Computational Intractability (Ch 8,9)

- **Problem Classes by Computational Requirements**
- **Polynomial Time Reduction**
- **P and NP Problems**
- **NP-completeness**
- **NP-complete Problems**
- **NP-hard and co-NP**
- **PSPACE, Class of Problems beyond NP**

The lecture notes/slides are adapted from those associated with the text book by J. Kleinberg and E. Tardos.

Problem Classes by Computational Requirements

- Three types of problems
  1. Easy problems, polynomial time algorithms are known.
  2. Difficult problems, no polynomial time algorithm proved.
  3. Problems in ”grey zone”: no polynomial time algorithm known, no proof that polynomial time algorithm does not exist.
- A large class of Type 3 problems have the equivalent property that if one of the problems can be solved in polynomial time, then every problem in the class can be solved in polynomial time.
- Polynomial time reduction is a tool to show the equivalent property.

Polynomial Time Reduction

- A problem Y is poly-time reducible to a problem X, denoted by $Y \leq_P X$, if there is an algorithm that solves any instance of Y using polynomial many primitive operations and polynomial many calls to an oracle which solves X.
- If $Y \leq_P X$ and X can be solved in polynomial time, then Y can be solved in polynomial time.

  This is useful for designing algorithms for Y.
- If $Y \leq_P X$ and Y can not be solved in polynomial time, then X can not be solved in polynomial time.

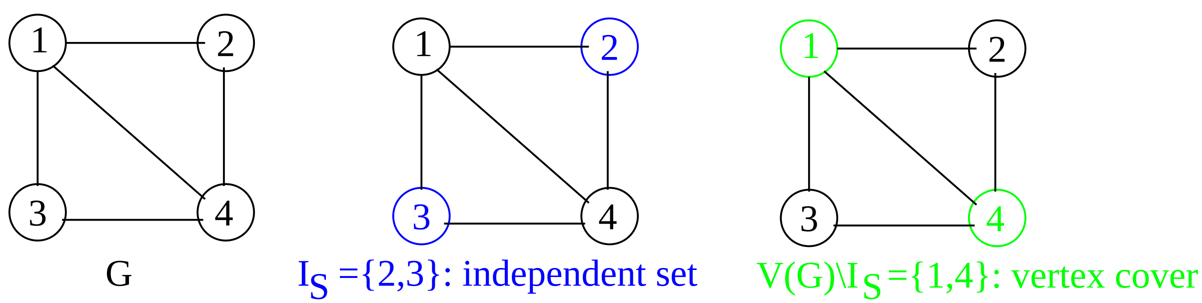
  This is commonly used to show the intractability of X.
- If $Y \leq_P X$ and $X \leq_P Y$ (denoted by $X \equiv_P Y$), then Y can be solved in polynomial time iff X can be solved in polynomial time.

- **Independent set problem: A set S of nodes in a graph G is independent if no two nodes in S is connected by an edge in G. Given G and integer $k > 0$, whether G has an independent set of size k or not.**

- **Vertex cover problem: A set S of nodes in a graph G is a vertex cover if each edge in G has at least one end node in S. Given G and integer $k > 0$, whether G has a vertex cover of size k or not.**

- **A set S of nodes in graph G is an independent set iff $V(G) \setminus S$ is a vertex cover of G.**

- **Independent-Set$\leq_P$Vertex-Cover and Vertex-Cover$\leq_P$Independent-Set.**



- **Satisfiability (SAT) problem**
  - **A Boolean variable is a variable takes a value from $\{0, 1\}$.**
  - **A literal is a Boolean variable x or its negation $\overline{x}$.**
  - **A clause is a disjunction of literals; the clause has size k if it has k literals.**
  - **A CNF (conjunctive normal form) is a conjunction of clauses; a k-CNF is a CNF with each clause of size at most k (or exact k).**

$$\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee x_3)$$

  - **A truth assignment for a CNF is a function $\sigma : X \rightarrow \{0, 1\}$ for each x in the CNF. The assignment satisfies a clause C if C has value 1, and satisfies a CNF if every clause in the CNF has value 1.**

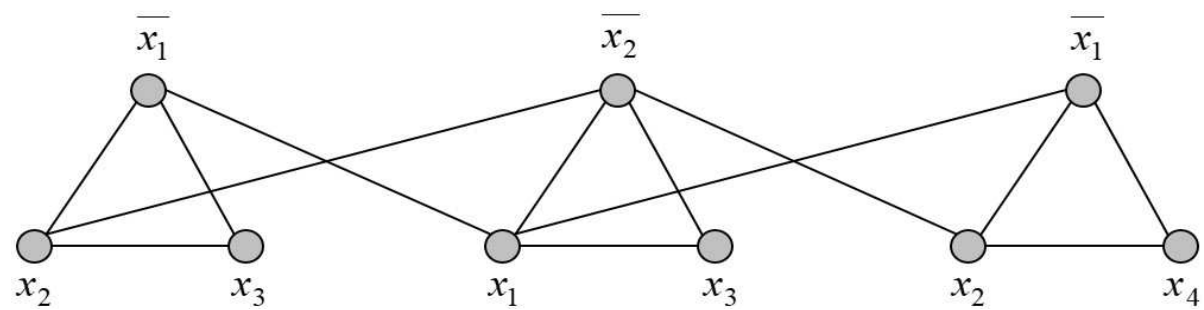
  Given a CNF Φ, is Φ satisfiable (is there a truth assignment satisfying Φ)?

- **k-satisfiability (k-SAT) problem: Given a k-CNF Φ, is Φ satisfiable?**

Theorem. 3-SAT$\leq_P$Independent-Set.

Proof. For a 3-CNF instance $\Phi = C_1..C_k$, let v_{i1}, v_{i2}, v_{i3} be the three literals in clause C_i. Two literals v_{ij} and $v_{i'j'}$, $1 \leq i, i' \leq k, 1 \leq j, j' \leq 3$, are conflict if one of them is variable x and the other is the negation $\overline{x}$ of x. We construct a graph G with $V(G) = \{v_{i1}, v_{i2}, v_{i3} | 1 \leq i \leq k\}$ and

$$E(G) = \{\{v_{ij}, v_{ij'}\} | v_{ij}, v_{ij'} \text{ in } C_i\} \cup \{\{v_{ij}, v_{i'j'}\} | v_{ij} \text{ and } v_{i'j'} \text{ are conflict}\}.$$



$$\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee x_4)$$

Claim: Φ is satisfiable iff G has an independent set of size k.

At most one node in each clause can be in an independent set, so the size of such a set is at most k. Assume there is an assignment satisfying Φ. Then there is a satisfied literal in each clause. The set consists of one satisfied literal from each clause gives an independent set of size k.

Assume there is an independent set S of size k. There is an assignment satisfying all literals in S. Since each clause has one literal in S, the assignment satisfies Φ. $\square$

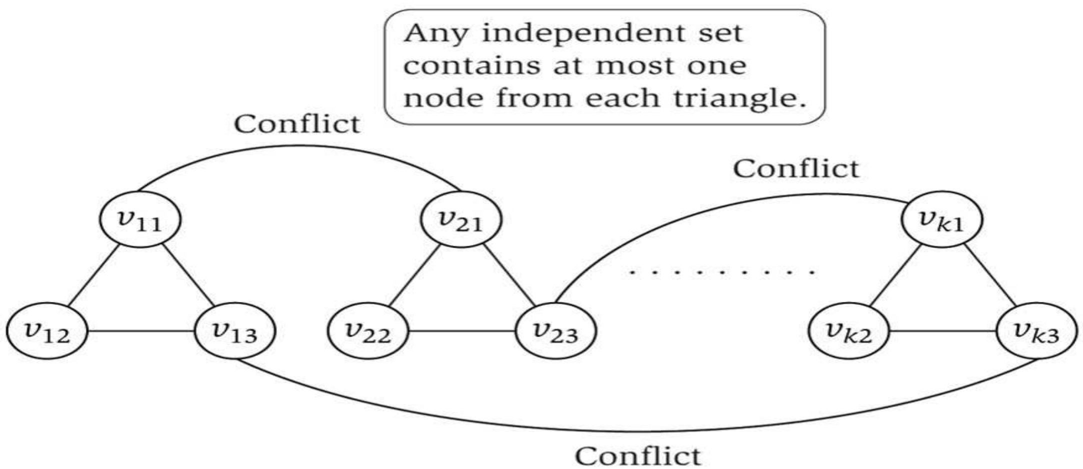


Figure 8.3 The reduction from 3-SAT to Independent Set.

P and NP Problems

- **Decision problem**
  - **For a set X of strings and an instance (string) s, decide if $s \in X$ or not. Example, $X = \{f | f$ is a satisfiable CNF$\}$; given a CNF s, decide if $s \in X$ or not (if s is satisfiable or not).**
  - **Algorithm A solves problem X if $A(s) =$YES for $s \in X$ and $A(s) =$NO for $s \notin X$.**
  - **Algorithm A runs in polynomial time if for every s, $A(s)$ terminates in polynomial time in the length of s.**
- **P problems: Set of decision problems for which there exists a poly-time algorithm.**

- **Algorithm C is a certifier for decision problem X if for every string s, $s \in X$ iff there exists a string t (certificate) s.t. $C(s,t) =$YES.**
- **NP problems: Set of decision problems for which there exists a certifier C:**
  - **$C(s,t)$ is a poly-time algorithm,**
  - **certificate t has size $|t| \leq \mathrm{Poly}(|s|)$.**
- **COMPOSITE, an NP problem example: if an integer s is composite or not?**
  - **Certificate, a nontrivial factor t of s. Such a t ($1 < t < s$) exists iff s is composite.**
  - **Certifier: check if $t > 1$ and $t < s$; if yes, then check if t divides s.**

  Example, $s = 437,669$; $t = 541$ or $t = 809$; $s = 437,669 = 541 \times 809$.

