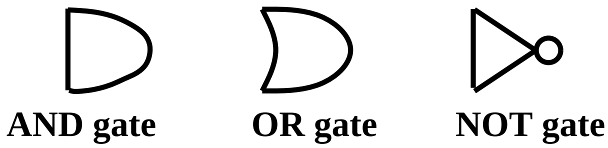
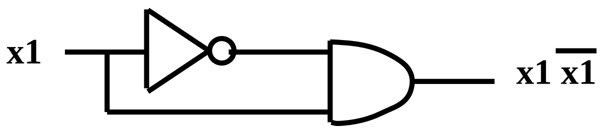
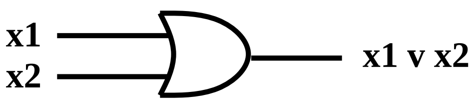
Circuit Satisfiability



Boolean Combinational Circuits



not satisfiable for any $x_1 \in \{0,1\}$



satisfiable for $x_1 x_2 = 01$, 10, or 11, thus satisfiable

Circuit-Satisfiability Problem

Given a boolean combination circuit composed of AND, OR, and NOT gates, is it satisfiable?

CIRCUIT-SAT = $\{\langle$C$\rangle$ | C is a satisfiable boolean combinational circuit$\}$

where $\langle$C$\rangle$ is a binary-string encoding of the circuit (e.g., as a graph)

Determining membership in CIRCUIT-SAT would require checking the 2^k possible binary assignments to the k inputs of a circuit.

There is strong evidence that CIRCUIT-SAT $\notin$ P.

CIRCUIT-SAT is NP-Complete

Proof:

1. CIRCUIT-SAT $\in$ NP
   Proof: Can verify an input assignment satisfies a circuit by computing the output of a finite number of gates, one of which will be the output of the circuit. This can be done in polynomial time. Thus, by definition of NP, CIRCUIT-SAT $\in$ NP.

2. CIRCUIT-SAT $\in$ NP-Hard
   I.e., L $\leq_P$ CIRCUIT-SAT for every L $\in$ NP

   Proof: Complex

   Show that any problem in NP can be computed using a boolean combination circuit (i.e., a computer).

   This circuit has a polynomial number of elements and can be constructed in polynomial time. Thus, L $\leq_P$ CIRCUIT-SAT for all L $\in$ NP.

   Thus, CIRCUIT-SAT $\in$ NP-Hard.

CIRCUIT-SAT is NP-Complete
Proof by Cook, 1971

NP-Completeness Proofs

Lemma 36.8

If L is a language such that L' $\leq_P$ L for some L' $\in$ NPC, then L is NP-Hard. If also L $\in$ NP, then L $\in$ NPC.

Strategy for proving L $\in$ NPC

1. Prove L $\in$ NP (poly-time verifiable)
2. Select L' $\in$ NPC
3. Describe poly-time algorithm computing a function f that maps instances of L' to instances of L
4. Prove that x $\in$ L' iff f(x) $\in$ L for all x $\in$ {0,1}*.
   Note: Showing L' $\leq_P$ spec(L) implies L' $\leq_P$ L.

Example: Boolean Formula Satisfiability

SAT: Given a Boolean formula in Conjunctive Normal Form (C.N.F.), does there exist a satisfying assignment?

SAT = { B : B is a boolean formula in CNF that is satisfiable by some truth assignment to its variables}

A CNF formula is a boolean formula composed of variables and connectives AND, OR, NOT, IMPLIES, and EQUIV, possibly separated by parentheses.

Let $B = (u_1 \vee \overline{u_2}) \wedge (\overline{u_1} \vee u_2)$.

This is an *instance* of SAT for which the answer is "yes". A satisfying truth assignment is given by $t(u_1) = t(u_2) = T$.

On the other hand, the expression $u_1 \wedge \overline{u_1}$ is an instance of SAT for which the answer is "no".

SAT $\in$ NPC

Proof:

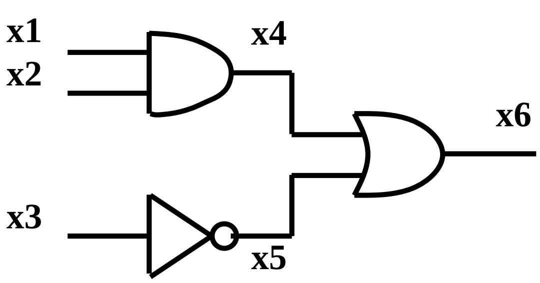
1. SAT $\in$ NP
   Replace each variable with 0 or 1 as specified by the certificate and evaluate (poly-time).

