NP hardness reductions

Lecture 14
Recap

- **P** = YES/NO questions that can be answered in polynomial time in input size (algorithm)

- **NP** = YES/No problems where YES instance can be verified in polynomial time

  - X is **NP-hard**: X in P implies P=NP

  - **Cook-Levin**: CircuitSAT NP hard
How to prove NP hardness

- To prove X is NP-hard:

  - **Step 1**: Pick a known NP-hard problem Y
  
  - **Step 2**: Assume for the sake of argument, a polynomial time algorithm for X.
  
  - **Step 3**: Derive a polynomial time algorithm for Y, using algorithm for X as subroutine.
  
  - **Step 4**: Contradiction
    
    Reduce FROM the problem I know about
    TO the problem I am curious about

Reduce Y to X
NP hardness

- Library of NP-hard problems

Let's assume the problem is easy and see what ridiculous consequences follow.
3SAT

- Look at boolean formulas in CNF

\[
\text{clause} \quad (a \lor b \lor c \lor d) \land (b \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor c \lor d) \land (a \lor \bar{b})
\]
3SAT

- Look at boolean formulas in CNF

\[
\begin{align*}
\text{clause} & : (a \lor b \lor c \lor d) \land (b \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor c \lor d) \land (a \lor \bar{b})
\end{align*}
\]

3SAT: exactly three literals per clause!

Every literal is a variable or the negation of a variable
3SAT

- Look at boolean formulas in CNF

\[
\text{clause} \quad \begin{array}{c}
(a \lor b \lor c \lor d) \\
(b \lor \bar{c} \lor \bar{d}) \\
(\bar{a} \lor c \lor d) \\
(a \lor \bar{b})
\end{array}
\land
\begin{array}{c}
\end{array}
\]

3SAT: exactly three literals per clause!
every literal is a variable or the negation of a variable

Not all boolean functions can be in the form \( a \lor b \lor c \lor d \)
3SAT

- Look at boolean formulas in CNF

\[(a \lor b \lor c \lor d) \land (b \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor c \lor d) \land (a \lor \bar{b})\]

Parse tree:

3SAT: exactly three literals per clause!

Not all boolean functions can be in the form

\[a \lor b \lor c \lor d\]
3SAT

- Look at Boolean formulas in CNF

\[(a \lor b \lor c \lor d) \land (b \lor \neg c \lor \neg d) \land (\neg a \lor c \lor d) \land (a \lor \neg b)\]

Parse tree:

3SAT special case of SAT. unlike when we are thinking about special cases in algorithms

2SAT there is algorithm!
NP hardness

• Poly time reduction from CircuitSAT.

• If there is a poly time algorithm to solve 3SAT, then there is poly time algorithm to solve CircuitSAT
NP hardness

- Poly time reduction from CircuitSAT.
- If there is a poly time algorithm to solve 3SAT, then there is poly time algorithm to solve CircuitSAT
Reduction Circuit SAT to 3SAT

• **Step 1**: Make gates binary (blows up size by at most 2x wires, if there were x wires). Poly time.

• **Step 2**: Transcribe

![Diagram of logical gates and expressions]
Reduction CircuitSAT to 3SAT

- **Step 1:** Make gates binary (blows up size by at most 2x wires, if there were x wires). Poly time.

- **Step 2:** Transcribe

\[ c = x \land y \]
\[ f = x \lor y \]
\[ g = \neg x \]
Reduction CircuitSAT to 3SAT

• **Step 3:** Make clauses in 3CNF
Reduction CircuitSAT to 3SAT

- **Step 3:** Make clauses in 3CNF

\[
\begin{align*}
a = b \land c & \iff (a \lor \bar{b} \lor \bar{c}) \land (\bar{a} \lor b) \land (\bar{a} \lor c) \\
a = b \lor c & \iff (\bar{a} \lor b \lor c) \land (a \lor \bar{b}) \land (a \lor \bar{c}) \\
a = \bar{b} & \iff (a \lor b) \land (\bar{a} \lor \bar{b}) \\
a \lor b & \iff (a \lor b \lor x) \land (a \lor b \lor \bar{x}) \\
a & \iff (a \lor x \lor y) \land (a \lor \bar{x} \lor y) \land (a \lor x \lor \bar{y}) \land (a \lor \bar{x} \lor \bar{y})
\end{align*}
\]
\[(y_1 \lor x_1 \lor \overline{x_4}) \land (y_1 \lor x_1 \lor z_1) \land (y_1 \lor x_1 \lor \overline{z_1}) \land (y_1 \lor x_4 \lor z_2) \land (y_1 \lor x_4 \lor \overline{z_2}) \land (y_2 \lor x_4 \lor z_3) \land (y_2 \lor x_4 \lor \overline{z_3}) \land (y_2 \lor x_4 \lor z_4) \land (y_2 \lor x_4 \lor \overline{z_4}) \land (y_3 \lor x_3 \lor \overline{y_2}) \land (y_3 \lor x_3 \lor z_5) \land (y_3 \lor x_3 \lor \overline{z_5}) \land (y_3 \lor y_2 \lor z_6) \land (y_3 \lor y_2 \lor \overline{z_6}) \land (y_4 \lor y_1 \lor x_2) \land (y_4 \lor \overline{x_2} \lor z_7) \land (y_4 \lor \overline{x_2} \lor \overline{z_7}) \land (y_4 \lor y_1 \lor \overline{z_8}) \land (y_4 \lor y_1 \lor z_8) \land (y_5 \lor x_2 \lor z_9) \land (y_5 \lor x_2 \lor \overline{z_9}) \land (y_5 \lor x_2 \lor z_{10}) \land (y_5 \lor x_2 \lor \overline{z_{10}}) \land (y_6 \lor x_5 \lor z_{11}) \land (y_6 \lor x_5 \lor \overline{z_{11}}) \land (y_6 \lor \overline{x_5} \lor z_{12}) \land (y_6 \lor \overline{x_5} \lor \overline{z_{12}}) \land (y_7 \lor y_3 \lor y_5) \land (y_7 \lor y_3 \lor \overline{z_{13}}) \land (y_7 \lor y_3 \lor \overline{z_{13}}) \land (y_7 \lor y_5 \lor z_{14}) \land (y_7 \lor y_5 \lor \overline{z_{14}}) \land (y_8 \lor y_4 \lor \overline{y_7}) \land (y_8 \lor y_4 \lor z_{15}) \land (y_8 \lor y_4 \lor \overline{z_{15}}) \land (y_8 \lor y_7 \lor z_{16}) \land (y_8 \lor y_7 \lor \overline{z_{16}}) \land (y_9 \lor y_8 \lor \overline{y_6}) \land (y_9 \lor y_8 \lor z_{17}) \land (y_9 \lor y_8 \lor \overline{z_{17}}) \land (y_9 \lor y_6 \lor z_{18}) \land (y_9 \lor y_6 \lor \overline{z_{18}}) \land (y_9 \lor \overline{z_{19}} \lor z_{20}) \land (y_9 \lor \overline{z_{19}} \lor \overline{z_{20}}) \land (y_9 \lor z_{19} \lor z_{20}) \land (y_9 \lor z_{19} \lor \overline{z_{20}})]

