Recap

- **NP**: languages that have polynomial time certifiers/verifiers
- A language $L$ is **NP-Complete** iff
  - $L$ is in **NP**
  - for every $L'$ in **NP**, $L' \leq_P L$
- $L$ is **NP-Hard** if for every $L'$ in **NP**, $L' \leq_P L$.
- Cook-Levin theorem...

**Theorem (Cook-Levin)**

*Circuit-SAT* and *SAT* are **NP-Complete**.

Recap contd

**Theorem (Cook-Levin)**

*Circuit-SAT* and *SAT* are **NP-Complete**.

- Establish **NP-Completeness** via reductions:
  - **SAT** $\leq_P 3\text{SAT}$ and hence **3SAT** is **NP-complete**
  - **3SAT** $\leq_P \text{Independent Set}$ (which is in **NP** and hence...
  - **Independent Set** is **NP-Complete**.
  - **Vertex Cover** is **NP-Complete**
  - **Clique** is **NP-Complete**.
  - **Set Cover** is **NP-Complete**.

Today

Prove
- **Hamiltonian Cycle** Problem is **NP-Complete**
- **3-Coloring** is **NP-Complete**
### Directed Hamiltonian Cycle

**Input** Given a directed graph $G = (V, E)$ with $n$ vertices

**Goal** Does $G$ have a Hamiltonian cycle?
- A Hamiltonian cycle is a cycle in the graph that visits every vertex in $G$ exactly once

### Directed Hamiltonian Cycle is NP-Complete

- Directed Hamiltonian Cycle is in $NP$
  - **Certificate**: Sequence of vertices
  - **Certifier**: Check if every vertex (except the first) appears exactly once, and that consecutive vertices are connected by a directed edge

### Hardness
We will show

$3$-$SAT \leq_P$ Directed Hamiltonian Cycle

### Reduction

1. **Given** $3$-$SAT$ formula $\varphi$ create a graph $G_\varphi$ such that
   - $G_\varphi$ has a Hamiltonian cycle if and only if $\varphi$ is satisfiable
   - $G_\varphi$ should be constructible from $\varphi$ by a polynomial time algorithm $A$

2. **Notation**: $\varphi$ has $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses $C_1, C_2, \ldots, C_m$.

### Reduction: First Ideas

1. **Viewing** $SAT$: Assign values to $n$ variables, and each clauses has 3 ways in which it can be satisfied.
2. **Construct** graph with $2^n$ Hamiltonian cycles, where each cycle corresponds to some boolean assignment.
3. **Then** add more graph structure to encode constraints on assignments imposed by the clauses.
The Reduction: Phase I

- Traverse path \( i \) from left to right iff \( x_i \) is set to true
- Each path has \( 3(m+1) \) nodes where \( m \) is number of clauses in \( \varphi \); nodes numbered from left to right (1 to \( 3m+3 \))

![Diagram of Reduction Phase I](image)

Correctness Proof

**Proposition**

\( \varphi \) has a satisfying assignment iff \( G_\varphi \) has a Hamiltonian cycle.

**Proof.**

\( \Rightarrow \) Let \( a \) be the satisfying assignment for \( \varphi \). Define Hamiltonian cycle as follows
- If \( a(x_i) = 1 \) then traverse path \( i \) from left to right
- If \( a(x_i) = 0 \) then traverse path \( i \) from right to left
- For each clause, path of at least one variable is in the “right” direction to splice in the node corresponding to clause \( \square \)

The Reduction: Phase II

- Add vertex \( c_j \) for clause \( C_j \). \( c_j \) has edge from vertex \( 3j \) and to vertex \( 3j+1 \) on path \( i \) if \( x_i \) appears in clause \( C_j \), and has edge from vertex \( 3j+1 \) and to vertex \( 3j \) if \( \neg x_i \) appears in \( C_j \).

