Lecture 23: May 7, 2001

Today (and next lecture):

- P and NP (or, interconnectedness of all things)
Have seen so far

- Algorithms for lots of problems
  - Running times $O(n) \ldots O(nm^2)$
  - I.e., polynomial in the input size
- Can we solve all (or most) interesting problems in polynomial time?
- Not really ...
Example problems

• Travelling Salesperson Problem (TSP):
  – Input: Undirected graph with lengths on edges.
  – Output: Smallest tour that visits every vertex exactly once.
  – Best running time: $O(n2^n)$.

• Clique:
  – Input: Undirected graph $(V, E)$
  – Output: Largest subset $C \subseteq V$ such that every pair of vertices in $C$ are adjacent to each other.
  – Best running time: $n^{O(n)}$. 
What can we do?

- Spend more time working on algorithms for those problems - unlikely to succeed
- Prove that there are no polynomial time algorithms for those problems
  - Would be great
  - Seems really hard
  - Best lower bound for any natural problem:
    - $\Omega(n^2)$ for restricted computational model
    - $4.5n$ for realistic computational model
- Show that those hard problems are computationally equivalent
  - If any of them can be solved in poly time, all of them can be solved in poly time
  - Works for at least 10,000 of hard problems
Benefits of showing the equivalence

- Combines research efforts
- Justifies inability to find fast algorithms. I.e., “I might be stupid, but then so are all the other famous people!”
- One day, somebody might show a lower bound for \textit{one} problem. Then we will \textit{know} that \textit{all} are hard.
Class of problems of interest

- Decision problems: answer YES or NO. E.g., “is there a TSP tour with length $\leq K$ ?”
- Solvable in *nondeterministic polynomial time* (NP). Intuitively: the solution can be *verified* in polynomial time. E.g., if someone gives us a tour $T$, we can verify if $T$ is a tour in $G$ of length $\leq K$. Therefore, TSP is in NP.
**P and NP**

A problem $\Pi$ is solvable in poly time (or $\Pi \in P$), if there is a poly time algorithm $V(\cdot)$, such that for any input $x$

$$\Pi(x) \text{ is YES if and only if } V(x) = \text{YES}$$

A problem $\Pi$ is solvable in nondeterministic poly time (or $\Pi \in NP$), if there is a poly time *verification* algorithm $V(\cdot, \cdot)$, such that for any input $x$

$$\Pi(x) \text{ is YES if and only if there exists certificate } y \text{ of size } \text{poly}(|x|) \text{ such that } V(x, y) = \text{YES}$$
Examples

• TSP: $V(x, y)$ interprets $x$ as a graph $G$, $y$ as a tour $T$, and checks if the length of $T$ is $\leq K$

• Clique: $V(x, y)$ interprets $x$ as a graph $G$, $y$ as a set $C$, and checks if all pairs of vertices in $C$ are adjacent

• ...

Thus, Clique and TSP are in NP.
Reductions

A problem $\Pi$ is (poly time) reducible to problem $\Pi'$, if there exists a poly time computable function $f$ which maps inputs $x$ of $\Pi$ into inputs of $\Pi'$, such that for any $x$

$$\Pi(x) = \Pi'(f(x))$$

Fact: if $\Pi'$ is poly time solvable, and $\Pi$ is poly time reducible to $\Pi'$, then $\Pi$ is also poly time solvable.

Thus, if $\Pi$ is poly time reducible to $\Pi'$, it means $\Pi$ is not harder than $\Pi'$. 
Recap

- A class of problems, i.e., NP
- Example problems in NP: TSP and Clique
- Notion of reduction

Our goal: show equivalence between “hard” problems.

\[ P_1 \quad P_2 \]
\[ P_4 \]
\[ P_3 \quad P_5 \]

Options:
- Reduce every problem to every other problem
- Spanning tree of reductions (each “edge” two way)
- Show all problems in NP are reducible to a fixed $\Pi$. To show $\Pi'$ is “hard”, reduce $\Pi$ to $\Pi'$. 
The problem $\Pi$

Satisfiability problems:

- **SAT:**
  - Input: $n$ Boolean variables $x_1, \ldots, x_n$ and a formula $\phi$ with $m$ clauses, e.g.,
    \[
    x_1 \lor x_2 \lor x_5, \ x_1 \lor \neg x_2, \ldots
    \]
  - Goal: Check if there exists a TRUE/FALSE assignments of variables such that every clause is satisfied, i.e., every clause above has one term set to TRUE.

- **3SAT:** as above, but at most 3 variables per clause
Cook’s theorem

**Theorem:** For any $\Pi$ in NP, $\Pi$ is poly time reducible to SAT.

**Definition:** a problem $\Pi$ such that any problem $\Pi' \in$ NP is poly time reducible to $\Pi$, is called *NP-hard*.

**Definition:** an NP-hard problem $\Pi$ which belongs to NP is called *NP-complete*.

**Corollary:** SAT is NP-complete.
Hardness of 3SAT

Reduce SAT to 3SAT:

• Need to take any SAT formula $\phi$, and transform them into 3SAT formula $f(\phi)$, so that $\phi$ is satisfiable if and only if $f(\phi)$ is satisfiable

• Reduction: for each clause $C_i = x_1 \lor x_2 \lor \ldots \lor x_k$, create

$f(C_i) = x_1 \lor y_2 \ldots k, \neg y_2 \ldots k \lor x_2 \lor y_3 \ldots k, \ldots, \neg y_k \ldots k \lor x_k$

(y’s are different for each clause) and define

$f(\phi) = f(C_1), \ldots, f(C_m)$

• Clearly, reduction is poly time

• Correctness: need to show $\phi$ is satisfiable if and only if $f(\phi)$ is satisfiable
Correctness

- Assume $\phi$ satisfiable. Then we can set $y$'s so that $\phi'$ is satisfiable. Consider $C = x_1 \lor \ldots \lor x_k$. Assume $x_i$ is set to TRUE. Then we set $y_{1..k}, \ldots y_{i..k}$ to TRUE, and the rest to FALSE. Check for yourself this satisfies the formula:

$$x_1 \lor y_{2..k}, \neg y_{2..k} \lor x_2 \lor y_{3..k}, \ldots, \neg y_{k..k} \lor x_k$$

- Assume $f(\phi)$ satisfiable. By contradiction, assume that the same assignment does not satisfy $\phi$. In particular, let $C_i = x_1 \lor \ldots \lor x_k$ be FALSE. Then

$$x_1 \lor y_{2..k}, \neg y_{2..k} \lor x_2 \lor y_{3..k}, \ldots, \neg y_{k..k} \lor x_k$$

is equivalent to

$$y_{2..k}, \neg y_{2..k} \lor y_{3..k}, \ldots, \neg y_{k..k}$$

which is always FALSE - contradiction!

Corollary: 3SAT is NP-hard (and since it is in NP, it is also NP-complete)