en.wikipedia.org/wiki/Karp%27s_21_NP-complete_problems
- https://en.wikipedia.org/wiki/List_of_NP-complete_problems, many from [Garey-Johnson `79] “Computers and Intractability”.

Why Many-One Reductions

- Special case of Turing reduction (invoke the algorithm for Y arbitrarily many times), so it's a stronger result
- More structured, so easier to prove correctness, as well as finding the proof
- Convention

Open, but holds for all known cases:

Is there always a many-one reduction to prove that a problem is NP-complete?

Lecture 20 Part 1 Summary

- Clique and Vertex Cover are NP-complete
- There are many NP-complete problems, many-one reductions suffice to all ones we know of to date
- What you should know from lecture 20 part 1
  - How to prove a problem is NP-complete using a polynomial time many-one reduction
- Next: a more intricate NP-completeness proof, to show that Hamiltonian cycle is NP-complete

Directed Hamiltonian Cycle

- Input: directed graph G = (V, E)
- Output: does G have a directed Hamiltonian Cycle?

(recall: Hamiltonian cycle is a cycle that visits each vertex exactly once)

Theorem: Directed Hamiltonian Cycle is NP-Complete

1. Directed Hamiltonian Cycle is in NP: exercise
2. 3-SAT $\leq_P$ Directed Hamiltonian Cycle: assume we have a polynomial time algorithm for Directed Hamiltonian Cycle, use it to solve 3-SAT.

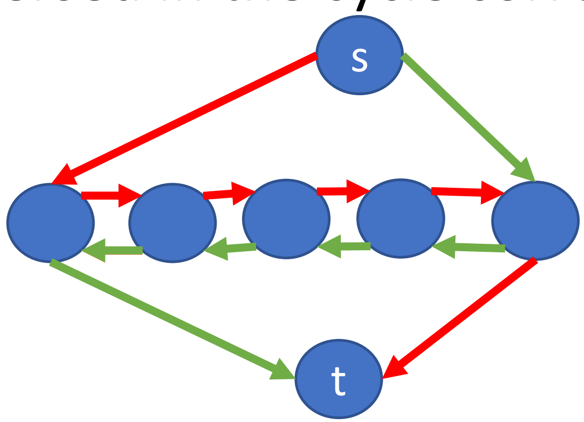
Need: a routine that takes a 3-SAT instance F, outputs a graph that has a Hamiltonian Cycle if and only if F is satisfiable.

From Boolean Variables to Graphs

3-SAT

- Input: n variables $x_1 \ldots x_n$, m clauses $c_1 \ldots c_m$, each the OR of three literals of the form x_i /$\neg x_i$.
- Output: whether we can assign T/F to the x_i s so that each clause has a literal satisfied

Idea: create a part of the graph for each variable, so that how it gets traversed in the cycle corresponds to whether x_i is true or false



