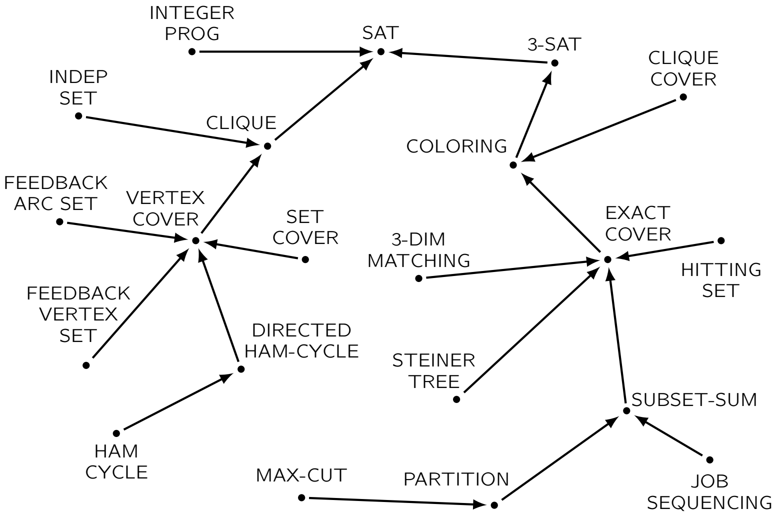
Complexity II

Martin Hoefer

(based on material by Walter Unger)

- More problems may not be solved in polynomial time.

Landscape with Karp's 20 reductions

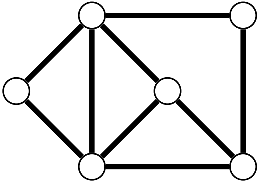


CLIQUE (1): Definition

Problem: CLIQUE

Instance: An undirected graph $G = (V, E)$; an integer k

Question: Does G contain a clique of size (at least) k?



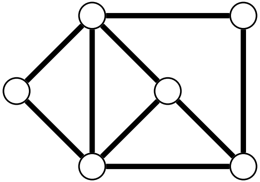
$k = 4$

CLIQUE (1): Definition

Problem: CLIQUE

Instance: An undirected graph $G = (V, E)$; an integer k

Question: Does G contain a clique of size (at least) k?



$k = 4$

Theorem

CLIQUE is NP-complete.

CLIQUE (2): According to our Cooking Recipe

1. We already know that CLIQUE is in NP.

2. We pick the NP-complete language $L^* = SAT$
   and we will show that SAT $\leq_p$ CLIQUE.

3. **(Reduction):**
   We construct a function f that translates a CNF-formula φ into a graph $G = (V, E)$ and a number $k \in \mathbb{N}$ such that:

   $$\varphi \text{ is satisfiable} \quad \Leftrightarrow \quad G \text{ contains } k\text{-clique}$$

(The other items of the cooking recipe will be settled later.)

CLIQUE (3): Description of Function f

- Let $c_1, \ldots, c_m$ be the clauses in formula φ.
  Let k_i denote the number of literals in clause c_i.
  Let $\ell_{i,1}, \ldots, \ell_{i,k_i}$ be the literals in clause c_i.

CLIQUE (3): Description of Function f

- Let $c_1, \ldots, c_m$ be the clauses in formula φ.
  Let k_i denote the number of literals in clause c_i.
  Let $\ell_{i,1}, \ldots, \ell_{i,k_i}$ be the literals in clause c_i.

- For every literal in every clause we create a corresponding vertex:
  $V = \{\ell_{i,j} \mid 1 \leq i \leq m,\ 1 \leq j \leq k_i\}$
- Two vertices are connected by an edge,
  if they come from distinct clauses and
  if there literals are not negations of each other.
- We set $k = m$.

CLIQUE (3): Description of Function f

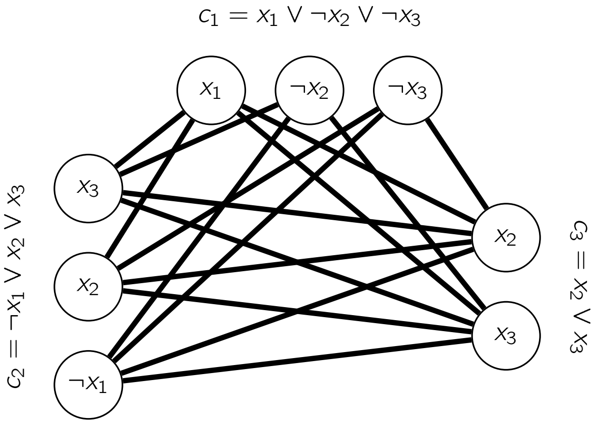
- Let $c_1, \ldots, c_m$ be the clauses in formula φ.
  Let k_i denote the number of literals in clause c_i.
  Let $\ell_{i,1}, \ldots, \ell_{i,k_i}$ be the literals in clause c_i.

