EECS 336: Introduction to Algorithms	**Lecture 16**
P vs. NP (cont.)	INDEP-SET, CIRCUIT-SAT

Reading: 8.1-8.4

Last time:

- $\mathcal{NP}$-completeness
- "notorious problem" NP.
- redutions from 3-SAT.

Today:

- INDEP-SET $\leq_{\mathcal{P}}$ 3-SAT
- NP $\leq_{\mathcal{P}}$ CIRCUIT-SAT $\leq_{\mathcal{P}}$ 3-SAT

Problem 1: Independent Set (INDEP-SET)

input: $G = (V, E)$

output: $S \subset V$

- satisfying $\forall v \in S,\ (u, v) \notin E$
- maximizing $|S|$

Problem 4: 3-SAT

input: boolean formula $f(\mathbf{z})$

- in <u>conjunctive normal form</u> (CNF)
- three literals per <u>or-clause</u>
- or-clauses <u>anded</u> together.

output:

- "Yes" if assignment $\mathbf{z}$ with $f(\mathbf{z}) = T$ exists
- "No" otherwise.

Independent Set

Recall: INDEP-SET (decision problem)

input: $G = (V, E)$, k

output: $S \subset V$

- satisfying $\forall v \in S,\ (u, v) \notin E$
- $|S| \geq k$

Lemma: INDEP-SET is $\mathcal{NP}$-hard.

Proof: (reduction from 3-SAT)

Step 1: convert 3-SAT instance f into INDEP-SET instance (G, k).

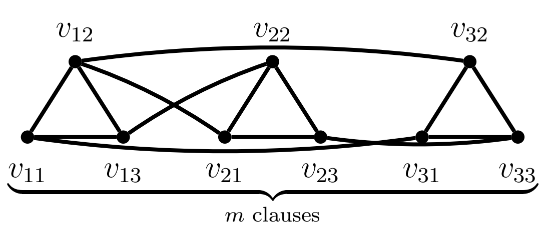
literal j in clause i

- vertices v_{ij} correspond to literals l_{ij}
- edges for:
  - clause (in triangle)

    "at most one vertex selected per clause"
  - conflicted literals.

    "vertices for conflicting literals cannot be selected"
- "vertex v_{ij} is selected" $\Rightarrow$ "literal l_{ij} is true".
- "indep set of size m $\Leftrightarrow$ "satisfying assignment"

Example: $f(z_1, z_2, z_3, z_4) = (z_1 \vee z_2 \vee z_3) \wedge (\bar{z}_2 \vee \bar{z}_3 \vee \bar{z}_4) \wedge (\bar{z}_1 \vee \bar{z}_2 \vee z_4)$



Step 2: construction is polynomial time.

one vertex per literal.

Step 3: show construction correct.

(a) if f is satisfiable then G has indep. set size $\geq m$.

- f is sat

  $\Rightarrow$ exists $\mathbf{z}$ so each clause is true.
- let S' be nodes in G corresponding to true literals.
- if more than one node in S' in same triangle drop all but one.

  $\Rightarrow$ S.
- $|S| = m$.
- for all $u, v \in S$,
  - u & v not in same triangle.
  - l_u and l_v both true

    $\Rightarrow$ must not conflict

    $\Rightarrow$ no (l_u, l_v) edge in G.
  - so S is independent.

(b) if G has indep. set S size $\geq m$ then f is satisfiable.

  (a) construct assignment $\mathbf{z}$ from S

  For each z_r

  - if nodes in S are labeled by z_r (but not $\bar{z}_r$)

    $\Rightarrow$ set $z_r = 1$
  - if nodes in S are labeled by $\bar{z}_r$ (but not z_r)

    $\Rightarrow$ set $z_r = 0$
  - if no $v \in S$ is labeled z_r or $\bar{z}_r$

    $\Rightarrow$ set $z_r = 1$ (or 0, doesn't matter)

Note: no two nodes $u, v \in S$ labeled by both z_r or $\bar{z}_r$, if so, there is (u, v) edge so S would not be independent.

(b) $f(\mathbf{z}) = T$:

- S has $|S| \geq m$
- can have at most one node from each triangle

  $\Rightarrow$ have exactly one from each triangle

  $\Rightarrow$ $|S| = m$
- $v \in S$ means literal l_v is true.

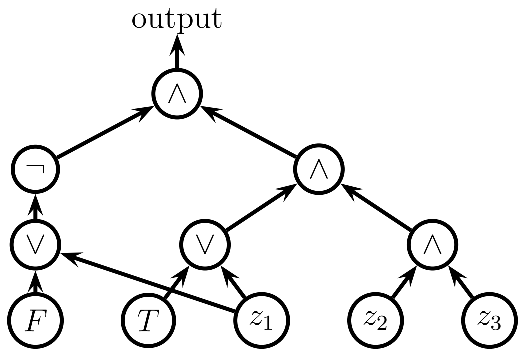
  $\Rightarrow$ one true literal per clause

  $\Rightarrow$ $f(\mathbf{z}) = T$.

QED

Circuit Satisfiability

Example:



Problem 4: CIRCUIT-SAT

input: boolean circuit $Q(\mathbf{z})$

- directed acyclic graph $G = (V, E)$
- internal nodes labeled by logical gates:

  "and", "or", or "not"

- leaves labeled by variables or constants

  $T, F, z_1, \ldots, z_n$.

- root r is output of circuit

output:

- "Yes" if exists $\mathbf{z}$ with $Q(\mathbf{z}) = T$
- "No" otherwise.

Lemma: CIRCUIT-SAT is $\mathcal{NP}$-hard.

Proof: (reduce from NP)

- goal: convert NP instance (VP, p, x) to CIRCUIT-SAT instance Q
- $VP(\cdot, \cdot)$ polynomial time

  $\Rightarrow$ computer can run it in poly steps.

- each step of computer is circuit.
- output of one step is input to next step
- unroll $p(|x|)$ steps of computation

  $\Rightarrow$ $\exists$ poly-size circuit $Q'(\mathbf{x}, \mathbf{c}) = VP(x, c)$

- hardcode $\mathbf{x}$: $Q(\mathbf{c}) = Q'(\mathbf{x}, \mathbf{c})$
- Conclusion: Q is sat iff exists c with $VP(x, c) =$ "verified".

QED

3-SAT

Problem 4: 3-SAT

input: boolean formula $f(\mathbf{z})$

- in conjunctive normal form (CNF)
- three literals per or-clause
- or-clauses anded together.

output:

- "Yes" if assignment $\mathbf{z}$ with $f(\mathbf{z}) = T$ exists
- "No" otherwise.

Problem 5: LE3-SAT

"like 3-SAT but at most 3 literals per or-clause"

Note: $\leq_{\mathcal{P}}$ is transitive: if $Y \leq_{\mathcal{P}} X$ and $X \leq_{\mathcal{P}} Z$ then $Y \leq_{\mathcal{P}} Z$.

Recall: NP $\leq_{\mathcal{P}}$ CIRCUIT-SAT

Plan: CIRCUIT-SAT $\leq_{\mathcal{P}}$ LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT

Lemma: LE3-SAT $\leq_{\mathcal{P}}$ 3-SAT

Step 1: convert LE3-SAT instance f' into 3-SAT instance f

- $f \leftarrow f'$
- add variables w_1, w_2
- add w_i to 1- and 2-clauses

  $(l_1) \Rightarrow (l_1 \vee w_1 \vee w_2)$.

  $(l_1 \vee l_2) \Rightarrow (l_1 \vee l_2 \vee w_1)$.
- ensure $w_i = 0$ add variables y_1, y_1 and clauses:

  $(\bar{w}_i \vee y_1 \vee y_2)$

  $(\bar{w}_i \vee \bar{y}_1 \vee y_2)$

  $(\bar{w}_i \vee y_1 \vee \bar{y}_2)$

  $(\bar{w}_i \vee \bar{y}_1 \vee \bar{y}_2)$

Step 2: construction is polynomial time.

Step 3: f is sat iff f' is sat.

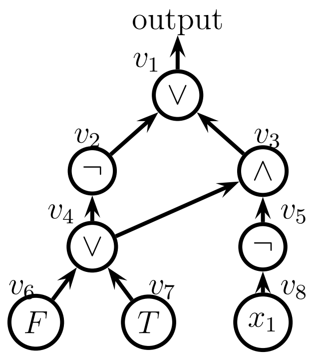
- given satisfying assignment $(\bar{z}, w_1, w_2, y_1, y_2)$ to f,

  $\Rightarrow$ $w_i = F$ by construction.

  $\Rightarrow$ $f(\bar{z}, F, F, y_1, y_2) \overset{\text{simplify}}{\Longrightarrow} f(\bar{z})$

  $\Rightarrow$ f is sat.
- given satisfying assignment $\bar{z}$ to f',
  - $f(\bar{z}, w_1, w_2, y_1, y_2) \overset{\text{simplify}}{\Longrightarrow}$ "clauses with only w_i and y_i"
  - set $w_i = F$ and $y_i = F$ (or anything) to satisfy. **QED**

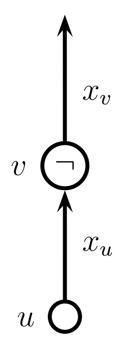
Example:



Proof: (reduce from CIRCUIT-SAT)

Step 1: convert CIRCUIT-SAT instance Q into 3-SAT instance f

- variables x_v for each vertex of Q.
- encode gates
  - **not**: if v not gate with input from u



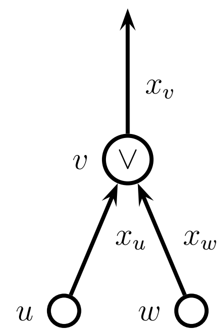
need $x_v = \bar{x}_u$

$x_v \setminus x_u$	0	1
0	0	1
1	1	0

$\Rightarrow$ add clauses $(x_v \vee x_u) \wedge (\bar{x}_v \vee \bar{x}_u)$

- **or**: if v is or gate from u to w

need $x_v = x_u \wedge x_w$



$x_v \setminus x_u x_w$	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$\Rightarrow$ add clauses $(\bar{x}_v \vee x_u \vee x_w) \wedge (x_v \vee \bar{x}_u) \wedge (x_v \vee \bar{x}_w)$

- **and:** if v is and gate from u to w

  $\Rightarrow$ add clauses $(x_v \vee \bar{x}_u \bar{x}_w) \wedge (\bar{x}_v \vee x_u) \wedge (\bar{x}_v \vee x_w)$.

- **0:** if v is 0 leaf.

  need $x_v = 0$

  $\Rightarrow$ add clause $(\bar{x}_v)$

  need $x_v = 1$

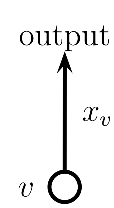
- **1:** if v is 1 leaf.

  $\Rightarrow$ add clause (x_v)

- **literal:** if v is literal z_j

  $\Rightarrow$ do nothing

- **root:** if v is root



need $x_v = 1$

$\Rightarrow$ add clause (x_v).

Step 2: construction is polynomial time.

- at most 3 clauses in f per node in Q.

Step 3: construction is correct (i.e., Q is sat iff f is sat.)

- f constrains variables v_i to "proper circuit outcomes".
- if exists $\mathbf{z}$ s.t. $f(\mathbf{z})$ is T,

  then can read $\mathbf{x}$ from $\mathbf{z}$ and $\mathbf{z}$ encodes proper circuit outcome to make Q output T for this $\mathbf{x}$.
- if Q outputs T for some $\mathbf{x}$

  then can map $\mathbf{x}$ and values at nodes to variables $\mathbf{z}$ such that $f(\mathbf{z})$ is true.

QED

Lemma: 3-SAT is in NP

Proof: Certificate is assignment $\mathbf{z}$.

Theorem: 3-SAT is NP-complete.

Proof: from lemmas.

Note: 2 steps to NP-completeness

1. $X \in \mathcal{NP}$
2. X is $\mathcal{NP}$-hard (via reduction)

3 steps to reduction

1. construction
2. runtime of construction
3. correctness of construction (iff)