Lecture # 19

CS 170
Spring 2021
Search Problems, P and NP

• Last time: Reductions \( A \rightarrow B \)
  • \( A \rightarrow B \) means can solve \( A \) using subroutine for \( B \)
  • \( B \) "easy" (poly-time) \( \Rightarrow \) \( A \) easy
  • \( A \) "hard" (no poly-time alg known) \( \Rightarrow \) \( B \) hard

• Goal—try to classify problems as easy or hard

• Def: A Binary Relation is a subset \( R \subseteq \{0,1\}^* \times \{0,1\}^* \) of pairs of finite bit strings, \((x,w)\) = \((\text{instance, witness})\)

• Def: \( \text{decide}(R) = \text{given instance } x, \text{ decide if } \exists w \text{ such that } (x,w) \in R \) (output = yes/no)

• Def: \( \text{search}(R) = \text{given instance } x, \text{ find a witness } w \text{ such that } (x,w) \in R \text{ if it exists, else "no"} \)

• \( \text{decide} \rightarrow \text{search}, \text{ other way too} \)
Search Problem - Example

• **Def**: A Binary Relation is a subset \( R \subseteq \{0,1\}^* \times \{0,1\}^* \) of pairs of finite bit strings, \((x,w)\) = (instance, witness)

• **Def**: \( \text{decide}(R) = \) given instance \( x \), decide if 
  \( \exists w \) such that \((x,w) \in R\) (output = yes/no)

• **Def**: \( \text{search}(R) = \) given instance \( x \), find a witness \( w \) such that \((x,w) \in R\) if it exists, else "no"

• **Ex**: Max Flow
  - **Instance**: \( x = (G,s,t) \), \( G = \text{network}, s = \text{source}, t = \text{sink} \)
  - **Witness**: \( w = \text{max}-\text{flow} \)
  - **Decide \( (R) = yes \) (nothing to do)
  - **Search \( (R) = solve, using Ford-Fulkerson \)**
Does decide(R) always exist?

• **No: Halting Problem**
  • instance $x =$ computer program, no witness
  • $R(x, \text{null}) = 1$ if $x$ halts, else 0
  • Undecidable (no algorithm exists, CS70)

• Focus on binary relations $R$ that are efficiently verifiable:
  • Given $(x, w) =$ (instance, witness) there exists an algorithm $V_R(x, w) = R(x, w) \in \{0, 1\}$ with runtime $O(\text{poly}(1 \cdot l))$, where $1 \cdot l = \text{size}(x)$

• New question: given $V_R$, how hard is decide($R$)?
  • Cost (decide($R$)) at most $2^{\text{poly}(1 \cdot l)}$
    For all $w \in \{0, 1\}^{\text{poly}(1 \cdot l)}$, if $V_R(x, w) = 1$ output yes
    Output no
Defining P and NP

• \( P \) = "complexity class" of all relations \( R \) such that \( \text{decide}(R) \) costs \( \text{poly}(\text{size}(x)) \) (\( P \) = "polynomial")

• \( NP \) = all relations \( R \) such that given \( x \), \( \exists \ w \) of size \( |w| = \text{poly}(\text{size}(x)) \), so \( V_R(x, w) \) costs \( \text{poly}(\text{size}(x)) \) when \( R(x, w) = 1 \) for some \( w \)

• Example: if \( V_R(x, w) \) costs \( \text{poly}(\text{size}(x)) \)

• \( P \leq NP \)

• Does \( P = NP ? \) Win $1M Millennium Prize!
Defining NP-hard and NP-complete

- \( P \) = "complexity class" of all relations \( R \) such that \( \text{decide}(R) \) costs \( \text{poly}(1 \times 1) \) (\( P \) = "polynomial")

- NP = all relations \( R \) such that given \( x \), \( \exists w \) of size \( |w| = \text{poly}(1 \times 1) \), so \( V_R(x, w) \) costs \( \text{poly}(1 \times 1) \) when \( R(x, w) = 1 \) for some \( w \)

- Example: if \( V_R(x, w) \) costs \( \text{poly}(1 \times 1) \)

- Def: problem \( A \) is NP-hard if \( B \to A \) for all \( B \in \text{NP} \)

- Def: problem \( A \) is NP-complete if \( A \) NP-hard and in \( \text{NP} \)

- NP-complete problems exist!
CSAT is NP-complete

- **Def**: CSAT is binary relation $R_{\text{CSAT}}$ where $(C=\text{circuit}, w) \in R_{\text{CSAT}}$ if $C(w) = 1$

- **Claim**: CSAT is NP-complete
  
  CSAT in NP because $C(w)$ efficiently computable (just evaluate circuit)
  