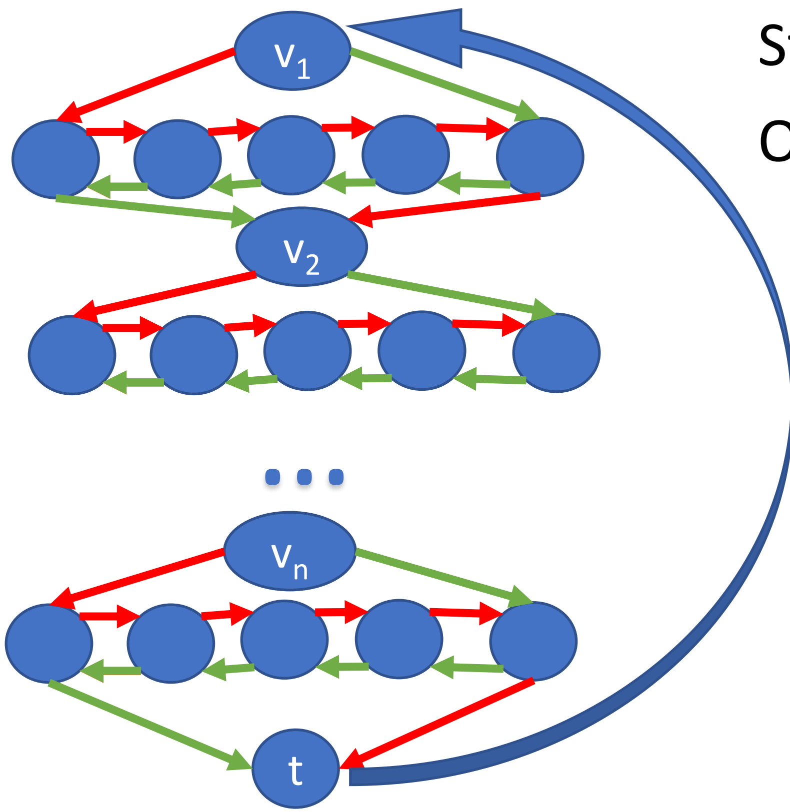
Going from s-to-t

- T: go left-to-right
- F: go right-to-left

From Boolean Variables to Graphs

Given 3-SAT formula F with n variables, m clauses

Create a graph that has a Hamiltonian cycle if and only if F is satisfiable



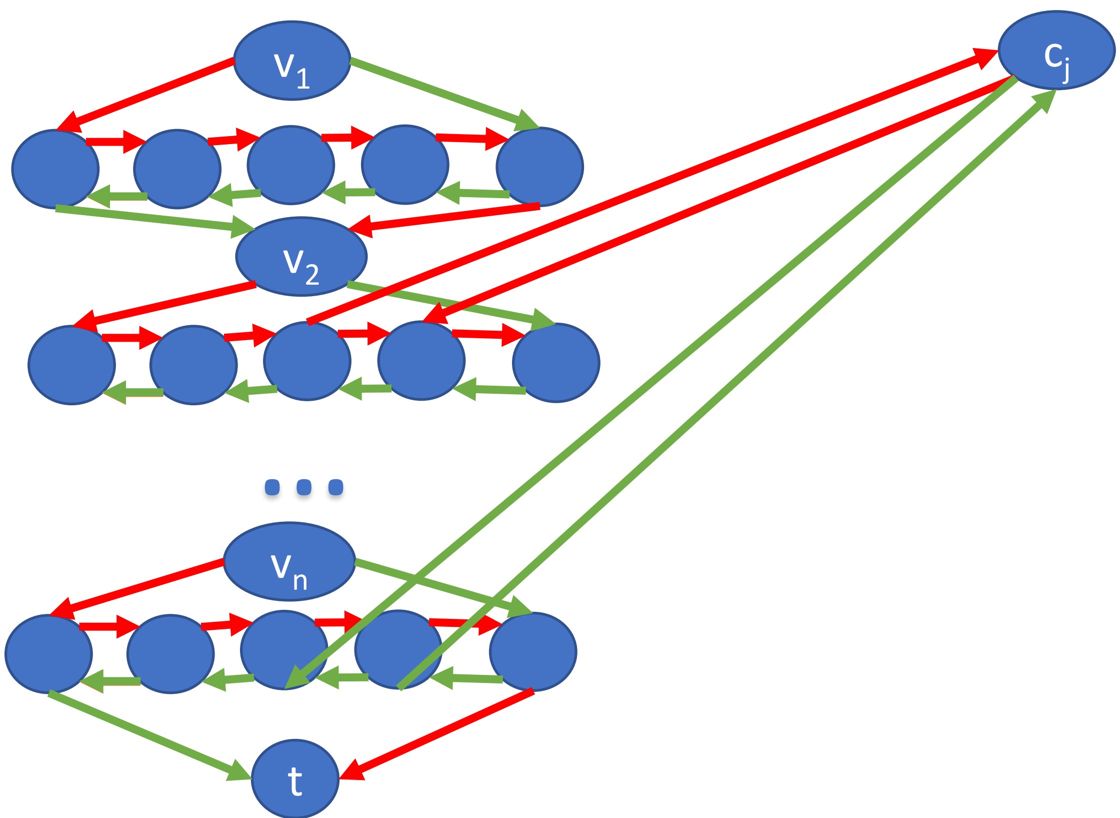
Stack these gadgets,

One per variable

t→1 edge to force a cycle

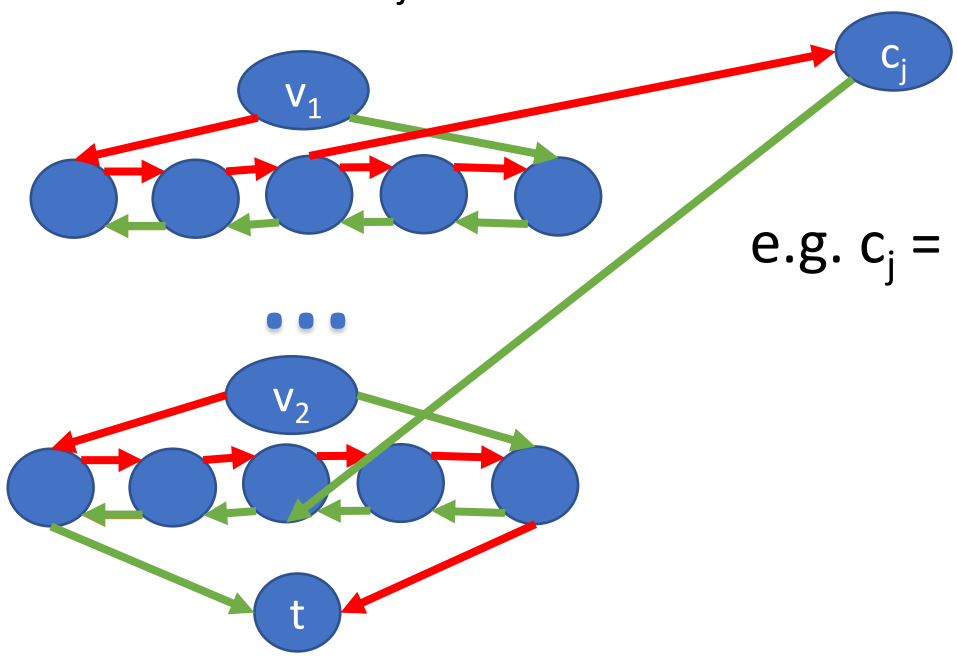
Encoding a Clause

- Make a vertex, c_j (which must be traversed)
- If it has x_i, connect c_j to the `path' of x_i in forward direction
- Otherwise (has $\neg x_i$), connect in reverse direction



‘Jumping around’ using Clauses

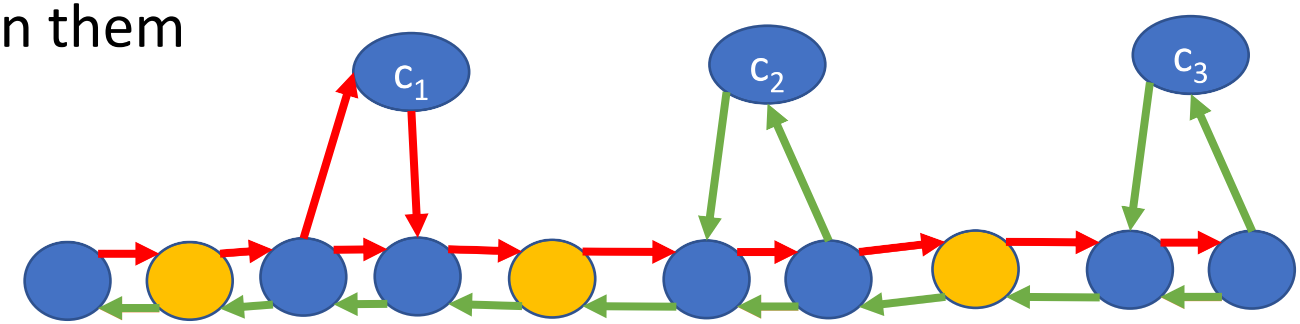
- Can go from row for x_i to row for x_j via two different literals in c_j



e.g. $c_j = (x_1 \lor \neg x_2 \lor x_3)$

Fix: insert spare vertices

- Make sure different `detour edges' have at least one vertex between them



- Formally: for variable $x_{i:}$
  - path of O(m) vertices $v_{i,1} \ldots v_{i,3m+3}$
  - If clause j contains x_i, add $v_{i,3j} \rightarrow c_j$, $c_j \rightarrow v_{i,3j+1}$
  - If clause j contains $\neg x_i$, add $v_{i,3j+1} \rightarrow c_j$, $c_j \rightarrow v_{i,3j}$

Correctness of Reduction

G has a Hamiltonian cycle iff F is satisfiable

- ← if F is satisfiable, traverse the variable paths in the True/False directions. For each clause, take a detour of one of its satisfied literals.
- → Suppose G has a directed Hamiltonian path.

