The Classes P and NP

Classes of Decision Problems

- **P**: Problems for which there exists a deterministic polynomial-time algorithm
- **NP**: Problems for which there exists a non-deterministic polynomial-time algorithm

- What is “non-deterministic polynomial-time”?
  - Solutions are small (== polynomial-size)
  - Guessing is free, but solutions must be checkable in polynomial time
    - “Succinct Certificate”

- **P ⊆ NP**, but whether **P = NP** is not known
  - Most people believe **P ≠ NP**, which would imply **P ⊊ NP**
NP-Hard, NP-Complete

- **NP-Hard**: Problem $X$ is NP-hard if *every* problem in NP is *polynomially reducible* to $X$

- **NP-Complete**: Problem $X'$ is NP-complete if:
  - $X'$ belongs to NP, and
  - $X'$ is NP-hard

  I.e., (1) Every problem in NP is poly-time reducible to $X'$; (2) “$X'$ is as hard as any problem in NP” (“as hard as it gets”)

- Also, $Y$ is **NP-complete** if $Y \in \text{NP}$ and some NP-complete problem $X$ is *polynomially reducible* to $Y$

- **NP-Complete problems are the *hardest* problems in NP**
Picture of $P$, $NP$, $NP$-Hard, $NP$-Complete

SAT, 3SAT, Vertex Cover, Independent Set, Dominating Set, Hamilton Cycle, CLIQUE, NumPart, ….

$P \subseteq NP$

$P = NP$?
SAT is NP-Complete (Cook, 1971)

• **Cook’s theorem (1971):** SAT is NP-complete
  - Levin (1972))

  ![Diagram of SAT being NP-complete]

  ![Diagram of polynomial reducibility]

• Once we know one problem to be NP-complete, proving other problems NP-complete becomes easier

• **A new problem $Y$ is NP-complete if:**
  - (1) $Y$ is in NP
  - (2) SAT or any other NP-complete problem is *polynomially reducible* to $Y$
    - I.e., “reduce from SAT to $Y$”
Two Parts To An NP-Completeness Proof

• Problem is in NP

• A known NP-complete problem reduces to it
NP-Complete Problems 0, 0’

- **Number Partition (NumPart):** Given a set of numbers \( S = \{x_1, x_2, \ldots, x_n\} \), is there a disjoint partition of \( S \) into \( S_1, S_2 \) such that sum of numbers in \( S_1 \) = sum of numbers of \( S_2 \)?

  disjoint partition: \( S_1 \cup S_2 = S \), and \( S_1 \cap S_2 = \emptyset \)

- **k-Tolerant Number Partition (kTNP):** Given a set of numbers \( S = \{x_1, x_2, \ldots, x_n\} \) and a number \( k \), is there a disjoint partition of \( S \) into \( S_1, S_2 \) such that \( |(\text{sum of numbers in } S_1) - (\text{sum of numbers of } S_2)| \leq k \)?
NP-Complete Problems 0, 0’

- **Number Partition (NumPart):** Given a set of numbers \( S = \{x_1, x_2, \ldots, x_n\} \), is there a disjoint partition of \( S \) into \( S_1, S_2 \) such that sum of numbers in \( S_1 = \) sum of numbers of \( S_2 \)?

  *disjoint partition: \( S_1 \cup S_2 = S \), and \( S_1 \cap S_2 = \emptyset \)*

- **k-Tolerant Number Partition (kTNP):** Given a set of numbers \( S = \{x_1, x_2, \ldots, x_n\} \) and a number \( k \), is there a disjoint partition of \( S \) into \( S_1, S_2 \) such that \(|(\text{sum of numbers in } S_1) - (\text{sum of numbers of } S_2)| \leq k|\)?

- **NumPart \( \leq_p \) kTNP**
NP-Complete Problems 0, 0’

- **Number Partition (NumPart):** Given a set of numbers $S = \{x_1, x_2, \ldots, x_n\}$, is there a disjoint partition of $S$ into $S_1, S_2$ such that sum of numbers in $S_1 = \text{sum of numbers of } S_2$?  
  
  _disjoint partition:_ $S_1 \cup S_2 = S$, and $S_1 \cap S_2 = \emptyset$

- **k-Tolerant Number Partition (kTNP):** Given a set of numbers $S = \{x_1, x_2, \ldots, x_n\}$ and a number $k$, is there a disjoint partition of $S$ into $S_1, S_2$ such that $|(\text{sum of numbers in } S_1) - (\text{sum of numbers of } S_2)| \leq k$?