  COMPOSITE is in NP

- **SAT: given a CNF formula Φ, is Φ satisfiable?**
  - **Certificate, an assignment of truth values to the n Boolean variables.**
  - **Certifier, check that each clause has at least one true literal.**

  Example, instance s:

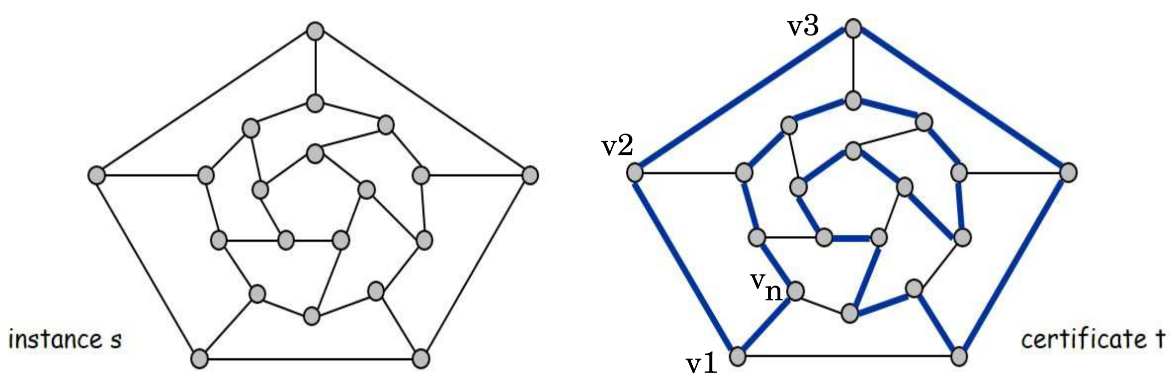
$$(\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3),$$

  certificate t: $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1$.

  SAT is in NP.

- **Hamiltonian cycle problem HAM-CYCLE: given a graph G, is there a simple cycle C that contains every node of G?**
  - **Certificate, a permutation of the n nodes of G.**
  - **certifier, check that the permutation contains each node of G exactly once and there is an edge between each pair of adjacent nodes in the permutation.**

HAM-CYCLE is in NP.



NP-Completeness

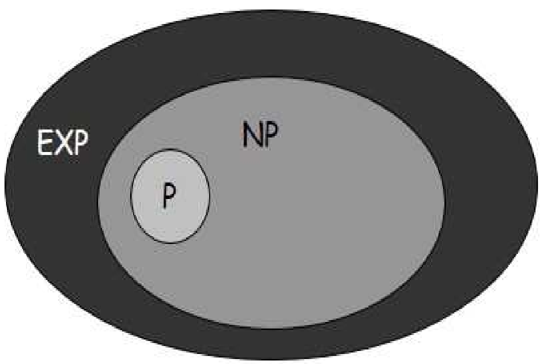
- **P is the class of problems for which there is a poly-time algorithm.**
- **NP is the class of problems for which there is a poly-time certifier.**
- **P$\subseteq$NP.**

  Proof. Certifier is a solution algorithm runs with an empty certificate. $\square$

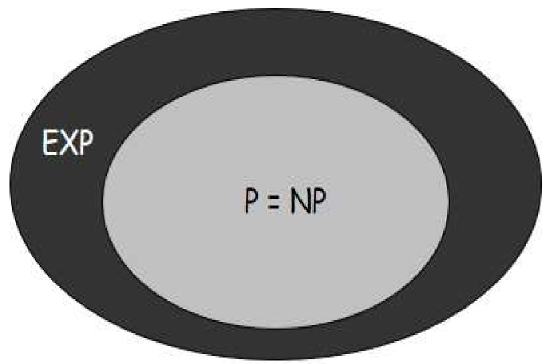
- **Problem X is NP-complete if X $\in$NP and for any Y $\in$NP, $Y \leq_P X$.**
- **If an NP-complete problem is solvable in polynomial time, then P$=$NP.**

Does P=NP?

- **Is the decision problem as easy as the certification problem?**
- **If YES then there are efficient algorithms for many hard problems.**
- **If NO then no efficient algorithms for these hard problems.**
- **Consensus opinion, probably NO.**



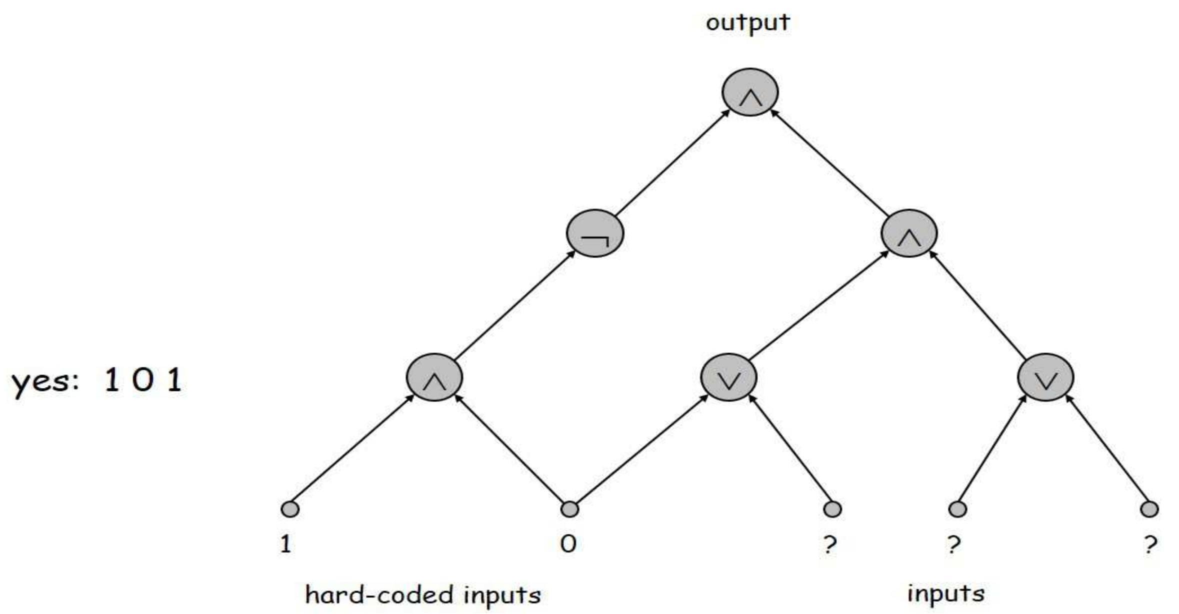
If $P \neq NP$



If $P = NP$

NP-Complete Problems

- **Circuit satisfiability problem:**
  - **A circuit C consists of inputs, wires, logic gates ($\wedge$ AND, $\vee$ OR, $\neg$ NOT) and output; C is satisfiable if there are values of the inputs s.t. the output is 1.**
  - **Given a circuit C, is C satisfiable?**



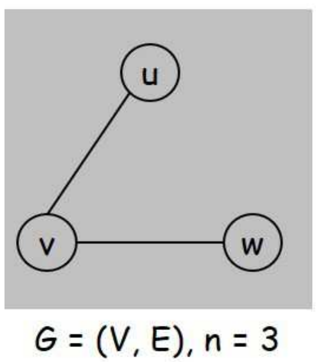
0 0 0
0 0 1
⋮
1 1 1

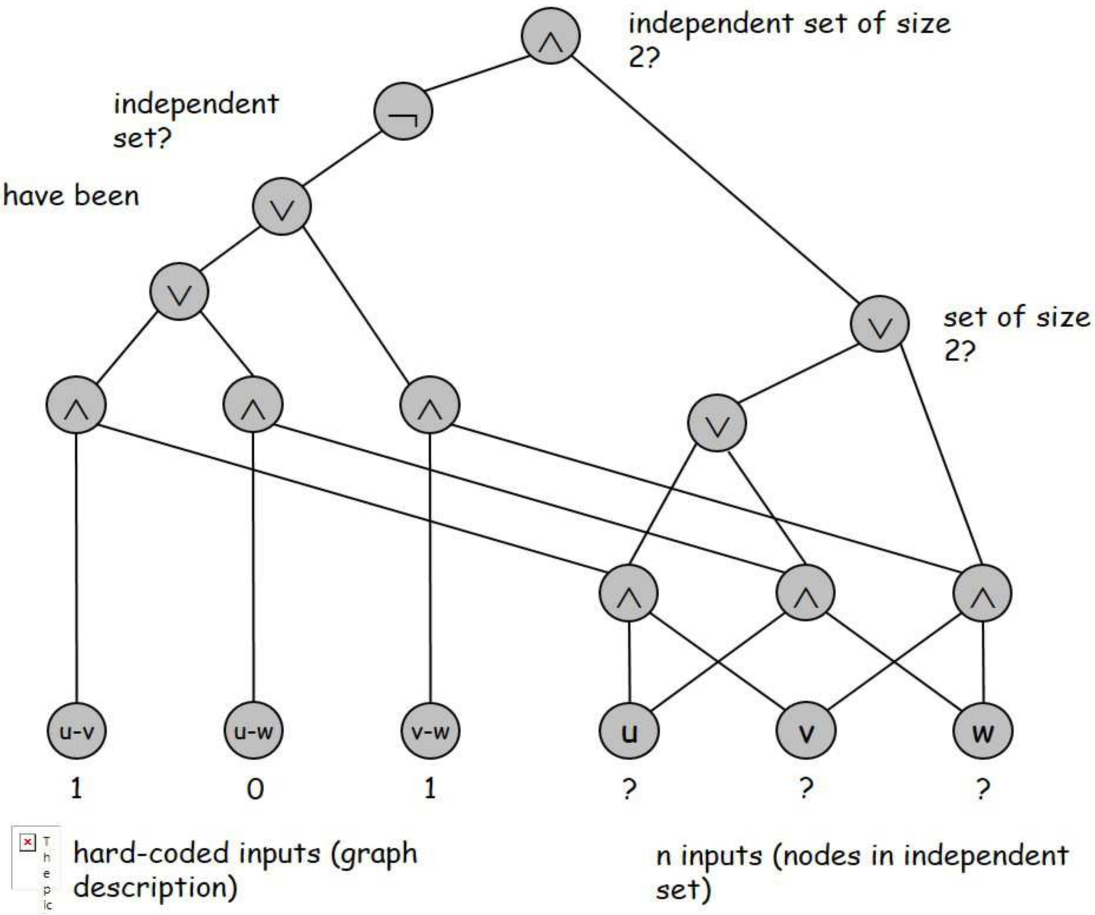
Theorem. Circuit satisfiability is NP-complete [Cook 1971, Levin 1973].

