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 \(\in\) NP. Proof?
  - Part 2: UHAMPATH is NP-hard.
    - reduce from?
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':
    s_out, (u₁)_in, (u₁)_mid, (u₂)_in, (u₂)_mid, (u₃)_in, ..., (uₖ)_in, (uₖ)_mid, (tₙ)

UHAMPATH is NP-complete

• NO maps to NO?
  – Hamilton path in G': s_out, v₁, v₂, v₃, v₄, v₅, ..., vₖ₋₁, vₖ, t_in
  – v₁ = (u₁)_in for some i₁ (only edges to ins)
  – v₂ = (u₁)_mid for some i₁ (only way to enter mid)
  – v₃ = (u₁)_out for some i₁ (only way to exit mid)
  – v₄ = (u₂)_in for some i₂ (only edges to ins)
  – ...
  – Hamilton path in G: s, uᵢ₁, uᵢ₂, uᵢ₃, ..., uᵢₖ, t

Undirected Hamilton Path

• 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 \}

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 \( v_1, v_2, \ldots, v_n \) 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 = \( \{\{(d_{ij} : 1 \leq i < j \leq n), k) : \text{these cities have a TSP tour of length } \leq k\} \)

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

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 \( \text{TIME}(B \cdot \text{poly}(k)) \)

**SUBSET-SUM is NP-complete**

**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.
  - reduce from?

**Our reduction had better produce super-polynomially large \( B \) (unless we want to prove \( P=NP \)).**

**SUBSET-SUM is NP-complete**

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

**SUBSET-SUM is NP-complete**

- every choice of one from each \( (x_i^{\text{TRUE}}, x_i^{\text{FALSE}}) \) pair must sum to \( B \)
  - every subset that sums to \( B \) must choose one from each \( (x_i^{\text{TRUE}}, x_i^{\text{FALSE}}) \) pair

**SUBSET-SUM is NP-complete**

- \( \phi = (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3) \land \ldots \land (\ldots) \)
- 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.
SUBSET-SUM is NP-complete

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

\[
\begin{array}{c}
 x_1^{\text{TRUE}} = 1000 \ldots 1 \\
 x_1^{\text{FALSE}} = 1000 \ldots 0 \\
 x_2^{\text{TRUE}} = 0100 \ldots 0 \\
 x_2^{\text{FALSE}} = 0100 \ldots 1 \\
 x_3^{\text{TRUE}} = 0010 \ldots 0 \\
 x_3^{\text{FALSE}} = 0010 \ldots 1 \\
 \vdots \\
 B = 1111 \ldots 1 ? ? ? \\
\end{array}
\]

– $B = 1111 \ldots 1 \ ? \ ? \ ?$ 
– 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 $i$: 
  – $\text{FILL}_1 = 0000 \ldots 0 \ 0 \ldots 0 \ 1 \ldots 0$ 
  – $\text{FILL}_2 = 0000 \ldots 0 \ 0 \ldots 0 \ 1 \ldots 0$ 

– bound $B$ has 3 in positions 1…k and 1 in positions k+1…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 $\phi$