Given this 3-SAT problem:

(x1 or x2 or x3) AND

(¬x1 or ¬x2 or ¬x2) AND

(¬x3 or ¬x1 or x2)

1. Draw the graph that you would use if you want to solve this problem using an algorithm for vertex cover.

2. What size of vertex cover would you look for in order to decide if this 3-SAT system is satisfiable?

Announcements

Final exam tutorial:

Monday Aug. 6, 10am, ECS 116.

If the building is locked, I will prop open the back door to ECS (the one that opens on to the campus).

Old finals are available from the course web page- the old finals page has linked to study guides and NP-completeness notes.

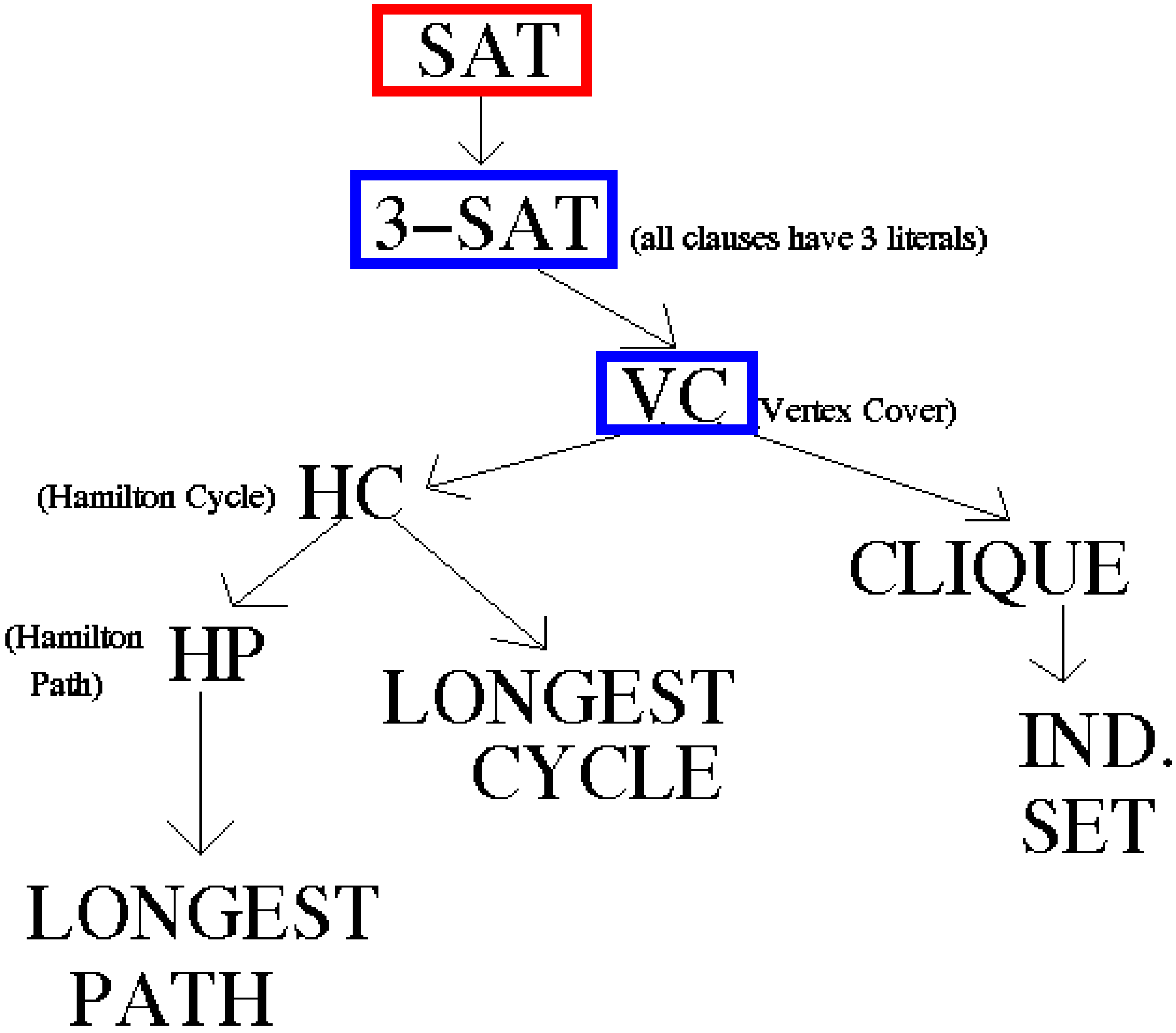
Remind me to post #5 solutions.