    Key claim: only way some c_i is visited is by a detour

    Need to show: if we take $a \to c_i$, must take $c_i \to b$:

    we don't use $a \to d$

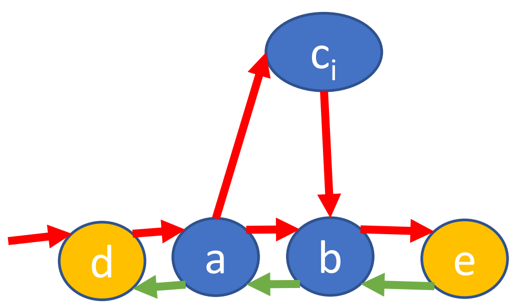
    so must use $d \to a$

    so $b \to a$ is not used

    so $b \to e$ must be used as 1 edge leaves b

    so $e \to b$ can't be used

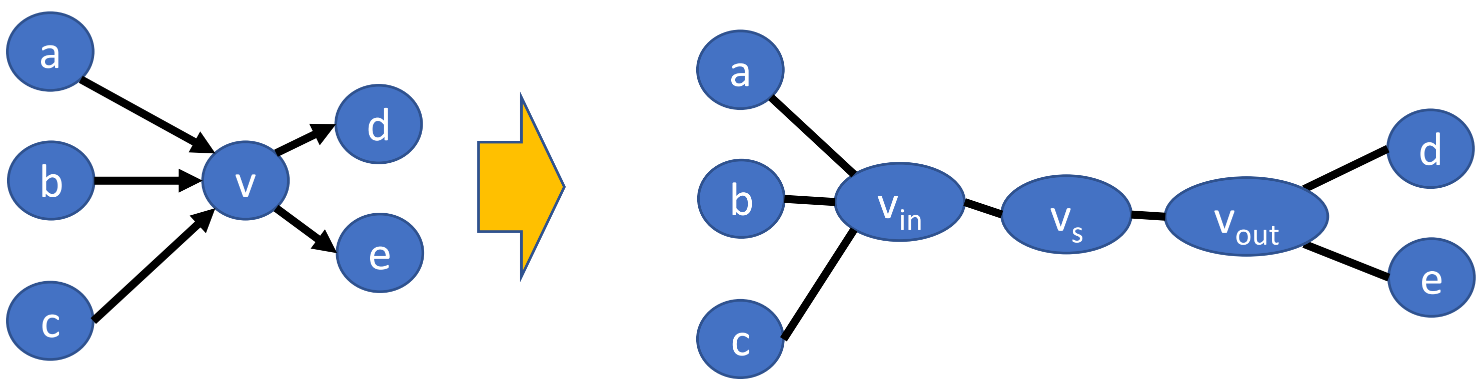
    so the edge into b must be $c_i \to b$



This means the path must traverse each x_i path in either T or F direction, so gives a satisfying assignment

Undirected Hamiltonian Cycle is NP-Complete

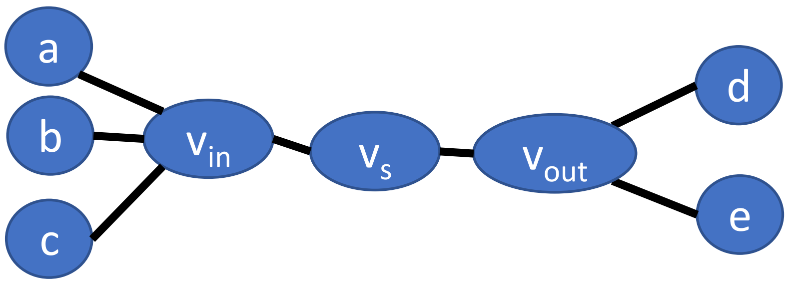
- Hamiltonian cycle is in NP.
- Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle
  - Given directed graph G, make undirected G' s.t. G has directed Hamiltonian cycle iff G' has undirected Hamiltonian cycle.
  - Plan: split vertices, v into v_{in} and v_{out}, with a spare vertex in between to ensure path goes from v_{in} to v_{out}



Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle

Form G' by:

- split vertices, v into v_{in} and v_{out}, with a spare vertex in between to ensure path goes from v_{in} to v_{out}
- Turn all u→v to (u_{out}, v_{in})



Runtime: O(m)

Correctness:

- Cycle in G → cycle in G'
- Cycle in G': cannot take both (a, v_{in}) and (b, v_{in}) because that would leave v_s with 1 neighbor
- So each vertex v must have v_{in}, v_s, v_{out} in order, with u_{out} before, and w_{in} after. This maps to directed edges in G.

Lecture 20 Summary

- NP-completeness of Independent Set, Clique, Vertex Cover, Hamiltonian Cycle, and TSP
- What you should know from Lecture 20: proving NP-completeness using many-one reductions
- Next: NP-completeness of subset sum, and circuit-SAT