Week 12.3, Friday, Nov 8

Homework 6 Due: November 14\textsuperscript{th} at 11:59PM (Gradescope)
7.11 Project Selection
Projects with prerequisites.
- Set $P$ of possible projects. Project $v$ has associated revenue $p_v$.
  - some projects generate money: create interactive e-commerce interface, redesign web page
  - others cost money: upgrade computers, get site license
- Set of prerequisites $E$. If $(v, w) \in E$, can't do project $v$ and unless also do project $w$.
- A subset of projects $A \subseteq P$ is feasible if the prerequisite of every project in $A$ also belongs to $A$.

Project selection. Choose a feasible subset of projects to maximize revenue.
Prerequisite graph.
- Include an edge from $v$ to $w$ if can't do $v$ without also doing $w$.
- $\{v, w, x\}$ is feasible subset of projects.
- $\{v, x\}$ is infeasible subset of projects.
Min cut formulation.

- Assign capacity $\infty$ to all prerequisite edge.
- Add edge $(s, v)$ with capacity $p_v$ if $p_v > 0$.
- Add edge $(v, t)$ with capacity $-p_v$ if $p_v < 0$.
- For notational convenience, define $p_s = p_t = 0$. 

![Graph diagram]
Claim. $(A, B)$ is min cut iff $A - \{s\}$ is optimal set of projects.

- Infinite capacity edges ensure $A - \{s\}$ is feasible.
- Max revenue because:

$$cap(A, B) = \sum_{v \in B: p_v > 0} p_v + \sum_{v \in A: p_v < 0} (-p_v)$$

$$= \sum_{v: p_v > 0} p_v - \sum_{v \in A} p_v$$

constant
Algorithm Design Patterns and Anti-Patterns

Algorithm design patterns.

- **Greedy.**
  - $O(n \log n)$ interval scheduling.

- **Divide-and-conquer.**
  - $O(n \log n)$ Closest Pair of Points.

- **Dynamic programming.**
  - $O(n^2)$ edit distance.

- **Duality.**
  - $O(n^3)$ bipartite matching.

- **Reductions.**
  - Circulation via Network Flow
  - Bipartite Matching via Network Flow
  - Baseball elimination
  - Project Selection

- **Local search.**
- **Randomization.**

Algorithm design anti-patterns.

- **NP-completeness.**
  - $O(n^k)$ algorithm unlikely.

- **PSPACE-completeness.**
  - $O(n^k)$ certification algorithm unlikely.

- **Undecidability.**
  - No algorithm possible.
8.1 Polynomial-Time Reductions
**Q.** Which problems will we be able to solve in practice?


Those with polynomial-time algorithms.

<table>
<thead>
<tr>
<th>Yes</th>
<th>Probably no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest path</td>
<td>Longest path</td>
</tr>
<tr>
<td>Matching</td>
<td>3D-matching</td>
</tr>
<tr>
<td>Min cut</td>
<td>Max cut</td>
</tr>
<tr>
<td>2-SAT</td>
<td>3-SAT</td>
</tr>
<tr>
<td>Planar 4-color</td>
<td>Planar 3-color</td>
</tr>
<tr>
<td>Bipartite vertex cover</td>
<td>Vertex cover</td>
</tr>
<tr>
<td>Primality testing</td>
<td>Factoring</td>
</tr>
</tbody>
</table>
Classify Problems

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

Provably requires exponential-time.
- 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?

Frustrating news. Huge number of fundamental problems have defied classification for decades.

This chapter. Show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one really hard problem.
Polynomial-Time Reduction

Desiderata'. Suppose we could solve X 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.

Notation. $X \leq_P Y$.

Example. Network Flow reduces to Linear Programming

Remarks.

- We pay for time to write down instances sent to black box $\Rightarrow$ instances of Y must be of polynomial size.
- Note: Cook reducibility.

in contrast to Karp reductions
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 $X \equiv_p Y$.

\[ \text{up to cost of reduction} \]
Reduction By Simple Equivalence

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 $S$?

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

**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.
Claim. \( \text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}. \)

Pf. We show \( S \) is an independent set iff \( V - S \) is a vertex cover.
Claim. \( \text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}. \)

Pf. We show \( S \) is an independent set iff \( V - S \) is a vertex cover.
\((G \text{ has VC of size } k \text{ iff } G \text{ has independent set of size } v-k)\)

\( \Rightarrow \)
- Let \( S \) be any independent set.
  - Consider an arbitrary edge \((u, v)\).
  - \( S \) independent \( \Rightarrow u \notin S \) or \( v \notin S \) \( \Rightarrow u \in V - S \) or \( v \in V - S \).
  - Thus, \( V - S \) covers \((u, v)\).