Proof. (Idea) Reduce every problem X $\in$NP to Circuit Satisfiability: Simulate steps of an efficient certifier $B(\cdot,\cdot)$ for X on inputs of fixed length by a circuit C s.t. C outputs 1 iff $B(\cdot,\cdot)$ outputs YES; and the size (number of gates) of C is O(running time of $B(\cdot,\cdot)$).

To decide if $s \in X$, we need to check if there is a string t of length $\mathrm{Poly}(|s|)$ s.t. $B(s,t)$ outputs YES. To simulate $B(\cdot,\cdot)$, transform $B(s,\cdot)$ into circuit $C(s)$ with s "hardwired" and $\mathrm{Poly}(|s|)$ inputs for possible t; then ask if $C(s)$ is satisfiable (call Circuit Satisfiability as an oracle); if satisfiable, then $s \in X$; otherwise, $s \notin X$. $\square$

Example: reduce independent set problem to Circuit Satisfiability, circuit C is satisfiable iff graph G has an independent set of size 2.





- **Prove a problem Y is NP-complete:**
  - **Show that Y $\in$NP;**
  - **Choose an NP-complete problem X;**
  - **Prove $X \leq_P Y$.**
- **If X is NP-complete, Y $\in$NP and $X \leq_P Y$, then Y is NP-complete.**

  Proof. For any problem Z $\in$NP, $Z \leq_P X$ as X is NP-complete. By $X \leq_P Y$ and the transitivity of polynomial time reduction, $Z \leq_P Y$. $\square$

- **Once the first NP-complete problem (Circuit Satisfiability) is proved, it is easier to prove others.**

Theorem. [Karp 1972] 3-SAT is NP-complete.

Proof. Given a 3-CNF Φ and a truth assignment, it takes linear time to check if Φ is satisfied by the assignment or not. So 3-SAT is in NP. Next we show Circuit-Satisfiability$\leq_P$3-SAT. For any circuit C, the inputs and output of each logical gate are considered as elements of C. For each circuit element i, a 3-SAT variable x_i is created. For each logical gate, CNF clauses are created as follows:

$$
\begin{array}{lcl}
x_i = \neg x_j & \rightarrow & (x_i \vee x_j)(\overline{x}_i \vee \overline{x}_j) \\
x_i = x_j \vee x_k & \rightarrow & (x_i \vee \overline{x}_j)(x_i \vee \overline{x}_k)(\overline{x}_i \vee x_j \vee x_k) \\
x_i = x_j \wedge x_k & \rightarrow & (\overline{x}_i \vee x_j)(\overline{x}_i \vee x_k)(x_i \vee \overline{x}_j \vee \overline{x}_k)
\end{array}
$$

If an input/output x_i is hard-coded 0, then create a clause $(\overline{x}_i)$; if hard coded 1, then create a clause (x_i). Make each clause of length$<$3 into a clause of length 3 (e.g., $(x_i \vee x_j) = (x_i \vee x_i \vee x_j)$. The construction takes polynomial time. Circuit C is satisfiable iff the constructed 3-CNF is satisfiable. So Circuit-Satisfiability$\leq_P$3-SAT.

Since Circuit-Satisfiability is NP-complete and Circuit-Satisfiability$\leq_P$3-SAT, for any problem Z in NP, $Z \leq_P$3-SAT. From this and 3-SAT is in NP, 3-SAT is NP-complete. $\square$

Circuit-Satisfiability$\leq_P$3-SAT example:

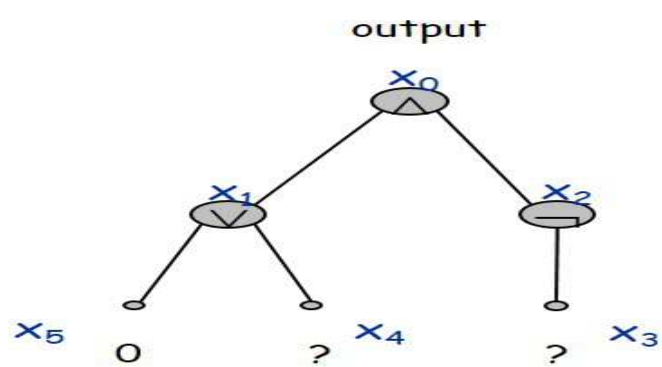
- **Make circuit compute correct values at each node**

$$x_2 = \overline{x}_3 \qquad \rightarrow \qquad (x_2 \vee x_3)(\overline{x}_2 \vee \overline{x}_3)$$
$$x_1 = x_4 \vee x_5 \qquad \rightarrow \qquad (x_1 \vee \overline{x}_4)(x_1 \vee x_5)(\overline{x}_1 \vee x_4 \vee x_5)$$
$$x_0 = x_1 \wedge x_2 \qquad \rightarrow \qquad (\overline{x}_0 \vee x_1)(\overline{x}_0 \vee x_2)(x_0 \vee \overline{x}_1 \vee \overline{x}_2)$$

- **Hard coded input values and output value**

$$x_5 = 0 \quad \rightarrow \quad (\overline{x}_5) \qquad\qquad x_0 = 1 \quad \rightarrow \quad (x_0)$$

- **CNF:** $(x_0)(x_2 \vee x_3)(\overline{x}_2 \vee \overline{x}_3)(x_1 \vee \overline{x}_4)(x_1 \vee x_5)(\overline{x}_1 \vee x_4 \vee x_5)$
$(\overline{x}_0 \vee x_1)(\overline{x}_0 \vee x_2)(x_0 \vee \overline{x}_1 \vee \overline{x}_2)(\overline{x}_5)$



- **Independent-Set is NP-complete.**

  Proof. Given a graph G and a subset S of $V(G)$ with $|S| = k$, it takes $O(n^2)$ time to check if S is an independent set of G or not. So, Independent-Set is in NP. Since, 3-SAT is NP-complete and 3-SAT$\leq_P$Independent-Set (proof in sildes 6-7), for any problem Z in NP, $Z \leq_P$Independent-Set. Therefore, Independent-Set is NP-complete. $\square$

- **Vertex-Cover is NP-complete.**

  Proof. Given a graph G and a subset S of $V(G)$ with $|S| = k$, it takes $O(m)$ time to check if S is a vertex cover of G or not. So, Vertex-Cover is in NP. Since, Independent-Set is NP-complete and Independent-Set$\leq_P$ Vertex-Cover (proof in silde 4), for any problem Z in NP, $Z \leq_P$Vertex-Cover. Therefore, Vertex-Cover is NP-complete. $\square$

- **Set-Cover problem: Given a set $U = \{a_1, ..., a_n\}$, subsets $S_1, .., S_m$ of U and integer k, is there a collection of k subsets $S_{i_1}, .., S_{i_k}$, $1 \leq i_1, .., i_k \leq m$, s.t. $\cup_{j=1}^{k} S_{i_j} = U$ (the union of the subsets in the collection equals U).**

- **Set-Cover is NP-complete.**

Proof. Given U and a collection of k subsets $S_{i_1}, .., S_{i_k}$, it takes $O(t)$ time, $t = |S_{i_1}| + \cdots + |S_{i_k}|$, to check if $\cup_{j=1}^{k} S_{i_j} = U$. So, Set-Cover is in NP.

Next, we show Vertex-Cover$\leq_P$Set-Cover. Given graph G and k, we construct a Set-Cover instance: $U = E(G)$. For every vertex $v \in V(G)$, let S_v be the set of edges incident to v. The construction takes polynomial time. There is vertex cover of size k for G iff there is a set cover of size k for U.

Since Vertex-Cover is NP-complete and Vertex-Cover$\leq_P$Set-Cover, for any problem Z in NP, $Z \leq_P$Set-Cover. From this and Set-Cover is in NP, Set-Cover is NP-complete. $\square$

Hamiltonian Cycle

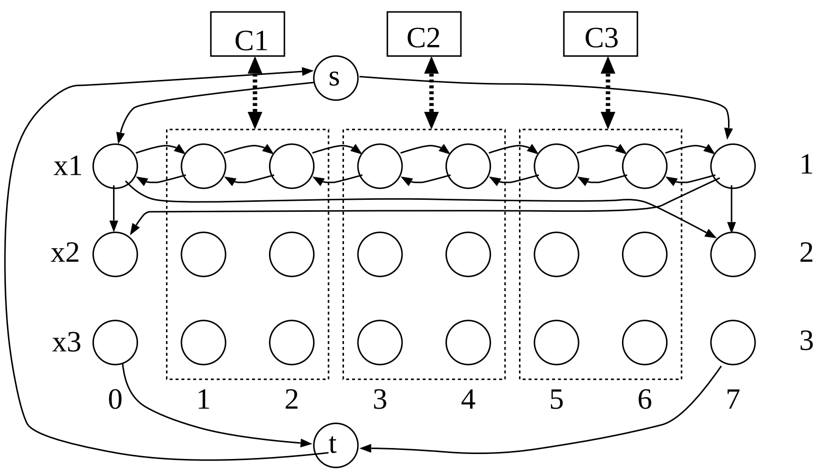
- **Ham-Cycle: Given a graph $G(V, E)$, is there a simple cycle Γ that contains every node of G?**
- **Directed-Ham-Cycle: Given a digraph $G(V, E)$, is there a simple directed cycle Γ that contains every node of G?**
- **Ham-Cycle and Directed-Ham-Cycle are NP-complete**

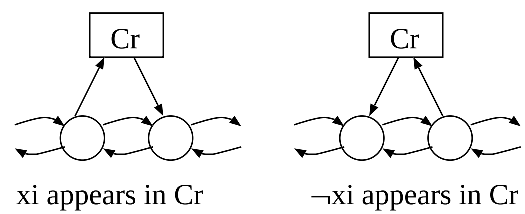
  3-SAT$\leq_P$Directed-Ham-Cycle, Directed-Ham-Cycle$\leq_P$Ham-Cycle.

