Algorithms

NP Completeness Continued:

Reductions

Review: **P** And **NP** Summary

- **P** = set of problems that can be solved in polynomial time
- **NP** = set of problems for which a solution can be verified in polynomial time
- **P** $\subseteq$ **NP**
- Open question: Does **P = NP**?

Review: Reduction

- A problem P can be *reduced* to another problem Q if any instance of P can be rephrased to an instance of Q, the solution to which provides a solution to the instance of P
  - This rephrasing is called a *transformation*
- Intuitively: If P reduces in polynomial time to Q, P is "no harder to solve" than Q

Review: NP-Hard and NP-Complete

- If P is *polynomial-time reducible* to Q, we denote this $P \leq_p Q$
- Definition of NP-Hard and NP-Complete:
  - If all problems $R \in$ **NP** are reducible to P, then P is *NP-Hard*
  - We say P is *NP-Complete* if P is NP-Hard and $P \in$ **NP**
- If $P \leq_p Q$ and P is NP-Complete, Q is also NP- Complete

Review: Proving NP-Completeness

- *What steps do we have to take to prove a problem Q is NP-Complete?*
  - Pick a known NP-Complete problem P
  - Reduce P to Q
    - Describe a transformation that maps instances of P to instances of Q, s.t. "yes" for Q = "yes" for P
    - Prove the transformation works
    - Prove it runs in polynomial time
  - Oh yeah, prove Q $\in$ **NP** (*What if you can't?*)

Directed Hamiltonian Cycle ⇒ Undirected Hamiltonian Cycle

- *What was the hamiltonian cycle problem again?*
- For my next trick, I will reduce the *directed hamiltonian cycle* problem to the *undirected hamiltonian cycle* problem before your eyes
  - *Which variant am I proving NP-Complete?*
- Draw a directed example on the board
  - *What transformation do I need to effect?*

Transformation: Directed ⇒ Undirected Ham. Cycle

- Transform graph G = (V, E) into G' = (V', E'):
  - Every vertex v in V transforms into 3 vertices v^1, v^2, v^3 in V' with edges (v^1,v^2) and (v^2,v^3) in E'
  - Every directed edge (v, w) in E transforms into the undirected edge (v^3, w^1) in E' (draw it)
  - *Can this be implemented in polynomial time?*
  - *Argue that a directed hamiltonian cycle in G implies an undirected hamiltonian cycle in G'*
  - *Argue that an undirected hamiltonian cycle in G' implies a directed hamiltonian cycle in G*

Undirected Hamiltonian Cycle

- Thus we can reduce the directed problem to the undirected problem
- *What's left to prove the undirected hamiltonian cycle problem NP-Complete?*
- *Argue that the problem is in* **NP**

Hamiltonian Cycle $\Rightarrow$ TSP

- The well-known *traveling salesman problem*:
  - Optimization variant: a salesman must travel to n cities, visiting each city exactly once and finishing where he begins. How to minimize travel time?
  - Model as complete graph with cost $c(i,j)$ to go from city i to city j
- *How would we turn this into a decision problem?*
  - A: ask if $\exists$ a TSP with cost $< k$

Hamiltonian Cycle ⇒ TSP

- The steps to prove TSP is NP-Complete:
  - Prove that TSP ∈ **NP** (*Argue this*)
  - Reduce the undirected hamiltonian cycle problem to the TSP
    - So if we had a TSP-solver, we could use it to solve the hamilitonian cycle problem in polynomial time
    - *How can we transform an instance of the hamiltonian cycle problem to an instance of the TSP?*
    - *Can we do this in polynomial time?*

The TSP

- Random asides:
  - TSPs (and variants) have enormous practical importance
    - E.g., for shipping and freighting companies
    - Lots of research into good approximation algorithms
  - Recently made famous as a DNA computing problem