

COMPSCI 311: Introduction to Algorithms  
Lecture 21: Intractability: Intro and Polynomial-Time Reductions

Dan Sheldon  
University of Massachusetts Amherst

## Algorithm Design

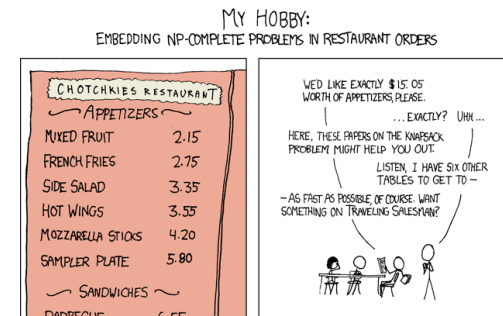
- ▶ Formulate the problem precisely
- ▶ Design an algorithm
- ▶ Prove correctness
- ▶ Analyze running time

Sometimes you can't find an efficient algorithm.

## Example: Network Design

- ▶ **Input:** undirected graph  $G = (V, E)$  with edge costs
- ▶ **Minimum spanning tree problem:** find min-cost subset of edges so there is a path between any  $u, v \in V$ .
  - ▶  $O(m \log n)$  greedy algorithm
- ▶ **Minimum Steiner tree problem:** find min-cost subset of edges so there is a path between any  $u, v \in W$  for specified terminal set  $W$ .
  - ▶ No polynomial-time algorithm is known.

## Example: Subset Sum / Knapsack

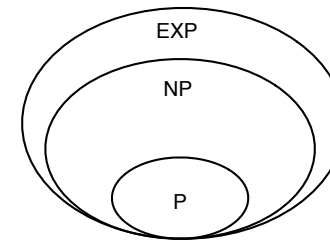


- ▶ **Input:**  $n$  items with weights, capacity  $W$
- ▶ **Goal:** maximize total weight without exceeding  $W$ 
  - ▶  $O(nW)$  pseudo-polynomial time algorithm (DP)
  - ▶ No polynomial time algorithm known!

## Tractability

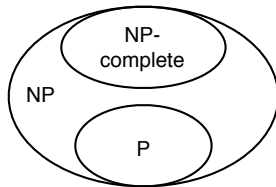
- ▶ Working definition of efficient: polynomial time
  - ▶  $O(n^d)$  for some  $d$ .
- ▶ Huge class of **natural and interesting** problems for which
  - ▶ We don't know any polynomial time algorithm
  - ▶ We can't prove that none exists
- ▶ **Goal**: develop mathematical tools to say when a problem is hard or "intractable"

## Preview of Landscape: Classes of Problems



- ▶ **P**: solvable in polynomial time
- ▶ **NP**: includes most problems we don't know about
- ▶ **EXP**: solvable in exponential time

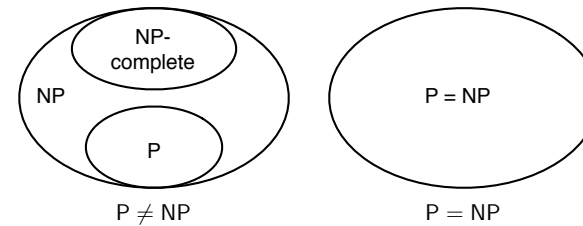
## NP-Completeness



- ▶ **NP-complete**: problems that are "as hard as" every other problem in NP.
- ▶ A polynomial time algorithm for any NP-complete problem implies one for *every problem in NP*

## $P \neq NP$ ?

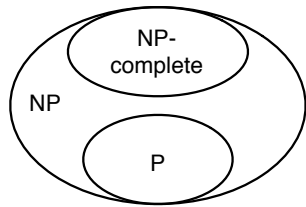
Two possibilities:



- ▶ We don't know which is true, but think  $P \neq NP$
- ▶ \$1M prize if you can find out (Clay Institute Millenium Problems)

## Outline

**Goal:** develop technical tools to make this precise



- ▶ **Polynomial-time reductions:** what it means for one problem to be “as hard as” another
- ▶ **Define NP:** characterize mystery problems
- ▶ **NP-completeness:** some problems in NP are “as hard as” all others

## Polynomial-Time Reduction

- ▶ Problem  $Y$  is **polynomial-time reducible** to Problem  $X$

```
solveY(yInput)
  Construct xInput          // poly-time
  foo = solveX(xInput)     // poly # of calls
  return yes/no based on foo // poly-time
```

- ▶ ... if any instance of Problem  $Y$  can be solved using
  1. A polynomial number of standard computational steps
  2. A polynomial number of calls to a black box that solves problem  $X$

- ▶ **Notation**  $Y \leq_P X$

## Clicker

Suppose that  $Y \leq_P X$ . Which of the following can we infer?