SAT is NP-complete

The fact that vertex cover is NP-complete is used to show that INDEPENDENT SET is NP-complete. Then we prove that CLIQUE is NP-complete.

An outline is given for the proof of Cook's theorem (SAT is NP-complete).

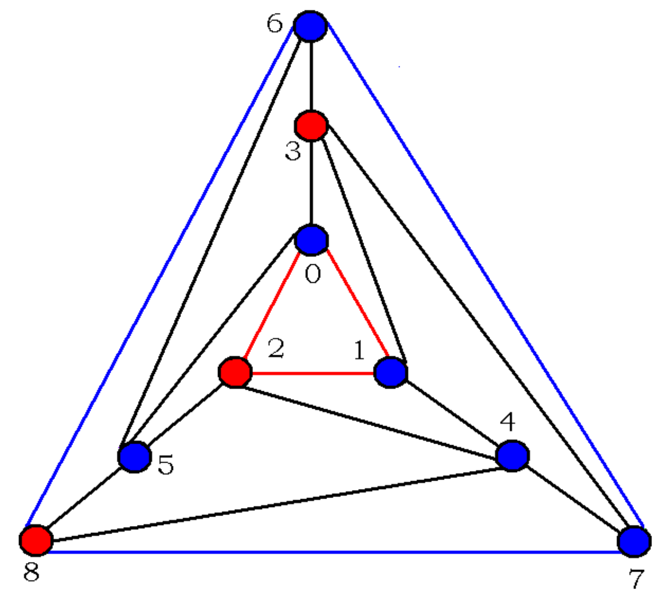
It is a direct proof. The tactic for showing other problems are NP-complete is to use a reduction preserving polynomial running time.



History of NP–completeness Reductions

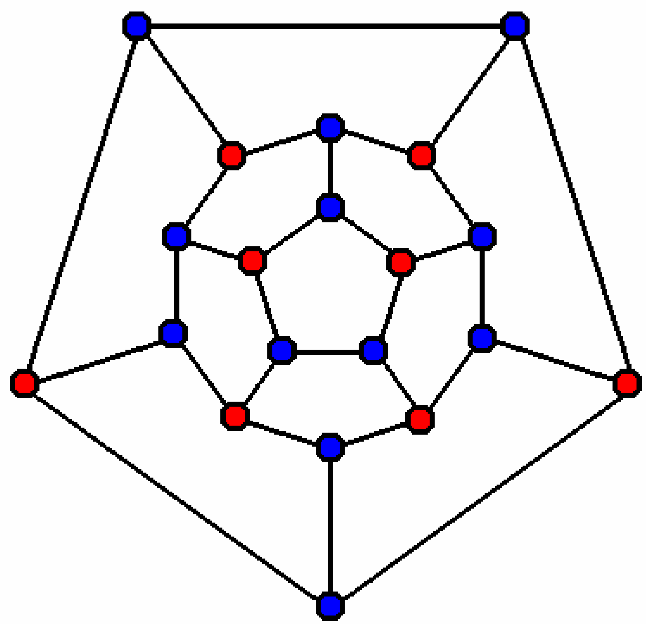
INDEPENDENT SET: Given G,k, does G have an independent set of order k?