![Diagram of Reduction Phase II](image)

Hamiltonian Cycle \( \Rightarrow \) Satisfying assignment

Suppose \( \Pi \) is a Hamiltonian cycle in \( G_\varphi \)
- If \( \Pi \) enters \( c_j \) (vertex for clause \( C_j \)) from vertex \( 3j \) on path \( i \) then it must leave the clause vertex on edge to \( 3j+1 \) on the same path \( i \)
  - If not, then only unvisited neighbor of \( 3j+1 \) on path \( i \) is \( 3j+2 \)
  - Thus, we don’t have two unvisited neighbors (one to enter from, and the other to leave) to have a Hamiltonian Cycle
- Similarly, if \( \Pi \) enters \( c_j \) from vertex \( 3j+1 \) on path \( i \) then it must leave the clause vertex \( c_j \) on edge to \( 3j \) on path \( i \)
Example

Hamiltionian Cycle

Problem

Input  Given undirected graph $G = (V, E)$
Goal  Does $G$ have a Hamiltonian cycle? That is, is there a cycle that visits every vertex exactly one (except start and end vertex)?

NP-Completeness

Theorem

Hamiltonian cycle problem for undirected graphs is NP-Complete.

Proof.

- The problem is in NP; proof left as exercise.
- Hardness proved by reducing Directed Hamiltonian Cycle to this problem
Reduction Sketch

Goal: Given directed graph $G$, need to construct undirected graph $G'$ such that $G$ has Hamiltonian Path iff $G'$ has Hamiltonian path

Reduction
- Replace each vertex $v$ by 3 vertices: $v_{in}$, $v$, and $v_{out}$
- A directed edge $(a, b)$ is replaced by edge $(a_{out}, b_{in})$

Graph Coloring

Problem: Graph Coloring

Instance: $G = (V, E)$: Undirected graph, integer $k$.
Question: Can the vertices of the graph be colored using $k$ colors so that vertices connected by an edge do not get the same color?

Graph 3-Coloring

Problem: 3 Coloring

Instance: $G = (V, E)$: Undirected graph.
Question: Can the vertices of the graph be colored using 3 colors so that vertices connected by an edge do not get the same color?

Reduction: Wrapup
- The reduction is polynomial time (exercise)
- The reduction is correct (exercise)
**Graph Coloring**

- **Observation:** If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$.
- $G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.
- Graph 2-Coloring can be decided in polynomial time.
- $G$ is 2-colorable iff $G$ is bipartite!
- There is a linear time algorithm to check if $G$ is bipartite using BFS (we saw this earlier).

**Graph Coloring and Register Allocation**

**Register Allocation**
Assign variables to (at most) $k$ registers such that variables needed at the same time are not assigned to the same register.

**Interference Graph**
Vertices are variables, and there is an edge between two vertices, if the two variables are “live” at the same time.

**Observations**
- [Chaitin] Register allocation problem is equivalent to coloring the interference graph with $k$ colors
- Moreover, $\text{3-COLOR} \leq_P \kappa$-Register Allocation, for any $k \geq 3$

**Class Room Scheduling**

- Given $n$ classes and their meeting times, are $k$ rooms sufficient?
- Reduce to Graph $k$-Coloring problem
- Create graph $G$
  - a node $v_i$ for each class $i$
  - an edge between $v_i$ and $v_j$ if classes $i$ and $j$ conflict
- Exercise: $G$ is $k$-colorable iff $k$ rooms are sufficient.

**Frequency Assignments in Cellular Networks**

- Cellular telephone systems that use Frequency Division Multiple Access (FDMA) (example: GSM in Europe and Asia and AT&T in USA)
  - Breakup a frequency range $[a, b]$ into disjoint bands of frequencies $[a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]$
  - Each cell phone tower (simplifying) gets one band
  - Constraint: nearby towers cannot be assigned same band, otherwise signals will interfere
- **Problem:** given $k$ bands and some region with $n$ towers, is there a way to assign the bands to avoid interference?
- Can reduce to $k$-coloring by creating interference/conflict graph on towers.
3-Coloring is NP-Complete

- **3-Coloring** is in NP.
  - **Certificate**: for each node a color from \( \{1, 2, 3\} \).
  - **Certifier**: Check if for each edge \((u, v)\), the color of \(u\) is different from that of \(v\).

