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

Reduction construction

From 3SAT to Hamiltonian cycle in directed graph

1. Given 3SAT 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 $\mathcal{A}$

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

The Reduction: By figure

More details were given in the previous lecture

3SAT formula $\varphi$:

$\varphi = (x_1 \lor \neg x_2 \lor x_4) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3)$
The Reduction: Assignment $\Rightarrow$ Hamiltonian cycle

$3SAT$ formula $\varphi$:

$$\varphi = \left( x_1 \lor \neg x_2 \lor x_4 \right) \land \left( \neg x_1 \lor \neg x_2 \lor \neg x_3 \right)$$

A satisfying assignment:

$x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$

Conclude: If $\varphi$ has a satisfying assignment then there is a Hamiltonian cycle in $G_\varphi$.

Reduction: Hamiltonian cycle $\Rightarrow$ Assignment

No shenanigan: Hamiltonian cycle cannot leave a row in the middle

Conclude: Hamiltonian cycle must go through each row completely from left to right, or right to left. As such, can be interpreted as a valid assignment.

Satisfying assignment: $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$

Conclude: If $\varphi$ has a satisfying assignment then there is a Hamiltonian cycle in $G_\varphi$. 

Reduction: Hamiltonian cycle $\Rightarrow$ Assignment

Drawing example
Correctness Proof

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

Proof.
⇒ 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 □

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 \( i \)

Hamiltonian Cycle \( \Rightarrow \) Satisfying assignment (contd)

Thus, vertices visited immediately before and after \( C_i \) are connected by an edge
- We can remove \( c_j \) from cycle, and get Hamiltonian cycle in \( G - c_j \)
- Consider Hamiltonian cycle in \( G - \{c_1, \ldots, c_m\} \); it traverses each path in only one direction, which determines the truth assignment

Hamiltonian 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 $(a_{out}, b_{in})$

Hamiltonian cycle reduction

Undirected to directed case

Reduction: Wrapup

- The reduction is polynomial time (exercise)
- The reduction is correct (exercise)
Graph Coloring

**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 Coloring and Register Allocation

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

---

Graph 3-Coloring

**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?

---

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, $3$-COLOR $\leq_p k$-Register Allocation, for any $k \geq 3$
Class Room Scheduling

1. Given $n$ classes and their meeting times, are $k$ rooms sufficient?
2. Reduce to Graph $k$-Coloring problem
3. 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
4. Exercise: $G$ is $k$-colorable iff $k$ rooms are sufficient.

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.

Frequency Assignments in Cellular Networks

1. 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
2. Problem: given $k$ bands and some region with $n$ towers, is there a way to assign the bands to avoid interference?
3. Can reduce to $k$-coloring by creating interference/conflict graph on towers.

Reduction Idea

1. $\phi$: Given 3SAT formula (i.e., 3CNF formula).
2. $\phi$: variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$.
3. Create graph $G_{\phi}$ s.t. $G_{\phi}$ 3-colorable $\iff$ $\phi$ satisfiable.
   - encode assignment $x_1, \ldots, x_n$ in colors assigned nodes of $G_{\phi}$.
   - 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)
Clause Satisfiability Gadget

1. 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

2. 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 $\overline{v_i}$ 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
Reduction

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

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

3 coloring of the clause gadget

Correctness of Reduction

$\varphi$ is satisfiable implies $G_\varphi$ is 3-colorable

- if $x_i$ is assigned True, color $v_i$ True and $\overline{v}_i$ 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!
Graph generated in reduction...

... from 3SAT to 3COLOR

\[(a \lor b \lor c) \land (b \lor c \lor d) \land (\overline{a} \lor c \lor d) \land (a \lor b \lor d)\]

Subset Sum

\textbf{Subset Sum}

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

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

Claim

\textbf{Subset Sum is NP-Complete.}

Vec Subset Sum

We will prove following problem is \textbf{NP-Complete}...

\textbf{Vec Subset Sum}

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

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

Reduction from \textbf{3SAT}.

Vec Subset Sum

Handling a single clause

Think about vectors as being lines in a table.

\textbf{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 \(x = 1\).
Handling a clause...

We will have a column for every clause...

<table>
<thead>
<tr>
<th>numbers</th>
<th>...</th>
<th>$C \equiv a \lor b \lor c$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>...</td>
<td>01</td>
<td>...</td>
</tr>
<tr>
<td>$\bar{a}$</td>
<td>...</td>
<td>00</td>
<td>...</td>
</tr>
<tr>
<td>$b$</td>
<td>...</td>
<td>01</td>
<td>...</td>
</tr>
<tr>
<td>$c$</td>
<td>...</td>
<td>00</td>
<td>...</td>
</tr>
<tr>
<td>$\tau$</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>

3SAT to Vec Subset Sum

<table>
<thead>
<tr>
<th>numbers</th>
<th>$a \lor \bar{a}$</th>
<th>$b \lor \bar{b}$</th>
<th>$c \lor \bar{c}$</th>
<th>$d \lor \bar{d}$</th>
<th>$D \equiv B \lor \bar{B}$</th>
<th>$C \equiv a \lor b \lor c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>$c$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>$d$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>$D$ fix-up 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>07</td>
</tr>
<tr>
<td>$D$ fix-up 2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>09</td>
</tr>
<tr>
<td>$D$ fix-up 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>09</td>
</tr>
<tr>
<td>TARGET</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

Vec Subset Sum to Subset Sum

<table>
<thead>
<tr>
<th>numbers</th>
<th>010101011010</th>
</tr>
</thead>
</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 / Hitting Set.
- Hamiltonian Cycle (in Directed/Undirected Graphs).
- 3Coloring.
- 3-D Matching.
- Subset Sum / Partition.

Subset Sum and Knapsack

1. 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 $B$?

2. Subset Sum is NP-Complete—see book.

3. 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 $P$?

4. Show Knapsack problem is NP-Complete via reduction from Subset Sum (exercise).

Subset Sum and Knapsack

1. Subset Sum can be solved in $O(nB)$ time using dynamic programming (exercise).

2. 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.

3. Number problems of the above type are said to be weakly NPComplete.