Reductions

PLEASE KEEP THE DOOR CLOSED!!!
THANK YOU!!!

Please don’t use Comic Sans—we are a Fortune 500 Company, not a Lemonade Stand.
Polynomial-Time Reductions
Classify Problems According to Computational Requirements

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>
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.

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

Reduction. Problem $X$ polynomial-time 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. $X \leq_p Y$.

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 (vs. Karp reducibility)

Means we can solve $X$ in polynomial time IF we can solve $Y$ 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 $X \equiv_p Y$. 
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.
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:**

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

$k = 2$

$S_1 = \{3, 7\}$

$S_2 = \{3, 4, 5, 6\}$

$S_3 = \{1\}$

$S_4 = \{2, 4\}$

$S_5 = \{5\}$

$S_6 = \{1, 2, 6, 7\}$
**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 \( \leq k \). □
Polynomial-Time Reduction

Basic strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.
8.2 Reductions via "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 exactly 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. \(3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \).

Pf. Given an instance \(\Phi\) of \(3\text{-SAT}\), we construct an instance \((G, k)\) of \(\text{INDEPENDENT-SET}\) 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.

\[
\Phi = \overline{x_1} \lor x_2 \lor x_3 \lor (x_1 \lor \overline{x_2} \lor x_3) \lor (\overline{x_1} \lor x_2 \lor x_4)
\]
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. \(\rightarrow\) 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 $k$. $\blacksquare$

\[\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)\]
Review

Basic reduction strategies.
- Simple equivalence: $\text{INDEPENDENT-SET} \equiv_p \text{VERTEX-COVER}$.
- Special case to general case: $\text{VERTEX-COVER} \leq_p \text{SET-COVER}$.
- Encoding with gadgets: $3\text{-SAT} \leq_p \text{INDEPENDENT-SET}$.

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

Pf idea. Compose the two algorithms.

Ex: $3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \leq_p \text{SET-COVER}$. 
Hamiltonian Cycle

**HAM-CYCLE:** given an undirected graph $G = (V, E)$, does there exist a simple cycle $\Gamma$ that contains every node in $V$.

NO: bipartite graph with odd number of nodes.
**Directed Hamiltonian Cycle**

**DIR-HAM-CYCLE:** given a digraph $G = (V, E)$, does there exists a simple directed cycle $\Gamma$ that contains every node in $V$?

**Claim.** $\text{DIR-HAM-CYCLE} \leq_p \text{HAM-CYCLE}$.  

**Pf.** Given a directed graph $G = (V, E)$, construct an undirected graph $G'$ with $3n$ nodes.
Claim. $G$ has a Hamiltonian cycle iff $G'$ does.

Pf. $\Rightarrow$

$\quad$ Suppose $G$ has a directed Hamiltonian cycle $\Gamma$.
$\quad$ Then $G'$ has an undirected Hamiltonian cycle (same order).

Pf. $\Leftarrow$

$\quad$ Suppose $G'$ has an undirected Hamiltonian cycle $\Gamma'$.
$\quad$ $\Gamma'$ must visit nodes in $G'$ using one of following two orders:

$\quad$ ..., B, G, R, B, G, R, B, G, R, B, ...
$\quad$ ..., B, R, G, B, R, G, B, R, G, B, ...

$\quad$ Blue nodes in $\Gamma'$ make up directed Hamiltonian cycle $\Gamma$ in $G$, or reverse of one. $
\Box$
Claim. 3-SAT $\leq_P$ DIR-HAM-CYCLE.

Pf. Given an instance $\Phi$ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamiltonian cycle iff $\Phi$ is satisfiable.

Construction. First, create graph that has $2^n$ Hamiltonian cycles which correspond in a natural way to $2^n$ possible truth assignments.
Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses. 
- Construct $G$ to have $2^n$ Hamiltonian cycles.
- Intuition: traverse path $i$ from left to right $\iff$ set variable $x_i = 1$. 

![Diagram of a graph showing Hamiltonian cycles](image-url)
Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.

- For each clause: add a node and 6 edges.

