# Objectives

- Computability
- Reducibility
- Conclusions



# 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

## What Were Our Goals In Finding a Solution?

- **Correctness**
- **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 According to Computational Requirements

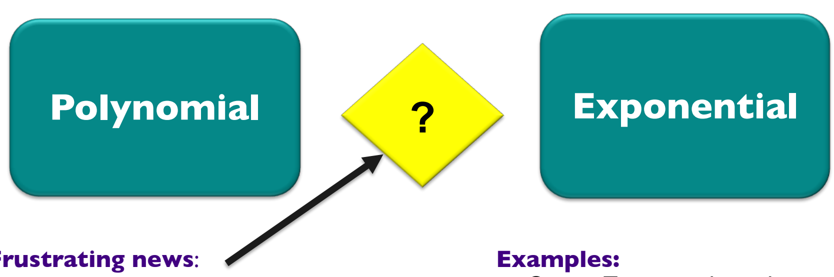
**Fundamental Question:**
Which problems will we be able to solve in practice?

**Working Answer:**
Those with polynomial runtimes.



## Classify Problems

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



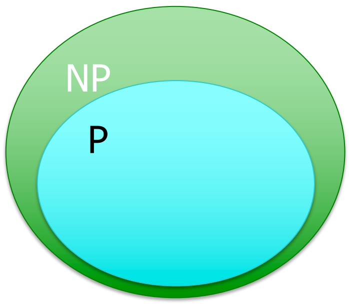

**Frustrating news**:
*Many* problems have defied classification.

**Chapter 8**. Show that problems are "computationally equivalent" and appear to be manifestations of one *really hard* problem.

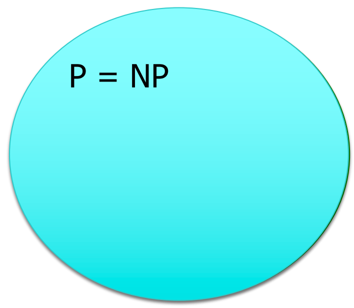
**Examples:**
- Given a Turing machine, does it halt in at most *k* steps?
- Given a board position in an n-by-n generalization of chess, can black guarantee a win?

## The Big Question

NP: "**n**ondeterministic **p**olynomial time"
We can verify that a solution solves the problem in polynomial time




P ⊆ NP



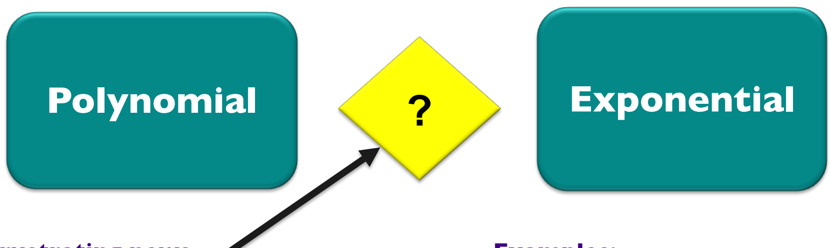

P = NP

Are there polynomial-time solutions to NP problems?




# In the mean time…

***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?



# 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

## Fun Fact: Connecting Chapters 7 and 8

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



## 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

**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.



## 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 $X \equiv_P Y$.

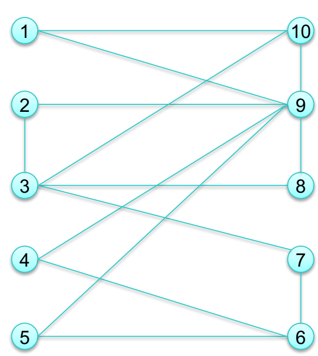
## Basic Reduction Strategies

- *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 $S \subseteq V$ such that $|S| \geq k$ and for each edge **at most one** of its endpoints is in $S$?



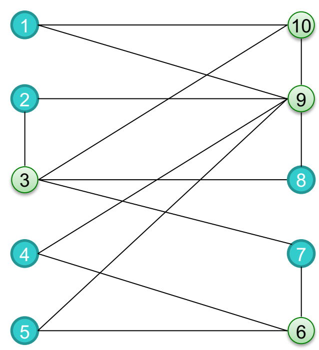

How is this different from the network flow problem?

Ex. Is there an independent set of size $\geq 6$?

Ex. Is there an independent set of size $\geq 7$?

## Independent Set

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



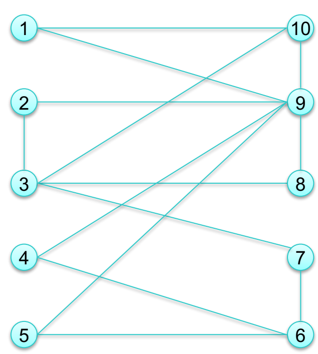

Ex. Is there an independent set of size $\geq$ 6? Yes
Ex. Is there an independent set of size $\geq$ 7? No

independent set



## Vertex Cover

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



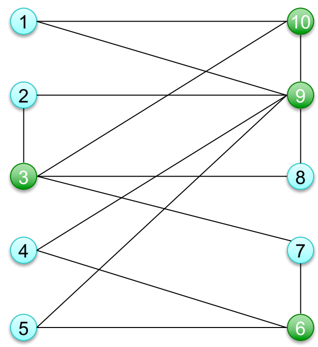

A vertex ***covers*** an edge.

