NP
CS 320, Fall 2017

Dr. Geri Georg, Instructor
georg@colostate.edu
NP-Complete

A class of problems where:

• No polynomial-time algorithm has been discovered

• No proof that one doesn’t exist
P & NP

Often a problem can have a Polynomial-time solution and a similar one may not:

P: Find a single-source shortest path
NP: Does a graph contain a simple path with at least $k$ edges?
Another example:

P: Find an Euler tour
• a cycle that traverses each edge once (can visit vertices more than once)

NP: Does a graph have a Hamiltonian cycle?
• a simple cycle containing every vertex
Examples of P & NP

Finally:

Is a 2-CNF satisfiable?
• \((x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3) \land (\neg x_2 \lor \neg x_3)\)

Is a 3-CNF satisfiable?
• \((x_1 \lor \neg x_1 \lor \neg x_2) \land (x_3 \lor x_2 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4)\)
Problem Classes

P – solvable in polynomial time

NP – verifiable in polynomial time

NPC (NP-complete) – in NP and as hard as any problem in NP

If a problem is NP-complete then there is good evidence that it is intractable.
Showing a problem is NPC

1. Convert an optimization problem into a related decision problem.

2. Reduce one problem to another to show that it is no harder or easier.
Diversion – Reduction

Where have we seen reduction before?

Cole gave an example the Friday before break – manipulating a new problem into a format where we can solve it with an algorithm we already know.
Reducibility

Uses a framework of formal language:
A language $L$ over an alphabet $\Sigma$ is any set of strings made up of symbols from $\Sigma$.
We can perform set operations (e.g. membership testing) on languages.

Notation: language $L'$ reduces to language $L$:

$L' \leq_p L$
Problem Language Example

PATH: decision problem – given a directed graph G, vertices u and v, and an integer k, does a path exist from u to v ≤ k?

Problem as a language over the alphabet \{0, 1\}:
PATH = \{⟨G, u, v, k⟩ where
  G = (V, E) is an undirected graph,
  u, v ∈ V,
  k ≥ 0 is an integer,
  decision is whether ∃ path from u to v in G consisting of at most k edges\}
Algorithms

An algorithm $A$ accepts the set of strings in $L$ for which it returns 1, and rejects the set of strings for which it returns 0.

Verification algorithm $A$:

2 arguments: input string and certificate

Input string: $x = \langle G, u, v, k \rangle$

Certificate: $y = \text{path } p \text{ from } u \text{ to } v$

$A (x, y) = 1$ if $y$ is in $G$ and is of length $\leq k$
Verification as a language

PATH verification algorithm verifies language
L = \{x \in \{0, 1\}* : A(x, y) = 1\}

NP:
A language L \in NP iff
L = \{x \in \{0, 1\}* : \exists \text{ a certificate } y
\quad \text{with } |y| = O(|x|^c) \text{ such that } A(x, y) = 1\}

Algorithm A verifies language L in polynomial time
Definition of NP-complete

A language $L \subseteq \{0, 1\}^*$ is NP-complete if
1. $L \in \text{NP}$
2. $L' \leq_{p} L$ for every $L' \in \text{NP}$

A language $L$ is NP-hard if #2 is satisfied, and #1 may not be satisfied.

All proofs have these 2 parts. Many start with #2, showing that a problem is NP-hard first, then going on to show #1.
Need an initial NP problem!

Circuit Satisfiability:

• Given a Boolean combinational circuit of AND, OR, and NOT gates, is it satisfiable?
• Circuit size is \# gates + \# wires. Encode as binary string of a mapping to a graph.
Circuit satisfiability as NP

Language CIRCUIT-SAT = \{ \langle C \rangle : C \text{ is a satisfiable Boolean combinational circuit} \}

#1: \( L \in NP \)

Construct polynomial time verification algorithm \( A(x, y) \)

where \( x = \text{encoded } C \),

\( y = \text{assignments of Boolean values to the wires in } C \)
Circuit satisfiability reducibility

Language CIRCUIT-SAT = \{\langle C \rangle : C \text{ is a satisfiable Boolean combinational circuit}\}

#2: L is NP-hard

Show that every language in NP is reducible in polynomial time to CIRCUIT-SAT
Circuit satisfiability reducibility

Every $L' \in \text{NP}$ has a verification algorithm $A$ that takes an input $x$ and certificate $y$ and outputs a decision.

$A$ is a sequence of program configurations that can all be computed by a single combinational circuit.

Creating the circuit can be done in polynomial time.
NPC

**Circuit satisfiability** is NP-complete

**Formula satisfiability (SAT):** is a given Boolean formula satisfiable?

#1: \( \text{SAT} \in \text{NP} \) – we can create a polynomial algorithm \( A \) to test a certificate

#2: \( \text{CIRCUIT-SAT} \leq_p \text{SAT} \)

**3-CNF satisfiability:** is a Boolean formula in 3-CNF satisfiable?

#1: \( \text{3-CNF-SAT} \in \text{NP} \) – we can create a polynomial algorithm \( A \) to test a certificate

#2: \( \text{SAT} \leq_p \text{3-CNF-SAT} \)
Finally to the examples in the book

**Clique:**

- Undirected graph $G$
- Subset of vertices where each pair is connected (complete subgraph of $G$)
- Social networks: subset of people who all know each other

**NP-complete:**

1. $\text{CLIQUE} \in \text{NP}$
2. $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
More book examples

**Vertex Cover:**
- Undirected graph $G$
- Subset of vertices that have all the edges in the graph as incident edges

**NP-complete:**
1. $\text{VERTEX-COVER} \in \text{NP}$
2. $\text{CLIQUE} \leq_{p} \text{VERTEX-COVER}$
More book examples

Hamiltonian Cycle:
- Undirected graph $G$
- Simple cycle containing every vertex

NP-complete:

1. $\text{HAM-CYCLE} \in \text{NP}$
2. $\text{VERTEX-COVER} \leq_p \text{HAM-CYCLE}$
More book examples

Traveling Salesman:

• Undirected graph G
• Simple cycle containing every vertex

NP-complete:

#1 TSP \in NP
#2: HAM-CYCLE
\leq_p TSP
And finally...

Subset-Sum:
Finite set $S$ of positive integers, integer target $t > 0$
$\exists S' \subset S$ where the elements sum to $t$?

NP-complete:

#1 $\text{SUBSET-SUM} \in \text{NP}$

#2: $3\text{-CNF-SAT} \leq_p \text{SUBSET-SUM}$
Reductions

- CIRCUIT-SAT \rightarrow SAT
- SAT \rightarrow 3-CNF-SAT
- 3-CNF-SAT \rightarrow CLIQUE
- 3-CNF-SAT \rightarrow SUBSET-SUM
- CLIQUE \rightarrow VERTEX-COVER
- VERTEX-COVER \rightarrow HAM-CYCLE
- HAM-CYCLE \rightarrow TSP
- TSP \rightarrow CIRCUIT-SAT
Image Credits

np.gif, cirsat.gif, clique.gif, vert-cover.gif:


penguins.png: http://www.cgl.uwaterloo.ca/csk/projects/tsp/

subsetsum.png, pulsars.gif: https://thquinn.github.io/projects/automaton.html