- For every literal in every clause we create a corresponding vertex:
  $V = \{\ell_{i,j} \mid 1 \leq i \leq m,\ 1 \leq j \leq k_i\}$
- Two vertices are connected by an edge,
  if they come from distinct clauses and
  if there literals are not negations of each other.
- We set $k = m$.

4. (Polynomial time):
The function f can be computed in polynomial time.

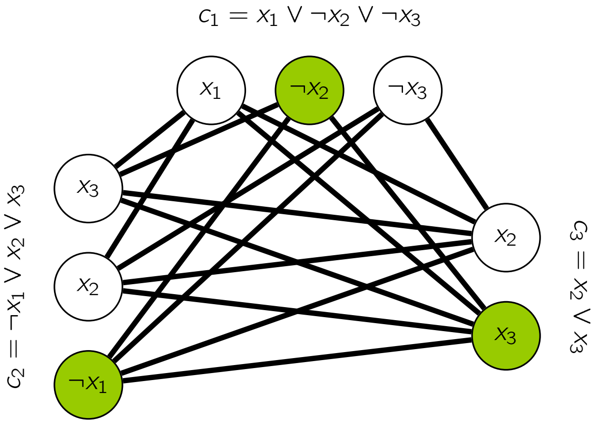
CLIQUE (4): Example

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



CLIQUE (4): Example

$\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3)$



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

CLIQUE (5a): Correctness

Lemma A: Formula φ satisfiable $\Rightarrow$ G has m-clique

- Consider a satisfying truth assignment of φ
- Form a set U by picking one true literal from every clause
- Claim: U forms m-clique

CLIQUE (5a): Correctness

Lemma A: Formula φ satisfiable $\Rightarrow$ G has m-clique

- Consider a satisfying truth assignment of φ
- Form a set U by picking one true literal from every clause
- Claim: U forms m-clique

Argument:

- By definition $|U| = m$
- Let ℓ and ℓ' be two distinct literals from U

CLIQUE (5a): Correctness

Lemma A: Formula φ satisfiable $\Rightarrow$ G has m-clique

- Consider a satisfying truth assignment of φ
- Form a set U by picking one true literal from every clause
- Claim: U forms m-clique

Argument:

- By definition $|U| = m$
- Let ℓ and ℓ' be two distinct literals from U

- By construction ℓ and ℓ' come from distinct clauses.
- As ℓ and ℓ' are true, they are not negations of each other.
- Hence there is an edge between ℓ and ℓ'.

CLIQUE (5b): Correctness

Lemma B: G has m-clique $\Rightarrow$ formula φ satisfiable

- Consider m-clique U in G
- Then the literals in U belong to m different clauses

CLIQUE (5b): Correctness

Lemma B: G has m-clique $\Rightarrow$ formula φ satisfiable

- Consider m-clique U in G
- Then the literals in U belong to m different clauses

- U contains one literal from every clause
- No literal occurs in positive and in negative form
- Hence: All these literals can be satisfied simultaneously
- Hence: φ is satisfiable

CLIQUE (5b): Correctness

Lemma B: G has m-clique $\Rightarrow$ formula φ satisfiable

- Consider m-clique U in G
- Then the literals in U belong to m different clauses

- U contains one literal from every clause
- No literal occurs in positive and in negative form
- Hence: All these literals can be satisfied simultaneously
- Hence: φ is satisfiable

5. (Correctness): $x \in L^* \Leftrightarrow f(x) \in L$

$\varphi \in \text{SAT} \Leftrightarrow f(\varphi) = \langle G; m \rangle \in \text{CLIQUE}$

Independent Set

Problem: Independent Set

Instance: An undirected graph $G' = (V', E')$; an integer k'

Question: Does G' contain an independent set with (at least) k' vertices?

- An independent set $S \subseteq V$ is a set that does not span any edges

Theorem

INDEPENDENT SET is NP-complete.

Independent Set

Problem: Independent Set

Instance: An undirected graph $G' = (V', E')$; an integer k'

Question: Does G' contain an independent set with (at least) k' vertices?

- An independent set $S \subseteq V$ is a set that does not span any edges

Theorem

INDEPENDENT SET is NP-complete.

