Recap

• Polynomial Time Reductions ($X \leq_P Y$)
  - Cook vs Karp Reductions
  - 3-SAT $\leq_P$ Independent Set (Gadgets)
• Decision Problems vs Search Problems
• Self-Reducibility

Complexity Classes

• Polynomial Time Certifier
• Definition of P, NP, EXP
• $P \subseteq NP \subseteq EXP$

8.4 NP-Completeness

**NP-Complete**

**NP-complete.** A problem $Y$ in NP with the property that for every problem $X$ in NP, $X \leq_P Y$.

**NP-hard.** A problem $Y$ (not necessarily in NP) with the property that for every problem $X$ in NP, $X \leq_P Y$

**Theorem.** Suppose $Y$ is an NP-complete problem. Then $Y$ is solvable in poly-time iff $P = NP$.

**Proof.**
- **$\Rightarrow$** If $P = NP$ then $Y$ can be solved in poly-time since $Y$ is in NP.
- **$\Leftarrow$** Suppose $Y$ can be solved in poly-time.
  - Let $X$ be any problem in NP. Since $X \leq_P Y$, we can solve $X$ in poly-time. This implies $NP \subseteq P$.
  - We already know $P \subseteq NP$. Thus $P = NP$. 

**Fundamental question.** Do there exist “natural” NP-complete problems?

**Polynomial Transformation**

**Def. Problem X polynomial reduces (Cook) 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.

**Def. Problem X polynomial transforms (Karp) to problem Y if given any input $x$ to X, we can construct an input $y$ such that $x$ is a yes instance of X iff $y$ is a yes instance of Y.**

**Note.** Polynomial transformation is polynomial reduction with just one call to oracle for Y, exactly at the end of the algorithm for X. Almost all previous reductions were of this form.

**Open question.** Are these two concepts the same with respect to NP? 

**NP-Complete Problems**

**Circuit Satisfiability**

**CIRCUIT-SAT.** Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?

**Q:** Why is CIRCUIT-SAT in NP?
The *First* NP-Complete Problem

**Theorem.** CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

**Proof (sketch).**
- Any algorithm that takes a fixed number of bits $n$ as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

  - Consider some problem $X$ in NP. It has a poly-time certifier $C(s, t)$.
  - To determine whether $s$ is in $X$, need to know if there exists a certificate $t$ of length $p(|s|)$ such that $C(s, t) = \text{yes}$.
  - View $C(s, t)$ as an algorithm on $|s| + p(|s|)$ bits (input $s$, certificate $t$) and convert it into a poly-size circuit $K$.
    - First $|s|$ bits are hard-coded with $s$.
    - Remaining $p(|s|)$ bits represent bits of $t$.
  - Circuit $K$ is satisfiable iff there exists $t$ s.t. $C(s, t) = \text{yes}$.

Example

**Construction below creates a circuit $K$ whose inputs can be set so that $K$ outputs true iff graph $G$ has an independent set of size 2.**

$$
\begin{align*}
\text{independent set of size 2?} & \\
\text{independent set?} & \\
\text{Hard-coded inputs (graph description)} & \\
\text{n inputs (nodes in independent set)} & \\
\end{align*}
$$

Establishing NP-Completeness

**Remark.** Once we establish first "natural" NP-complete problem, others fall like dominoes.

**Recipe to establish NP-completeness of problem $Y$.**
- Step 1. Show that $Y$ is in NP.
- Step 2. Choose an NP-complete problem $X$.
- Step 3. Prove that $X \leq_P Y$.

**Justification.** If $X$ is an NP-complete problem, and $Y$ is a problem in NP with the property that $X \leq_P Y$ then $Y$ is NP-complete.

**Proof.** Let $W$ be any problem in NP. Then $W \leq_P X \leq_P Y$.
- By transitivity, $W \leq_P Y$.
- Hence $Y$ is NP-complete.
  - by definition of NP-complete
  - by assumption

Some NP-Complete Problems

**Six basic genres of NP-complete problems and paradigmatic examples.**
- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

**Practice.** Most NP problems are either known to be in P or NP-complete.

**Notable exceptions.** Factoring, graph isomorphism, Nash equilibrium.
Extent and Impact of NP-Completeness

Extent of NP-completeness [Papadimitriou 1995]
- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year (title, abstract, keywords).
- More than “compiler”, “operating system”, “database”
- Broad applicability and classification power.
- “Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly.”