- $kTNP \leq_p \text{NumPart}$
NP-Complete Problems 0, 0’

• Reduction of k-Tolerant Number Partition to Number Partition
NumPart, Bin-Packing

- **NumPart(S):** Given a set of numbers $S = \{x_1, x_2, \ldots, x_n\}$ with $0 < x_i \leq 1$, there a disjoint partition of $S$ into $S_1, S_2$ (i.e., where $S_1 \cup S_2 = S$, and $S_1 \cap S_2 = \emptyset$) such that $\sum_{x_i \in S_1} = \sum_{x_j \in S_2} \ ?$

- **Bin-Packing(L,B):** Given a list of items $L = \{x_1, x_2, \ldots, x_n\}$ with $0 < x_i \leq 1$, and a number $B$, can $L$ be packed into $B$ unit-capacity bins?

Given that NumPart is NP-complete, prove that Bin-Packing is NP-complete.
NumPart is NP-C: Prove Bin-Packing is NP-C

• **NumPart(S):** Given a set of numbers $S = \{x_1, x_2, \ldots, x_n\}$ with $0 < x_i \leq 1$, there a disjoint partition of $S$ into $S_1, S_2$ (i.e., where $S_1 \cup S_2 = S$, and $S_1 \cap S_2 = \emptyset$) such that $\sum_{x_i \in S_1} = \sum_{x_j \in S_2}$?

• **Bin-Packing(L,B):** Given a list of items $L = \{x_1, x_2, \ldots, x_n\}$ with $0 < x_i \leq 1$, and a number $B$, can $L$ be packed into $B$ unit-capacity bins?
NP-Complete Problems #1, #2, #3

The following five problems are NP-complete
(Manber, Introduction to Algorithms: A Creative Approach)

• **Definition:** A *vertex cover of* $G = (V, E)$ is $V' \subseteq V$ such that every edge in $E$ is incident to some $v \in V'$.

• **VC(G,k):** Given undirected $G = (V, E)$ and integer $k$, does $G$ have a vertex cover with $\leq k$ vertices?

• **CLIQUE(G,k):** Does $G$ contain a clique of size $\geq k$?

• **3COLOR(G):** Is $G$ properly colorable with 3 colors?
NP-Complete Problems #1, #2, #3

• **VC(G,k):** Given undirected $G = (V, E)$ and integer $k$, does $G$ have a vertex cover with $\leq k$ vertices?

• **CLIQUE(G,k):** Does $G$ contain a clique of size $\geq k$?

• **3COLOR(G):** Is $G$ properly colorable with 3 colors?
NP-Complete Problems #4, #5

- **Definition:** A dominating set $D$ of $G = (V, E)$ is $D \subseteq V$ such that every $v \in V$ is either in $D$ or adjacent to at least one vertex of $D$.

- **$DS(G,k):$** given $G$, $k$, does $G$ have a dominating set of size $\leq k$?

- **$3SAT(F):$** Given Boolean expression $F$ in CNF such that each clause has exactly 3 variables, is $F$ satisfiable?
To prove NP-completeness of a new problem:
- Prove that the problem belongs to NP
- Show polynomial reducibility from a known NP-complete problem

This slide shows the order in which “classic” NP-completeness reductions are often presented.
NP-Completeness Proof: Vertex Cover (VC)

• **Problem:** Given undirected $G = (V, E)$ and integer $k$, does $G$ have a vertex cover with $\leq k$ vertices?

• **Theorem:** The VC problem is NP-complete.

• **Proof:**
  - **Approach:** Reduction from CLIQUE
  - I.e., given that CLIQUE is NP-complete, we will prove that VC is NP-complete
  - **VC is in NP.** This is true since we can guess a cover of size $\leq k$ and check it in polynomial time
  - **Goal:** Transform arbitrary CLIQUE instance into VC instance such that CLIQUE answer is “yes” iff VC answer is “yes”
NP-Complete Problems #1, #2

• **Definition:** A vertex cover of $G = (V, E)$ is $V' \subseteq V$ such that every edge in $E$ is incident to some $v \in V'$.

• **$VC(G,k)$:** Given undirected $G = (V, E)$ and integer $k$, does $G$ have a vertex cover with $\leq k$ vertices?

• **$CLIQUE(G,k)$:** Does $G$ contain a clique of size $\geq k$?

• “Given that $CLIQUE$ is NP-complete, prove that $VC$ is NP-complete” → Reduce from _____ to ______
NP-Complete Problems #1, #2

• **Definition:** A vertex cover of $G = (V, E)$ is $V' \subseteq V$ such that every edge in $E$ is incident to some $v \in V'$.

