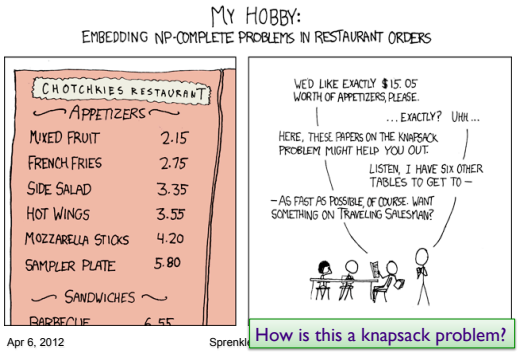
Objectives

- Reducibility
- Conclusions

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?
- What does NP-Complete mean?

Now you "get" this xkcd comic



How is this a knapsack problem?

"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.
…

Fun Fact: Connecting Chapters 7 and 8

- Karp, of the Edmonds-Karp algorithm (max-flow problem on networks), published a paper in complexity theory on "Reducibility Among Combinatorial Problems", in which he proved 21 Problems to be NP-complete

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 $X \equiv_P Y$.

Considering $X \leq_P Y$

- Need to be careful putting X in terms of Y
- Make sure you're not putting an easy problem (X) in terms of a hard problem (Y)
  - While you could do that, what does that do for you?
  - Just because Y is hard to solve does *not* mean that X is hard to solve

Review: Basic Reduction Strategies

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

Basic Reduction Strategies

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

Set Cover

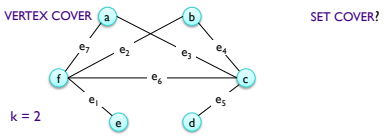
- SET COVER: Given a set U of elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and an integer k, does there exist a collection of size ≤ k of these sets whose union is equal to U?
- Sample application
  - m available pieces of software
  - Set U of n capabilities that we would like our system to have
  - The i^{th} piece of software provides the set $S_i \subseteq U$ of capabilities
  - **Goal**: achieve all n capabilities using fewest pieces of software
- Ex:

U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2
S_1 = {3, 7}
S_2 = {3, 4, 5, 6}
S_3 = {1}
S_4 = {2, 4}
S_5 = {5}
S_6 = {1, 2, 6, 7}

Choose S_2 and S_6

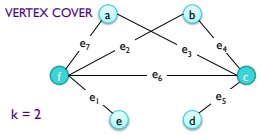
Vertex Cover Reduces to Set Cover

- Claim. VERTEX-COVER $\leq_P$ SET-COVER
- Pf. Given a VERTEX-COVER instance G = (V, E), k, we construct a set cover instance whose size equals the size of the vertex cover instance.
  - …



Vertex Cover Reduces to Set Cover

- Claim. VERTEX-COVER $\leq_P$ SET-COVER
- Pf. Given a VERTEX-COVER instance G = (V, E), k, we construct a set cover instance whose size equals the size of the vertex cover instance.
  - …

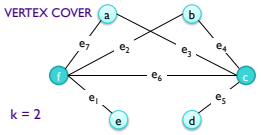


SET COVER

U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2
S_a = {3, 7}
S_b = {2, 4}
S_c = {3, 4, 5, 6}
S_d = {5}
S_e = {1}
S_f = {1, 2, 6, 7}

Vertex Cover Reduces to Set Cover

- Claim. VERTEX-COVER $\leq_P$ SET-COVER
- Pf. Given a VERTEX-COVER instance G = (V, E), k, we construct a set cover instance whose size equals the size of the vertex cover instance.
- Construction.
  - Create SET-COVER instance:
    - k = k, U = E, S_v = {e ∈ E : e incident to v }
  - Set-cover of size ≤ k iff vertex cover of size ≤ k. ▪



SET COVER

U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2
S_a = {3, 7}
S_b = {2, 4}
S_c = {3, 4, 5, 6}
S_d = {5}
S_e = {1}
S_f = {1, 2, 6, 7}

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 an independent set or a vertex cover for a graph

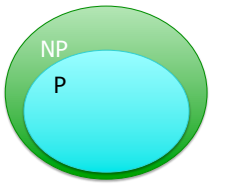
Nondeterministic Algorithm

- Input: instance of a decision problem

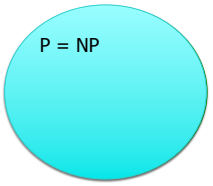
1. Nondeterministic "guessing" stage: guess a solution to problem
2. Deterministic "verification" stage: outputs yes if solution is a solution to the problem

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?

Exam 2

Average Pct	Median Pct
83.5	81.1

- Common issues
  - Missing analysis of runtime
  - Missing proofs
    - How do you prove a greedy algorithm?
  - Incorrect/inefficient/unclear algorithms
    - What needs to be returned/output?

PS8

- Dynamic programming
- Expected followed examples from class/book
  - Use memoization
  - Process to find *solution* after finding *value*

Final

- Usual rules
- Due next Friday, 5 p.m. (end of exams)
- Can use book, notes, handouts, my lecture notes, me (limited)
  - "The status of the P versus NP problem"
  - No other outside resources
- Office hours: Monday and Tuesday afternoons
  - Others by appointment
- Evaluations due Monday at midnight on Sakai (tests and quizzes)
  - Last checked: 4 submissions of evaluations