2. Select L' = CIRCUIT-SAT

3. Reduction from CIRCUIT-SAT to SAT.
   Straight-forward technique of computing the formula of each gate output as a combination of the input formulae may cause exponential instantiations of a variable as outputs are copied to multiple inputs.
   Instead, let each gate output be a variable.
   AND the output variable with expressions for each gate describing the equivalence between the gate's output and input variables.

Example

Circuit C



Formula

$$\phi = x_6 \wedge (x_4 \leftrightarrow (x_1 \wedge x_2)) \wedge (x_5 \leftrightarrow \neg x_3) \wedge (x_6 \leftrightarrow (x_4 \vee x_5))$$

Constructing this formula takes polynomial time.

SAT ∈ NPC

4. Prove x ∈ L' iff f(x) ∈ L, where L' is CIRCUIT-SAT, L is SAT, and f is the construction above.

   x ∈ CIRCUIT-SAT → f(x) ∈ SAT

   If C has a satisfying assignment, then each wire is well-defined and the output is 1.

   Therefore, each conjunct of ϕ is 1, and ϕ will evaluate to 1.

A satisfying assignment to ϕ yields a valid circuit C whose output is 1.

CNF Satisfiability

When the full power of SAT is not required to prove a language is in NPC, 3-CNF provides a more constrained alternative.

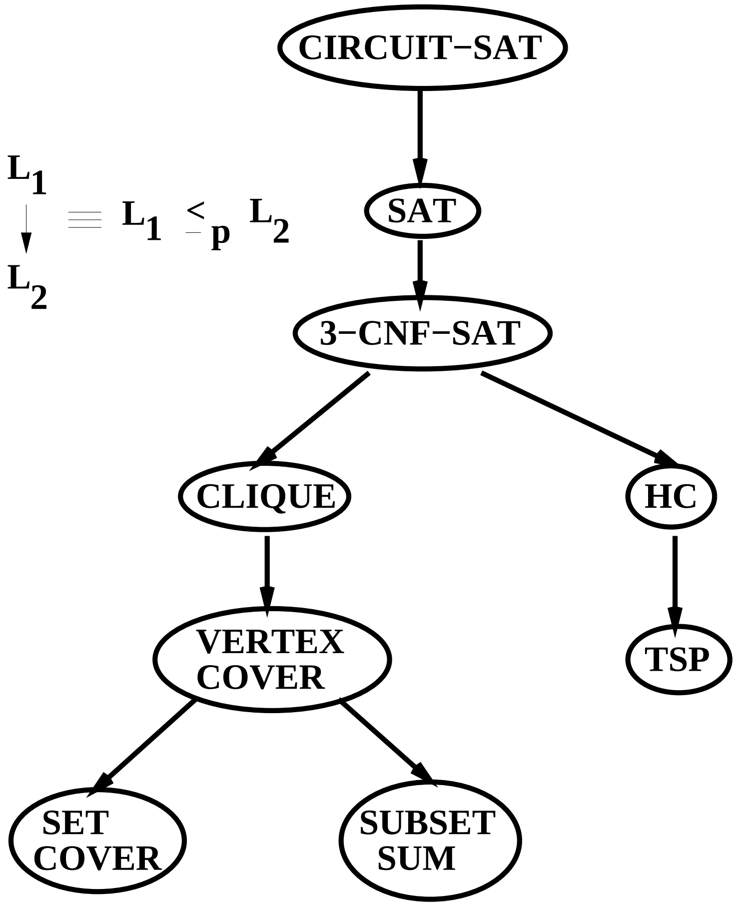
k-CNF (Conjunctive Normal Form) is a formula having a conjunction of **clauses**, where each clause is a disjunction of exactly k literals (variable or its negation).

Example 3-CNF: $(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee \neg x_2 \vee x_4)$

Theorem 36.10

3-CNF-SAT ∈ NPC

Some NP-Complete Problems

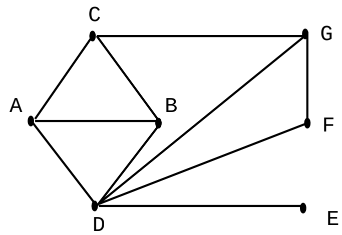


Clique Problem

CLIQUE: Given graph $G = (V, E)$, find largest subset $V' \subseteq V$ such that $\forall u, v \in V'$, $(u, v) \in E$.
I.e., V' forms a complete subgraph of G (usually want largest).