• **VC($G,k$):** Given undirected $G = (V, E)$ and integer $k$, does $G$ have a vertex cover with $\leq k$ vertices?

• **CLIQUE($G,k$):** Does $G$ contain a clique of size $\geq k$?

• “Given that CLIQUE is NP-complete, prove that VC is NP-complete” → Reduce from ______ to ______
Step 1: VC is in NP

• This is true since we can guess a cover of size $\leq k$ and check it in polynomial time.

• **Step #2 is Reduction:** Transform arbitrary CLIQUE instance into VC instance such that CLIQUE answer is “yes” iff VC answer is “yes.”
Step 2: Reduce CLIQUE to VC

Transform arbitrary CLIQUE instance into VC instance such that CLIQUE answer is “yes” iff VC answer is “yes”

Clique = \{u, v, x, y\}  
Vertex cover = \{w, z\}

CLIQUE(G, k) ⇒ VC(G’, n-k)

- **Claim**: CLIQUE(G, k) has **same** answer as VC \((\overline{G}, n - k)\), where \(n = |V|\).
- **Observe**: If \(U \subseteq V\) is a clique in \(G\), then \(V \setminus U\) is VC in \(\overline{G}\).
VC(\(G',n-k\)) \Rightarrow\text{CLIQUE}(G,k)

- **Observe**: If \(D \subseteq V\) is a VC in \(\overline{G}\), then \(\overline{G}\) has no edge between vertices in \(V \setminus D\).

\[
\begin{align*}
\text{Clique} &= \{u, v, x, y\} \\
\text{Vertex cover} &= \{w, z\}
\end{align*}
\]

**Transform** is *polynomial time*. 
“Given that X is NP-C, prove that Y is NP-C”

• Remember: reduce from X to Y

• Step 1: Y is in NP

• Step 2: Transform arbitrary instance \( \text{inst}_X \) of X into an instance \( \text{inst}_Y \) of Y such that \( X(\text{inst}_X) \) is YES iff \( Y(\text{inst}_Y) \) is YES

• Practice problems are on HW and in slides; will “consolidate” a week from today
“Given that $X$ is NP-C, prove that $Y$ is NP-C”

- $X = \text{Hamilton Path, } Y = \text{Hamilton Cycle}$
  - ($\text{Rudrata} = \text{Hamilton}$)

- $X = \text{Hamilton Cycle, } Y = \text{Hamilton Path}$

- $X = \text{SAT, } Y = \text{3-SAT}$

- $X = \text{CLIQUE, } Y = \text{DOM\_SET}$
Techniques for Dealing with Hard Problems

• Implicit enumeration
  – Backtrack
  – Branch-and-Bound

• Ad hoc methods (heuristics)
  – Provably good (bounded performance ratio)
    • Christofides’ method for metric Traveling Salesman Problem
    • Engineer’s method for Number Partitioning
  – Not provably good – “metaheuristics”
    • Genetic Optimization (evolutionary algorithms)
    • Simulated Annealing
    • Tabu Search
Backtrack

- This development is from T. C. Hu, *Combinatorial Algorithms*, Addison-Wesley, 1982

- Set up a 1-1 correspondence between configurations and possible solution sequences (or, partial solution vectors).

- **Decision Tree**: The root corresponds to the initial state of the problem (usually is null, means no decision is made), and each branch corresponds to a decision concerning one parameter.
A graph is legally colored with $k$ colors if each vertex has a color (label) between 1 and $k$ (inclusive), and no adjacent vertices have the same color.
A graph is legally colored with \( k \) colors if each vertex has a color (label) between 1 and \( k \) (inclusive), and no adjacent vertices have the same color.

\[ \text{Three colors: R, G, B} \]

Does the ordering of the vertices (1, 2, 3, 4, 5) matter?
“Domino Effect”

• Suppose a **solution** is represented by a **vector**. A **partial solution** is represented by a **partial vector**.

• If a partial vector does not satisfy the solution requirements, there is no point in extending the partial vector into a more complete solution.