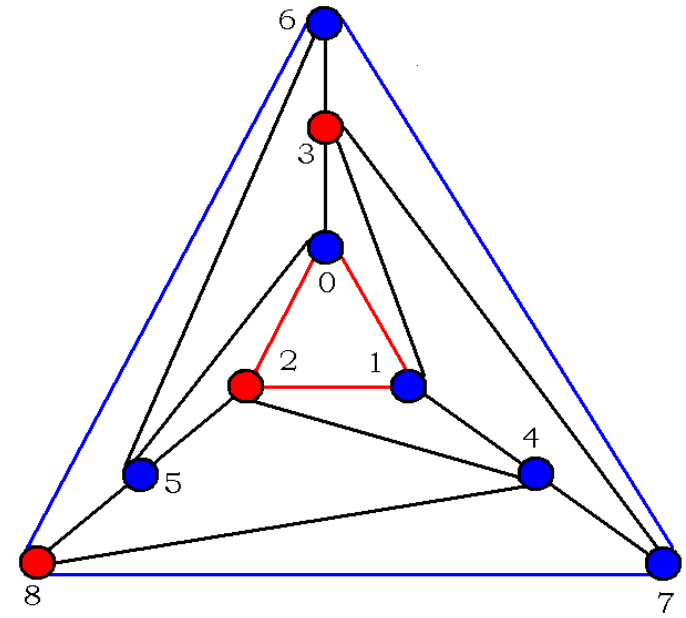
Theorem: IND. SET is NP-complete.



Proof: Certificate: vertex numbers of vertices in ind. set.

To check: Time $O(k^2)$ to make sure edges are missing between each pair of vertices in the independent set.

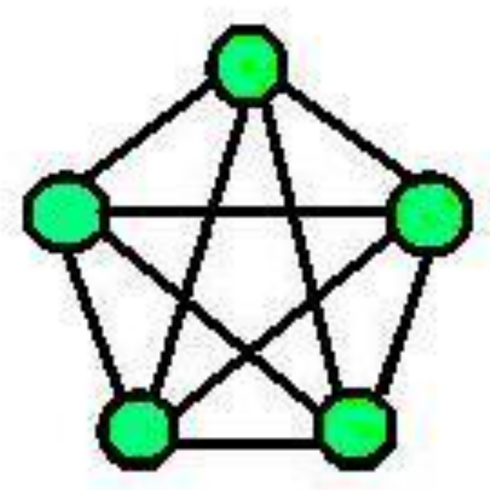




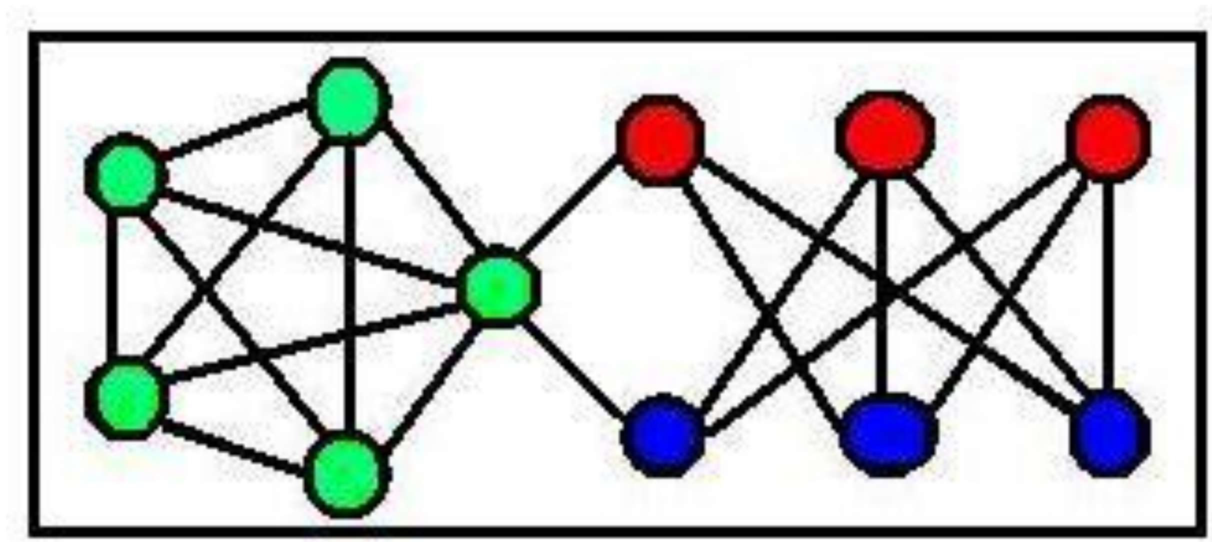
To determine if G has a vertex cover of order k: ask if G has an independent set of order n-k. The complement of every vertex cover is an independent set.