Proof:

- We show CLIQUE $\leq_p$ INDEPENDENT-SET
- Set $V' = V$ and $E' = V \times V - E$ and $k' = k$

Vertex Cover (1)

Problem: Vertex Cover (VC)

Instance: An undirected graph $G'' = (V'', E'')$; an integer k''

Question: Does G'' contain a vertex cover with (at most) k'' vertices?

- Vertex cover $S \subseteq V$ touches all edges

Theorem

VC is NP-complete.

Vertex Cover (1)

Problem: Vertex Cover (VC)

Instance: An undirected graph $G'' = (V'', E'')$; an integer k''

Question: Does G'' contain a vertex cover with (at most) k'' vertices?

- Vertex cover $S \subseteq V$ touches all edges

Theorem

VC is NP-complete.

Proof:

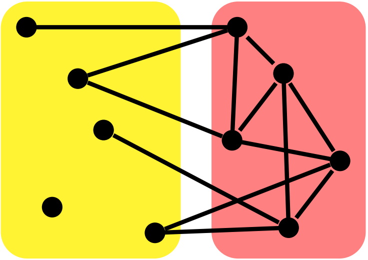
- We show INDEPENDENT-SET $\leq_p$ VC
- Set $V'' = V'$ and $E'' = E'$ and $k'' = |V'| - k'$

Vertex Cover (2)

Observation

In an undirected graph $G = (V, E)$ all subsets $S \subseteq V$ satisfy:

- S is independent set $\Leftrightarrow$ $V - S$ is vertex cover
- S is vertex cover $\Leftrightarrow$ $V - S$ is independent set



D-Ham-Cycle (1): Definition

Problem: Directed Hamilton cycle (D-Ham-Cycle)

Instance: A directed graph $G = (V, A)$

Question: Does G contain a directed Hamilton cycle?

Theorem

D-Ham-Cycle is NP-complete.

D-Ham-Cycle (2): According to our Cooking Recipe

1. D-Ham-Cycle is in NP.

2. We pick the NP-complete language $L^* = SAT$
and we will show that SAT $\leq_p$ D-Ham-Cycle.

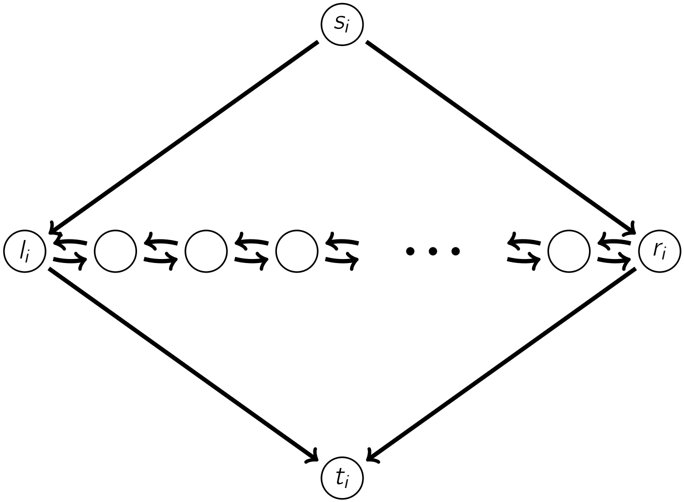
3. **(Reduction):**
We construct a function f that translates a CNF-formula φ into a directed graph $G = (V, A)$ such that:

$$\varphi \text{ is satisfiable} \quad \Leftrightarrow \quad G \text{ has directed Hamilton cycle}$$

The CNF-formula φ consists of clauses $c_1, \ldots, c_m$ with Boolean variables $x_1, \ldots, x_n$.

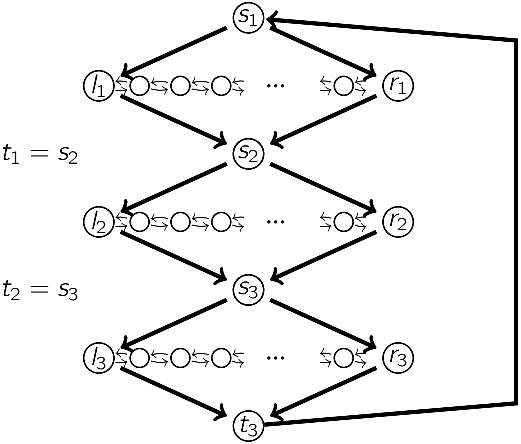
D-Ham-Cycle (3a): Reduction / Diamond-Gadgets

For every variable x_i, we create a corresponding **diamond-gadget** G_i:



D-Ham-Cycle (3b): Reduction / Diamond-Gadgets

These n diamond-gadgets are connected to each other by identifying vertex t_i and vertex s_{i+1} (for $1 \leq i \leq n-1$) as well as t_n and s_1 with each other:



D-Ham-Cycle (3c): Reduction / Diamond-Gadgets

In the resulting graph: If a Hamilton cycle starts in vertex s_1 then it must visit the diamond-gadgets in the ordering $G_1, G_2, \ldots, G_n$.

