NP-Completeness

Definition SAT = { $\langle\phi\rangle$ | ϕ is a satisfiable Boolean formula }

Example $((x \wedge \neg y) \vee \neg x)$ is satisfiable by the assignment x = 1 and y = 0.

Theorem SAT is NP-complete [proof next week]

Definition 3SAT = { $\langle\phi\rangle$ | ϕ is a satisfiable 3CNF formula }

A 3CNF formula has the form $(s_{11} \vee s_{12} \vee s_{13}) \wedge (s_{21} \vee s_{22} \vee s_{23}) \wedge \ldots$
where literals s_{ij} are Boolean variables or negated Boolean variables.

Theorem 3CNF is NP-complete [proof next week]

Theorem $L \in$ NPC

Proof

I. Show that $L \in$ NP
   - A. Certificate: [Describe certificate]
   - B. Algorithm: [Construct verification algorithm V]
   - C. Runtime: [Argue that V runs in polynomial time]

II. Show that $S \leq_P L$ [Find a reduction from S to L, where $S \in$ NPC]
   - A. Algorithm: [Construct reduction algorithm M]
     1. Input: [Give form of input]
     2. Output: [Give form of output]
   - B. Reduction: [Show that M is a reduction]
     1. ($\Rightarrow$): [$x \in S \Rightarrow M(x) \in L$]
     2. ($\Leftarrow$): [$x \notin S \Rightarrow M(x) \notin L$]
   - C. Runtime: [Argue that M runs in polynomial time]

Clique

A **clique** is a complete subgraph of undirected graph G.

CLIQUE = { $\langle G, k\rangle$: G is a graph with a clique of size k }

Theorem CLIQUE $\in$ NPC

Proof
(1) CLIQUE $\in$ NP

Certificate: A subset $V' \subseteq V[G]$ of vertices that forms a clique

Verification algorithm V:
V = "On input $\langle G, V'\rangle$:
for each pair of vertices u, v $\in$ V′
  check if u, v $\in$ V[G]
  check if (u, v) $\in$ E
if both checks succeed **then** accept **else** reject

Runtime: V runs in O(E[G]) time

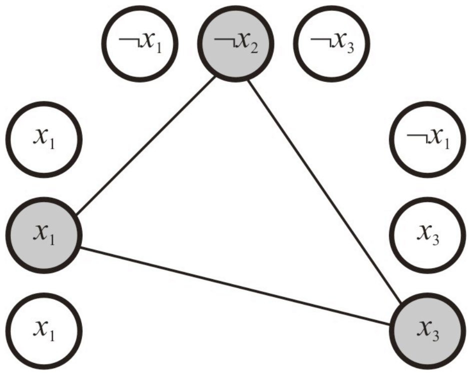
(2) CLIQUE is NP-hard (Show 3-CNF-SAT $\leq_P$ CLIQUE)

Let $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ be a 3-CNF formula where $C_r = (s_1^r \vee s_2^r \vee s_3^r)$.

Reduction algorithm M: $\langle \phi \rangle \rightarrow \langle G, k\rangle$

Construct G as follows:
for each clause $C_r = (s_1^r \vee s_2^r \vee s_3^r)$ in ϕ
  add v_1^r, v_2^r, v_3^r to V[G]
for each pair of vertices v_i^r and v_j^s in G
  place an edge between them if both:
    (1) v_i^r and v_j^s are in different triples ($r \neq s$), **and**
    (2) their corresponding literals are consistent ($s_i^r \neq \neg s_i^s$)
Halt with G on tape.

Example
$\phi = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_3)$



[change the middle clause to $(\neg x_1 \vee \neg x_1 \vee \neg x_3)$ and you won't have a clique of size 3]

Runtime: M runs in $O(k) + O(k^2) = O(k^2)$ time.

Correctness of M: $\langle \phi \rangle \in$ 3-CNF-SAT $\Leftrightarrow \langle G, k \rangle \in$ CLIQUE

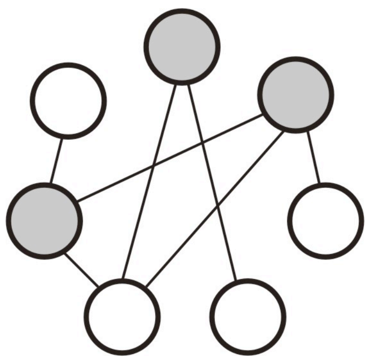
($\Rightarrow$)
ϕ has a satisfying assignment.
Each clause C_r contains at least one "true" literal.
Let S be a set containing one "true" literal from each clause.
Claim: the set of vertices V′ in G that corresponds to S is a clique.
Let u and v be arbitrary vertices in V′
We know that:
(1) their corresponding literals come from different clauses, and
(2) their corresponding literals are consistent since they both map to "true"
Therefore, there must be an edge between the vertices
Since u and v are arbitrary,
  for all u, v in V′, (u, v) is in G
V′ is a clique of size k for G