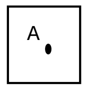
$$\text{CLIQUE} = \{\langle G, k\rangle : \exists V' \subseteq V \text{ of size} \geq k \text{ and } \forall u, v \in V', (u, v) \in E\}$$

Example

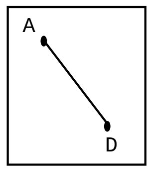


Original Graph G

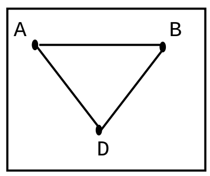
K = 1



K = 2



K = 3



K = 4 NONE

The running time of the CLIQUE algorithm is $\Omega(k^2 \binom{\mid V \mid}{k})$

Theorem 36.11: CLIQUE $\in$ NPC

1. CLIQUE $\in$ NP

To show CLIQUE in NP, we use set V' of vertices as a certificate.

Verifying is polynomial time, check whether for every pair u,v in V', the edge is in E ($\mid V' \mid^2$ pairs).

2. L' = 3-CNF-SAT

3. 3-CNF-SAT $\leq_P$ CLIQUE

Start with instance of 3-CNF-SAT (also called 3CNF).

Let f be 3CNF with k clauses, $(C_{11} \vee C_{12} \vee C_{13}) \wedge (C_{21} \vee C_{22} \vee C_{23}) \wedge (C_{31} \vee C_{32} \vee C_{33}) \wedge \ldots (C_{k1} \vee C_{k2} \vee C_{k3})$.

For r = 1,2,..,k, each clause has three distinct literals l_1^r, l_2^r, l_3^r.

Construct a graph G such that f is satisfiable iff G has a clique of size k.

For each C_r in f, put triple of vertices v_1^r, v_2^r, v_3^r in V.

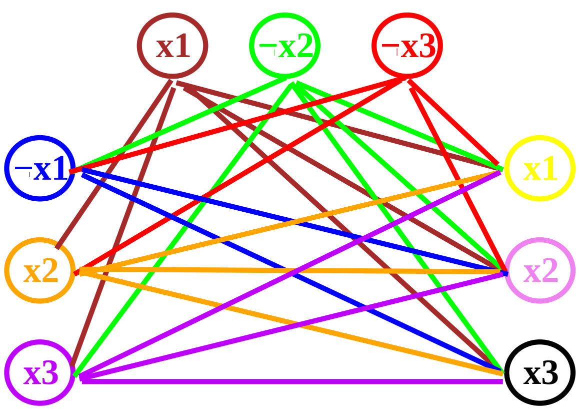
Add edge (v_i^r, v_j^s) if

1. v_i^r and v_j^s are in different triples (r $\neq$ s), and
2. their corresponding literals are consistent (l_i^r is not the negation of l_j^s).

Proof (cont.)

This graph is constructed in polynomial time.

If f = $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$, then the graph is



Proof (cont.)

4. Is this a reduction?

Suppose f has a satisfying assignment. Then each clause C_r contains at least one literal l_i^r that is assigned 1, and each such literal corresponds to a vertex v_i^r.

Picking one such "true" literal from each clause yields a set V' of k vertices.

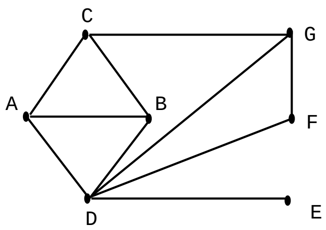
Is V' a clique? For any two vertices v_i^r, v_j^s, r ≠ s, the corresponding literals are mapped to 1 by the satisfying assignment and thus the literals cannot be complements. By the construction of G, the edge (v_i^r, v_j^s) belongs in E.

Proving the other direction, if G has a clique V' of size k, no edges in G connect vertices in the same triple, so V' contains exactly one vertex per triple. Assign a 1 to each literal l_i^r such that v_i^r in V' without fear of assigning 1 to a literal and its complement. Each clause is satisfied, and f is satisfied.

Vertex-Cover Problem

A **vertex cover** of an undirected graph G = (V, E) is a subset V' ⊆ V such that each edge in E is incident on at least one of the vertices in V'.