D-Ham-Cycle (3c): Reduction / Diamond-Gadgets

In the resulting graph: If a Hamilton cycle starts in vertex s_1 then it must visit the diamond-gadgets in the ordering $G_1, G_2, \ldots, G_n$.

The Hamilton cycle may choose for every gadget G_i whether it traverses

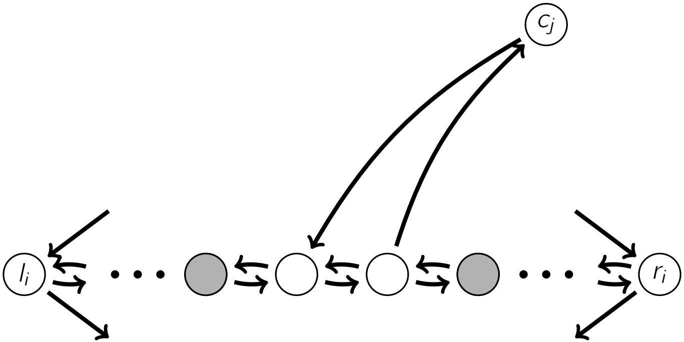
- the gadget *from left to right* (that is: from l_i to r_i)
- or *from right to left* (that is: from r_i to l_i).

The LR variant is interpreted as truth assignment $x_i = 0$,
and the RL variant as truth assignment $x_i = 1$.

D-Ham-Cycle (4a): Reduction / Clause-Vertices

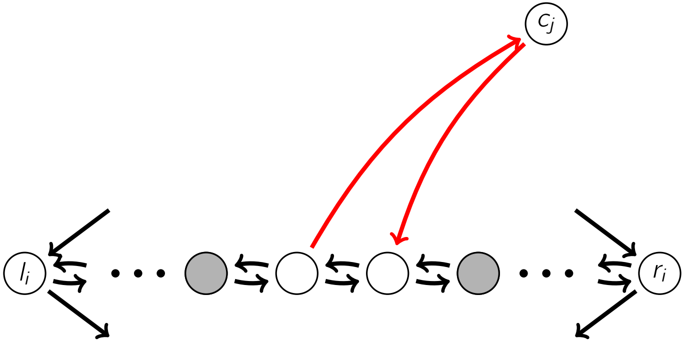
In the next step we create one more vertex for every clause c_j.

(a) If literal x_i occurs in clause c_j,
then connect gadget G_i in the following way with clause-vertex c_j:



D-Ham-Cycle (4b): Reduction / Clause-Vertices

(b) If literal $\bar{x}_i$ occurs in clause c_j,
then connect gadget G_i in the following way with clause-vertex c_j:



D-Ham-Cycle (4c): Reduction / Clause-Vertices

Question

Is it possible that after the creation of all these clause-vertices some Hamiltonian cycle jumps forward and backward between the diamond-gadgets instead of traversing them in the natural order?

D-Ham-Cycle (4c): Reduction / Clause-Vertices

Question

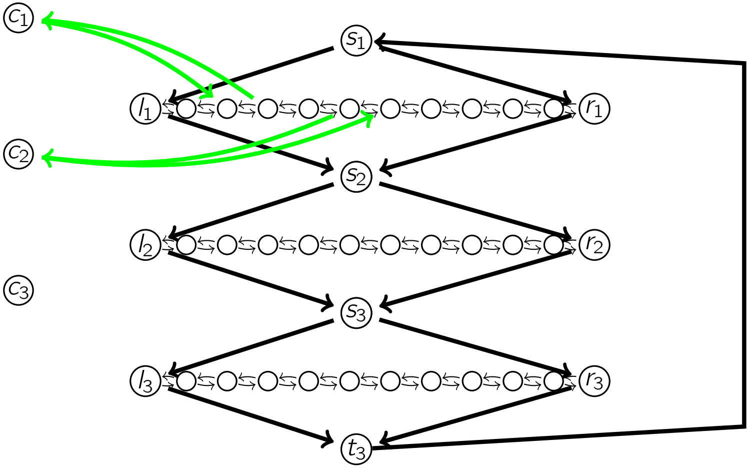
Is it possible that after the creation of all these clause-vertices some Hamiltonian cycle jumps forward and backward between the diamond-gadgets instead of traversing them in the natural order?