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

Ex. Is there a vertex cover of size $\leq$ 4?
Ex. Is there a vertex cover of size $\leq$ 3?

## Vertex Cover

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




Ex. Is there a vertex cover of size $\leq$ 4? Yes

Ex. Is there a vertex cover of size $\leq$ 3? No

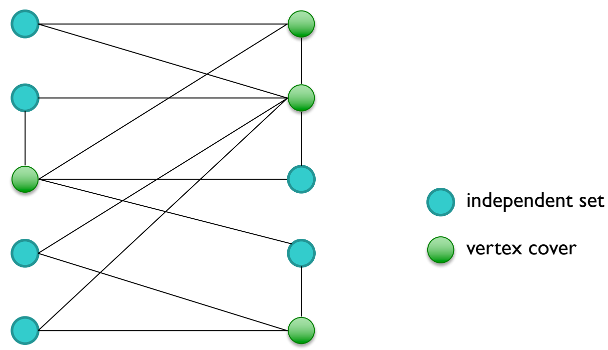


## Problem

- Not known if finding 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. S is an independent set iff V – S is a vertex cover
- $\Rightarrow$

## 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 an independent set
  - Consider an arbitrary edge (u, v)
  - Since S is an independent set $\Rightarrow$ u $\notin$ S or v $\notin$ S or both $\notin$ S $\Rightarrow$ u $\in$ V – S or v $\in$ V – S or both $\in$ V – S
  - Thus, V – S covers (u, v)
    - Every edge has at least 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

## Using the Previous Result

- 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

How do we show polynomial reduction for the independent set and vertex cover?



## Summary

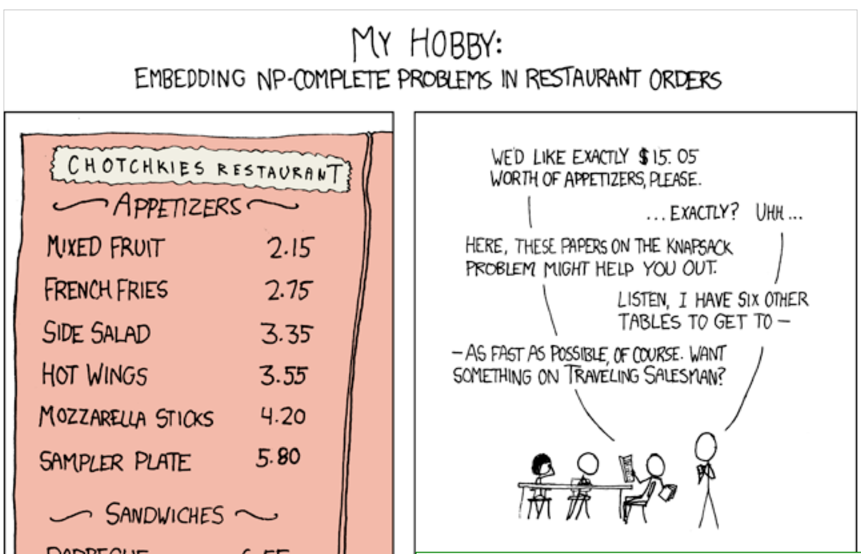
- If we have a black box to solve Vertex Cover, can decide whether G has an independent set of size at least *k* by asking the black box whether G has a vertex cover of size at most *n* – *k*
- If we have a black box to solve Independent Set, can decide whether G has a vertex cover of size at most *k* by asking the block box whether G has an independent set of size at least *n* - *k*

# Basic Reduction Strategies

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



# Now you "get" this xkcd comic




How is this a knapsack problem?

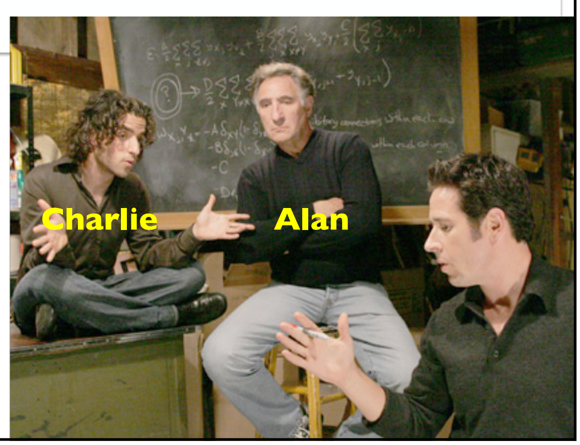
## From Patrick: Oracle of Bacon

- "Say you wanted to have each of the 1000 best centers of the Hollywood universe in your movie collection at least 20 times. What is the least number of movies you would have to own?"
- Are you trolling me, random Oracle of Bacon user? I'm pretty sure that's an example of the set-cover problem, with n equal to the ~100,000 movies the top 1000 best centers have appeared in. 2^10^6 is really big.
  - NP-Complete trolls are the trolliest trolls. At least the problem is decideable.



## P vs NP in *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.
…



## My Algorithms Approach

- Why problems?
  - Why no implementation until the end?
- Why wiki?
- Research to support decisions



## Final

- Due next Friday, noon (end of exams)
- Can use book, notes, handouts, my lecture notes, Sakai solutions, me (limited)
  - "The status of the P versus NP problem", article
  - No other outside resources
- Office hours: see BoxNote
  - Starting this afternoon
- Evaluations due Sunday at midnight on Sakai