NP-completeness can guide scientific inquiry.
- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.

More Hard Computational Problems

- Aerospace engineering: optimal mesh partitioning for finite elements.
- Biology: protein folding.
- Chemical engineering: heat exchanger network synthesis.
- Civil engineering: equilibrium of urban traffic flow.
- Economics: computation of arbitrage in financial markets with friction.
- Electrical engineering: VLSI layout.
- Environmental engineering: optimal placement of contaminant sensors.
- Financial engineering: find minimum risk portfolio of given return.
- Game theory: find Nash equilibrium that maximizes social welfare.
- Genomics: phylogeny reconstruction.
- Mechanical engineering: structure of turbulence in shear flows.
- Medicine: reconstructing 3-D shape from biplane angiogram.
- Operations research: optimal resource allocation.
- Physics: partition function of 3-D Ising model in statistical mechanics.
- Politics: Shapley-Shubik voting power.
- Pop culture: Minesweeper consistency.
- Statistics: optimal experimental design.

Asymmetry of NP

Asymmetry of NP. We only need to have short proofs of yes instances.

Ex 1. SAT vs. TAUTOLOGY.
- Can prove a CNF formula is satisfiable by giving such an assignment.
- How could we prove that a formula is not satisfiable?

Ex 2. HAM-CYCLE vs. NO-HAM-CYCLE.
- Can prove a graph is Hamiltonian by giving such a Hamiltonian cycle.
- How could we prove that a graph is not Hamiltonian?

Remark. SAT is NP-complete and SAT \(\equiv\) P TAUTOLOGY, but how do we classify TAUTOLOGY?

\[\text{not even known to be in NP}\]

NP and co-NP

NP. Decision problems for which there is a poly-time certifier.
Ex. SAT, HAM-CYCLE, COMPOSITES.

Def. Given a decision problem \(X\), its complement \(\overline{X}\) is the same problem with the yes and no answers reverse.
Ex. \(X = \{0, 1, 4, 6, 8, 9, 10, 12, 14, 15, \ldots\}\)
\(\overline{X} = \{2, 3, 5, 7, 11, 13, 17, 23, 29, \ldots\}\)

co-NP. Complements of decision problems in NP.
Ex. TAUTOLOGY, NO-HAM-CYCLE, PRIMES.

NP = co-NP?

Fundamental question. Does NP = co-NP?
- Do yes instances have succinct certificates if no instances do?
- Consensus opinion: no.

Theorem. If NP = co-NP, then P = NP.
Pf idea.
- P is closed under complementation.
- If P = NP, then NP is closed under complementation.
- In other words, NP = co-NP.
- This is the contrapositive of the theorem.
### Good Characterizations

**Good characterization.** [Edmonds 1965] $\text{NP} \cap \text{co-NP}$.
- If problem $X$ is in both $\text{NP}$ and $\text{co-NP}$, then:
  - for yes instance, there is a succinct certificate
  - for no instance, there is a succinct disqualifier
- Provides conceptual leverage for reasoning about a problem.

**Ex.** Given a bipartite graph, is there a perfect matching.
- If yes, can exhibit a perfect matching.
- If no, can exhibit a set of nodes $S$ such that $|N(S)| < |S|$.

### Observation. $\text{P} \subseteq \text{NP} \cap \text{co-NP}$

**Proof of max-flow min-cut theorem led to stronger result**
that max-flow and min-cut are in $\text{P}$.

- Many examples where problem found to have a non-trivial good characterization, but only years later discovered to be in $\text{P}$.
  - [Khachiyan, 1979] linear programming
  - [Agrawal-Kayal-Saxena, 2002] primality testing

**Fact.** Factoring is in $\text{NP} \cap \text{co-NP}$, but not known to be in $\text{P}$.

- If poly-time algorithm for factoring, can break RSA cryptosystem.

### 8.5 Sequencing Problems

**Basic genres:***
- Packing problems: SET-PACKING, INDEPENDENT-SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.
**Hamiltonian Cycle**

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

**YES**: vertices and faces of a dodecahedron.

**NO**: bipartite graph with odd number of nodes.

---

**Directed Hamiltonian Cycle**

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

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

**Pf.** Given a directed graph $G = (V, E)$, construct an undirected graph $G'$ with $3n$ nodes.

![Graph transformation from directed to undirected]

**Claim**: $G$ has a Hamiltonian cycle iff $G'$ does.

**Pf.**

- **$\Rightarrow$** Suppose $G$ has a directed Hamiltonian cycle $\Gamma$.
- Then $\Gamma'$ has an undirected Hamiltonian cycle (same order).
- For each node $v$ in directed path cycle replace $v$ with $v_{in}, v, v_{out}$.

**Pf.**

- **$\Leftarrow$** Suppose $G'$ has an undirected Hamiltonian cycle $\Gamma'$.
- $\Gamma'$ must visit nodes in $G'$ using one of following two orders:
- Blue nodes in $\Gamma'$ make up directed Hamiltonian cycle $\Gamma$ in $G$, or reverse of one.

---

**3-SAT Reduces to Directed Hamiltonian Cycle**

**Claim**: $3\text{-SAT} \leq \text{DIR-HAM-CYCLE}$.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance of $\text{DIR-HAM-CYCLE}$ 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.
3-SAT Reduces to Directed Hamiltonian Cycle

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 $x_i = 1$.

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

**Pf.**
- Suppose $\Phi$ has a satisfying assignment $x^*$.
- If $x^*_i = 1$, traverse row $i$ from left to right.
- If $x^*_i = 0$, traverse row $i$ from right to left.

3-SAT Reduces to Directed Hamiltonian Cycle

**Pf.**

1. Redo proof for $\text{DIR-HAM-CYCLE}$, ignoring back-edge from $t$ to $s$.
2. Show $\text{HAM-CYCLE} \leq_p \text{LONGEST-PATH}$.
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 33,638 cities in US with a population of at least 600
Reference: http://www.tsp.gatech.edu

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

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

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

Claim. HAM-CYCLE $\leq_p$ TSP.

Pf.
• Given instance $G = (V, E)$ of HAM-CYCLE, create $n$ cities with distance function
\[
d(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 2 & \text{if } (u, v) \not\in E \end{cases}
\]
• TSP instance has tour of length $\leq n$ if and only if $G$ is Hamiltonian.

Remark. TSP instance in reduction satisfies \(\Delta\)-inequality.
8.7 Graph Coloring

Basic genres:
- Packing problems: SET-PACKING, INDEPENDENT SET,
- Covering problems: SET-COVER, VERTEX-COVER,
- Constraint satisfaction problems: SAT, 3-SAT,
- Sequencing problems: HAMILTONIAN-CYCLE, TSP,
- Partitioning problems: 3D-MATCHING, 3-COLOR,
- Numerical problems: SUBSET-SUM, KNAPSACK.

Register Allocation

Register allocation. Assign program variables to machine register so that no more than k registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables names, edge between u and v if there exists an operation where both u and v are “live” at the same time.

Observation. [Chaitin 1982] Can solve register allocation problem iff interference graph is k-colorable.

Fact. 3-COLOR \leq_k k-REGISTER-ALLOCATION for any constant k \geq 3.

3-Colorability

Claim. 3-SAT \leq_p 3-COLOR.

Pf. Given 3-SAT instance \varphi, we construct an instance of 3-COLOR that is 3-colorable iff \varphi is satisfiable.

Construction.
- For each literal, create a node.
- Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.
- Connect each literal to its negation.
- For each clause, add gadget of 6 nodes and 13 edges.
- To be described next.

Claim. Graph is 3-colorable iff \varphi is satisfiable.

Pf. Suppose graph is 3-colorable.
- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.

Claim. Graph is 3-colorable iff \varphi is satisfiable.

Pf. Suppose graph is 3-colorable.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.
Claim. Graph is 3-colorable iff $\phi$ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.
   - Consider assignment that sets all T literals to true.
   - (i) ensures each literal is T or F.
   - (ii) ensures a literal and its negation are opposites.
   - (iv) ensures at least one literal in each clause is T.

3-Colorability

Claim. Graph is 3-colorable iff $\phi$ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT formula $\phi$ is satisfiable.
   - Color all true literals T.
   - Color node below green node F, and node below that B.
   - Color remaining middle row nodes B.
   - Color remaining bottom nodes T or F as forced.