Answer

No. (Why??)

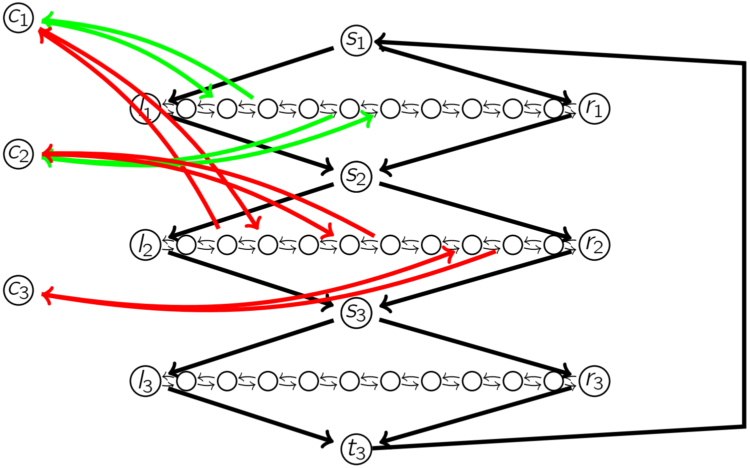
D-Ham-Cycle (5): Illustration

$$\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3)$$



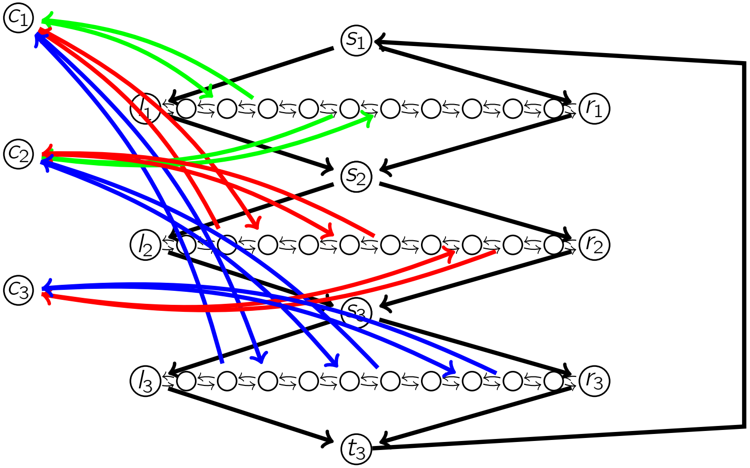
D-Ham-Cycle (5): Illustration

$$\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3)$$



D-Ham-Cycle (5): Illustration

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



D-Ham-Cycle (6a): Correctness

Lemma A: G has directed Hamilton cycle $\Rightarrow$ φ satisfiable

D-Ham-Cycle (6a): Correctness

Lemma A: G has directed Hamilton cycle $\Rightarrow$ φ satisfiable

- If a clause-vertex c_j is visited from a gadget G_i that is traversed **left-to-right**, then according to our construction clause c_j must contain literal $\bar{x}_i$.
- Consequently this clause c_j will be satisfied, whenever variable x_i is set to $x_i = 0$ (LR).

D-Ham-Cycle (6a): Correctness

Lemma A: G has directed Hamilton cycle $\Rightarrow$ φ satisfiable

- If a clause-vertex c_j is visited from a gadget G_i that is traversed **left-to-right**, then according to our construction clause c_j must contain literal $\bar{x}_i$.
- Consequently this clause c_j will be satisfied, whenever variable x_i is set to $x_i = 0$ (LR).
- If a clause-vertex c_j is visited from a gadget G_i that is traversed **right-to-left**, then according to our construction clause c_j must contain literal x_i.
- Consequently this clause c_j will be satisfied, whenever variable x_i is set to $x_i = 1$ (RL).

D-Ham-Cycle (6a): Correctness

Lemma A: G has directed Hamilton cycle $\Rightarrow$ φ satisfiable

- If a clause-vertex c_j is visited from a gadget G_i that is traversed **left-to-right**, then according to our construction clause c_j must contain literal $\bar{x}_i$.
- Consequently this clause c_j will be satisfied, whenever variable x_i is set to $x_i = 0$ (LR).
- If a clause-vertex c_j is visited from a gadget G_i that is traversed **right-to-left**, then according to our construction clause c_j must contain literal x_i.
- Consequently this clause c_j will be satisfied, whenever variable x_i is set to $x_i = 1$ (RL).
- Summarizing: The truth setting associated with the Hamilton cycle does indeed satisfy formula φ.