Directed Ham-Cycle is NP-complete

Proof. Given a digraph G of n nodes and a permutation $(v_1, .., v_n)$ of the nodes of G, we can check if there is an edge (v_i, v_{i+1}) for every $1 \leq i < n$ and edge (v_n, v_1) in polynomial time. So, the problem is in NP. Next we show 3-SAT$\leq_P$Directed-Ham-Cycle. Given a 3-CNF Φ of n variables $\{x_1, .., x_n\}$ and m clauses $C_1, .., C_m$, we construct a digraph G with m nodes $c_1, .., c_m$ for clauses; n rows of nodes $v(i, j)$, each row has $2m + 2$ nodes, row i for x_i, $v(i, 2r - 1)$ and $v(i, 2r)$ for connecting to c_r if clause C_r has x_i or $\overline{x}_i$, and a source node s and destination node t.

Example Φ of 3 variables x_1, x_2, x_3 and 3 clauses $C_1C_2C_3$

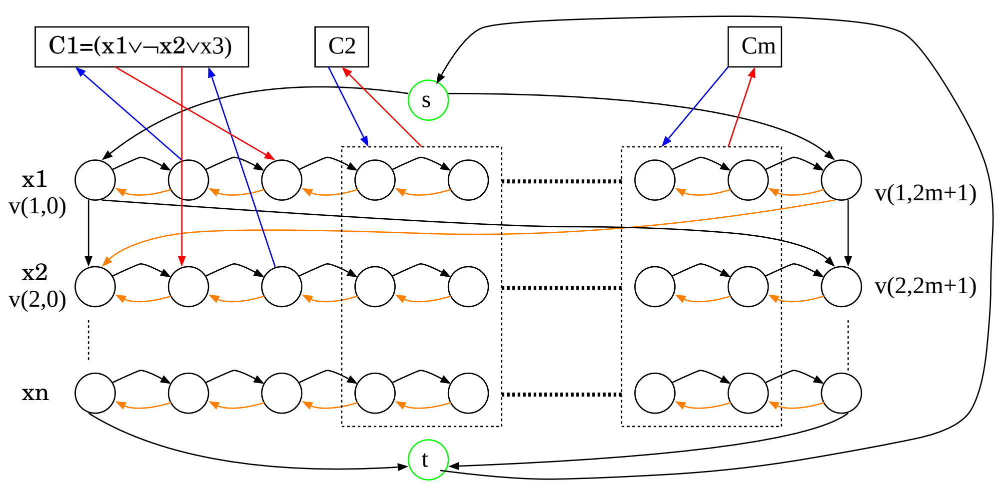




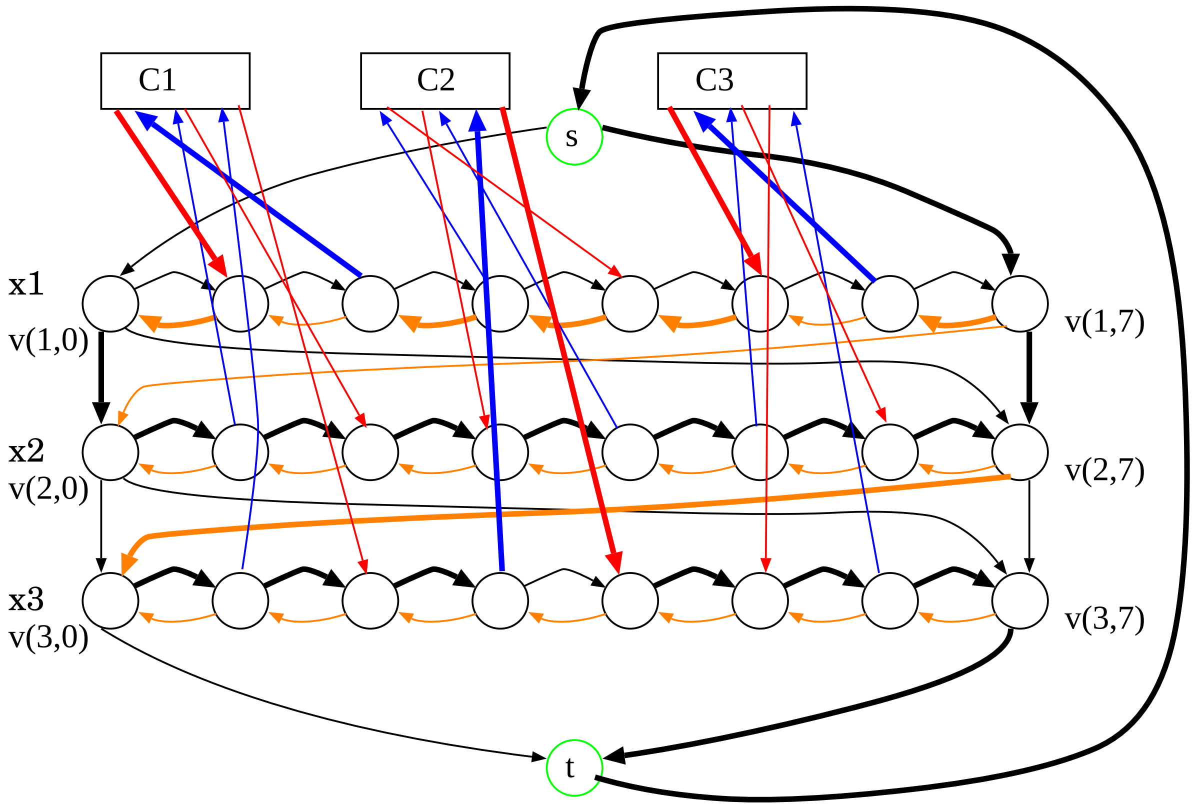
Formally,

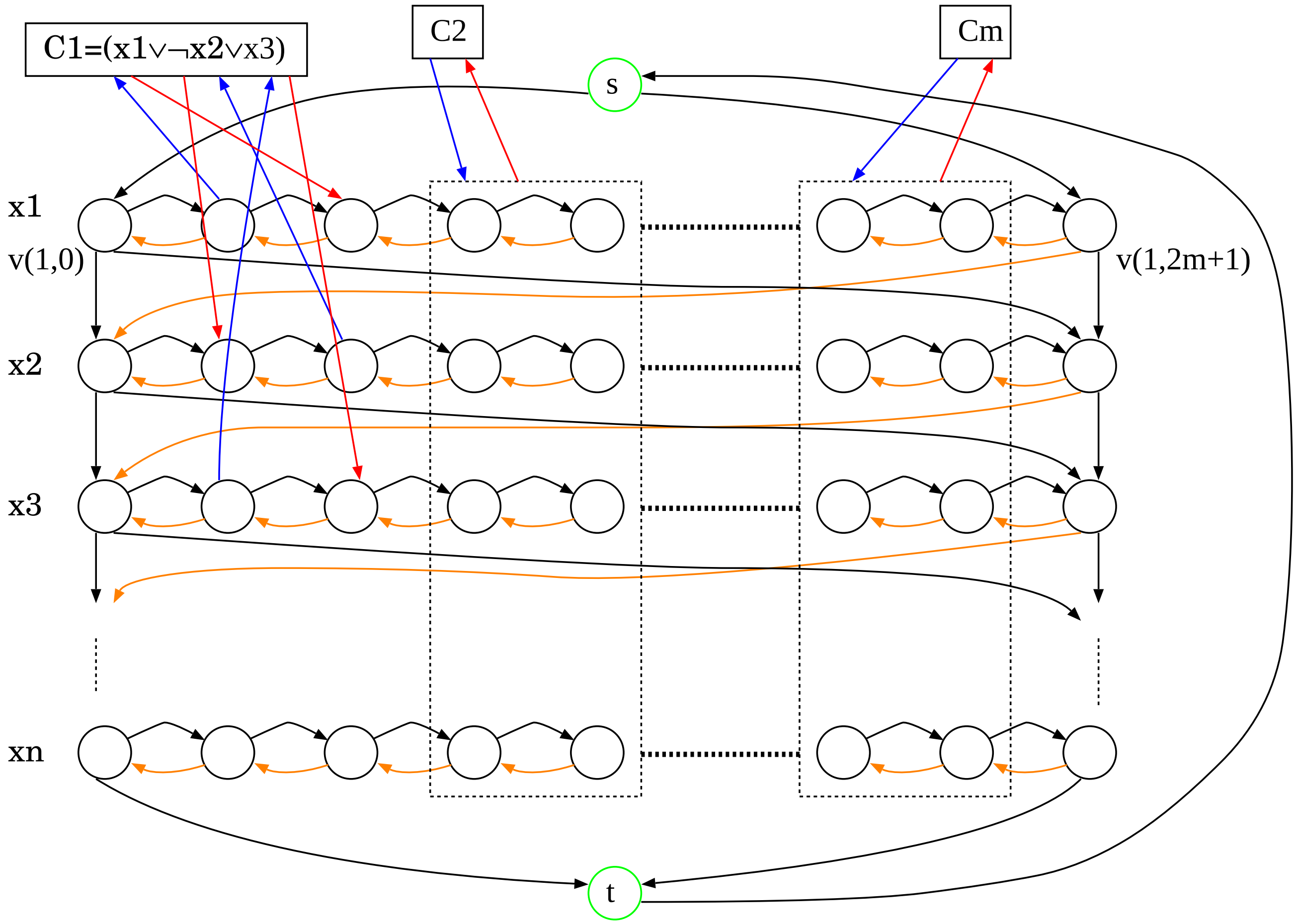
$$
\begin{aligned}
V(G) &= \{v(i,j) | 1 \le i \le n, 0 \le j \le 2m+1\} \cup \{c_j | 1 \le j \le m\} \cup \{s,t\} \\
E(G) &= \{(v(i,j), v(i,j+1)), (v(i,j+1), v(i,j)) | 1 \le i \le n, 0 \le j \le 2m\} \\
&\cup \{(v(i,0), v(i+1,0)), (v(i,0), v(i+1,2m+1)), \\
&\quad (v(i,2m+1), v(i+1,0)), (v(i,2m+1), v(i+1,2m+1)) | 1 \le i \le n-1\} \\
&\cup \{(v(i,2r-1), c_r), (c_r, v(i,2r)) | \text{if } C_r \text{ has } x_i\} \\
&\cup \{(v(i,2r), c_r), (c_r, v(i,2r-1)) | \text{if } C_r \text{ has } \overline{x}_i\} \\
&\cup \{(s, v(1,0)), (s, v(1,2m+1)), (v(n,0), t), (v(n,2m+1), t), (t,s)\}.
\end{aligned}
$$