\( \Leftarrow \)
- 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 \( \Rightarrow S \) is an independent set. \( \Box \)
Reduction from Special Case to General Case

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 $\leq k$ 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:**

<table>
<thead>
<tr>
<th>$U$ = {1, 2, 3, 4, 5, 6, 7}</th>
<th>$k$ = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1 = {3, 7}$</td>
<td>$S_4 = {2, 4}$</td>
</tr>
<tr>
<td>$S_2 = {3, 4, 5, 6}$</td>
<td>$S_5 = {5}$</td>
</tr>
<tr>
<td>$S_3 = {1}$</td>
<td>$S_6 = {1, 2, 6, 7}$</td>
</tr>
</tbody>
</table>
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 \( \leq k \). •

VERTEX COVER

- \( k = 2 \)
  - \( f \)
  - \( e \)
  - \( d \)
  - \( c \)
  - \( e_7 \)
  - \( e_2 \)
  - \( e_3 \)
  - \( e_4 \)
  - \( e_6 \)
  - \( e_5 \)

SET COVER

- \( U = \{ 1, 2, 3, 4, 5, 6, 7 \} \)
- \( k = 2 \)
- \( S_a = \{ 3, 7 \} \)
- \( S_b = \{ 2, 4 \} \)
- \( S_c = \{ 3, 4, 5, 6 \} \)
- \( S_d = \{ 5 \} \)
- \( S_e = \{ 1 \} \)
- \( S_f = \{ 1, 2, 6, 7 \} \)
Polynomial-Time Reduction

**Basic 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 via "gadgets."
Satisfiability

**Literal:** A Boolean variable or its negation.  \( x_i \) or \( \overline{x_i} \)

**Clause:** A disjunction of literals.  \( C_j = x_1 \lor \overline{x_2} \lor x_3 \)

**Conjunctive normal form:** A propositional formula \( \Phi \) that is the conjunction of clauses.  \( \Phi = C_1 \land C_2 \land C_3 \land C_4 \)

**SAT:** Given CNF formula \( \Phi \), does it have a satisfying truth assignment?

**3-SAT:** SAT where each clause contains (at most) 3 literals.

Each corresponds to a different variable

**Ex:**  \((\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3})\)

**Yes:**  \( x_1 = \text{true}, x_2 = \text{true} \), \( x_3 = \text{false} \).
Claim. $\text{3-SAT} \leq_P \text{INDEPENDENT-SET}.$

Pf. Given an instance $\Phi$ of 3-SAT, we construct an instance $(G, k)$ of INDEPENDENT-SET that has an independent set of size $k$ iff $\Phi$ is satisfiable.

Construction.
- $G$ contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

$$\begin{align*}
\Phi &= \left( \overline{x_1} \lor x_2 \lor x_3 \right) \land \left( x_1 \lor \overline{x_2} \lor x_3 \right) \land \left( \overline{x_1} \lor x_2 \lor x_4 \right) \\
G &= \text{Graph representation of } \Phi
\end{align*}$$
Claim. $G$ contains independent set of size $k = |\Phi|$ iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Let $S$ be independent set of size $k$.
- $S$ must contain exactly one vertex in each triangle.
- Set these literals to true. and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.

$$
\Phi = (\overline{x_1} \vee x_2 \vee x_3) \land (x_1 \vee \overline{x_2} \vee x_3) \land (\overline{x_1} \vee x_2 \vee x_4)
$$
Claim. $G$ contains independent set of size $k = |\Phi|$ iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Let $S$ be independent set of size $k$.
- $S$ must contain exactly one vertex in each triangle.
- Set these literals to true. and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.

Independent Set $S$ has size $k=3$
$\rightarrow$ Set $x_3$=true, $x_1$=false and $x_2$ = (arbitrary)

$$\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)$$
Claim. \( G \) contains independent set of size \( k = |\Phi| \) iff \( \Phi \) is satisfiable.

Pf. \( \Rightarrow \) Let \( S \) be independent set of size \( k \).
- \( S \) must contain exactly one vertex in each triangle.
- Set these literals to true. \( \iff \) and any other variables in a consistent way.
- Truth assignment is consistent and all clauses are satisfied.

Pf. \( \Leftarrow \) Given satisfying assignment, select one true literal from each triangle. This is an independent set of size \( k \). \( \blacksquare \)
Review

Basic reduction strategies.

- Simple equivalence: INDEPENDENT-SET $\equiv_p$ VERTEX-COVER.
- Special case to general case: VERTEX-COVER $\leq_p$ SET-COVER.
- Encoding with gadgets: 3-SAT $\leq_p$ INDEPENDENT-SET.

Transitivity. If $X \leq_p Y$ and $Y \leq_p Z$, then $X \leq_p Z$.

Pf idea. Compose the two algorithms.

Ex: 3-SAT $\leq_p$ INDEPENDENT-SET $\leq_p$ VERTEX-COVER $\leq_p$ SET-COVER.
Self-Reducibility

**Decision problem.** Does there exist a vertex cover of size \( \leq k \)?

**Search problem.** Find vertex cover of minimum cardinality.

**Self-reducibility.** Search problem \( \leq_p \) decision version.
- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

**Ex:** to find min cardinality vertex cover.
- (Binary) search for cardinality \( k^* \) of min vertex cover.
- Find a vertex \( v \) such that \( G - \{ v \} \) has a vertex cover of size \( \leq k^* - 1 \).
  - any vertex in any min vertex cover will have this property
- Include \( v \) in the vertex cover.
- Recursively find a min vertex cover in \( G - \{ v \} \).