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

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:

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

- Proof:
  - Part 1: UHAMPATH \( \in \) NP. Proof?
  - Part 2: UHAMPATH is NP-hard.
    - reduce from?

UHAMPATH is NP-complete

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

  to the language:

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

• Does the reduction run in poly-time?

• YES maps to YES?
  – Hamilton path in G: s, u₁, u₂, u₃, ..., uₖ, t
  – Hamilton path in G':
    s₁out, (u₁)₁in, (u₁)₁mid, (u₂)₂in, (u₂)₂mid, (u₂)₂out, ...
    (uₖ)ₖin, (uₖ)ₖmid, (uₖ)ₖout, t₁out

• NO maps to NO?
  – Hamilton path in G':
    s₁out, v₁, v₂, v₃, v₄, v₅, v₆, ..., vₖ, t₁out
  – H. path in G:
    s₁out, u₁₁, u₁₂, u₁₃, ..., u₁ₖ, t₁out

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

Traveling Salesperson Problem

• Definition: given n cities v₁, v₂, ..., vₙ and inter-city distances dᵢⱼ, a TSP tour in G is a permutation 𝜋 of {1...n}. The tour’s length is \( \Sigma_i = 1...n d_{\pi(i)\pi(i+1)} \) (where n+1 means 1).
• A search problem:
given the \( \{dᵢⱼ\} \), find the shortest TSP tour
• corresponding language (decision problem):
  TSP = \( \{\{dᵢⱼ : 1 ≤ i < j ≤ n\}, k\} : \) these cities have a TSP tour of length ≤ 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 \(\in\) NP. Proof?
- Part 2: TSP is NP-hard.
  - reduce from?

TSP is NP-complete

- We are reducing from the language:
  \[ \text{UHAMCYCLE} = \{ G : G \text{ has a Hamilton cycle} \} \]

  to the language:

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

TSP is NP-complete

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

Subset Sum

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

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

Subset Sum

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

- Is this problem NP-complete? in P?

- Problem set: in \( \text{TIME}(B \cdot \text{poly}(k)) \)
Theorem: the following language is NP-complete:

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

Proof:
- Part 1: \( \text{SUBSET-SUM} \in \text{NP} \)
- Part 2: \( \text{SUBSET-SUM} \) is NP-hard.

We are reducing from the language:

\( 3\text{SAT} = \{\phi : \phi \text{ is a 3-CNF formula that has a satisfying assignment}\} \)

To the language:

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

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

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

\( x_1^{\text{TRUE}} = 10000 \ldots 0 \)
\( x_1^{\text{FALSE}} = 10000 \ldots 0 \)
\( x_2^{\text{TRUE}} = 01000 \ldots 0 \)
\( x_2^{\text{FALSE}} = 01000 \ldots 0 \)
\( \ldots \)
\( x_m^{\text{TRUE}} = 00000 \ldots 1 \)
\( x_m^{\text{FALSE}} = 00000 \ldots 1 \)
\( B = 11111 \ldots 1 \)

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

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.

\( x_1^{\text{TRUE}} = 10000 \ldots 0 \)
\( x_1^{\text{FALSE}} = 10000 \ldots 0 \)
\( x_2^{\text{TRUE}} = 01000 \ldots 0 \)
\( x_2^{\text{FALSE}} = 01000 \ldots 0 \)
\( x_3^{\text{TRUE}} = 00100 \ldots 0 \)
\( x_3^{\text{FALSE}} = 00100 \ldots 0 \)
\( \ldots \)
\( B = 11111 \ldots 1 \ldots ? \ldots ? \ldots ? \)
SUBSET-SUM is NP-complete

- \( B = 1 1 1 1 \ldots 1 ? ? ? ? \)
- if clause \( i \) is satisfied sum might be 1, 2, or 3 in corresponding column.
- want \( ? \) to "mean" \( \geq 1 \)
- solution: set \( ? = 3 \)
- add two "filler" elements for each clause \( i \):
  - \( \text{FILL1}_i = 0 0 0 0 \ldots 0 0 1 0 \ldots 0 \)
  - \( \text{FILL2}_i = 0 0 0 0 \ldots 0 0 1 0 \ldots 0 \)
- column for clause \( i \)

SUBSET-SUM is NP-complete

- Reduction: \( m \) variables, \( k \) clauses
  - for each variable \( x_i \):
    - \( x_i^{\text{TRUE}} \) has ones in positions \( k + i \) and \( \{j : \text{clause } j \text{ includes literal } x_i \} \)
    - \( x_i^{\text{FALSE}} \) has ones in positions \( k + i \) and \( \{j : \text{clause } j \text{ includes literal } \neg x_i \} \)
  - for each clause \( i \):
    - \( \text{FILL1}_i \) and \( \text{FILL2}_i \) have one in position \( i \)
  - bound \( B \) has 3 in positions 1…\( k \) and 1 in positions \( k+1 \ldots k+m \)

SUBSET-SUM is NP-complete

- Reduction computable in poly-time?
- YES maps to YES?
  - choose one from each \((x_i^{\text{TRUE}}, x_i^{\text{FALSE}})\) pair corresponding to a satisfying assignment
  - choose 0, 1, or 2 of filler elements for each clause \( i \) depending on whether it has 3, 2, or 1 true literals
  - first \( m \) digits add to 1; last \( k \) digits add to 3

SUBSET-SUM is NP-complete

- NO maps to NO?
  - at most 5 ones in each column, so no carries to worry about
  - first \( m \) digits of \( B \) force subset to choose exactly one from each \((x_i^{\text{TRUE}}, x_i^{\text{FALSE}})\) pair
  - last \( k \) digits of \( B \) require at least one true literal per clause, since can only sum to 2 using filler elements
  - resulting assignment must satisfy \( \varphi \)

MAX CUT

- Given graph \( G = (V, E) \)
  - a cut is a subset \( S \subseteq V \)
  - an edge \((x, y)\) crosses the cut if \( x \in S \) and \( y \in V - S \) or \( x \in V - S \) and \( y \in S \)
  - search problem:
    find cut maximizing number of edges crossing the cut
MAX CUT

Theorem: the following language is NP-complete:

\[ \text{MAX CUT} = \{(G = (V, E), k) : \text{there is a cut } S \subseteq V \text{ with at least } k \text{ edges crossing it}\} \]

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