Objectives

- Discussion of "The Status of P and NP"

Review

- What does "polynomial time reducible" mean?
  - What is it relating?
- What is a way of showing that one algorithm is polynomial time reducible to another?

Review: Polynomial-Time Reduction

Suppose we could solve Y in polynomial-time. What else could we solve in polynomial time?

- Reduction. Problem *X polynomial reduces to* problem *Y* if arbitrary instances of problem *X* can be solved using:
  - Polynomial number of standard computational steps, ***plus***
  - Polynomial number of calls to **oracle** that solves problem Y
    - Assume have a black box that can solve Y

For X + Y

- Notation: $X \leq_P Y$
  - "X is polynomial-time reducible to Y"
- Conclusion: If Y can be solved in polynomial time and $X \leq_P Y$, then X can be solved in polynomial time.

Review: Polynomial-Time Reduction

- Purpose. Classify problems according to *relative difficulty*.
- Design algorithms. If $X \leq_P Y$ and Y can be solved in polynomial-time, then X can also be solved in polynomial time.
- Establish intractability. If $X \leq_P Y$ and X cannot be solved in polynomial-time, then Y cannot be solved in polynomial time.
- Establish equivalence. If $X \leq_P Y$ and $Y \leq_P X$, we use notation $X \equiv_P Y$.

Review: Basic Reduction Strategies

- Reduction by simple equivalence
- Reduction from special case to general case
- Reduction by encoding with gadgets

Status of the P vs NP Problem

From *Numbers*

Charlie: Dad, uhm.. I've been working on a problem. P vs. NP. It can't be solved.
Alan: I think you knew that when you started.
Charlie: I could work on it forever. Constantly pushing forward, still never reaching an end.
…

Status of the P vs NP Problem

- What is P? NP? What is the problem?
- What is a common feature of NP?

NP

- Problems that no polytime algorithm has been found AND have not proven that no polytime algorithm exists
  - A little more …
- Examples:

Name	Description
Hamiltonian circuit	Determine whether a given graph has a Hamiltonian circuit (a path that starts & ends at the same vertex and passes through all other vertices exactly once)
Traveling salesman	Find the shortest tour through n cities with known positive integer distances between them (each city once)
Graph coloring	Find a graph's chromatic number: smallest # of colors that need to be assigned to the graph's vertices so that no 2 adjacent vertices are assigned the same color.

Common Feature

- Computationally difficult BUT checking if a proposed solution solves problem *can* be solved in polynomial time
- Example: easy to check if a proposed list of vertices is a Hamiltonian circuit for a given graph with n vertices

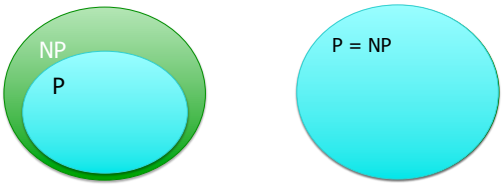
Nondeterministic Algorithm

- Input: instance of a decision problem

1. Nondeterministic "guessing" stage: guess a solution to problem I
2. Deterministic "verification" stage: outputs yes if solution is a solution to the problem

NP: A nondeterministic algorithm whose verification stage has a polynomial runtime.

What We're Trying To Figure Out



P ⊆ NP P = NP

Status of the P vs NP Problem

- What are the consequences of NP Completeness?
- What if P = NP?
- How have people tried to prove P ≠ NP?
  - Limitations? Still in progress?

Final

- Usual rules
- Due next Friday, 5 p.m. (end of exams)
- Can use book, notes, handouts, my lecture notes, me (limited)
- No outside resources