  CSAT NP-hard because everything in NP can be reduced to it:

  \[ B \in \text{NP} \implies \exists \text{efficient verifier } V_B(x_B, w_B) \]

  **Reduction**: $x_B \rightarrow$ preprocess to make output circuit for $V_B(x_B, \ast) \rightarrow$ CSAT $\rightarrow$ $w$ that makes $C(w)=1 \rightarrow x_B$ solvable or not
Reducing CSAT to simpler problems: SAT

- Recall what a circuit is: DAG of gates
- Convert circuit to CNF = conjunctive normal form = and of clauses like \((x_1 \lor \overline{x_2} \lor x_3)\)

- One variable per gate in DAG:
  - \(x \lor y\) becomes \((z \lor \overline{x})\)
  - \(x \land y\) becomes \((z \land \overline{y})\)
  - \(\overline{x} \lor \overline{y}\) becomes \((\overline{x} \lor \overline{y})\)
  - \(\overline{\overline{x}}\) becomes \((z \lor \overline{x})\)
  - \(\frac{1}{T}, \frac{1}{F}\) become \((x), (\overline{y})\)

SAT NP-complete
Reducing SAT to simpler case: 3SAT

• Want to show "simple" problems are NP-complete, to make them easier to use to show others are

• 3SAT: SAT with \( \leq 3 \) variables per clause

• Ex: \( (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_4 \lor x_5) \land (x_2 \lor x_6) \land \ldots \)

• Trick to convert \( (a_1 \lor a_2 \lor a_3 \ldots \lor a_k) \) to 3SAT

• Introduce new variables \( y_1, \ldots, y_{k-3} \)

• Convert to

\[
(a_1 \lor a_2 \lor y_1) \land (\overline{y}_1 \lor a_3 \lor y_2) \land (\overline{y}_2 \lor a_4 \lor y_3) \ldots (\overline{y}_{i-2} \lor a_i \lor y_{i-1}) \ldots (\overline{y}_{k-3} \lor a_{k-1} \lor y_{k-2})
\]

• If all \( a_i = F \), making above expression = \( T \) \( \Rightarrow \) \( y_1 = T \Rightarrow y_2 = T \Rightarrow \ldots \overline{y}_{k-3} = F \Rightarrow \) expression = \( F \)

• If \( a_i = T \), set \( y_i = y_{i-1} = \ldots = y_{i-2} = T \), \( y_{i-1} = y_i = \ldots = F \) to make all clauses = \( T \)
More NP-complete problems

All of NP

\[ \text{CSAT} \]

\[ \text{SAT} \]

\[ \text{3SAT} \]

- Independent Set
- Vertex Cover
- Clique

- 3D Matching
- ZOE
- ILP
- Rudrata/Hamiltonian Cycle

- TSP
Reducing 3SAT to Independent Set (IS)

- **IS**: Does graph $G$ have $\geq g$ unconnected vertices?

- **Ex**: $(\overline{x} \lor y \lor \overline{z}) \land (x \lor \overline{y} \lor z) \land (x \lor y \lor \overline{z}) \land (\overline{x} \lor y \lor z)$

- Transform to graph where
  - each variable is a vertex
  - each clause is a clique
  \[ \Rightarrow \text{pick at most 1 vertex per clause for IS} \]
  - add edge between every $(v, \overline{v})$
  \[ \Rightarrow \text{can choose either } v \text{ or } \overline{v} \text{ for IS} \]

- Is there an IS of size # clauses?
  - Yes $\Rightarrow$ one vertex per clause, set $= T$
  \[ \Rightarrow \text{each clause } = T \]

- Is expression satisfiable?
  - Yes $\Rightarrow \geq 1$ vertex per clause $= T$
  \[ \Rightarrow \text{choose 1 from each clause, get IS} \]
Reducing Independent Set (IS) to...

• Vertex Cover (VC): Subset $S \subseteq V$ that touch every edge
  - Fact: $S$ is a VC iff $V-S$ is an IS

• Clique (Cl): Subset $S \subseteq V$ that is fully connected
  - Fact: $S$ is a clique in $G=(V,E)$ iff $S$ is an IS in $G'=(V,\overline{E})$, $\overline{E}$ = all edges not in $E$
Did I forget to prove anything?

• Need to confirm all NP-complete problems are in NP, not just NP-hard

• Easy to confirm a witness $w$ is correct for an instance $x$:
  - (3)SAT: plug values into formula, evaluate it
  - IS: given a list of vertices, confirm no edges connect them
  - VC: given a list of vertices, confirm all edges touch one of them
  - Clique: given a list of vertices, confirm each pair connected