CLIQUE: Given G,k, does G have a clique of order k?

Theorem: CLIQUE is NP-complete.

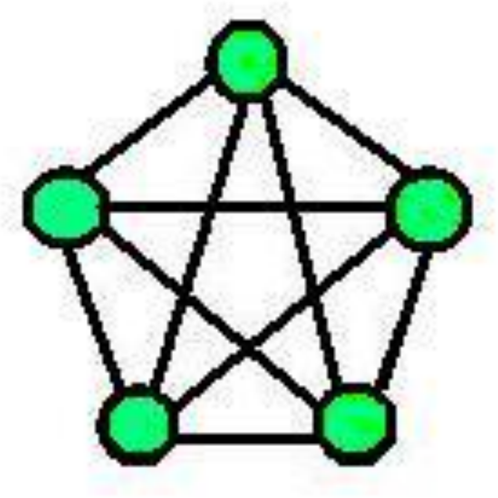


Clique of order 5.



Graph which has a clique of order 5.

The complement of G: has an edge (u, v) for each pair of vertices in G which are not connected by an edge in G.



To solve IND. SET using CLIQUE: create the complement of G in $O(n^2)$ time and then ask if it has a clique of order k.

A problem Q in NP is NP-complete if the existence of a polynomial time algorithm for Q implies the existence of a polynomial time algorithm for all problems in NP.

How do we prove SAT is NP-complete?

Theorem: **SAT (Satisfiability) is in NP.**

Proof.

Variables: $u_1, u_2, u_3, \ldots u_n$.

Certificate: sequence $b_1, b_2, b_3, \ldots b_n$ of true/false values.

Check for each clause if at least one literal is true.

Time: $O(n+k)$ where k is the number of occurrences of a literal in a clause.

Every problem Q which is in NP has a non-deterministic TM M which:

1. Guesses a certificate of length n.
2. Checks the certificate in p(n) time.

Start state- q_0

Final state- the computation will terminate in a special state q_N when the certificate is not correct and in a state q_Y if the certificate is correct.

Cook's Theorem: SAT is NP-complete.

Idea of proof:

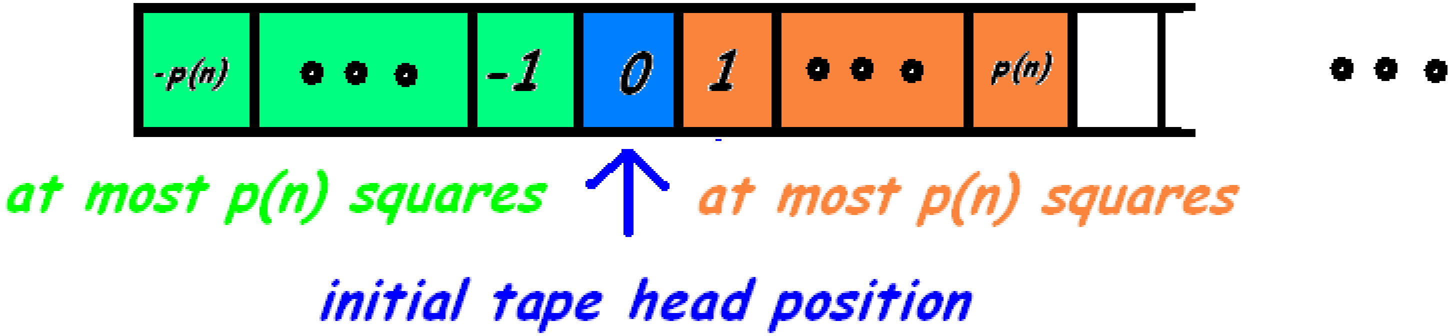
Construct a collection of clauses whose length is polynomial in p(n) and hence is also polynomial in n.