G has $O(mn)$ nodes and $O(mn)$ edges. Φ is satisfiable iff G has a Hamiltonian cycle. $\square$



Example: $\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = C_1 C_2 C_3$.
$(x_1 = 0, x_2 = 1, x_3 = 1)$ **satisfies** Φ.





Graph constructed for 3-CNF of n variabls and m clauses: create one node for each clause; n rows ($1 \le i \le n$) and 2m+2 cloumns ($0 \le j \le 2m+1$) black nodes; row i for xi, cloumns 2r-1 and 2r for connecting to clause Cr

Directed Ham-Cycle$\leq_P$Ham-Cycle

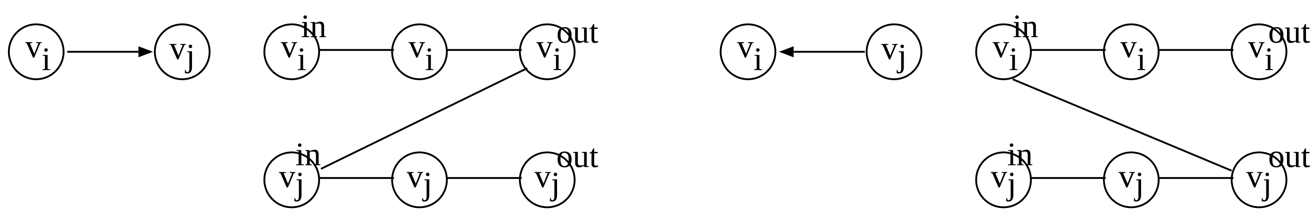
Proof. For a digraph G of n nodes, construct a graph H of $3n$ nodes: for each node $v_i \in V(G)$, $V(H)$ has three nodes v_i^{in}, v_i, v_i^{out}; $E(H)$ has edges $\{v_i^{in}, v_i\}$, $\{v_i, v_i^{out}\}$ for each v_i and has edge $\{v_i^{out}, v_j^{in}\}$ for each arc $(v_i, v_j) \in E(G)$. Now we show that G has a directed Ham-cycle iff H has a Ham-cycle.

Assume G has a Ham-cycle $v_{i_1}, v_{i_2}, ...$ Then H has a Ham-cycle $v_{i1}^{in}, v_{i1}, v_{i1}^{out}, v_{i_2}^{in}, v_{i_2}, v_{i_2}^{out}, ...$

Assume H has a Ham-cycle Γ. Then Γ has one of the two orders:

1. $v_{i_1}^{in}, v_{i_1}, v_{i_1}^{out}, v_{i_2}^{in}, v_{i_1}, v_{i_2}^{out}, ..$
2. $v_{i_1}^{out}, v_{i_1}, v_{i_1}^{in}, v_{i_2}^{out}, v_{i_1}, v_{i_2}^{in}, ..$

(1) gives a Ham-cycle with arc (v_{i1}, v_{i_2}) in G and (2) gives a Ham-cycle with arc (v_{i_2}, v_{i_1}). □



Travelling Salesperson Problem (TSP)

- **Given a number $D \geq 0$ and a weighted complete graph (digraph) G with each edge (arc) assigned a distance≥ 0, is there a cycle C containing every node of G s.t. the length of C is at most D.**
- **TSP is NP-complete.**

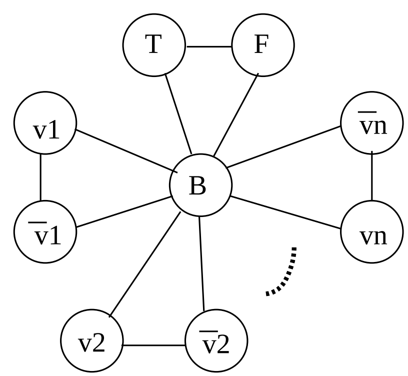
Proof. Given G and a cycle C, it takes $O(n)$ time if C contains every node of G and has length at most D. So, TSP is in NP.

Next, we show Ham-Cycle$\leq_P$TSP. Given an instance G of Ham-Cycle, we construct a TSP instance: create a weighted complete graph H with $V(H) = V(G)$ and assign each edge $\{u, v\}$ distance 1 if $\{u, v\} \in E(G)$ and assign distance 2 otherwise. Then TSP distance$\leq n$ iff G has a Ham-cycle. The construction takes $O(n^2)$ time. $\square$

Graph Coloring

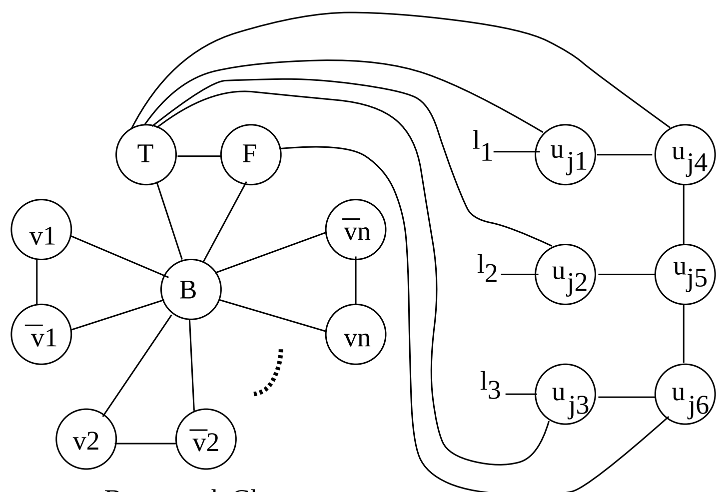
- **k-Colorability: Given graph G and integer k, is there a way to color the nodes of G by k colors s.t. every pair of adjacent nodes are colored by different colors?**
- **k-colorability is NP-complete.**

Proof. Given a coloring for nodes of G with m edges, we can check if every pair of adjacent nodes are colored by different colors in $O(m)$ time. So k-colorability is in NP. Next, we show 3-SAT$\leq_P$3-Colorability. Given a 3-CNF Φ of n variables and m clauses, we construct a graph G as follows: a base graph G_b with
$V(G_b) = \{T, F, B, v_i, \overline{v}_i | 1 \leq i \leq n\}$,
$E(G_b) = \{\{T, F\}, \{T, B\}, \{F, B\}, \{v_i, \overline{v}_i\}, \{v_i, B\}, \{\overline{v}_i, B\} | 1 \leq i \leq n\}$;



Base graph Gb

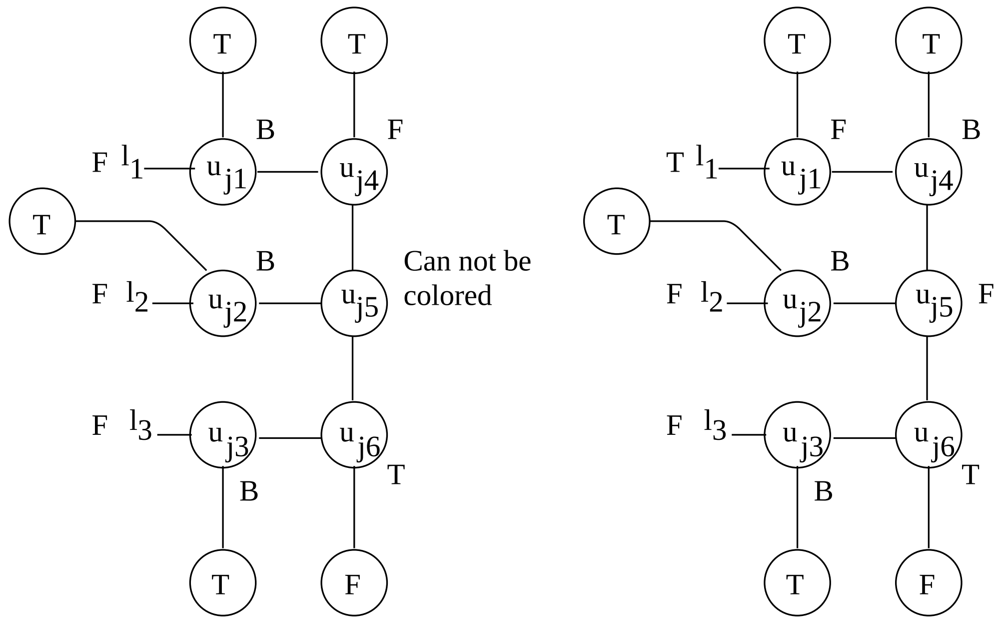
a gadget graph G_j ($1 \leq j \leq m$) for each clause C_j of Φ with
$V(G_j) = \{u_{j1}, u_{j2}, u_{j3}, u_{j4}, u_{j5}, u_{j6}\}$,
$E(G_j) = \{\{u_{j1}, u_{j4}\}, \{u_{j2}, u_{j5}\}, \{u_{j3}, u_{j6}\}, \{u_{j4}, u_{j5}\}, \{u_{j5}, u_{j6}\}\}$;
for $C_j = (l_1 \vee l_2 \vee l_3)$, if $l_p = x_i$ ($1 \leq p \leq 3$), connect v_i to u_{jp}; if $l_p = \overline{x}_i$, connect $\overline{v}_i$ to u_{jp}; connect T to $u_{j1}, u_{j2}, u_{j3}, u_{j4}$; connect F to u_{j6}.