($\Leftarrow$)
G has a clique V′ of size k
Let S be the set of literals corresponding to V′
We know that:
(1) S contains exactly one literal from each clause in ϕ
  (there are k clauses and k vertices, and there cannot be
  more than one vertex from the same triple)
(2) All the literals in S are consistent.
Therefore, we can safely assign "true" to all the literals in S
The assignment makes all clauses true,
  which in turn makes ϕ true.
ϕ has a satisfying assignment.

Vertex cover

Let G = (V, E) be an undirected graph.
A **vertex cover** is a subset V' $\subseteq$ V of vertices such that
if (u, v) $\in$ E then u $\in$ V' or v $\in$ V' (or both)

Example



VERTEX-COVER = { $\langle G, k \rangle$: graph G has a vertex cover of size k }

Theorem VERTEX-COVER $\in$ NPC

Proof
(1) VERTEX-COVER $\in$ NP
certificate: subset of V' $\subseteq$ V of vertices
algorithm:
check of |V'| = k
for each edge (u, v) in E
  check if u $\in$ V or v $\in$ V
poly-time: size check takes O(V); edge check takes O(E)

(2) VERTEX-COVER is NP-hard
(we show CLIQUE $\leq_P$ VERTEX-COVER)

(A) Algorithm F: $\langle G, k \rangle \rightarrow \langle G', k' \rangle$
construct G' as G* = (V, E*), E* is the compliment of E [set of all edges not in E]
k' = |V| – k

(B) F runs in poly time
O(V^2) time to change 0 $\leftrightarrow$ 1 in adjacency matrix.

(C) F computes a reduction
$\langle G, k \rangle \in$ CLIQUE $\Leftrightarrow \langle G^*, |V| - k \rangle \in$ VERTEX-COVER

($\Rightarrow$)
G has clique V' of size k
Let (u, v) be an arbitrary edge in G*
Since (u, v) is not an edge in G,
  at least one of u, v must be outside clique V'
Therefore, at least one of u, v must be in V – V'
V – V' covers (u, v)
Since (u, v) is arbitrary, V – V' covers all (u, v) in G*
V – V' is a vertex cover for G* – it has size |V| – k

($\Leftarrow$)
G* has vertex cover V' with size |V| – k
Let both u and v be arbitrary vertices in V – V'
Suppose (u, v) is in G*
Then at least one of u, v must be in vertex cover V' ($\Rightarrow\Leftarrow$)
So (u, v) is in G
Since u and v are arbitrary,
  for all u, v in V – V', (u, v) is in G
Therefore, V – V' is a clique of size k for G

[return to the picture – the complement graph has a clique of size four]

Hamiltonian paths and cycles

HAMPATH = { $\langle$G, s, t$\rangle$: G has a Hamiltonian path from s to t }

<u>Theorem</u> HAMPATH $\in$ NPC

[see the text for a reduction VERTEX-COVER $\leq_P$ HAMPATH]

HAMCYCLE = { $\langle$G$\rangle$: G has a Hamiltonian cycle }

<u>Theorem</u> HAMCYCLE $\in$ NPC

HAMCYCLE $\in$ NP
[certificate is a Hamiltonian cycle, similar to proof for HAMPATH $\in$ NP]

HAMCYCLE is NP-hard (Show HAMPATH $\leq_P$ HAMCYCLE)

[try adding an edge from t to s $\Rightarrow$ doesn't work: You can have $\langle$G, s, t$\rangle$ $\notin$ HAMPATH and $\langle$G$'\rangle$ $\in$ HAMCYCLE]

[try adding an edge and eliminating edges that are not part of the Hamiltonian path; doesn't work: you need to know which edges are in the path which may take exponential time]

[Add a vertex k, add edge (t, k), and add edge (k, s). this works!]

