Intro to Theory of Computation

LECTURE 24

Last time
• Polynomial-time reductions
• NP-completeness

Today
• Cook-Levin Theorem
• Examples of NP-complete languages

Sofya Raskhodnikova

3/17/2016
Implication of poly-time reductions

**Theorem.** If

- $B$ is $\textbf{NP}$-complete,
- $C \in \textbf{NP}$ and
- $B \leq_{p} C$

then $C$ is $\textbf{NP}$-complete.

**Theorem.** If $B$ is $\textbf{NP}$-complete and $B \in \textbf{P}$ then $\textbf{P} = \textbf{NP}$.

(So, if $B$ is $\textbf{NP}$-complete and $\textbf{P} \neq \textbf{NP}$ then there is no poly-time algorithm for $B$.)
The class NP

**NP** is the class of languages that have polynomial-time verifiers.
Recall: The running time of a verifier $V(\langle w, c \rangle)$ is measured only in terms of length of $w$.

If we allowed a verifier to run in time polynomial in the length of $\langle w, c \rangle$, the class NP would

A. be smaller
B. be the same
C. contain more (decidable only) languages
D. contain more (even undecidable) languages
E. none of the above
A language $B$ is **NP-complete** if

1. $B \in \text{NP}$
2. $B$ is **NP-hard**, i.e., every language in NP is poly-time reducible to $B$. 

A diagram illustrating the relationship between $P$, $NP$, and $B$ (NP-complete).
An NP-complete problem

$\text{BA}_{\text{NTM}} = \{ \langle M, x, t \rangle \mid M \text{ is an NTM that accepts } x \text{ in at most } t \text{ steps} \}$

Theorem. $\text{BA}_{\text{NTM}}$ is NP-Complete.

1. $\text{BA}_{\text{NTM}} \in \text{NP}$:
   
   The list of guesses $M$ makes to accept $x$ in $t$ steps is the certificate that $\langle M, x, t \rangle \in \text{BA}_{\text{NTM}}$.

2. For all $A \in \text{NP}$, $A \leq_p \text{BA}_{\text{NTM}}$.

   $A \in \text{NP}$ iff there is an NTM $N$ for $A$ that runs in time $O(n^k)$.
   Let $f_A(w) = \langle N, w, c \mid w \mid^k \rangle$.
   
   $\langle N, w, c \mid w \mid^k \rangle \in \text{BA}_{\text{NTM}} \iff N \text{ accepts } w \iff w \in A$. 

Technical detail: $n$ denotes $1^n$. 

11/28/2018
CIRCUIT-SAT is similar to 3SAT, but it is about circuits, not formulas.

**Theorem.** CIRCUIT-SAT is NP-complete. [Cook `71, Levin `73]

A circuit is built from AND, OR and NOT gates.

A circuit is **satisfiable** if one can set the circuit inputs, so that the output is 1.

CIRCUIT-SAT = \{\langle C \rangle | C \text{ is a satisfiable circuit} \}. 

```
output

?·?

?·?

?·?

?·?

?·?
```

yes: 1 0 1
Theorem. 3SAT is NP-complete. (Book)
Establishing NP-completeness

- Once we establish first "natural" NP-complete problems, others fall like dominoes.

- Recipe to establish NP-completeness of problem Y.
  - Step 1. Show that Y is in NP.
  - Step 2. Choose an NP-complete problem X (e.g., 3SAT) and prove that $X \leq_p Y$. 

All problems below are NP-complete and hence poly-time reduce to one another!
A clique in a graph is a set of nodes, where every two nodes are connected by an edge.

\[
\text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ is an undirected graph that contains a clique with } k \text{ nodes} \}
\]
Prove: CLIQUE is NP-complete

• CLIQUE = \{ \langle G, k \rangle \mid G \text{ is an undirected graph that contains a clique with } k \text{ nodes} \}

1. Show that CLIQUE is in NP.
   
   Certificate: clique of size \( k \)

2. Show that 3SAT \leq_p CLIQUE.
   
   “On input \( \langle \phi \rangle \), where \( \phi \) is a 3cnf formula, Output a graph \( G \) and a number \( k \), where… ”  

(on the board)
SUBSET-SUM

- $SSUM = \{ \langle S, t \rangle \mid S = \{ x_1, \ldots, x_r \},$
  and for some $\{ y_1, \ldots, y_{r'} \} \subseteq \{ x_1, \ldots, x_r \},$
  we have $y_1 + \cdots + y_{r'} = t \}$

- Examples: $\langle \{5, 7, 23\}, 28 \rangle \in SSUM$
  $\langle \{5, 7, 25\}, 28 \rangle \notin SSUM$
Prove: SSUM is NP-complete

1. Show that SSUM is in NP.
   
   **Certificate**: subset \( y_1, \ldots, y_{r'} \) of \( S \) that sums up to \( t \).

2. Show that 3SAT \( \leq_p \) SSUM.
   “On input \( \langle \phi \rangle \), where \( \phi \) is a 3cnf formula, with variables \( x_1, \ldots, x_\ell \) and clauses \( c_1, \ldots, c_k \)
   
   • Output a set \( S \) of numbers and a target number \( t \ldots \)”

   (on the board)
Some NP-Complete Problems

- Six basic genres of NP-complete problems and paradigmatic examples.
  - Packing problems: SET-PACKING, INDEPENDENT SET.
  - Covering problems: SET-COVER, VERTEX-COVER.
  - Constraint satisfaction problems: SAT, 3-SAT.
  - Sequencing problems: HAMILTONIAN-CYCLE, TSP.
  - Partitioning problems: 3D-MATCHING, 3-COLOR.
  - Numerical problems: SUBSET-SUM, KNAPSACK.

- Most NP problems are either known to be in P or NP-complete.
- Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.
An encyclopedia of NP-complete problems
``We need an efficient algorithm that constructs a design of iThingy that meets the maximum # of requirements at lowest cost.''

``I thought about it for weeks, but I can’t come up with an efficient algorithm. I guess I’m just too dumb.’’

“I can’t find an efficient algorithm, but neither can all these famous people.”
3SAT is an example of a problem that cannot be solved by an algorithm.

A. True
B. False
C. It is an open question
D. None of the above
3SAT is an example of a problem that cannot be solved by a **polynomial time** algorithm.

A. True  
B. False  
C. It is an open question  
D. None of the above