Outline

• NP-complete problems: independent set, vertex cover, clique…
• NP-complete problems: Hamilton path and cycle, Traveling Salesperson Problem
• NP-complete problems: Subset Sum
• NP-complete problems: NAE-3-SAT, max cut

HAMPATH is NP-complete

\[ \phi = (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3) \land \ldots \]

• What can go wrong?
  – path has "intended form" unless return from clause gadget to different variable gadget

Case 1 (positive occurrence of v in clause):

- path must visit y
- must enter from x, z, or c
- must exit to z (x is taken)
- x, c are taken: can't happen

Case 2 (negative occurrence of v in clause):

- path must visit y
- must enter from x or z
- must exit to z (x is taken)
- x is taken: can't happen
Undirected Hamilton Path

• HAMPATH refers to a directed graph.
• Is it easier on an undirected graph?
• A language (decision problem):
  \( \text{UHAMPATH} = \{(G, s, t) : \text{undirected } G \text{ has a Hamilton path from } s \text{ to } t\} \)

UHAMPATH is NP-complete

**Theorem:** the following language is NP-complete:

\( \text{UHAMPATH} = \{(G, s, t) : \text{undirected graph } G \text{ has a Hamilton path from } s \text{ to } t\} \)

• Proof:
  – Part 1: UHAMPATH ∈ NP. Proof?
  – Part 2: UHAMPATH is NP-hard.
  • reduce from?

UHAMPATH is NP-complete

• We are reducing from the language:
  HAMPATH = \{(G, s, t) : directed graph G has a Hamilton path from s to t\}

to the language:
  UHAMPATH = \{(G, s, t) : undirected graph G has a Hamilton path from s to t\}

UHAMPATH is NP-complete

• The reduction:
  \begin{align*}
  s & \rightarrow s^\text{out} \\
  u_1 & \rightarrow u_1^\text{in} \\
  u_1 & \rightarrow u_1^\text{mid} \\
  u_1 & \rightarrow u_1^\text{out} \\
  v_1 & \rightarrow \ldots \\
  v_k & \rightarrow \ldots \\
  t & \rightarrow t^\text{in}
  \end{align*}

UHAMPATH is NP-complete

• Does the reduction run in poly-time?
  • YES maps to YES?
  – Hamilton path in G: \( s, u_1, u_2, u_3, \ldots, u_k, t \)
  – Hamilton path in G': \( s_{\text{out}}, u_1, u_2, v_1, v_2, v_3, v_4, v_5, \ldots, v_k-1, v_k^\text{mid}, v_k^\text{out} \)
  – \( u_i = (u_1)_{i^{\text{in}}} \) for some \( i \) (only edges to ins)
  – \( v_i = (u_1)_{i^{\text{mid}}} \) for some \( i \) (only way to enter mid)
  – \( v_i = (u_1)_{i^{\text{out}}} \) for some \( i \) (only way to exit mid)
  – \( u_i = (u_1)_{i^{\text{out}}} \) for some \( i \) (only edges to ins)
  – Hamilton path in G: \( s, u_1, u_2, u_3, \ldots, u_k, t \)
Undirected Hamilton Cycle

- Definition: given a undirected graph $G = (V, E)$, a Hamilton cycle in $G$ is a cycle in $G$ that touches every node exactly once.
- Is finding one easier than finding a Hamilton path?
- A language (decision problem): $UHAMCYCLE = \{G : G$ has a Hamilton cycle$\}$

UHAMCYCLE is NP-complete

**Theorem**: the following language is NP-complete:

$UHAMCYCLE = \{G : G$ has a Hamilton cycle$\}$

**Proof**:
- Part 1: $UHAMCYCLE \in$ NP. Proof?
- Part 2: $UHAMCYCLE$ is NP-hard.
  - reduce from?

Traveling Salesperson Problem

- Definition: given $n$ cities $v_1, v_2, \ldots, v_n$ and inter-city distances $d_{ij}$, a TSP tour in $G$ is a permutation $\pi$ of $\{1 \ldots n\}$. The tour's length is $\sum_{i=1}^{n} d_{\pi(i) \pi(i+1)}$ (where $n+1$ means 1).
- A search problem: given the $\{d_{ij}\}$, find the shortest TSP tour
- corresponding language (decision problem): $TSP = \{((d_{ij} : 1 \leq i < j \leq n), k) :$ these cities have a TSP tour of length $\leq k$\}$

TSP is NP-complete

**Theorem**: the following language is NP-complete:

$TSP = \{((d_{ij} : 1 \leq i < j \leq n), k) :$ these cities have a TSP tour of length $\leq k$\}$