VC = $\{\langle G, k\rangle : G = (V, E)$ is a graph, and $\exists V' \subseteq V$ such that $\mid V' \mid \leq k$ and $\forall (u, v) \in E$, either $u \in V'$ or $v \in V'$ (or both) $\}$



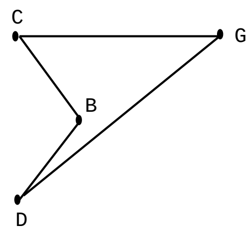
Original Graph G

K = 1 NONE

K = 2 NONE

K = 3 NONE

K = 4



VERTEX COVER

$$\text{VC} = \{\langle \text{G, k} \rangle \mid \text{graph G has vertex cover of size k}\}$$

Theorem 36.12: VC $\in$ NPC

Proof Sketch:

1. VC $\in$ NP
   Given V', check |V'| = k, and for each edge (u,v) $\in$ E, check that either u $\in$ V' or v $\in$ V'.

2. L' = CLIQUE

3. CLIQUE $\leq_P$ VC
   If graph G = (V, E) has clique V', then graph $\overline{G}$ has vertex cover V - V'.
   $\overline{G}$ = (V,$\overline{E}$) is the *complement* of G = (V,E), where $\overline{E} = \{(u, v) | (u, v) \notin$

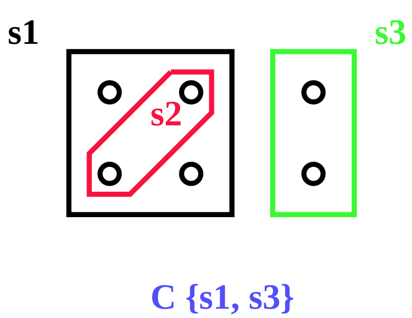
$E\}$
Reduction: G $\rightarrow \overline{G}$ (poly-time)

4. x $\in$ CLIQUE(G) = V' $\longrightarrow$ f(x) $\in$ VC($\overline{G}$) = V - V' ($|V'| = k$)
   Every edge (u,v) $\in \overline{E}$ implies (u,v) $\notin$ E, thus at least one of u and v $\notin$ V'. Thus, at least one of u,v belongs to V - V', which means edge (u,v) is covered by V - V'. Similar argument for other direction.

Set-Covering Problem

Given a finite set X and a family F of subsets of X, X = $\cup_{S \in F} S$, find a minimum-size subset C $\subseteq$ F whose members cover all of X.

SC = $\{\langle$X, F, k$\rangle$ | there exists a set cover C $\subseteq$ F covering X with size $\leq$ k$\}$



Theorem: SC $\in$ NPC

Proof:

1. Given C, check that all elements of X are members of some set in C and that $| C | \leq k$.

2. L' = VC

3. Given $\langle$G, k$\rangle$ $\in$ VC, define F such that each element of F is a subset for a vertex v in G containing v and all vertices reachable by an edge from v.
   Let X = V. Then $\langle$X,F, k$\rangle$ $\in$ SC.

4. If C is the vertex cover of $\langle$G, k$\rangle$ $\in$ VC, then every vertex u in G is incident from an edge (u,v) where either u $\in$ C or v $\in$ C. Thus all vertices will appear in some set in F, and the sets in F corresponding to the vertices in C make up the set covering of $\langle$X, F, k$\rangle$ $\in$ SC.

Subset Sum Problem

SUBSET-SUM = {$\langle$S, t$\rangle$ | there exists S' $\subseteq$ S $\subset$ N such that $\Sigma_{s \in S'} s$ = t $\in$ N},
N = set of natural numbers

Theorem 36.13: SUBSET-SUM $\in$ NPC

Proof:

1. SUBSET-SUM $\in$ NP. Just add up elements of S' and compare sum to t.

2. L' = VC

3. VC $\leq_P$ SUBSET-SUM

4. x $\in$ VC $\leftrightarrow$ f(x) $\in$ SS
   Proof is complex.

Hamiltonian Cycle Problem

A Hamiltonian Cycle is a simple cycle in a graph going through each vertex exactly once.

HC = { ⟨ G ⟩| G has a Hamiltonian cycle}

HC ∈ NPC

Proof:

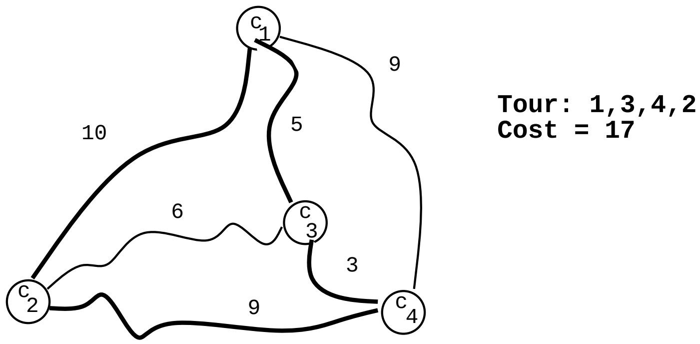
1. Done earlier.
2. L' = 3-CNF-SAT
3. 3-CNF-SAT $\leq_P$ HC
4. x ∈ 3-CNF-SAT ↔ f(x) ∈ HC
   Proof is complex.