D-Ham-Cycle (6b): Correctness

Lemma B: φ satisfiable $\Rightarrow$ G has directed Hamilton cycle

D-Ham-Cycle (6b): Correctness

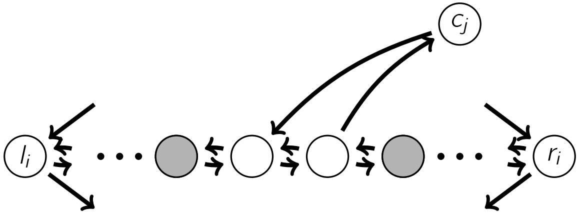
Lemma B: φ satisfiable $\Rightarrow$ G has directed Hamilton cycle

- A satisfying truth assignment for the variables determines for every diamond-gadget $G_1, \ldots, G_n$, whether it is traversed left-to-right or right-to-left.

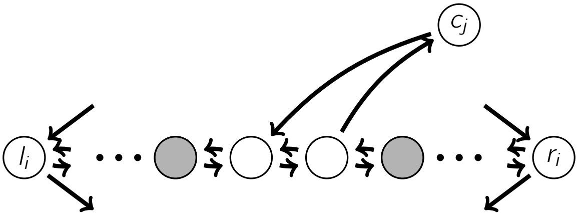
D-Ham-Cycle (6b): Correctness

Lemma B: φ satisfiable $\Rightarrow$ G has directed Hamilton cycle

- A satisfying truth assignment for the variables determines for every diamond-gadget $G_1, \ldots, G_n$, whether it is traversed left-to-right or right-to-left.
- The clause-vertex c_j can be built into the traversal: We pick a variable x_i that makes clause c_j true, and we visit c_j by a short excursion from the diamond-gadget G_i.

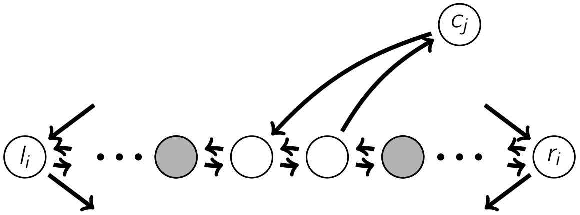


D-Ham-Cycle (6c): Correctness



- If c_j is satisfied for $x_i = 1$, then x_i occurs in **positive** form in c_j. A right-to-left traversal of diamond-gadget G_i allows a short excursion to c_j.
- If c_j is satisfied for $x_i = 0$, then x_i occurs in **negative** form in c_j. A left-to-right traversal of diamond-gadget G_i allows a short excursion to c_j.

D-Ham-Cycle (6c): Correctness



- If c_j is satisfied for $x_i = 1$, then x_i occurs in **positive** form in c_j. A right-to-left traversal of diamond-gadget G_i allows a short excursion to c_j.
- If c_j is satisfied for $x_i = 0$, then x_i occurs in **negative** form in c_j. A left-to-right traversal of diamond-gadget G_i allows a short excursion to c_j.
- Hence all the clause-vertices can be integrated into the traversal.

D-Ham-Cycle (7): Wrapping Up

4. (Polynomial time):
Show that f can be computed in polynomial time.

- The construction uses n diamond-gadgets, each with $O(m)$ vertices
- The construction uses m clause-vertices

5. (Correctness):
Show that for $x \in \{0,1\}^*$ we have $x \in L^*$ if and only if $f(x) \in L$.

$\varphi \in$ SAT $\Leftrightarrow$ $f(\varphi) = \langle G \rangle \in$ D-Ham-Cycle

Ham-Cycle (1): Definition

Problem: Hamilton cycle (Ham-Cycle)

Instance: An undirected graph $G = (V, E)$

Question: Does G contain a Hamilton cycle?

Theorem

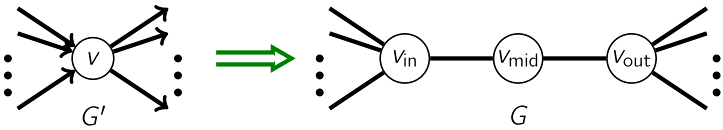
Ham-Cycle is NP-complete.