**Proof**:
- Part 1: $TSP \in$ NP. Proof?
- Part 2: $TSP$ is NP-hard.
  - reduce from?
TSP is NP-complete

- The reduction:
  - given $G = (V, E)$ with $n$ nodes
  produce:
  - $n$ cities corresponding to the $n$ nodes
  - $d_{u,v} = 1$ if $(u, v) \in E$
  - $d_{u,v} = 2$ if $(u, v) \notin E$
  - set $k = n$

TSP is NP-complete

- YES maps to YES?
  - if $G$ has a Hamilton cycle, then visiting cities in that order gives TSP tour of length $n$

- NO maps to NO?
  - if TSP tour of length $\leq n$, it must have length exactly $n$.
  - all distances in tour are 1. Must be edges between every successive pair of cities in tour.

Hamilton Path

- Definition: given a directed graph $G = (V, E)$, a Hamilton path in $G$ is a directed path that touches every node exactly once.

- A language (decision problem):
  $\text{HAMPATH} = \{(G, s, t) : G$ has a Hamilton path from $s$ to $t\}$

HAMPATH is NP-complete

**Theorem:** the following language is NP-complete:

$\text{HAMPATH} = \{(G, s, t) : G$ has a Hamilton path from $s$ to $t\}$

**Proof:**
- Part 1: $\text{HAMPATH} \in \text{NP}$. Proof?
- Part 2: $\text{HAMPATH}$ is NP-hard.
  - reduce from?

HAMPATH is NP-complete

- We are reducing from the language:
  $3\text{SAT} = \{ \varphi : \varphi$ is a 3-CNF formula that has a satisfying assignment $\}$

  to the language:
  $\text{HAMPATH} = \{(G, s, t) : G$ has a Hamilton path from $s$ to $t\}$

HAMPATH is NP-complete

- We want to construct a graph from $\varphi$ with the following properties:
  - a satisfying assignment to $\varphi$ translates into a Hamilton Path from $s$ to $t$
  - a Hamilton Path from $s$ to $t$ can be translated into a satisfying assignment for $\varphi$

- We will build the graph up from pieces called gadgets that "simulate" the clauses and variables of $\varphi$. 
HAMPATH is NP-complete

- The variable gadget (one for each $x_i$):
  - $x_i$ true:
  - $x_i$ false:

$\phi = (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3) \land \cdots \land (\ldots)$

- How to ensure that all $k$ clauses are satisfied?
- need to add nodes
  - can be visited in path if the clause is satisfied
  - if visited in path, implies clause is satisfied by the assignment given by path through variable gadgets

- One clause gadget for each of $k$ clauses:
  - for clause 1
  - for clause 2

$\phi = (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3) \land \cdots \land (\ldots)$

- Clause gadget allows “detour” from “assignment path” for each true literal in clause

- $f(\phi)$ is this graph (edges to/from clause nodes not pictured)
- $f$ poly-time computable?
- # nodes = $O(km)$
HAMPATH is NP-complete

φ = (x₁ ∨ x₂) x₁ (x₂ ∨ x₃) x₁ (x₃ ∨ x₄) ...

- YES maps to YES?
  - first form path from satisfying assign.
  - pick true literal in each clause and add detour

- NO maps to NO?
  - try to translate path into satisfying assignment.
  - if path has "intended" form, OK.

What can go wrong?
- path has "intended form" unless return from clause gadget to different variable gadget

Case 1 (positive occurrence of v in clause):
  - path must visit y
  - must enter from x, z, or c
  - must exit to z (x is taken)
  - x, c are taken. can't happen

Case 2 (negative occurrence of v in clause):
  - path must visit y
  - must enter from x or z
  - must exit to z (x is taken)
  - x is taken. can't happen

HAMPATH refers to a directed graph.
Is it easier on an undirected graph?

A language (decision problem):
UHAMPATH = {(G, s, t) : undirected G has a Hamilton path from s to t}
Theorem: the following language is NP-complete:

\[ \text{UHAMPATH} = \{(G, s, t) : \text{undirected graph } G \text{ has a Hamilton path from } s \text{ to } t\} \]

Proof:
- Part 1: UHAMPATH ∈ NP. Proof?
- Part 2: UHAMPATH is NP-hard.

reduce from?  

We are reducing from the language:

\[ \text{HAMPATH} = \{(G, s, t) : \text{directed graph } G \text{ has a Hamilton path from } s \text{ to } t\} \]

to the language:

\[ \text{UHAMPATH} = \{(G, s, t) : \text{undirected graph } G \text{ has a Hamilton path from } s \text{ to } t\} \]

The reduction:

- replace each node with three (except s, t)
  - \((u_{in}, u_{mid})\)
  - \((u_{mid}, u_{out})\)
  - \((u_{out}, v_{in})\) iff \(G\) has \((u,v)\)

Does the reduction run in poly-time?