- **Hardness**: We will show 3-SAT \(\leq_P\) 3-Coloring.

Reduction Idea

- \(\varphi\): Given 3SAT formula (i.e., 3CNF formula).
- \(\varphi\): variables \(x_1, \ldots, x_n\) and clauses \(C_1, \ldots, C_m\).
- Create graph \(G_\varphi\) s.t. \(G_\varphi\) 3-colorable \(\iff\) \(\varphi\) satisfiable.
  - encode assignment \(x_1, \ldots, x_n\) in colors assigned nodes of \(G_\varphi\).
  - create triangle with node True, False, Base
  - for each variable \(x_i\) two nodes \(v_i\) and \(\bar{v}_i\) connected in a triangle with common Base
  - If graph is 3-colored, either \(v_i\) or \(\bar{v}_i\) gets the same color as True. Interpret this as a truth assignment to \(v_i\)
  - Need to add constraints to ensure clauses are satisfied (next phase)

Figure

Clause Satisfiability Gadget

- For each clause \(C_j = (a \lor b \lor c)\), create a small gadget graph
  - gadget graph connects to nodes corresponding to \(a, b, c\)
  - needs to implement OR
- OR-gadget-graph:
OR-Gadget Graph

Property: if \( a, b, c \) are colored False in a 3-coloring then output node of OR-gadget has to be colored False.

Property: if one of \( a, b, c \) is colored True then OR-gadget can be 3-colored such that output node of OR-gadget is colored True.

Reduction

- create triangle with nodes True, False, Base
- for each variable \( x_i \) two nodes \( v_i \) and \( \bar{v}_i \) connected in a triangle with common Base
- for each clause \( C_j = (a \lor b \lor c) \), add OR-gadget graph with input nodes \( a, b, c \) and connect output node of gadget to both False and Base

Claim

No legal 3-coloring of above graph (with coloring of nodes \( T, F, B \) fixed) in which \( a, b, c \) are colored False. If any of \( a, b, c \) are colored True then there is a legal 3-coloring of above graph.
Reduction Outline

Example

\( \varphi = (u \lor \neg v \lor w) \land (v \lor x \lor \neg y) \)

Correctness of Reduction

\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- if \( x_i \) is assigned True, color \( v_i \) True and \( \bar{v}_i \) False
- for each clause \( C_j = (a \lor b \lor c) \) at least one of \( a, b, c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- if \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment
- consider any clause \( C_j = (a \lor b \lor c) \). it cannot be that all \( a, b, c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!

Subset Sum

**Problem:** Subset Sum

**Instance:** \( S \) - set of positive integers, \( t \): - an integer number (Target)

**Question:** Is there a subset \( X \subseteq S \) such that \( \sum_{x \in X} x = t \)?

**Claim**

Subset Sum is NP-Complete.
Vec Subset Sum

We will prove following problem is NP-Complete...

**Problem: Vec Subset Sum**

**Instance:** $S$ - set of $n$ vectors of dimension $k$, each vector has non-negative numbers for its coordinates, and a target vector $\vec{t}$.

**Question:** Is there a subset $X \subseteq S$ such that $\sum_{x \in X} x = \vec{t}$?

Reduction from 3SAT.

---

Handling a single clause

Think about vectors as being lines in a table.

**First gadget**

Selecting between two lines.

<table>
<thead>
<tr>
<th>Target</th>
<th>??</th>
<th>??</th>
<th>01</th>
<th>???</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>??</td>
<td>??</td>
<td>01</td>
<td>??</td>
</tr>
<tr>
<td>$a_2$</td>
<td>??</td>
<td>??</td>
<td>01</td>
<td>??</td>
</tr>
</tbody>
</table>

Two rows for every variable $x$: selecting either $x = 0$ or $x = 1$.