Proof:

- We show D-Ham-Cycle $\leq_p$ Ham-Cycle
- Let $G' = (V', A')$ be an instance of D-Ham-Cycle
- We construct in polynomial time an undirected graph $G = (V, E)$, so that: $G' \in$ D-Ham-Cycle $\Leftrightarrow$ $G \in$ Ham-Cycle

Ham-Cycle (2): Reduction

- Let $G' = (V', A')$ be an instance of D-Ham-Cycle
- The undirected graph G results from G' by local replacements:



Interpretation:

- v_{in} is entrance-vertex for v_{mid}
- v_{out} is exit-vertex for v_{mid}

Ham-Cycle (3): Correctness

G' has directed Hamilton cycle $\Leftrightarrow$ G has Hamilton cycle

(A) Each directed Hamilton cycle in G' can easily be translated into a Hamilton cycle in G.

(B) What about the reverse statement?

Ham-Cycle (3): Correctness

G' has directed Hamilton cycle $\Leftrightarrow$ G has Hamilton cycle

(A) Each directed Hamilton cycle in G' can easily be translated into a Hamilton cycle in G.

(B) What about the reverse statement?

- Every Hamilton cycle in G visits vertex v_{mid} right between the two vertices v_{in} and v_{out}
- Either: $v_{\text{in}} - v_{\text{mid}} - v_{\text{out}}$ Or: $v_{\text{out}} - v_{\text{mid}} - v_{\text{in}}$

Ham-Cycle (3): Correctness

G' has directed Hamilton cycle $\Leftrightarrow$ G has Hamilton cycle

(A) Each directed Hamilton cycle in G' can easily be translated into a Hamilton cycle in G.

(B) What about the reverse statement?

- Every Hamilton cycle in G visits vertex v_{mid} right between the two vertices v_{in} and v_{out}
- Either: $v_{\text{in}} - v_{\text{mid}} - v_{\text{out}}$ Or: $v_{\text{out}} - v_{\text{mid}} - v_{\text{in}}$
- There are no edges between u_{in} and v_{in}
- There are no edges between u_{out} and v_{out}
- Hence every Hamilton cycle in G can be translated into a directed Hamilton cycle for G'.

TSP (1): Definitions

Traveling Salesman Problem (TSP)

Instance: Cities $1, \dots, n$; distances $d(i,j)$; a bound B
Question: Does there exist a roundtrip of length at most B?

Two special cases:

Problem: Δ-TSP

Instance: Cities $1, \dots, n$; symmetric distances $d(i,j)$ that satisfy the triangle inequality $d(i,j) \leq d(i,k) + d(k,j)$; a bound B
Question: Does there exist a roundtrip of length at most B?

Problem: $\{1,2\}$-TSP

Instance: Cities $1, \dots, n$; symmetric distances $d(i,j) \in \{1,2\}$; a bound B
Question: Does there exist a roundtrip of length at most B?

TSP (2): Proof of NP-Completeness

Theorem

TSP and Δ-TSP and $\{1, 2\}$-TSP are NP-hard.

- It is enough to show that $\{1, 2\}$-TSP is NP-hard.
- We show: Ham-Cycle $\leq_p$ $\{1, 2\}$-TSP

TSP (2): Proof of NP-Completeness

Theorem

TSP and Δ-TSP and $\{1, 2\}$-TSP are NP-hard.

- It is enough to show that $\{1, 2\}$-TSP is NP-hard.
- We show: Ham-Cycle $\leq_p$ $\{1, 2\}$-TSP

- From an undirected graph $G = (V, E)$ for Ham-Cycle we construct a TSP instance.
- Each vertex $v \in V$ becomes a city
- The distance between city u and city v is

$$d(u, v) = \begin{cases} 1 & \text{in case } \{u, v\} \in E \\ 2 & \text{in case } \{u, v\} \notin E \end{cases}$$

- We set $B := |V|$

TSP (2): Proof of NP-Completeness

Theorem

TSP and Δ-TSP and $\{1,2\}$-TSP are NP-hard.

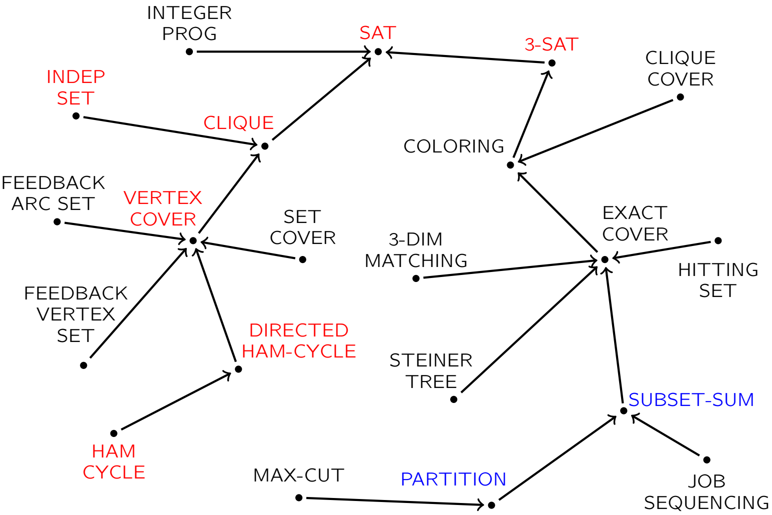
- It is enough to show that $\{1,2\}$-TSP is NP-hard.
- We show: Ham-Cycle $\leq_p$ $\{1,2\}$-TSP

