Intro to Theory of Computation

LECTURE 25

Last time
• Polynomial-time reductions
• NP-completeness

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

Sofya Raskhodnikova
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 $B$. 
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 = \{ ⟨C⟩ | C is a satisfiable circuit \}.

```
output

hard-coded inputs

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 = \{<G, k> | G is an undirected graph that contains a clique with k 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 = \{ ⟨S, t⟩ \mid S = \{x_1, \ldots, x_r\},
               and for some \{y_1, \ldots, y_{r′}\} ⊆ \{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: \textit{SSUM} is NP-complete

- \textit{SSUM} = \{ \langle S, t \rangle \mid S = \{x_1, \ldots, x_r\}, \text{ and for some } \{y_1, \ldots, y_r\} \subseteq \{x_1, \ldots, x_r\}, \text{ we have } y_1 + \ldots + y_r = t \}

1. Show that \textit{SSUM} is in NP.

   \textbf{Certificate:} subset \( y_1, \ldots, y_r \) of \( S \) that sums up to \( t \).

2. Show that \textit{3SAT} \( \leq_p \textit{SSUM} \).
   “On input \( \langle \phi \rangle \), where \( \phi \) is a 3cnf formula, with variables \( x_1, \ldots, x_\ell \) and clauses \( c_1, \ldots, c_k \)

   \begin{itemize}
   \item Output a set \( S \) of numbers and a target number \( t \)…”
   \end{itemize}

   (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.