Traveling salesman problem (TSP)

G = (V, E) is a *complete* graph. [all vertices have edges between them]
All edges have an integer cost.
Problem: Find a min-cost tour (ham-cycle)

TSP = { $\langle G, c, k \rangle$: G = (V, E) is a complete graph,
c is a cost function $V \times V \to \mathbf{Z}$
$k \in \mathbf{Z}$, and
G has a tour with cost $\leq k$ }

Theorem TSP $\in$ NPC

Proof
(1) TSP $\in$ NP
certificate: sequence of |V| vertices
algorithm:
check that vertices don't occur more than once
sum up edge cost and check that it is less than k
poly time:
O(V) to check dup vertices
O(V) to get edge costs

(2) TSP is NP-hard
(we show HAM-CYCLE $\leq_P$ TSP)

(A) algorithm F: $\langle G \rangle \to \langle G', c, k \rangle$
1. construct G' = (V, E') where G' is a complete graph.
2. c(i, j) = 0 if $(i, j) \in E$; c(i, j) = 1 if $(i, j) \notin E$
3. k = 0

(B) F runs in poly time
constructing G' takes $O(V^2)$ time
constructing c takes $O(V^2)$ time

(C) F computes a reduction
$\langle G \rangle \in$ HAM-CYCLE $\Leftrightarrow \langle G', c, 0 \rangle \in$ TSP

($\Rightarrow$)

G has hamiltonian cycle h
All edges in the cycle are also in G
These edges will all have cost 0 in G'
Therefore G' has a tour with cost ≤ 0

($\Leftarrow$)
G' has a tour with cost ≤ 0.
All edges in G' have cost 0 or 1
Therefore, all edges in the tour must have cost 0
So all edges in the tour must also be edges in G
G has a hamiltonian cycle

<u>Example</u> Show that 3-COLOR $\leq$P 3-COLOR-PLUS.

3-COLOR = { $\langle G \rangle$: G is 3-colorable (vertices can be colored with 3 colors; no two adjacent vertices have the same color) }

3-COLOR-PLUS = { $\langle G \rangle$: G is 3-colorable and every vertex is adjacent to at least one vertex colored with each of the other two colors }

Reduction algorithm M: $\langle G \rangle \rightarrow \langle G' \rangle$
M = "On input $\langle G \rangle$:
$G' = G$
for each v in V[G]
  add two new vertices u and w to V[G′]
  add three new edges (v, u), (v, w), and (u, w) to E[G′]
output G′

Runtime
M takes time O(V)

Reduction: ($\langle G \rangle \in$ 3-COLOR $\Leftrightarrow$ M(G) $\in$ 3-COLOR-PLUS)

($\Rightarrow$) Assume G $\in$ 3-COLOR
$\Rightarrow$ any vertex v in G must be colored with one of three colors.
$\Rightarrow$ the corresponding vertex v′ in G′ can be colored with the same color as v
$\Rightarrow$ the two neighbors of v added in algorithm F can be colored with the two other colors.
$\Rightarrow$ G′ is 3-colorable and each vertex is adjacent to a vertex of each of the other 2 colors.
$\Rightarrow$ G′ $\in$ 3-COLOR-PLUS

($\Leftarrow$) Assume G′ $\in$ 3-COLOR-PLUS
G is a subgraph of G′, so if G′ is 3-colorable, G must also be 3-colorable

Example Show that the subgraph isomorphism problem in NP-complete.

SUB-ISO = { ⟨G1, G2⟩ | G1 is isomorphic to a subgraph of G2 }

Step 1. Show SUB-ISO ∈ NP

certificate: An isomorphic mapping *m* from the vertices of G1 to a subset of the vertices of G2.

Verification algorithm V
V = "On input ⟨⟨G1, G2⟩, y⟩:
for each vertex v in G1
    for each vertex u in G1
        if (u, v) is in G1 and (f(u), f(v)) is not in G2 **then** return false
        if (u, v) is not in G1 and (f(u), f(v)) is in G2 **then** return true
return true
Runtime: V takes O(V2) time

Step II. Show CLIQUE $\leq_P$ SUB-ISO

Reduction algorithm M: ⟨G, k⟩ → ⟨G1, G2⟩
M = "On input ⟨G, k⟩:
Construct G2 = G
Construct G1 = complete graph with k vertices

Runtime of M: Polynomial in E + V

Correctness of M: ⟨G, k⟩ ∈ CLIQUE ⇔ ⟨G1, G2⟩ ∈ SUB-ISO
Assume ⟨G, k⟩ ∈ CLIQUE
⇔ G has a clique of size k
⇔ G has a subgraph of size k
⇔ G2 has a subgraph of size k isomorphic to G1
⇔ ⟨G1, G2⟩ ∈ SUB-ISO

Definition CLIQUE = { ⟨G, k⟩ | G is an undirected graph with a clique (a complete subgraph) of k nodes }

Definition VERTEX-COVER = { ⟨G, k⟩ | G is an undirected graph with a vertex cover (a set of nodes that touches every edge of G) of k nodes }

Definition HAMPATH = { ⟨G, s, t⟩ | G has a path from s to t that goes through each node exactly once }

Definition HAMCYCLE = { ⟨G ⟩ | G contains a cycle that goes through each node exactly once }

Assume 3SAT is NP-Complete