Traveling Salesman Problem

Given a complete graph with weights on the edges, find a cycle of least total weight that visits each vertex exactly once.

Decision Problem:

TSP = $\{\langle G, k\rangle |$ G is a complete graph with weights on edges that contains a cycle of total weight $\leq k$ visiting each vertex exactly once}



TRAVELING SALESMAN PROBLEM

Theorem 36.15: TSP $\in$ NPC

Variant of proof in textbook.

Proof sketch:

1. TSP $\in$ NP
   Given a tour, check that each vertex is visited exactly once and the sum of costs $\leq$ k
2. L' = HC
3. HC $\leq_P$ TSP
   Given graph $G = (V, E)$, transformation f outputs complete graph with vertices V.
   Weights of edges = 1 if $e \in E$, or (—V— + 1) if $e \notin E$
   Also outputs the number —V—.
   f is clearly implementable in polynomial time.
4. Then there exists a tour in this complete graph of size $\leq |V|$ iff there exists a Hamiltonian Cycle in original graph.

Click mouse to advance to next frame.

Partition Problem

Given a finite set A and a "size" $s(a) \in Z^+$ for each $a \in A$, find a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in (A-A')} s(a)$$

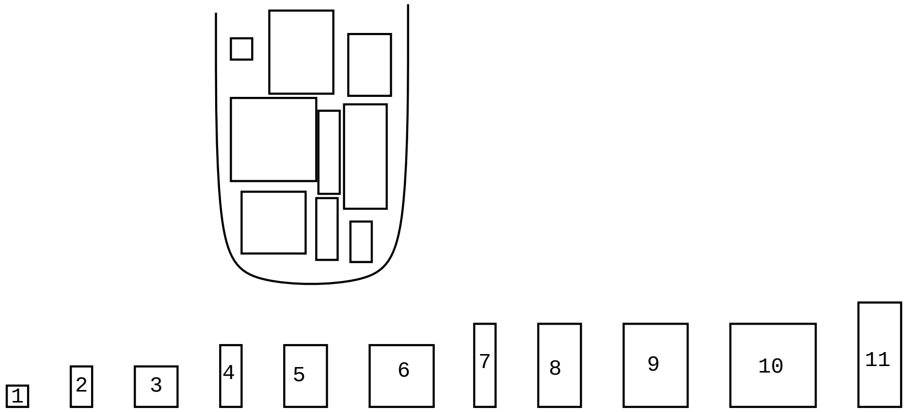
PARTITION = $\{\langle A, s(a) \rangle : \exists A' \subseteq A$ such that the sums of A' and $(A - A')$ are equal$\}$

For example, if $A = \{a = 1, b = 2, c = 3, d = 4, e = 5, f = 7, g = 8\}$, then one possible partition is $A' = \{a, b, c, d, e\}$ and $A - A' = \{f, g\}$. The sum of both subsets is 15.

Knapsack Problem

KNAPSACK: Given a finite set U, a "size" $s(u) \in Z^+$ and a "value" $v(u) \in Z^+$ for each $u \in U$, a size constraint $B \in Z^+$, and a value goal $K \in Z^+$, is there a subset $U' \subseteq U$ such that $\Sigma_{u \in U'} s(u) \leq B$ and $\Sigma_{u \in U'} v(u) \geq K$?

This can be seen as a knapsack, which has a size limit for the objects, as in the picture below.



KNAPSACK PROBLEM

The goal is to pick a collection of objects that will fit in the knapsack and whose total value is at least K (K is input)

KNAPSACK = $\{(U, s, v, B, K)$: $\exists$ subset U' of U such that the sum of s values is at most B, and the sum of v values is at least $K\}$

KNAPSACK is NP-Complete

Proof: We will show that the KNAPSACK problem is NP-complete by polynomial-time restricting it in a way that makes it equal to the PARTITION problem, or PARTITION $\leq_P$ spec(KNAPSACK).

We can restrict KNAPSACK to PARTITION by allowing only instances in which $s(u) = v(u)$ for all $u \in U$ and $B = K = 1/2 \sum_{u \in U} s(u)$.

NP-Complete Problems