Base graph Gb

gadget graph Gj for clause Cj=(l1∨l2∨l3)

G can be constructed in poly-time and Φ is satisfiable iff G is 3-colorable.



For $k > 3$, we show 3-colorability$\leq_P k$-colorability. Given a graph G of n nodes, we construct a graph G' by adding a clique C of size $k-3$ and connecting every node of C to every node of G. Then G is 3-colorable iff G' is k colorable. $\square$

Subset Sum: Given a set of integers $I = \{w_1, .., w_n\}$ and integer W, is there a subset $S \subseteq I$ s.t. $w(S) = \sum_{w_i \in S} w_i = W$.

Optimization problem of Subset Sum, find a subset $S \subseteq I$ s.t. $w(S) \leq W$ and $w(S)$ maximized. The optimization problem is a special case of Knapsack problem (with value v_i equal to the weight w_i for every item i) and can be solved in $O(nW)$ time.

Subset Sum is NP-complete.

Proof. Given a subset $S \subseteq I$, it takes $O(n)$ additions of w_i to compute $w(S)$, each addition takes $O(\log w_i) = O(n)$ time. So, it takes Poly(n) time to check a certificate and the problem is in NP.

Next, we show 3-SAT$\leq_P$Subset Sum. Given a 3-CNF Φ of n variables $x_1, .., x_n$ and m clauses $C_1, .., C_m$, we construct a Subset Sum instance: for each x_i, create numbers t_i and f_i,

$$t_i = 10^{m+i} + \sum_{j:C_j \text{ has } x_i} 10^j$$

$$f_i = 10^{m+i} + \sum_{j:C_j \text{ has } \overline{x}_i} 10^j$$

Example: $\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = C_1C_2C_3$.

$x_1 : t_1 = 10^{3+1} + 10^2 = 0010100 = 10100$

$f_1 = 10^{3+1} + 10^3 + 10^1 = 0011010 = 11010$

$x_2 : t_2 = 10^{3+2} + 10^3 + 10^1 = 0101010 = 101010$

$f_2 = 10^{3+2} + 10^2 = 0100100 = 100100$

$x_3 : t_3 = 10^{3+3} + 10^2 + 10^1 = 1000110$

$f_3 = 10^{3+3} + 10^3 = 1001000$

For each clause C_j, create $a_j = 10^j$ and $b_j = 10^j$.
Example: $\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = C_1 C_2 C_3$.
$C_1 : \ a_1 = 10^1 = 10, b_1 = 10^1 = 10$
$C_2 : \ a_2 = 10^2 = 100, b_2 = 10^2 = 100$
$C_3 : \ a_3 = 10^3 = 1000, b_3 = 10^3 = 1000$.

Let $I = \{t_i, f_i, a_j, b_j | 1 \leq i \leq n, 1 \leq j \leq m\}$ and

$$W = \sum_{i=1}^{n} 10^{m+i} + 3 \sum_{j=1}^{m} 10^j$$

$$W = \underbrace{11 \cdots 1}_{n1's} \underbrace{33 \cdots 3}_{m3's} 0$$

Every number of I and W is an $m + n + 1$ digits decimal.

For $\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = C_1 C_2 C_3$,
$W = 10^{3+3} + 10^{3+2} + 10^{3+1} + 3 \cdot (10^3 + 10^2 + 10^1) = 1113330$.

Example: Values of $(m+i)$th and jth (in increasing order of significance) decimal digits for $\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = C_1C_2C_3$.

variable	number	$i=3$	$i=2$	$i=1$	$j=3$	$j=2$	$j=1$
x_1	t_1	0	0	1	0	1	0
	f_1	0	0	1	1	0	1
x_2	t_2	0	1	0	1	0	1
	f_2	0	1	0	0	1	0
x_3	t_3	1	0	0	0	1	1
	f_3	1	0	0	1	0	0
C_1	a_1	0	0	0	0	0	1
	b_1	0	0	0	0	0	1
C_2	a_2	0	0	0	0	1	0
	b_2	0	0	0	0	1	0
C_3	a_3	0	0	0	1	0	0
	b_3	0	0	0	1	0	0
	W	1	1	1	3	3	3

Assume σ is an assignment satisfying Φ. Put $t_i \in S$ if $\sigma(x_i) = 1$, otherwise $f_i \in S$; put $a_j \in S$ if C_j has at most 2 literals assigned 1; put $b_j \in S$ if C_j has exactly 1 literal assigned 1. Then $w(S) = W$:

- Since S has either t_i or f_i but not both for every x_i, the most significant n digits of $w(S)$ meet these of W.

- For each j of the m least significant digits,

  if C_j has 3 literals assigned 1, then digit j has 3 from t_i or f_i of the 3 literals, 0 from a_j and 0 from b_j, total 3.

  If C_j has 2 literals assigned 1, then digit j has 2 from t_i or f_i of the 2 literals, 1 from a_j and 0 from b_j, total 3.

  If C_j has 1 literal assigned 1, then digit j has 1 from t_i or f_i of the literal, 1 from a_j and 1 from b_j, total 3.

  So, the least significant m digits of $w(S)$ meet these of W.

Example: $\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = C_1C_2C_3$.
$(x_1 = 0, x_2 = 1, x_3 = 1)$ satisfies Φ. $S = \{f_1, t_2, t_3, a_2, b_2, a_3\}$ and $w(S) = W$.

variable	number	$i=3$	$i=2$	$i=1$	$j=3$	$j=2$	$j=1$
x_1	t_1						
	f_1	0	0	1	1	0	1
x_2	t_2	0	1	0	1	0	1
	f_2						
x_3	t_3	1	0	0	0	1	1
	f_3						
C_1	a_1						
	b_1						
C_2	a_2	0	0	0	0	1	0
	b_2	0	0	0	0	1	0
C_3	a_3	0	0	0	1	0	0
	b_3						
	W	1	1	1	3	3	3

Assume there is a subset $S \subseteq \{t_i, f_i, a_j, b_j | 1 \leq i \leq n, 1 \leq j \leq m\}$ with $w(S) = W$.

- For each of x_i, exactly one of t_i and f_i is in S, otherwise $w(S) \neq W$ because the most significant n digits of $w(S)$ do not meet these of W ($10^{m+i} \notin \{0 \cdot 10^{m+i}, 2 \cdot 10^{m+i}\}$).
- For each clause C_j, the corresponding digit in the least significant m digits of $w(S)$ is 3, implying at least one of t_i or f_i with the jth least significant digit$= 1$ (10^j) is in S.

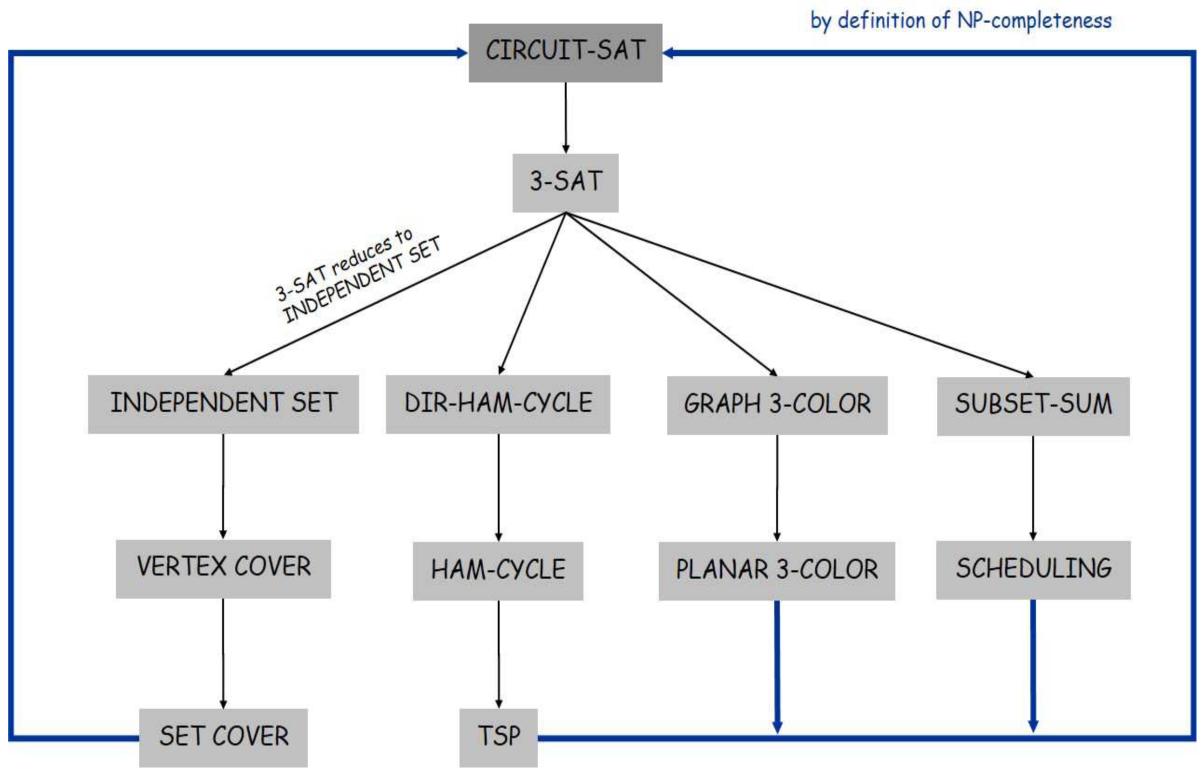
So, assign $x_i = 1$ if $t_i \in S$, otherwise $\overline{x}_i = 1$. This assignment satisfies Φ.