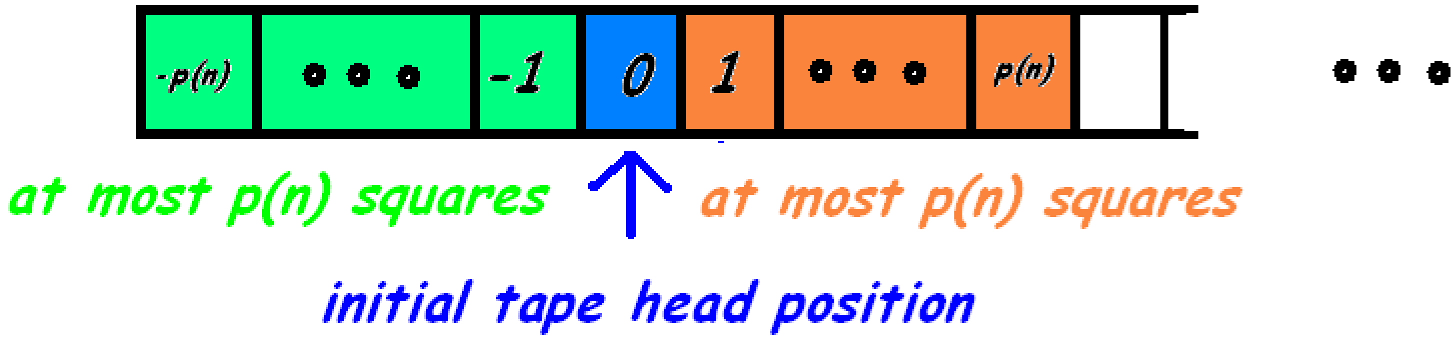
A satisfying truth assignment corresponds to a computation of M which non-deterministically guesses a correct certificate and checks it.

The number of tape squares used is bounded:

p(n)= maximum number of steps that M takes on an input of size n.

Tape squares used in this computation:





Variable	Range	Meaning
Q[i, k]	$0 \le i \le p(n)$, $1 \le k \le$ # states	At time i, M is in state q_k
H[i,j]	$0 \le i \le p(n)$, $-p(n) \le j \le p(n)$	Scanning square j at time i
S[i, j, k]	$0 \le i \le p(n)$, $-p(n) \le j \le p(n)$, $1 \le k \le$ # symbols	Time i, square j contains symbol s_k

Clauses to make sure that satisfying assignments correspond to valid computations:

Recall: Q[i, k]- At time i, M is in state q_k

Group G1: At time i, M is in exactly one state:

(a) M is in at least one state at time i:

for each i, $0 \leq i \leq p(n)$, add a clause (s= # states):

(Q[i, 1] or Q[i, 2] or ... or Q[i, s])

(b) M is in at most one state at time i: For each time i and pair j, k of states,

$0 \leq i \leq p(n)$, $1 \leq j < k \leq s$, add a clause

(not Q[i, j] or not Q[i, k])

Clause Group	Restriction
G1	At time i, M is in exactly one state.
G2	At time i, the tape head scans exactly one square.
G3	At time i, each square contains exactly one symbol.
G4	At time 0, M is in initial checking stage.
G5	By time p(n), M enters state q_Y.
G6	Configuration at time i+1 follows the one at time i by one application of the transition relation.

Original problem: Input of size n.

The size of this SAT system has length which is at most some polynomial $q(n)$.

If there is an $O(n^c)$ algorithm for SAT then there is an $O(q(n)^c)$ algorithm for this other problem which is in NP.

Therefore, if there is a polynomial time algorithm for SAT, there is a polynomial time algorithm for every problem in NP.

Conclusion: SAT is NP-complete.

Summary of class contents: