CS 344
LECTURE 19
COMPLEXITY
<table>
<thead>
<tr>
<th>Hard problems (NP-complete)</th>
<th>Easy problems (P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3SAT</td>
<td>2SAT, Horn SAT</td>
</tr>
<tr>
<td>TSP</td>
<td>Min. spanning tree</td>
</tr>
<tr>
<td>Longest path</td>
<td>Shortest path</td>
</tr>
<tr>
<td>3D matching</td>
<td>Bipartite matching</td>
</tr>
<tr>
<td>Knapsack</td>
<td>Unary knapsack</td>
</tr>
<tr>
<td>Indep. set</td>
<td>Indep. set on trees</td>
</tr>
<tr>
<td>ILP</td>
<td>LP</td>
</tr>
<tr>
<td>Hamiltonian cycles</td>
<td>Euler paths</td>
</tr>
<tr>
<td>Balanced cut</td>
<td>Minimum cut</td>
</tr>
</tbody>
</table>
WHAT ARE P AND NP?

- P: problems solvable in polynomial time
- NP: problems verifiable in polynomial time
REDUCING A TO B

Algorithm for $A$

Instance $I$ → $f$ → Instance $f(I)$ → Algorithm for $B$ → Solution $S$ of $f(I)$ → $h$ → Solution $h(S)$ of $I$

No solution to $f(I)$

No solution to $I$
REDUCING A TO B

\[ A \xrightarrow{\text{Alg}_A} S_A \]
\[ B \xrightarrow{\text{Alg}_B} S_B \]
\[ f \quad h \]
If we reduce $A$ to $B$, we write $A \rightarrow B$.

- If we could efficiently solve $B$, we could solve $A$.
- If we know $A$ cannot be efficiently solved, then neither can $B$!
- ($B$ is at least as hard as $A$)

Sometimes a reduction is written as $A \leq_p B$
REDUCING A TO B

\[ A \xrightarrow{\text{Alg}_A} S_A \]

\[ A \xleftarrow{f} B \xrightarrow{\text{Alg}_B} S_B \]

\[ B \xrightarrow{h} S_B \]
REDUCING A TO B

The diagram shows the relationship between sets $A$ and $B$ with the following mappings:

- $A$ is mapped to $S_A$ by $\text{Alg}_A$.
- $B$ is mapped to $S_B$ by $\text{Alg}_B$.
- $S_A$ is mapped to $S_B$ by $h$.
- $f$ maps from $B$ to $A$.
If $A \to B$ and $B \to C$, then $A \to C$.

Given $(f_{AB}, h_{AB})$ and $(f_{BC}, h_{BC})$, we can compose them:

$$(f_{BC} \circ f_{AB}, h_{AB} \circ h_{BC})$$
If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$. 
NP-COMPLETENESS

A problem is NP-complete if:

- it's in NP
- all other problems in NP reduce to it

(NP-complete problems are the "hardest" in NP)
To show $B$ is NP-complete:

- Given an NP-complete problem $A$,
- Reduce $A$ to $B$
All of **NP**

- **SAT**
  - **3Sat**
    - **INDEPENDENT SET**
    - **3D MATCHING**
      - **CLIQUE**
      - **ZOE**
        - **SUBSET SUM**
        - **ILP**
        - **RUDRATA CYCLE**
          - **TSP**
HAM. PATH $\rightarrow$ HAM. CYCLE

Is it easier to find a cycle than an $(s, t)$ path?
HAM. PATH $\rightarrow$ HAM. CYCLE
**RUDRATA \((s, t)\)-PATH**

**Instance:**
\(G = (V, E)\) nodes \(s, t\)

Add node \(x\) and edges \(\{s, x\}, \{x, t\}\)

\(G' = (V', E')\)

**RUDRATA CYCLE**

Solution: cycle \(\{s, x\}, \{x, t\}\)

Delete edges \(\{s, x\}, \{x, t\}\)

No solution

Solution: path

No solution
• If a cycle is found
  - Delete edges \((t, x)\) and \((x, s)\)
• If not
  - Then there's no path
  - (Contrapositive: \((s, t)\)-path \(\Rightarrow\) cycle)
INDEPENDENT SET → CLIQUE
INDEPENDENT SET $\rightarrow$ CLIQUE
INDEPENDENT SET → CLIQUE
INDEPENDENT SET $\rightarrow$ CLIQUE
INDEP. SET $\rightarrow$ VERTEX COVER
3SAT $\rightarrow$ INDEP. SET

$$(\bar{x} \lor y \lor \bar{z})(x \lor \bar{y} \lor z)(x \lor y \lor z)(\bar{x} \lor \bar{y})$$
• If an independent set is found
  ▪ Make those values true
• If not
  ▪ Then there's no satisfying assignment
  ▪ (Contrapositive: assignment $\Rightarrow$ independent set)
SAT $\rightarrow$ 3SAT

$$(\bar{x} \lor y \lor z)(x \lor y \lor \bar{z} \lor \bar{w} \lor u \lor v)(w \lor z)(\bar{y} \lor \bar{w})$$
SAT $\rightarrow$ 3SAT

$$(a_1 \lor a_2 \lor y_1) (\overline{y}_1 \lor a_3 \lor y_2) (\overline{y}_2 \lor a_4 \lor y_3) \cdots (\overline{y}_{k-3} \lor a_{k-1} \lor a_k)$$
We need to convert the second clause:

\[(\overline{x} \lor y \lor z)(x \lor y \lor \overline{z} \lor \overline{w} \lor u \lor v)(w \lor z)(\overline{y} \lor \overline{w})\]
\[(x \lor y \lor \bar{z} \lor \bar{w} \lor u \lor v)\]

becomes:

\[(x \lor y \lor y_1)(\bar{y}_1 \lor \bar{z} \lor y_2)(\bar{y}_2 \lor \bar{w} \lor y_3)(\bar{y}_3 \lor u \lor v)\]
\[(\bar{x} \lor y \lor z)(x \lor y \lor \bar{z} \lor \bar{w} \lor u \lor v)(w \lor z)(\bar{y} \lor \bar{w})\]

becomes:
A satisfying assignment:

\[
(\overline{x} \lor y \lor z) \\
(x \lor y \lor y_1) \\
(y_1 \lor \overline{z} \lor y_2) \\
(y_2 \lor \overline{w} \lor y_3) \\
(y_3 \lor u \lor v) \\
(w \lor z) \\
(\overline{y} \lor \overline{w})
\]

\[
x := false \\
y := true \\
z := true \\
y_1 := false \\
y_2 := false \\
w := false \\
y_3 := false \\
u := false \\
v := false
\]
A satisfying assignment:

\[(\bar{x} \lor y \lor z)\]  \[x := false\]  \[w := false\]

\[(x \lor y \lor \bar{z} \lor \bar{w} \lor u \lor v)\]  \[y := true\]  \[u := false\]

\[(w \lor z)\]  \[z := true\]  \[v := false\]

\[(\bar{y} \lor \bar{w})\]
\[
\left\{ (a_1 \lor a_2 \lor \cdots \lor a_k) \right. \\
\text{is satisfied} \quad \iff \quad \left\{ \begin{array}{l}
\text{there is a setting of the } y_i \text{'s for which} \\
(a_1 \lor a_2 \lor y_1) (\overline{y}_1 \lor a_3 \lor y_2) \cdots (\overline{y}_{k-3} \lor a_{k-1} \lor a_k) \\
\text{are all satisfied}
\end{array} \right. 
\]

- If the RHS is satisfied
  - At least one of the \( a_i \) values must be true
- If the LHS is satisfied
  - Then some \( a_i \) is true
  - Set \( y_1, \ldots, y_{i-2} \) to true
  - Set the other \( y_j \) to false
ANY NP PROBLEM $\rightarrow$ SAT

- Any NP problem $\rightarrow$ Circuit SAT
- Circuit SAT $\leftrightarrow$ SAT
CIRCUIT SAT

output

AND

NOT

OR

AND

true

?

?

?
SAT $\rightarrow$ CIRCUIT SAT

- AND gates at the top
- OR gates in each clause
- literals are (NOT of) unknowns
CIRCUIT SAT $\rightarrow$ SAT

Gate $g$

$\text{true}$

$(g)$

$\text{false}$

$(\overline{g})$
\[
\begin{align*}
g & \quad \text{OR} \\
\quad h_1 & \quad h_2 \\
(g \lor \overline{h_2}) & \\
(g \lor \overline{h_1}) & \\
(\overline{g} \lor h_1 \lor h_2) & \\
\end{align*}
\]

\[
\begin{align*}
g & \quad \text{AND} \\
\quad h_1 & \quad h_2 \\
(\overline{g} \lor h_1) & \\
(\overline{g} \lor h_2) & \\
(g \lor \overline{h_1} \lor \overline{h_2}) & \\
\end{align*}
\]

\[
\begin{align*}
g & \quad \text{NOT} \\
\quad h & \\
(g \lor h) & \\
(\overline{g} \lor \overline{h}) & \\
\end{align*}
\]
NP PROBLEM $\rightarrow$ CIRCUIT SAT

- Given problem $A$ in NP
- $C(I, S)$ verifies solution $S$ in polynomial time
- Can be converted to a circuit with polynomial number of gates
- Bits of $S$ become unknowns
- Then satisfying assignments to unknowns $\iff$ solutions of $I$
NP-COMPLETENESS

A problem is NP-complete if:

- it's in NP
- all other problems in NP reduce to it

(NP-complete problems are the "hardest" in NP)
NP-COMPLETENESS

If we could solve even one NP-complete problem in polynomial time,

- We could solve any other problem in NP!
- Then $P = NP$
We showed:

- Hamiltonian path $\rightarrow$ Hamiltonian cycle
- Independent set $\rightarrow$ Vertex cover
- Independent set $\rightarrow$ Clique
- 3SAT $\rightarrow$ Independent set
- SAT $\rightarrow$ 3SAT
- Anything in NP $\rightarrow$ SAT
Others in DPV:

- 3SAT $\rightarrow$ 3D matching
- 3D matching $\rightarrow$ ZOE
- ZOE $\rightarrow$ Subset sum
- ZOE $\rightarrow$ ILP
- ZOE $\rightarrow$ Hamiltonian cycle
- Hamiltonian cycle $\rightarrow$ TSP
Given a program $P$ and input $x$, can we write an algorithm to see if $P$ will halt on input $x$?

$$\text{terminates}(p, x)$$
Let's define another function:

```
paradox(z):
    if terminates(z, z):
        infinite_loop()
```
What if we then run this?

paradox(paradox)
What if we then run this?

paradox(paradox)

- If terminate says paradox halts, it runs forever
- If terminate says paradox runs forever, it halts
UNSOLVABLE PROBLEMS

- The halting problem is undecidable
- If Halting $\rightarrow A$,  
  - if we had an algorithm for $A$,
  - we could use that to solve the halting problem
  - so $A$ must also be undecidable!
UNSOLVABLE PROBLEMS

$A$: Will program $P$ reference memory location 0x8000?

- Can we reduce Halting $\rightarrow A$?