YES maps to YES?
- Hamilton path in \(G\): \(s, u_{i1}, u_{i2}, ..., u_{ik}, t\)
- Hamilton path in \(G'\):
  \(s_{out}, (u_{i1})_{in}, (u_{i1})_{mid}, (u_{i1})_{out}, (u_{i2})_{in}, (u_{i2})_{mid}, (u_{i2})_{out}, ..., (u_{ik})_{in}, (u_{ik})_{mid}, (u_{ik})_{out}, t\in\)

NO maps to NO?
- Hamilton path in \(G\):
  \(s_{out}, v_{1}, v_{2}, v_{3}, v_{4}, v_{5}, v_{6}, ..., v_{k-2}, v_{k-1}, v_{k}, t\)
- \(v_{1} = (u_{i1})_{in}\) for some \(i_{1}\) (only edges to ins)
- \(v_{2} = (u_{i1})_{mid}\) for some \(i_{1}\) (only way to enter mid)
- \(v_{3} = (u_{i1})_{out}\) for some \(i_{1}\) (only way to exit mid)
- \(v_{4} = (u_{i2})_{in}\) for some \(i_{2}\) (only edges to ins)
- ...
- Hamilton path in \(G\): \(s, u_{i1}, u_{i2}, u_{i3}, ..., u_{ik}, t\)

Undirected Hamilton Cycle

Definition: given a undirected graph \(G = (V, E)\), a Hamilton cycle in \(G\) is a cycle in \(G\) that touches every node exactly once.
- Is finding one easier than finding a Hamilton path?
- A language (decision problem):
  \[ \text{UHAMCYCLE} = \{(G : G \text{ has a Hamilton cycle})\} \]
UHAMCYCLE is NP-complete

**Theorem:** the following language is NP-complete:

\[ \text{UHAMCYCLE} = \{ G : \text{G has a Hamilton cycle} \} \]

- **Proof:**
  - Part 1: UHAMCYCLE ∈ NP. Proof?
  - Part 2: UHAMCYCLE is NP-hard.
    - reduce from?

Traveling Salesperson Problem

- **Definition:** given n cities \( v_1, v_2, ..., v_n \) and inter-city distances \( d_{ij} \), a TSP tour in G is a permutation \( \pi \) of \( \{1, 2, ..., n\} \). The tour’s length is \( \sum_{i=1}^{n} d_{\pi(i)\pi(i+1)} \) (where \( n+1 \) means 1).
- **A search problem:**
  - given the \( \{d_{ij}\} \), find the shortest TSP tour
- **corresponding language (decision problem):**
  - \( \text{TSP} = \{(d_{ij} : 1 \leq i < j \leq n), k \} : \text{these cities have a TSP tour of length} \leq k \}

TSP is NP-complete

- **Theorem:** the following language is NP-complete:
  - \( \text{TSP} = \{(d_{ij} : 1 \leq i < j \leq n), k \} : \text{these cities have a TSP tour of length} \leq k \}
- **Proof:**
  - Part 1: TSP ∈ NP. Proof?
  - Part 2: TSP is NP-hard.
    - reduce from?
TSP is NP-complete

- YES maps to YES?
  - if G has a Hamilton cycle, then visiting cities in that order gives TSP tour of length n
- NO maps to NO?
  - if TSP tour of length ≤ n, it must have length exactly n.
  - all distances in tour are 1. Must be edges between every successive pair of cities in tour.

Subset Sum

- A language (decision problem):
  \( \text{SUBSET-SUM} = (\{a_1, a_2, a_3, \ldots, a_k\}, B) \)
  - there is a subset of \( S \) that sums to \( B \)

  - example:
    - \( S = \{1, 7, 28, 3, 2, 5, 9, 32, 41, 11, 8\} \)
    - \( B = 30 \)
    - \( 30 = 7 + 3 + 9 + 11 \). yes.

\[ \text{SUBSET-SUM is NP-complete} \]

**Theorem:** the following language is NP-complete:

\( \text{SUBSET-SUM} = (\{a_1, a_2, a_3, \ldots, a_k\}, B) \)
  - there is a subset of \( S \) that sums to \( B \)

**Proof:**

- Part 1: SUBSET-SUM ∈ NP. Proof?
- Part 2: SUBSET-SUM is NP-hard. reduce from?
  - our reduction had better produce super-polynomially large \( B \) (unless we want to prove P=NP)

\[ \text{SUBSET-SUM is NP-complete} \]

- \( \phi = (x_1 \lor x_2 \lor \neg x_3) \land \neg (x_4 \lor x_5 \lor x_6) \land \ldots \land (\ldots) \)

