Objectives

- Oh, the places you've been!
- Oh, the places you'll go!

Now, everything comes down to expert knowledge of **algorithms** and **data structures**. If you don't speak fluent **O-notation**, you may have trouble getting your next job at the technology companies in the forefront.
-- Larry Freeman

Algorithm Design Patterns

- What are some approaches to solving problems?
- How do they compare in terms of difficulty?

Algorithm Design Patterns

- Greedy
- Divide-and-conquer
- Dynamic programming
- Duality/network flow

Course Objectives: Given a problem…

You'll recognize when to try an approach
-AND, when to bail out and try something different
Know the steps to solve the problem using the approach
- e.g., breaking it into subproblems, sorting possibilities in some order
Know how to ***analyze*** the run time of the solution
- e.g., solving recurrence relation

Algorithm Design Patterns

- Greedy
- Divide-and-conquer
- Dynamic programming
- Duality/network flow
- Reductions – Chapter 8
- Local search – Chapter 12
- Randomization – Chapter 13

What Was Our Goal In Finding a Solution?

Polynomial Time → Efficient

POLYNOMIAL-TIME REDUCTIONS

Classify Problems According to Computational Requirements

Fundamental Question:
Which problems will we be able to solve in practice?

Classify Problems According to Computational Requirements

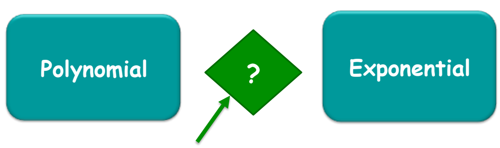
Which problems will we be able to solve in practice?

- Working definition. [Cobham 1964, Edmonds 1965, Rabin 1966] Those with polynomial-time algorithms.

Yes	Probably no
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover
Primality testing	Factoring

Classify Problems

Classify problems according to those that can be solved in polynomial-time and those that cannot.



Frustrating news: *Many* problems have defied classification.
Chapter 8. Show that problems are "computationally equivalent" and appear to be manifestations of one *really hard* problem.

Examples:
- *Given a Turing machine, does it halt in at most k steps?*
- *Given a board position in an n-by-n generalization of chess, can black guarantee a win?*

Polynomial-Time Reduction

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

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

NP Complete Problems

- Problems from many different domains whose complexity is unknown
- NP-completeness and proof that all problems are equivalent is **POWERFUL**!
  - All open complexity questions ➔ **ONE** open question!
- What does this mean?
  - "Computationally hard for practical purposes but we can't prove it"
  - If you find an NP-Complete problem, you can stop looking for an efficient solution
    - Or figure out efficient solution for ALL NP-complete problems

Polynomial-Time Reduction

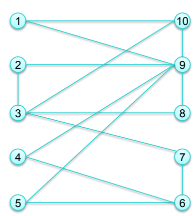
- Purpose. Classify problems according to *relative difficulty*.
- Design algorithms. If X $\le_P$ Y and Y can be solved in polynomial-time, then X can also be solved in polynomial time.
- Establish intractability. If X $\le_P$ Y and X cannot be solved in polynomial-time, then Y cannot be solved in polynomial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

Basic Reduction Strategies

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

Independent Set

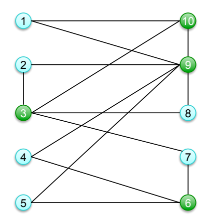
- Given a graph G = (V, E) and an integer *k*, is there a subset of vertices S ⊆ V such that |S| ≥ *k* and for each edge at most one of its endpoints is in *S*?



Ex. Is there an independent set of size ≥ 6?
Ex. Is there an independent set of size ≥ 7?

Independent Set

- Given a graph G = (V, E) and an integer *k*, is there a subset of vertices S ⊆ V such that |S| ≥ *k* and for each edge at most one of its endpoints is in *S*?

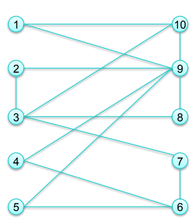


Ex. Is there an independent set of size ≥ 6? Yes
Ex. Is there an independent set of size ≥ 7? No

independent set

Vertex Cover

- Given a graph G = (V, E) and an integer *k*, is there a subset of vertices S ⊆ V such that |S| ≤ *k* and for each edge, at least one of its endpoints is in S?



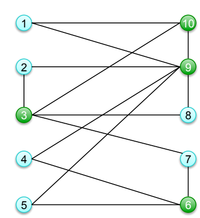
A vertex ***covers*** an edge.

Application: place guards within an art gallery so that all corridors are visible at any time

Ex. Is there a vertex cover of size ≤ 4?
Ex. Is there a vertex cover of size ≤ 3?

Vertex Cover

- Given a graph G = (V, E) and an integer *k*, is there a subset of vertices S ⊆ V such that |S| ≤ *k* and for each edge, at least one of its endpoints is in S?



Ex. Is there a vertex cover of size ≤ 4? Yes
Ex. Is there a vertex cover of size ≤ 3? No

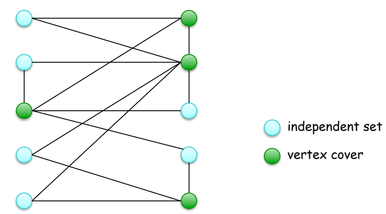
vertex cover

Problem

- Not known if either Independent Set or Vertex Cover can be solved in polynomial time
- BUT, what can we say about their relative difficulty?

Vertex Cover and Independent Set

- Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET
- Pf. We show *S* is an independent set iff V - S is a vertex cover



Vertex Cover and Independent Set

- Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET
- Pf. We show S is an independent set iff V - S is a vertex cover
- $\Rightarrow$
  - Let S be any independent set
  - Consider an arbitrary edge (u, v)
  - Since S is an independent set $\Rightarrow$ $u \notin S$ or $v \notin S$ $\Rightarrow$ $u \in V - S$ or $v \in V - S$
  - Thus, V - S covers (u, v)
    - Every edge has one end in V-S
  - V-S is a vertex Cover

Vertex Cover and Independent Set

- Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET
- Pf. We show S is an independent set iff V - S is a vertex cover
- $\Leftarrow$
  - Let V - S be any vertex cover
  - Consider two nodes $u \in S$ and $v \in S$
  - Observe that $(u, v) \notin E$ since V - S is a vertex cover
  - Thus, no two nodes in S are joined by an edge $\Rightarrow$ S independent set

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, . . . , S_m of subsets of U, and an integer k, does there exist a collection of $\leq$ 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:

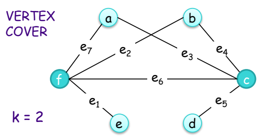
U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2

S_1 = {3, 7}	S_4 = {2, 4}
S_2 = {3, 4, 5, 6}	S_5 = {5}
S_3 = {1}	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.
  - …

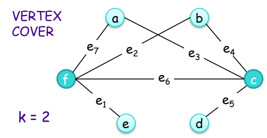


SET COVER

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, $S_v = \{e \in E : e \text{ incident to } v\}$
  - Set-cover of size $\leq$ k iff vertex cover of size $\leq$ 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}

For Friday

- Problem set
- Post on Sakai:
  - Brief overview statement (what is the article about)
  - 3 most important points
  - Questions: either for discussion or for understanding