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

• We are reducing from the language:
UHAMCYCLE = \{G : G has a Hamilton cycle\}
to the language:
TSP = {((d_{ij}: 1 \leq i < j \leq n), k) : 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 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 \( P=NP \))

- 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

\( \varphi = (x_1 \lor x_2 \lor \neg x_3)(\neg x_1 \lor x_4 \lor x_5) \land \ldots \land (...) \)

\( x_i^{\text{TRUE}} = 1 \ 0 \ 0 \ 0 \ \ldots \ 0 \)
\( x_i^{\text{FALSE}} = 1 \ 0 \ 0 \ 0 \ \ldots \ 0 \)
\( x_2^{\text{TRUE}} = 0 \ 1 \ 0 \ 0 \ \ldots \ 0 \)
\( x_2^{\text{FALSE}} = 0 \ 1 \ 0 \ 0 \ \ldots \ 0 \)
\( \ldots \)
\( x_m^{\text{TRUE}} = 0 \ 0 \ 0 \ \ldots \ 1 \)
\( x_m^{\text{FALSE}} = 0 \ 0 \ 0 \ \ldots \ 1 \)
\( B = 1 \ 1 \ 1 \ 1 \ \ldots \ 1 \)

- 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
SUBSET-SUM is NP-complete

- \( \varphi = (x_1 \lor x_2 \lor \neg x_3)(\neg x_1 \lor x_4 \lor x_5) \land \ldots \land (\ldots) \)
- 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.

\[
\begin{align*}
x_1^\text{TRUE} &= 1 \ 0 \ 0 \ 0 \ldots \ 0 \ 1 \\
x_1^\text{FALSE} &= 1 \ 0 \ 0 \ 0 \ldots \ 0 \\
x_2^\text{TRUE} &= 0 \ 1 \ 0 \ 0 \ldots \ 0 \\
x_2^\text{FALSE} &= 0 \ 1 \ 0 \ 0 \ldots \ 0 \\
x_3^\text{TRUE} &= 0 \ 0 \ 1 \ 0 \ldots \ 0 \\
x_3^\text{FALSE} &= 0 \ 0 \ 1 \ 0 \ldots \ 0 \\
\ldots & \ \\
B &= 1 \ 1 \ 1 \ 1 \ldots \ 1 \ ? \ ? \ ? \ ? \ ? \\
\end{align*}
\]

Σ-clause: clause 1
clause 2
clause 3
clause k

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 \( k+1 \ldots k+m \)

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

---

Not-All-Equal 3SAT

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

Theorem: the following language is NP-complete:

\[\text{NAE3SAT} = \{ \phi : \phi \text{ is a 3-CNF formula for which there exists a truth assignment in which every clause has at least one true literal and at least one false literal} \}\]

- Proof:
  - Part 1: NAE3SAT \( \in \) NP. Proof?
  - Part 2: NAE3SAT is NP-hard. Reduce from?
NAE3SAT is NP-complete

- Recall reduction to 3SAT
  - variables \( x_1, x_2, \ldots, x_n \), gates \( g_1, g_2, \ldots, g_m \)
  - produce clauses:
    \[
    \begin{align*}
    &\neg z_1 \land x_1 \land v \\
    &\neg z_2 \land v \land w \\
    &\neg g_1 \land z_1 \land v \\
    &\neg g_2 \land z_2 \land v \\
    &\neg g_3 \land v \land w
    \end{align*}
    \]
    output gate \( g_0 \):
    \[
    \begin{align*}
    &\neg z_1 \land v \land z_2 \\
    &\neg z_2 \land v \land z_1 \\
    &\neg g_1 \land z_1 \land v \\
    &\neg g_2 \land z_2 \land v \\
    &\neg g_3 \land v \land w
    \end{align*}
    \]

NAE3SAT is NP-complete

- NO maps to NO
  - given NAE assignment \( A \)
  - complement \( A' \) is a NAE assignment
  - \( A \) or \( A' \) has \( w = \text{FALSE} \)
  - must have TRUE BLUE variable in every clause
  - we know this implies \( C \) satisfiable

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

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

• We are reducing from the language:

\[ \text{NAE3SAT} = \{ \phi : \phi \text{ is a 3-CNF formula for which there exists a truth assignment in which every clause has at least 1 true literal and at least 1 false literal} \} \]

to the language:

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

31

MAX CUT is NP-complete

• The reduction:
  – given instance of NAE3SAT (n nodes, m clauses):
    \[ (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_5) \land \ldots \land (\neg x_2 \lor x_3 \lor x_3) \]
  – produce graph \( G = (V, E) \) with node for each literal
  • triangle for each 3-clause
  • parallel edges for each 2-clause

32

• YES maps to YES
  – take cut to be TRUE literals in a NAE truth assignment
  – contribution from clause gadgets: \( 2m \)
  – contribution from \( (x_i, \neg x_i) \) parallel edges: \( 3m \)
  – set \( k = 5m \)

33

• NO maps to NO
  – Claim: if cut has \( x_i, \neg x_i \) on same side, then can move one to opposite side without decreasing # edges crossing cut
  • Proof:
    \[ a + n \leq 2n \]

34

35

36