  - Need integers to play the role of truth assignments
  - For each variable \( x_i \) include two integers in our set \( S \):
    - \( x_i^{\text{true}} \) and \( x_i^{\text{false}} \)
  - set \( B \) so that exactly one must be in sum

\[ \text{SUBSET-SUM is NP-complete} \]
SUBSET-SUM is NP-complete

\[ \begin{align*}
x_1^{\text{true}} &= 1000\ldots0 \\
x_1^{\text{false}} &= 1000\ldots0 \\
x_2^{\text{true}} &= 0100\ldots0 \\
x_2^{\text{false}} &= 0100\ldots0 \\
\vdots \\
x_n^{\text{true}} &= 0000\ldots1 \\
x_n^{\text{false}} &= 0000\ldots1 \\
B &= 1111\ldots1
\end{align*} \]

- every choice of one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair sums to \(B\)

- every subset that sums to \(B\) must choose one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

\[ \phi = (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3) \land \ldots \land (\ldots) \]

- \(\phi\) has one true literal from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

- Need to force subset to "choose" at least one true literal from each clause

- Idea:
  - add more digits
  - one digit for each clause
  - set \(B\) to force each clause to be satisfied.

\[ \begin{align*}
x_1^{\text{true}} &= 1000\ldots0 \\
x_1^{\text{false}} &= 1000\ldots0 \\
x_2^{\text{true}} &= 0000\ldots1 \\
x_2^{\text{false}} &= 0000\ldots1 \\
\vdots \\
B &= 1111\ldots1
\end{align*} \]

- every choice of one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair sums to \(B\)

- every subset that sums to \(B\) must choose one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

- bound \(B\) has 3 in positions 1\(\ldots\)k and 1 in positions k+1\(\ldots\)k+m

\[ \begin{align*}
x_1^{\text{true}} &= 1000\ldots0 \\
x_1^{\text{false}} &= 1000\ldots0 \\
x_2^{\text{true}} &= 0100\ldots0 \\
x_2^{\text{false}} &= 0100\ldots0 \\
\vdots \\
B &= 1111\ldots1
\end{align*} \]

- \(\phi\) has one true literal from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

- Need to force subset to "choose" at least one true literal from each clause

- Idea:
  - add more digits
  - one digit for each clause
  - set \(B\) to force each clause to be satisfied.

\[ \begin{align*}
x_1^{\text{true}} &= 1000\ldots0 \\
x_1^{\text{false}} &= 1000\ldots0 \\
x_2^{\text{true}} &= 0000\ldots1 \\
x_2^{\text{false}} &= 0000\ldots1 \\
\vdots \\
B &= 1111\ldots1
\end{align*} \]

- every choice of one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair sums to \(B\)

- every subset that sums to \(B\) must choose one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

- bound \(B\) has 3 in positions 1\(\ldots\)k and 1 in positions k+1\(\ldots\)k+m

\[ \begin{align*}
x_1^{\text{true}} &= 1000\ldots0 \\
x_1^{\text{false}} &= 1000\ldots0 \\
x_2^{\text{true}} &= 0000\ldots1 \\
x_2^{\text{false}} &= 0000\ldots1 \\
\vdots \\
B &= 1111\ldots1
\end{align*} \]

- every choice of one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair sums to \(B\)

- every subset that sums to \(B\) must choose one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

- bound \(B\) has 3 in positions 1\(\ldots\)k and 1 in positions k+1\(\ldots\)k+m

\[ \begin{align*}
x_1^{\text{true}} &= 1000\ldots0 \\
x_1^{\text{false}} &= 1000\ldots0 \\
x_2^{\text{true}} &= 0000\ldots1 \\
x_2^{\text{false}} &= 0000\ldots1 \\
\vdots \\
B &= 1111\ldots1
\end{align*} \]

- every choice of one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair sums to \(B\)

- every subset that sums to \(B\) must choose one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

- bound \(B\) has 3 in positions 1\(\ldots\)k and 1 in positions k+1\(\ldots\)k+m

\[ \begin{align*}
x_1^{\text{true}} &= 1000\ldots0 \\
x_1^{\text{false}} &= 1000\ldots0 \\
x_2^{\text{true}} &= 0000\ldots1 \\
x_2^{\text{false}} &= 0000\ldots1 \\
\vdots \\
B &= 1111\ldots1
\end{align*} \]

- every choice of one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair sums to \(B\)

- every subset that sums to \(B\) must choose one from each \((x_i^{\text{true}}, x_i^{\text{false}})\) pair

- bound \(B\) has 3 in positions 1\(\ldots\)k and 1 in positions k+1\(\ldots\)k+m