$C_1 = x_1 \lor \overline{x_2} \lor x_3$

clause node

$C_2 = \overline{x_1} \lor x_2 \lor \overline{x_3}$

clause
3-SAT Reduces to Directed Hamiltonian Cycle

Claim. \( \Phi \) is satisfiable iff \( G \) has a Hamiltonian cycle.

Pf. \( \Rightarrow \)

- Suppose 3-SAT instance has satisfying assignment \( x^* \).
- Then, define Hamiltonian cycle in \( G \) as follows:
  - if \( x^*_i = 1 \), traverse row \( i \) from left to right
  - if \( x^*_i = 0 \), traverse row \( i \) from right to left
  - for each clause \( C_j \), there will be at least one row \( i \) in which we are going in "correct" direction to splice node \( C_j \) into tour
Claim. \( \Phi \) is satisfiable iff \( G \) has a Hamiltonian cycle.

Pf. \( \Leftarrow \)

- Suppose \( G \) has a Hamiltonian cycle \( \Gamma \).
- If \( \Gamma \) enters clause node \( C_j \), it must depart on mate edge.
  - thus, nodes immediately before and after \( C_j \) are connected by an edge \( e \) in \( G \)
  - removing \( C_j \) from cycle, and replacing it with edge \( e \) yields Hamiltonian cycle on \( G - \{ C_j \} \)
- Continuing in this way, we are left with Hamiltonian cycle \( \Gamma' \) in \( G - \{ C_1, C_2, \ldots, C_k \} \).
- Set \( x^*_i = 1 \) iff \( \Gamma' \) traverses row \( i \) left to right.
- Since \( \Gamma \) visits each clause node \( C_j \), at least one of the paths is traversed in "correct" direction, and each clause is satisfied. \( \blacksquare \)
Longest Path

SHORTEST-PATH. Given a digraph $G = (V, E)$, does there exists a simple path of length at most $k$ edges?

LONGEST-PATH. Given a digraph $G = (V, E)$, does there exists a simple path of length at least $k$ edges?

Claim. $3$-SAT $\leq_p$ LONGEST-PATH.

Pf 1. Redo proof for DIR-HAM-CYCLE, ignoring back-edge from $t$ to $s$.
Pf 2. Show HAM-CYCLE $\leq_p$ LONGEST-PATH.
Traveling Salesperson Problem

TSP. Given a set of n cities and a pairwise distance function \( d(u, v) \), is there a tour of length \( \leq D \)?

All 13,509 cities in US with a population of at least 500
Reference: http://www.tsp.gatech.edu
**Traveling Salesperson Problem**

**TSP.** Given a set of n cities and a pairwise distance function \( d(u, v) \), is there a tour of length \( \leq D \)?

Optimal TSP tour
Reference: [http://www.tsp.gatech.edu](http://www.tsp.gatech.edu)
Traveling Salesperson Problem

**TSP.** Given a set of \( n \) cities and a pairwise distance function \( d(u, v) \), is there a tour of length \( \leq D \)?

11,849 holes to drill in a programmed logic array
Reference: http://www.tsp.gatech.edu
Traveling Salesperson Problem

**TSP.** Given a set of $n$ cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?

Optimal TSP tour
Reference: http://www.tsp.gatech.edu
Traveling Salesperson Problem

**TSP.** Given a set of \( n \) cities and a pairwise distance function \( d(u, v) \), is there a tour of length \( \leq D \)?

**HAM-CYCLE:** given a graph \( G = (V, E) \), does there exists a simple cycle that contains every node in \( V \)?

**Claim.** \( HAM\text{-}CYCLE \leq_p TSP \).

**Pf.**
- Given instance \( G = (V, E) \) of \( HAM\text{-}CYCLE \), create \( n \) cities with distance function

\[
d(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 2 & \text{if } (u, v) \notin E \end{cases}
\]

- TSP instance has tour of length \( \leq n \) iff \( G \) is Hamiltonian.

**Remark.** TSP instance in reduction satisfies \( \Delta \)-inequality.