---

3SAT to Vec Subset Sum

<table>
<thead>
<tr>
<th>numbers</th>
<th>$a \lor b \lor \overline{c}$</th>
<th>$a \lor b \lor c$</th>
<th>$b \lor c \lor \overline{d}$</th>
<th>$D \lor b \lor c \lor \overline{D}$</th>
<th>$C \lor a \lor b \lor \overline{C}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$c$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$\overline{c}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$c$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$D$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$C$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Target**

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
</table>

---

Handling a clause...

We will have a column for every clause...

<table>
<thead>
<tr>
<th>numbers</th>
<th>...</th>
<th>$C \equiv a \lor b \lor \overline{c}$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>...</td>
<td>01</td>
<td>...</td>
</tr>
<tr>
<td>$\overline{a}$</td>
<td>...</td>
<td>00</td>
<td>...</td>
</tr>
<tr>
<td>$b$</td>
<td>...</td>
<td>01</td>
<td>...</td>
</tr>
<tr>
<td>$b$</td>
<td>...</td>
<td>00</td>
<td>...</td>
</tr>
<tr>
<td>$c$</td>
<td>...</td>
<td>00</td>
<td>...</td>
</tr>
<tr>
<td>$\overline{c}$</td>
<td>...</td>
<td>01</td>
<td>...</td>
</tr>
<tr>
<td>$C$ fix-up 1</td>
<td>000</td>
<td>07</td>
<td>000</td>
</tr>
<tr>
<td>$C$ fix-up 2</td>
<td>000</td>
<td>08</td>
<td>000</td>
</tr>
<tr>
<td>$C$ fix-up 3</td>
<td>000</td>
<td>09</td>
<td>000</td>
</tr>
<tr>
<td>TARGET</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vec Subset Sum to Subset Sum

<table>
<thead>
<tr>
<th>numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>010000000001</td>
</tr>
<tr>
<td>010000000000</td>
</tr>
<tr>
<td>001000000001</td>
</tr>
<tr>
<td>000100001000</td>
</tr>
<tr>
<td>000001000100</td>
</tr>
<tr>
<td>000001000001</td>
</tr>
<tr>
<td>000000100000</td>
</tr>
<tr>
<td>000000101011</td>
</tr>
<tr>
<td>000000000007</td>
</tr>
<tr>
<td>000000000008</td>
</tr>
<tr>
<td>000000000009</td>
</tr>
<tr>
<td>000000000070</td>
</tr>
<tr>
<td>000000000080</td>
</tr>
<tr>
<td>000000000090</td>
</tr>
<tr>
<td>010101011010</td>
</tr>
</tbody>
</table>

Other NP-Complete Problems

- 3-Dimensional Matching
- Subset Sum

Read book.

Need to Know NP-Complete Problems

- 3SAT.
- Circuit-SAT.
- Independent Set.
- Vertex Cover.
- Clique.
- Set Cover.
- Hamiltonian Cycle (in Directed/Undirected Graphs).
- 3Coloring.
- 3-D Matching.
- Subset Sum.

Subset Sum and Knapsack

- **Subset Sum Problem:** Given $n$ integers $a_1, a_2, \ldots, a_n$ and a target $B$, is there a subset of $S$ of $\{a_1, \ldots, a_n\}$ such that the numbers in $S$ add up precisely to $B$?
- **Subset Sum is NP-Complete**— see book.
- **Knapsack:** Given $n$ items with item $i$ having size $s_i$ and profit $p_i$, a knapsack of capacity $B$, and a target profit $P$, is there a subset $S$ of items that can be packed in the knapsack and the profit of $S$ is at least $P$?
- Show Knapsack problem is NP-Complete via reduction from Subset Sum (exercise).
Subset Sum and Knapsack

- Subset Sum can be solved in $O(nB)$ time using dynamic programming (exercise).
- Implies that problem is hard only when numbers $a_1, a_2, \ldots, a_n$ are exponentially large compared to $n$. That is, each $a_i$ requires polynomial in $n$ bits.
- *Number problems* of the above type are said to be **weakly NPhard**.