$\square$

Example: $\Phi = (\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) = C_1C_2C_3$. $S = \{t_1, t_2, f_3, a_1, b_1, a_2, b_2, b_3\}, w(S) = W$. $\sigma : x_1 = 1, x_2 = 1, x_3 = 0$ **satisfies** Φ.

variable	number	$i = 3$	$i = 2$	$i = 1$	$j = 3$	$j = 2$	$j = 1$
x_1	t_1	**0**	**0**	**1**	**0**	**1**	**0**
	f_1						
x_2	t_2	**0**	**1**	**0**	**1**	**0**	**1**
	f_2						
x_3	t_3						
	f_3	**1**	**0**	**0**	**1**	**0**	**0**
C_1	a_1	**0**	**0**	**0**	**0**	**0**	**1**
	b_1	**0**	**0**	**0**	**0**	**0**	**1**
C_2	a_2	**0**	**0**	**0**	**0**	**1**	**0**
	b_2	**0**	**0**	**0**	**0**	**1**	**0**
C_3	a_3						
	b_3	**0**	**0**	**0**	**1**	**0**	**0**
	W	**1**	**1**	**1**	**3**	**3**	**3**

More NP-complete problems



Classification of NP-complete problems

To prove an NP problem Y NP-complete, select an NP-complete problem X and show $X \leq_P Y$. Classifying well-known NP-complete problems may suggest how to choose X.

- **Packing problems**

  Common structure: Given a collection of objects, choose at least k of them for some goal. Constraints among objects make the choice difficult.

  - **Independent set problem: Given graph G and $k > 0$, does G have an independent set of size at least k?**
  - **Set packing problem: Given a set U of elements, a collection $S_1, .., S_m$ of subsets of U, and $k > 0$, does there exist a collection of at least k of these sets s.t. no two of them intersect?**

- **Covering problems**

  Common structure: Given a collection of objects, choose a subset of objects to cover some goal. Upper bound on the number of subsets makes the choice difficult.

  - **Vertex cover problem: Given a graph G and $k > 0$, does G has a vertex cover of size at most k?**
  - **Set cover problem: Given a set U of n elements, a collection $S_1, .., S_m$ of subsets of U, and $k > 0$, does there exist a collection of at most k subsets whose union equals to U?**

- **Partition problems**

  Common structure: Given a collection of objects, partition objects into subsets s.t. each object is in exactly one subset subject to some contraints which make the partition difficult.

  - **3-Dimensional matching problem: Given disjoint sets X, Y, Z, each of size n, and a subset $T \subseteq X \times Y \times Z$ of ordered triples, does there exist a set of n triples in T s.t. each element of $X \cup Y \cup Z$ is in exactly one triple.**
  - **Graph coloring problem: Given a graph G and $k > 0$, does G have a k-coloring?**

- **Sequencing problems**

  Common structure: find an ordered sequence of n objects satisfying certain properties from $n!$ sequences.

  - **Hamiltonian cycle/path problem: Given a graph/digraph G, does G have a Hamiltonian cycle/path?**
  - **Traveling salesperson problem: Given n cities and distances between them, and $D > 0$, does there exist a tour to visit all cities with length at most D?**

- **Numerical problems**

  Common structure: Given a set of integers and $W > 0$, find a subset of integers of sum value exactly W.

  - **Subset sum problem: Given n positive integers and $W > 0$, is there a subset of the integers s.t. the sum of the integers of the subset equals to W.**
  - **Knapsack problem: Given n objects, each object has a value and weight, $W > 0$ and $D > 0$, does there exist a subset of objects with total weight at most W and total value at least D?**

- **Constraint satisfaction problems**

  Common structure: Given a Boolean formula/circuit, find a truth value assignment to variables to make the forumla/circuit output true.

  - **SAT problem: Given a CNF formula f of n variables, does there exist an assignment satisfying f?**
  - **3-SAT problem: Given a 3-CNF formula f of n variables, does there exist an assignment satisfying f?**

NP-Hard and co-NP

NP-hard problems

- **Problem X is NP-complete if X $\in$NP and for any Y $\in$NP, $Y \leq_P X$.**
- **Problem X is NP-hard if for any Y $\in$NP, $Y \leq_P X$.**
- **An NP-complete problem is NP-hard.**
- **An NP-hard problem may not be in NP (thus not NP-complete).**
- **NP-complete problem examples, SAT, TSP, Set Cover.**
- **NP-hard problem examples, UN-SAT, No-Hamiltonian-Cycle, Quantified SAT, Competitive Facility Location, Halting problem.**

co-NP

- **Asymmetry of NP, only a short certificate for yes instance.**

  Example 1, SAT vs UN-SAT

  - **SAT, given a CNF formula Φ, is Φ satisfiable?**

    $X = \{f | f$ **is satisfiable**$\}$**, given** ϕ **if** $\phi \in X$**?**

    Can be proved by a certificate (truth assignment).

  - **UN-SAT, given a CNF formula Φ, is Φ not satisfiable?**

    $\overline{X} = \{g | g$ **is not satisfiable**$\}$**, given** ϕ **if** $\phi \in \overline{X}$**?**

    How to prove this?

Example 2, Hamiltonian-Cycle vs No-Hamiltonian-Cycle

- **Hamiltonian-Cycle, given a graph G, is there a simple cycle that contains every node of G?**

  $X = \{H | H$ **has a Hamiltonian-Cycle**$\}$**, given G if $G \in X$?**

  Can be proved by a certificate (a permutation of nodes in G).

- **No-Hamiltonian-Cycle, given a graph G, is there no simple cycle that contains every node of G?**

  $\overline{X} = \{H | H$ **does not have a Hamiltonian-Cycle**$\}$**, given G if $G \in \overline{X}$?**

  How to prove this?

- **Decision problem X: given an instance of X, is there a YES answer for the instance? Example, $X = \{f | f$ is satisfiable$\}$.**

  Complement $\overline{X}$ of X: given an instance of X, is there no YES answer for the instance? Example, $\overline{X} = \{g | g$ is not satisfiable$\}$.

- **co-NP: set of complements of decision problems in NP (co-NP=$\{\overline{X} | X \in$ NP$\}$).**

  Examples, UN-SAT, No-Hamiltonian-Cycle.

- **Does NP$=$co-NP? Consensus opinion: No.**

- **If NP$\neq$co-NP then P$\neq$NP.**

- **P$\subseteq$NP$\cap$co-NP.**

- **Does P$=$NP$\cap$co-NP? Not known.**

PSPACE

- **Problem examples**
- **PSPACE definition**
- **PSPACE problems**
- **PSPACE completeness**

Problem examples

- **Geography**

  Alice names a capital city x_1, then Bob names a capital city x_2 that starts with the letter on which x_1 ends; Alice and Bob repeat this game until one player is unable to continue. Does Alice have a winning strategy?

  Example, Budapest$\rightarrow$Tokyo$\rightarrow$Ottawa$\rightarrow$Ankara$\rightarrow$Amsterdam $\rightarrow$Moscow$\rightarrow$Washington$\rightarrow$Nairobi$\rightarrow$...

- **Geography on graphs**

  Given a digraph G and start node s, two players move in turn from one node to an unvisited node following one arc of G until one player is unable to make any move. Does the 1st player has a winning strategy?

- **Is Geography in NP?**

PSPACE definition

- **PSPACE: class of decision problems solvable in polynomial space.**
- **P$\subseteq$PSPACE**

  Each problem in P can be solved in Poly-time and Poly-time algorithm can use only Poly-space.
- **3-SAT is in PSPACE.**

  For a 3-CNF of n Boolean variables, create 2^n truth assignment one by one, and check if each assignment satisfies the 3-CNF. This uses Poly-space.
- **NP$\subseteq$PSPACE.**

  For any problem $X \in$NP, since $X \leq_P$3-SAT, there is an algorithm which solves X in Poly-time plus Poly-number of calls to 3-SAT oracle which uses Poly-space.

PSPACE problems

- **Quantified Satisfiability (QSAT)**

  Given a CNF $\Phi(x_1, .., x_n)$ (n odd), is the propositional formula true?

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 .. \forall x_{n-1} \exists x_n \Phi(x_1, .., x_n).$$

- **Intuition, Alice set the truth value for x_1, then Bob for x_2, and so on. Can Alice satisfy Φ no matter what Bob does?**

- **Example 1:** $(x_1 \vee x_2)(x_2 \vee \overline{x}_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3)$

  Yes. Alice set $x_1 = 1$, Bob set x_2, Alice set $x_3 = x_2$.

- **Example 2:** $(x_1 \vee x_2)(\overline{x}_2 \vee \overline{x}_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3)$

  No. Alice set x_1, Bob set $x_2 = x_1$ and Alice loses.