- A. If  $X$  can be solved in polynomial time, then so can  $Y$ .
- B. If  $Y$  cannot be solved in polynomial time, then neither can  $X$ .
- C. Both A and B.
- D. Neither A nor B.

## Polynomial-Time Reduction

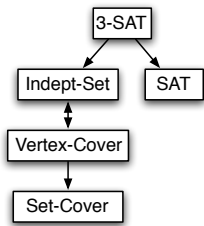
- ▶  $Y \leq_P X$

```
solveY(yInput)
  Construct xInput          // poly-time
  foo = solveX(xInput)     // poly # of calls
  return yes/no based on foo // poly-time
```

- ▶ Statement about **relative hardness**. Suppose  $Y \leq_P X$ , then:
  1. If  $X$  is solvable in poly-time, so is  $Y$
  2. If  $Y$  is *not* solvable in poly-time, neither is  $X$
- ▶ 1: design algorithms, 2: prove hardness

## Preview

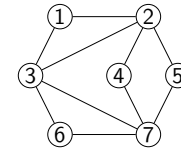
Partial map of problems we can use to solve others in polynomial time, through **transitivity** of reductions:



▶  $Y \rightarrow X$   
means  $Y \leq_P X$ .

## First Reduction: Independent Set and Vertex Cover

Given a graph  $G = (V, E)$ ,



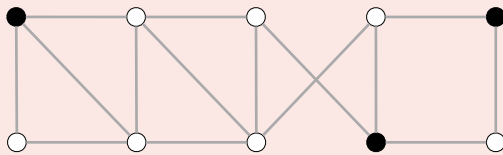
- ▶  $S \subset V$  is an **independent set** if no nodes in  $S$  share an edge. **Examples:**  $\{3, 4, 5\}$ ,  $\{1, 4, 5, 6\}$ .
- ▶  $S \subset V$  is a **vertex cover** if every edge has at least one endpoint in  $S$ . **Examples:**  $\{1, 2, 6, 7\}$ ,  $\{2, 3, 7\}$

INDEPT-SET Does  $G$  have independent set of size **at least**  $k$ ? VERTEX-COVER Does  $G$  have a vertex cover of size **at most**  $k$ ?

### Intractability: quiz 3

Consider the following graph  $G$ . Which are true?

- A. The white vertices are a vertex cover of size 7.
- B. The black vertices are an independent set of size 3.
- C. Both A and B.
- D. Neither A nor B.



slide credit: Kevin Wayne / Pearson

15

## Independent Set and Vertex Cover

- ▶ **Claim:**  $S$  is independent set if and only if  $V - S$  is a vertex cover.
1.  $S$  independent set  $\Rightarrow V - S$  vertex cover
    - ▶ Consider any edge  $(u, v)$
    - ▶  $S$  independent  $\Rightarrow$  either  $u \notin S$  or  $v \notin S$
    - ▶ i.e., either  $u \in V - S$  or  $v \in V - S$
    - ▶  $\Rightarrow V - S$  is a vertex cover
  2.  $V - S$  vertex cover  $\Rightarrow S$  independent set
    - ▶ Similar.