Although this formula may look a lot more ugly and complicated than the original circuit at first glance, it’s actually only a constant factor larger—every binary gate in the original
• Algo runs in poly time
  Proof:
  • Circuit satisfiable implies formula satisfiable
  • formula satisfiable implies circuit satisfiable

  even though the reduction goes one direction, the proof needs to go both directions
MAX Independent Set

- Input: a graph $G(V,E)$
- Output: Largest subset of vertices with no edges between them. (enough to find size)
How to prove NP hardness

• To prove X is NP-hard:

• **Step 1**: Pick a known NP-hard problem Y

• **Step 2**: Assume for the sake of argument, a polynomial time algorithm for X.

• **Step 3**: Derive a polynomial time algorithm for Y, using algorithm for X as subroutine.

• **Step 4**: Contradiction

Reduce FROM the problem I know about
TO the problem I am curious about
NP hardness

- Library of NP-hard problems

CircuitSAT

SAT

3SAT

Let’s assume the problem is easy and see what ridiculous consequences follow.
MAX Independent Set

- Input: a graph G(V,E)
- Output: Largest subset of vertices with no edges between them. (enough to find size)

Prove this is NP hard by reduction from 3SAT
MAX Independent Set

- Given an arbitrary 3CNF formula

- Build a graph G as follows

1) For every clause 3 vertices connected in a triangle

\[ x \lor y \lor \neg z \]

2) add edges between a literal and its negation

\[ z \quad \neg z \]
MAX Independent Set

- Input: a graph $G(V,E)$
- Output: Largest subset of vertices with no edges between them. (enough to find size)

$k$ clauses $\rightarrow$ $3k$ vertices

Graph has IS of size $k$ if and only if the formula is satisfiable
Claim:
Graph has IS of size $k$ if and only if the formula is satisfiable

2 steps to proof:
Step 1) Assume formula satisfiable
- Choose satisfying assignment $(a=1, b=1, c=1, d=0)$

$$(a \lor b \lor c) \land (b \lor \bar{c} \lor \bar{d}) \land (ar{a} \lor \bar{c} \lor d) \land (a \lor \bar{b} \lor d)$$

No long edges because every selected literal is true
and no edges between each triangle

G Has IS of size $k$!
MAX Independent Set

- Input: a graph G(V,E)
- Output: Largest subset of vertices with no edges between them. (enough to find size)

\[(a \lor b \lor c) \land (b \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor c \lor d) \land (a \lor \bar{b} \lor \bar{d})\]

k clauses \(\rightarrow\) 3k vertices

The graph has IS of size k if and only if the formula is satisfiable.
Claim:
Graph has IS of size k if and only if the formula is satisfiable

2 steps to proof:

Step 2) Suppose G has IS of size k. Then this IS contains at most one node per triangle so it has exactly one node per triangle. These nodes provide a satisfying assignment to the formula.
When you write reductions

- **Step 1**: Describe the algorithm (and it runs in poly time)
- **Step 2**: Prove one way
- **Step 3**: Prove the other way
NP hardness

- Library of NP-hard problems

Let’s assume the problem is easy and see what ridiculous consequences follow.
MAX Clique

- Input: a graph $G(V,E)$
- Output: Largest subset of vertices that are all pairwise connected

- Reduction from MAX-IS
- Assume poly time algorithm for MAX Clique
- Derive poly time algorithm for MAX IS
what is $G'$? Not the same graph as $G$
NP hardness

- Library of NP-hard problems

CircuitSAT

SAT

3SAT

MAX IS

MAX Clique

Let’s assume the problem is easy and see what ridiculous consequences follow
MIN Vertex Cover

- Input: a graph $G(V,E)$
- Output: Smallest set of vertices that touch every edge

- Reduction from MAX-IS
- Assume poly time algorithm for MIN Vertex Cover
what is $G'$? same graph as $G$
Output is different