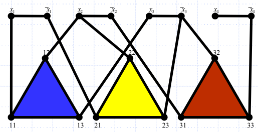
NP-Completeness (2)

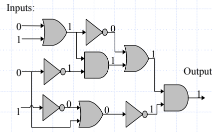


Outline and Reading

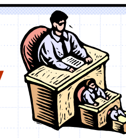
- Definitions (§13.1-2)
  - NP is the set of all problems (languages) that can be
    - accepted non-deterministically (using "choose" operation) in polynomial time.
    - verified in polynomial-time given a certificate y.
- Some NP-complete problems (§13.3)
  - Problem reduction
  - CNF-SAT and 3SAT
  - Vertex Cover
  - Clique
  - Hamiltonian Cycle

Problem Reduction

- A language M is polynomial-time **reducible** to a language L if an instance x for M can be transformed in polynomial time to an instance x' for L such that x is in M if and only if x' is in L.
  - Denote this by M⇒L.
- A problem (language) L is **NP-hard** if every problem in NP is polynomial-time reducible to L.
- A problem (language) is **NP-complete** if it is in NP and it is NP-hard.
- CIRCUIT-SAT is NP-complete:
  - CIRCUIT-SAT is in NP
  - For every M in NP, M⇒CIRCUIT-SAT.

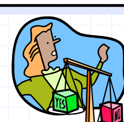


Transitivity of Reducibility



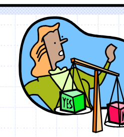
- If A⇒B and B⇒C, then A⇒C.
  - An input x for A can be converted to x' for B, such that x is in A if and only if x' is in B. Likewise, for B to C.
  - Convert x' into x" for C such that x' is in B iff x" is in C.
  - Hence, if x is in A, x' is in B, and x" is in C.
  - Likewise, if x" is in C, x' is in B, and x is in A.
  - Thus, A⇒C, since polynomial-time is closed under composition.
- Types of reductions:
  - **Local replacement:** Show A⇒B by dividing an input to A into components and show how each component can be converted to a component for B.
  - **Component design:** Show A⇒B by building special components for an input of B that enforce properties needed for A, such as "choice" or "evaluate."

SAT

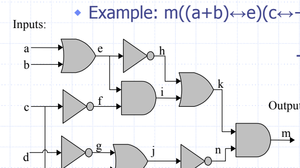


- A Boolean formula is a formula where the variables and operations are Boolean (0/1):
  - (a+b+¬d+e)(¬a+¬c)(¬b+c+d+e)(a+¬c+¬e)
  - OR: +, AND: (times), NOT: ¬
- SAT: Given a Boolean formula S, is S satisfiable, that is, can we assign 0's and 1's to the variables so that S is 1 ("true")?
  - Easy to see that CNF-SAT is in NP:
    - Non-deterministically choose an assignment of 0's and 1's to the variables and then evaluate each clause. If they are all 1 ("true"), then the formula is satisfiable.

SAT is NP-complete



- Reduce CIRCUIT-SAT to SAT.
  - Given a Boolean circuit, make a variable for every input and gate.
  - Create a sub-formula for each gate, characterizing its effect. Form the formula as the output variable AND-ed with all these sub-formulas:
    - Example: m((a+b)↔e)(c↔¬f)(d↔¬g)(e↔¬h)(ef↔i)…

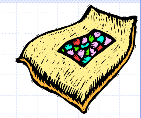


The formula is satisfiable if and only if the Boolean circuit is satisfiable.

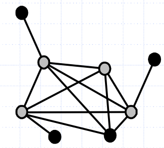
3SAT

- The SAT problem is still NP-complete even if the formula is a conjunction of disjuncts, that is, it is in conjunctive normal form (CNF).
- The SAT problem is still NP-complete even if it is in CNF and every clause has just 3 literals (a variable or its negation):
  - (a+b+¬d)(¬a+¬c+e)(¬b+d+e)(a+¬c+¬e)
- Reduction from SAT (See §13.3.1).

Vertex Cover



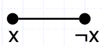
- A vertex cover of graph G=(V,E) is a subset W of V, such that, for every edge (a,b) in E, a is in W or b is in W.
- VERTEX-COVER: Given an graph G and an integer K, is does G have a vertex cover of size at most K?



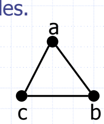
- VERTEX-COVER is in NP: Non-deterministically choose a subset W of size K and check that every edge is covered by W.

Vertex-Cover is NP-complete

- Reduce 3SAT to VERTEX-COVER.
  - Let S be a Boolean formula in CNF with each clause having 3 literals.
  - For each variable x, create a node for x and ¬x, and connect these two:

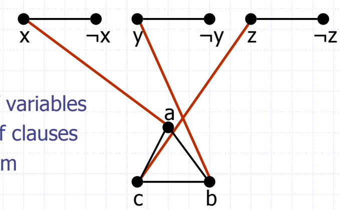


  - For each clause (a+b+c), create a triangle and connect these three nodes.



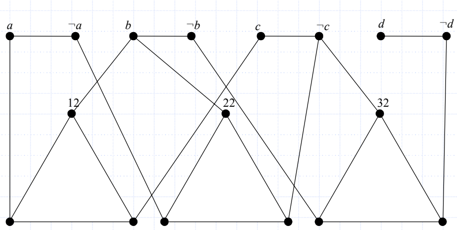
Vertex-Cover is NP-complete

- Completing the construction
  - Connect each literal in a clause triangle to its copy in a variable pair.
  - E.g., a clause (¬x+y+z)
  - Let n=# of variables
  - Let m=# of clauses
  - Set K=n+2m



Vertex-Cover is NP-complete

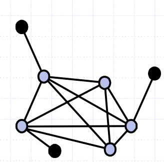
- Example: (a+b+c)(¬a+b+¬c)(¬b+¬c+¬d)
- Graph has vertex cover of size K=4+6=10 iff formula is satisfiable.



Clique

- A **clique** of a graph G=(V,E) is a subgraph C that is fully-connected (every pair in C has an edge).
- CLIQUE: Given a graph G and an integer K, is there a clique in G of size at least K?

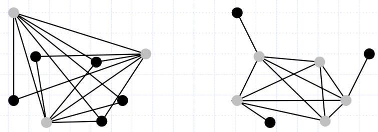
This graph has a clique of size 5



- CLIQUE is in NP: non-deterministically choose a subset C of size K and check that every pair in C has an edge in G.

CLIQUE is NP-Complete

- Reduction from VERTEX-COVER.
- A graph G has a vertex cover of size K if and only if it's complement has a clique of size n-K.



G G'

Some Other NP-Complete Problems

- SET-COVER: Given a collection of m sets, are there K of these sets whose union is the same as the whole collection of m sets?
  - NP-complete by reduction from VERTEX-COVER
- SUBSET-SUM: Given a set of integers and a distinguished integer K, is there a subset of the integers that sums to K?
  - NP-complete by reduction from VERTEX-COVER

Some Other NP-Complete Problems

- 0/1 Knapsack: Given a collection of items with weights and benefits, is there a subset of weight at most W and benefit at least K?
  - NP-complete by reduction from SUBSET-SUM
- Hamiltonian-Cycle: Given an graph G, is there a cycle in G that visits each vertex exactly once?
  - NP-complete by reduction from VERTEX-COVER
- Traveling Saleperson Tour: Given a complete weighted graph G, is there a cycle that visits each vertex and has total cost at most K?
  - NP-complete by reduction from Hamiltonian-Cycle.