• **Domino Effect:** $P_k$ false $\rightarrow P_{k+1}$ false

*Domino effect must hold for backtrack to work!*
Backtrack Example: Non-Attacking Queens

• Put N queens on an N x N chessboard such that no queen attacks another queen

A queen in chess can *attack* any square on the same row, column, or diagonal. Given an n x n chessboard, we seek to place n queens onto squares of the chessboard, such that no queen attacks another queen. The example shows a placement (red squares) of four mutually non-attacking queens.
Branch-and-Bound (B&B)

• Variant of backtrack with costs
  – Associate a cost with a partial solution, such that the cost of a parent is always less than or equal to the cost of its child in the decision tree
  – do not branch from an internal node whose cost is higher than the current bound = cost of the minimum-cost complete solution found so far
  – the bound is updated if a better solution is found

• Key points
  – Used for optimization problems
  – Cost-driven
  – Bounding prunes the decision tree, saves time
What Does B&B Look Like for the TSP?
What Does B&B Look Like for the TSP?
Branch-and-Bound Example: Game Tree

• In games (e.g., chess) can model the different stages of the game by a rooted tree
  – Don’t need to consider all possible situations of a game
  – Can predict the outcome of the game using the concept of branch-and-bound

• Questions
  – Who is the winner if both players play optimally?
  – How much is the payoff?
    • E.g., we may initially only know the payoff of the terminal nodes (end-states) of the game.
Two-Player Game Tree

• Payoff of the Game = amount the 1st player receives at the end of the game

1st player wants to \textit{maximize} payoff (Call her Max)

2nd player wants to \textit{minimize} payoff (Call her Min)
We can see that the value of the game is 1 (node A) by examining (bottom-up) the entire game tree. In general, (bottom-up) examination of an entire game tree is not feasible. Applying DFS with branch-and-bound using “α-β pruning” improves efficiency.
NP-Completeness Proof: DS

Definition: A dominating set $D$ of $G = (V, E)$ is $D \subseteq V$ such that every $v \in V$ is either in $D$ or adjacent to at least one vertex of $D$

- **Problem**: given $G$, $k$ does $G$ have a dominating set of size $\leq k$?

- **Theorem**: DS is NP-complete

- **Proof**: (reduction from VC)
  - $DS \in NP$
  - Given instance $(G,k)$ of VC, make every edge of $G$ into a triangle and answer DS.
NP-Completeness Proof: DS

G: \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} G':

\[
\begin{array}{ccc}
\text{v} & \text{w} & \text{u} \\
\text{z} & & \text{u} & \text{w} \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{v} & \text{w} & \text{u} \\
\text{z} & & \text{u} & \text{w} \\
\end{array}
\]

- G’ has DS D of size k \textbf{iff} G has VC C of size k.
  - Without loss of generality, assume \( D \subseteq V \), D is a DS \( \Rightarrow \) every edge hits some vertex in D \( \Rightarrow \) D is a VC in G
  - C is a VC \( \Rightarrow \) new and old vertices dominated
NP-Completeness Proof: 3SAT

• **3SAT**: Given a Boolean expression in CNF such that each clause has exactly 3 variables, determine satisfiability.

• **Theorem**: 3SAT is NP-Complete

• **Proof**:
  - **3SAT ∈ NP**
  - Let E be an arbitrary instance of SAT
    
    \[
    C = (x_1 + x_2 + \ldots + x_k)
    \]
    
    arbitrary clause of E
    
    For \(k \geq 4\)
    
    \[
    C' = (x_1 + x_2 + y_1) \cdot (x_3 + \overline{y_1} + y_2) \cdot (x_4 + \overline{y_2} + y_3) \cdot \ldots \cdot (x_{k-1} + x_k + \overline{y_{k-3}})
    \]
NP-Completeness Proof: 3SAT

C’ is satisfiable iff C is satisfiable:
– C’= 1 ⇒ at least one \( x_i = 1 \) ⇒ C = 1
– C = 1 ⇒ at least one \( x_i = 1 \) ⇒ can set \( y_i \)'s so that C’ = 1

\[
C = (x_1 + x_2)
\]

\[
C = (x_1 + x_2 + y) \cdot (x_1 + x_2 + \bar{y})
\]

\[
C = (x_1)
\]

\[
C = (x_1 + y_1 + y_2) \cdot (x_1 + \bar{y}_1 + y_2) \cdot (x_1 + y_1 + \bar{y}_2) \cdot (x_1 + \bar{y}_1 + \bar{y}_2)
\]
NP-Completeness Proof: CLIQUE

• **Problem:** Does G=(V,E) contain a clique of size \( k \)?
• **Theorem:** Clique is NP-Complete. (reduction from SAT)
• **Idea:** Make “column” of \( k \) vertices for each clause with \( k \) variables.
  – No edge within a column.
  – All other edges present except between \( x \) and \( \overline{x} \)
NP-Completeness Proof: CLIQUE

• Example: \[ E = (x + y + \overline{z}) \cdot (\overline{x} + \overline{y} + z) \cdot (y + \overline{z}) \]

\[ G = \]

\[ G \text{ has } m\text{-clique (} m \text{ is the number of clauses in } E), \text{ iff } E \text{ is satisfiable.} \]

(Assign value 1 to all variables in clique)