## Independent Set $\leq_P$ Vertex Cover

**Claim:** INDEPENDENT SET  $\leq_P$  VERTEX COVER. **Reduction:**

- ▶ On INDEPENDENT SET instance  $\langle G, k \rangle$
- ▶ Construct VERTEX COVER instance  $\langle G, n - k \rangle$
- ▶ Return YES iff  $\text{solveVC}(\langle G, n - k \rangle) = \text{YES}$

**Correctness** for YES output:

- ▶ Suppose  $G$  has independent set  $S$  with  $\geq k$  nodes
- ▶ Then  $T = V - S$  is a vertex cover with  $\leq n - k$  nodes
- ▶ The algorithm correctly outputs YES

**Correctness** for NO output:

- ▶ Suppose  $G$  has no independent set  $S$  with  $\geq k$  nodes
- ▶ Then there is no vertex cover with  $T$  with  $\leq n - k$  nodes, otherwise  $S = V - T$  is an independent set with  $\geq k$  nodes.
- ▶ The algorithm correctly outputs NO

## Vertex Cover $\leq_P$ Independent Set

▶ **Claim:** VERTEX COVER  $\leq_P$  INDEPENDENT SET

▶ **Reduction:**

- ▶ On VERTEX COVER input  $\langle G, k \rangle$
- ▶ Construct INDEPENDENT SET input  $\langle G, n - k \rangle$
- ▶ Return YES if  $\text{solveIS}(\langle G, n - k \rangle) = \text{YES}$

▶ **Proof:** similar

## Aside: Decision versus Optimization

- ▶ For intractability and reductions we will focus on decision problems (YES/NO answers)
- ▶ Algorithms have typically been for optimization (find biggest/smallest)
- ▶ Can reduce optimization to decision and vice versa. [Discuss](#).

## Reduction Strategies

- ▶ Reduction by equivalence
- ▶ Reduction to a more general case
- ▶ Reduction by "gadgets"

## Reduction to General Case: Set Cover

**Problem.** Given a set  $U$  of  $n$  elements, subsets  $S_1, \dots, S_m \subset U$ , and a number  $k$ , does there exist a collection of at most  $k$  subsets  $S_i$  whose union is  $U$ ?

- ▶ Example:  $U = \{A, B, C, D, E\}$  is the set of all skills, there are five people with skill sets:

$$S_1 = \{A, C\}, \quad S_2 = \{B, E\}, \quad S_3 = \{A, C, E\}$$

$$S_4 = \{D\}, \quad S_5 = \{B, C, E\}$$

- ▶ Find a small team that has all skills.  $S_1, S_4, S_5$

**Theorem.**  $\text{VERTEXCOVER} \leq_P \text{SETCOVER}$

## Intractability: quiz 4



Given the universe  $U = \{1, 2, 3, 4, 5, 6, 7\}$  and the following sets, which is the minimum size of a set cover?

- A. 1
- B. 2
- C. 3
- D. None of the above.

$$\begin{aligned} U &= \{1, 2, 3, 4, 5, 6, 7\} \\ S_a &= \{1, 4, 6\} & S_b &= \{1, 6, 7\} \\ S_c &= \{1, 2, 3, 6\} & S_d &= \{1, 3, 5, 7\} \\ S_e &= \{2, 6, 7\} & S_f &= \{3, 4, 5\} \end{aligned}$$

slide credit: Kevin Wayne / Pearson

20

## Clicker

Vertex Cover is a special case of Set Cover with:

- A.  $U = V$  and  $S_e =$  the two endpoints of  $e$  for each  $e \in E$ .
- B.  $U = E$  and  $S_v =$  the set of edges incident to  $v$  for each  $v \in V$ .
- C.  $U = V \cup E$  and  $S_v =$  the set of neighbors of  $v$  together with edges incident to  $v$  for each  $v \in V$ .

## Reduction of Vertex Cover to Set Cover

**Theorem.**  $\text{VERTEXCOVER} \leq_P \text{SETCOVER}$

### Reduction.

- ▶ Given VERTEX COVER instance  $\langle G, k \rangle$
- ▶ Construct SET COVER instance  $\langle U, S_1, \dots, S_m, k \rangle$  with  $U = E$ , and  $S_v =$  the set of edges incident to  $v$
- ▶ Return YES iff  $\text{solveSC}(\langle U, S_1, \dots, S_m, k \rangle) = \text{YES}$

### Proof

- ▶ Straightforward to see that  $S_{v_1}, \dots, S_{v_\ell}$  is a set cover of size  $\ell$  if and only if  $v_1, \dots, v_\ell$  is a vertex cover of size  $\ell$
- ▶ This implies the algorithm correctly outputs YES if  $G$  has a vertex cover of size  $\leq k$  and NO otherwise
- ▶ Polynomial # of steps outside of solveSC
- ▶ Only one call to solveSC