A “hard” $a \rightarrow [B \overset{b}{\rightarrow} (p_{N}) \overset{y}{\rightarrow} (p_{N})] \quad \rightarrow y_{N}$

$\downarrow B$ must also be “hard”
The Classes P and NP

Classes of Decision Problems

- **P**: Problems for which there exists a \textit{deterministic} polynomial-time algorithm
- **NP**: Problems for which there exists a \textit{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 \subseteq NP, but whether P = NP is not known  
  - Most people believe P \neq NP, which would imply P \subset 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:
  1. $X'$ belongs to NP, and
  2. $X'$ 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 $Y \in NP$ and some NP-complete problem $X'$ is *polynomially reducible* to $Y$ (because of polynomials being closed under composition)

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

Figure 8.6 The space NP of all search problems, assuming P ≠ NP.

- P
- NP-complete

Increasing difficulty

- Graph Non-Isomorphism
- Shortest Path
- NP-hard
- Halting Problem
- SAT, 3SAT, Vertex Cover, Independent Set, Dominating Set, Hamilton Cycle, CLIQUE, NumPart, ….

√ P ⊆ C ⊆ NP

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

The mapping \( V \leftrightarrow V' \) "guess for free" check in \( \Theta(F) \) time
SAT is NP-Complete (Cook, 1971)

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

- 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

1. Problem is in NP

2. 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 \(|(\text{sum of numbers in } S_1) - (\text{sum of numbers of } S_2)| \leq k\)?

  \[
  \text{kTNP}(S, k) = \begin{cases} 
  \checkmark & \text{if } |(\text{sum of numbers in } S_1) - (\text{sum of numbers of } S_2)| \leq k \\
  \times & \text{otherwise}
  \end{cases}
  \]
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 $|$(sum of numbers in $S_1$) – (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 $\leq_p$ NumPart**

\[
(S, k) \rightarrow \begin{array}{c}
S' \\
(\text{NumPart})
\end{array} \quad \Rightarrow \\
\Downarrow
\]

\[
kTNP (\{5, 3, 2, 8\}; 1) \rightarrow \text{N} \\
kTNP (\{5, 3, 2, 8\}; 2) \rightarrow \text{Y} \quad \{5, 3\} \quad \{2, 8\}
\]
NP-Complete Problems 0, 0’

- Reduction of k-Tolerant Number Partition (kTNP) to Number Partition (NumPart)

\[ S' = \{ x_1, \ldots, x_n, k \} \]

- \( T_{kTNP} \): 
  - \( S' = S \cup \{ k \} \)
  - \( d = 0, 1, \ldots, k \)
  - if \( Y \) for some \( d \), then \( Y \)
  - otherwise, \( N \)

- \( kTNP(1,1000,3,1) = N \)
- \( kTNP(1,13,1000) = Y \)
- \( NumPart\{1,1,1000\} = N \)
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 \( \sum_{x_i \in S_1} = \sum_{x_j \in S_2} \) ?

\[ S = \{ 5, 3, 2, 2, 8 \} \quad \text{(online)} \]

\[ L = \{ 5/10, 3/10, 2/10, 2/10, 8/10 \} \]

• **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? e.g., grocery bags

\[ \checkmark \text{Bin-Packing is in NP.} \]

\[ \text{NumPart} \leq_p \text{BinPack} \]

Given that **NumPart** is NP-complete, prove that **Bin-Pack**ing is NP-complete
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 #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?
NP-Completeness Proofs: Reductions

All NP problems

SAT

Clique

Vertex Cover

Dominating Set

3SAT

3-Colorability

• 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 $CLIQUE$ to $VC$
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\}

\[
\text{CLIQUE}(G, k) = \text{?} \quad \Rightarrow \quad \text{VC}(\overline{G}, n-k) = \text{?}
\]

- **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 \(\overline{G}\).
\[ VC(G', n-k) \Rightarrow CLIQUE(G, k) \]

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

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 = Hamilton Path, Y = Hamilton Cycle
  – (Rudrata = Hamilton)

• X = Hamilton Cycle, Y = Hamilton Path

• X = SAT, Y = 3-SAT

• X = CLIQUE, Y = 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.

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: \( P_k \text{ false} \rightarrow P_{k+1} \text{ false} \)

*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.
Backtrack Example: 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 1\textsuperscript{st} player receives at the end of game

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

2\textsuperscript{nd} player wants to \textbf{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

- $G'$ has DS $D$ of size $k$ 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

  \[ C = (x_1 + x_2 + ... + x_k) \] arbitrary clause of E

  For k ≥ 4

  \[ C' = (x_1 + x_2 + y_1) \cdot (x_3 + \overline{y_1} + y_2) \cdot (x_4 + \overline{y_2} + y_3) \cdot \]

  \[ ... \cdot (x_{k-1} + x_k + \overline{y_{k-3}}) \]
NP-Completeness Proof: 3SAT

C’ is satisfiable iff C is satisfiable:
- C’ = 1 \implies \text{at least one } x_i = 1 \implies C = 1
- C = 1 \implies \text{at least one } x_i = 1 \implies \text{can set } y_i\text{'s so that } C’ = 1

\[ C = (x_1 + x_2) \]
\[ C = (x_1 + x_2 + y)(x_1 + x_2 + \overline{y}) \]
\[ C = (x_1) \]
\[ C = (x_1 + y_1 + y_2)(x_1 + \overline{y_1} + y_2)(x_1 + y_1 + \overline{y_2})(x_1 + \overline{y_1} + \overline{y_2}) \]
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 \( \bar{x} \).
NP-Completeness Proof: CLIQUE

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

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

\[ x \quad \overline{x} \quad y \quad \overline{y} \quad z \quad \overline{z} \]

• \( G \) has \( m \)-clique (\( m \) is the number of clauses in \( E \)), iff \( E \) is satisfiable.

(Assign value 1 to all variables in clique)