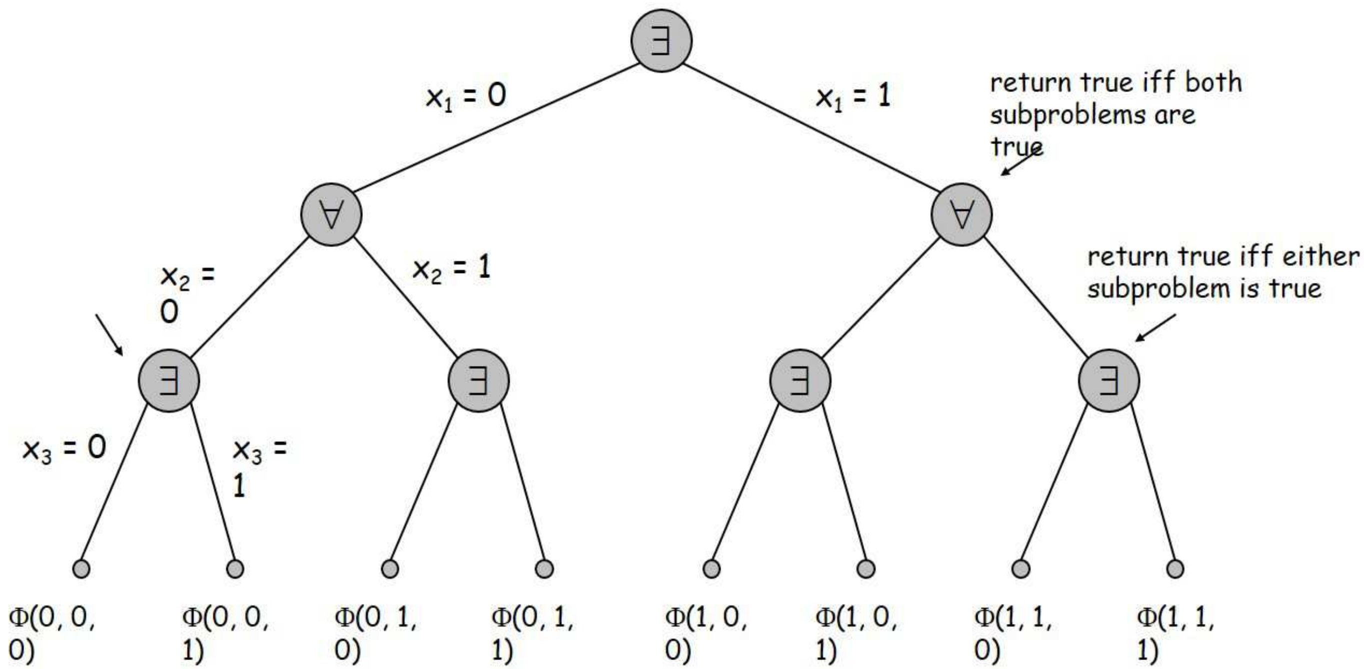
- **QSAT is in PSPACE.**

Proof. Basic idea: $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdots \exists x_n \Phi \rightarrow$
$0 \forall x_2 \exists x_3 \forall x_4 \cdots \exists x_n \Phi$ and $1 \forall x_2 \exists x_3 \forall x_4 \cdots \exists x_n \Phi$
$0 \forall x_2 \exists x_3 \forall x_4 \cdots \exists x_n \Phi \rightarrow 00 \exists x_3 \forall x_4 \cdots \exists x_n \Phi$ and $01 \exists x_3 \forall x_4 \cdots \exists x_n \Phi$.

Algorithm to check if quantified CNF Φ is satisfiable:

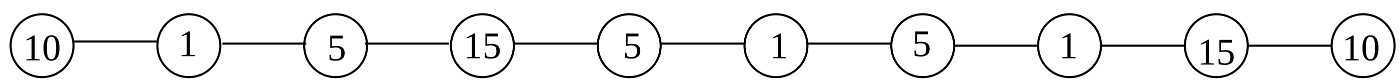
- If the 1st quantifier is $\exists x_i$ then
  - set $x_i = 0$; recursively call on Φ over the remaining variables; save the result (0 or 1) and delete all interim work;
  - set $x_i = 1$; recursively call on Φ over the remaining variables; save the result (0 or 1) and delete all interim work;
  - If either outcome from $x_i = 0$ or $x_i = 1$ is true, then output 1, otherwise 0;
- If the 1st quantifier is $\forall x_i$ then
  - set $x_i = 0$; recursively call on Φ over the remaining variables; save the result (0 or 1) and delete all interim work;
  - set $x_i = 1$; recursively call on Φ over the remaining variables; save the result (0 or 1) and delete all interim work;
  - If both outcomes from $x_i = 0$ and $x_i = 1$ are true, then output 1, otherwise 0;

The space used by the algorithm is proportional to the depth of recursion. $\square$


∃
$x_1 = 0$
$x_1 = 1$
return true iff both subproblems are true
∀
∀
$x_2 = 0$
$x_2 = 1$
return true iff either subproblem is true
∃
∃
∃
∃
$x_3 = 0$
$x_3 = 1$
Φ(0, 0, 0)
Φ(0, 0, 1)
Φ(0, 1, 0)
Φ(0, 1, 1)
Φ(1, 0, 0)
Φ(1, 0, 1)
Φ(1, 1, 0)
Φ(1, 1, 1)

Competitive facility location problem

- **Input: graph G, each node has a positive profit, and target B.**
- **Game: two players alternatively select a node, not allowed to select a node if any of its neighbors has been selected.**
- **Can the 2nd player guarantee at least B units of profit?**
- **The problem is in PSPACE: can be solved in poly-space by a recursion similar to that for QSAT with each step has up to n choices instead of 2.**



YES for B=20; No for B=25

PSPACE-completeness

- **Problem Y is PSPACE-Complete if Y is in PSPACE and for every problem X in PSPACE, $X \leq_P Y$.**
- **QSAT is PSPACE-complete (Stockmeyer and Meyer 1973)**
- **Competitive facility location (CFL) is PSPACE-complete (QSAT$\leq_P$CFL).**
- **EXTIME: class of problems solvable in exponential time.**
- **PSPACE$\subseteq$EXTIME.**

  QSAT is PSPACE-complete and can be solved in exponential time
- **Conjectures: P$\subseteq$NP$\subseteq$PSPACE$\subseteq$EXTIME**

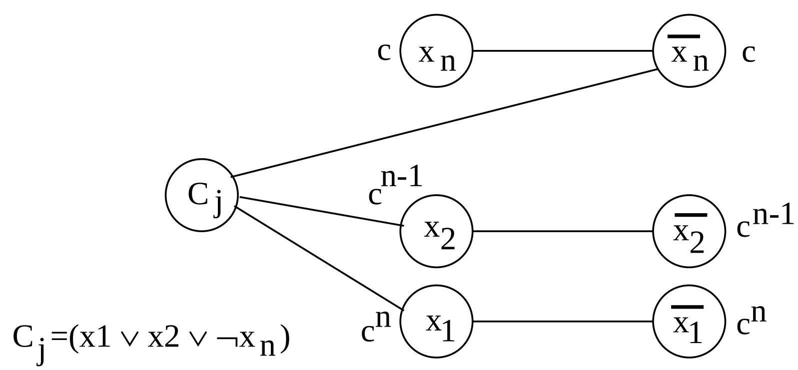
  It is known P$\neq$EXTIME, but not known which inclusion is strict; conjectured all the inclusions are.

Theorem. Competitive facility location (CFL) is PSPACE-complete.

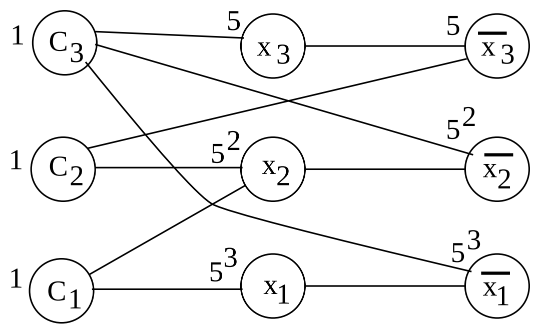
Proof. The problem is in PSPACE. We show QSAT$\leq_P$CFL. Given a QSAT instance $\exists x_1 \forall x_2 .. \exists x_n \Phi(x_1, .., x_n) = C_1 \wedge .. \wedge C_k$, we construct an instance of the problem:

- For each x_i, create nodes x_i and $\overline{x}_i$ with profit c^{n-i+1} ($c \geq k+2$) and edge $\{x_i, \overline{x}_i\}$.
- For each $C_j = (l_{j_1} \vee .. \vee l_{j_r})$, create node C_j with profit 1 and edges $\{C_j, l_{j_1}\}, .., \{C_j, l_{j_r}\}$.
- $B = c^{n-1} + c^{n-3} + .. + c^4 + c^2 + 1$ (assume n is odd).

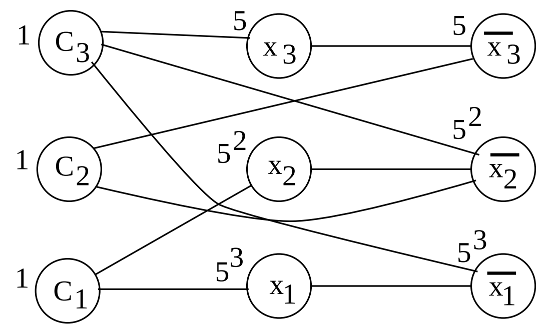
Player 2 can force a win in the instance iff player 1 can not satisfy Φ. □



- **Example 1:** $(x_1 \vee x_2)(x_2 \vee \overline{x}_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3)$
- **Example 2:** $(x_1 \vee x_2)(\overline{x}_2 \vee \overline{x}_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3)$



Example 1



Example 2

x_i	x_j	$x_i = \neg x_j$	$(x_i \vee x_j)(\overline{x}_i \vee \overline{x}_j)$
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**1**	**1**
1	**1**	**0**	**0**

x_i	x_j	x_k	$x_i = x_j \vee x_k$	$(x_i \vee \overline{x}_j)(x_i \vee \overline{x}_k)(\overline{x}_i \vee x_j \vee x_k)$
0	0	0	1	1
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

x_i	x_j	x_k	$x_i = x_j \wedge x_k$	$(\overline{x}_i \vee x_j)(\overline{x}_i \vee x_k)(x_i \vee \overline{x}_j \vee \overline{x}_k)$
0	**0**	**0**	**1**	**1**
0	**0**	**1**	**1**	**1**
0	**1**	**0**	**1**	**1**
0	**1**	**1**	**0**	**0**
1	**0**	**0**	**0**	**0**
1	**0**	**1**	**0**	**0**
1	**1**	**0**	**0**	**0**
1	**1**	**1**	**1**	**1**