- From an undirected graph $G = (V, E)$ for Ham-Cycle we construct a TSP instance.
- Each vertex $v \in V$ becomes a city
- The distance between city u and city v is

$$d(u, v) = \begin{cases} 1 & \text{in case } \{u, v\} \in E \\ 2 & \text{in case } \{u, v\} \notin E \end{cases}$$

- We set $B := |V|$
- Graph G has a Hamilton cycle,
  if and only if the constructed TSP instance has a tour of length $\leq B$.

Landscape with Karp's 20 reductions



SUBSET-SUM (1): Definition

SUBSET-SUM

Instance: Positive integers $a_1, \ldots, a_n$; a bound b

Question: Does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = b$?

Example: Instance for SUBSET-SUM

Numbers $1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024$ and $b = 999$

SUBSET-SUM (1): Definition

SUBSET-SUM

Instance: Positive integers $a_1, \ldots, a_n$; a bound b

Question: Does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = b$?

Example: Instance for SUBSET-SUM

Numbers $1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024$ and $b = 999$

Theorem

SUBSET-SUM is NP-complete.

SUBSET-SUM (2): Reduction

Theorem

SUBSET-SUM is NP-complete.

Proof:

- SUBSET-SUM lies in NP
- We show 3-SAT $\leq_p$ SUBSET-SUM
- Consider Boolean formula φ in 3-CNF as instance of 3-SAT
- The formula has clauses $c_1, \ldots, c_m$ with variables $x_1, \ldots, x_n$

The reduction works with decimal numbers with $n + m$ digits.
The k-th digit of an integer z will always be denoted $z(k)$.

SUBSET-SUM (3a): Var-Numbers / Definition

We define:

$$S^+(i) = \{j \in \{1, \ldots, m\} \mid \text{clause } c_j \text{ contains literal } x_i\}$$
$$S^-(i) = \{j \in \{1, \ldots, m\} \mid \text{clause } c_j \text{ contains literal } \bar{x}_i\}$$

SUBSET-SUM (3a): Var-Numbers / Definition

We define:

$$S^+(i) = \{j \in \{1,\ldots,m\} \mid \text{clause } c_j \text{ contains literal } x_i\}$$
$$S^-(i) = \{j \in \{1,\ldots,m\} \mid \text{clause } c_j \text{ contains literal } \bar{x}_i\}$$

For every Boolean variable x_i with $1 \leq i \leq n$, we create two corresponding Var-numbers a_i^+ and a_i^- with the following digits:

$$a_i^+(i) = 1 \quad \text{and for all } j \in S^+(i): \; a_i^+(n+j) = 1$$
$$a_i^-(i) = 1 \quad \text{and for all } j \in S^-(i): \; a_i^-(n+j) = 1$$

All other digits in these decimal representations are 0.

SUBSET-SUM (3b): Var-Numbers / Example

As an example, consider the formula

$$(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$

The following Var-numbers are created:

$$
\begin{aligned}
a_1^+ &= 100010 \\
a_1^- &= 100000 \\
a_2^+ &= 010011 \\
a_2^- &= 010000 \\
a_3^+ &= 001010 \\
a_3^- &= 001001 \\
a_4^+ &= 000100 \\
a_4^- &= 000101
\end{aligned}
$$

SUBSET-SUM (3c): Dummy-Numbers

- For every clause c_j we introduce two corresponding **Dummy-numbers** d_j and d'_j.
- Dummy-numbers have digit 1 at position $n+j$; all other digits are 0.

Example, continued

Once again consider the formula

$$(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_4})$$

The Dummy-numbers for the two clauses are:

$$d_1 = 000010$$
$$d'_1 = 000010$$
$$d_2 = 000001$$
$$d'_2 = 000001$$

SUBSET-SUM (3d): Goal Value

The **goal value** b is defined as follows:

- $b(k) = 1$ for $1 \leq k \leq n$,
- $b(k) = 3$ for $n+1 \leq k \leq n+m$.

Example, completed

Once again consider the formula

$$(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$

The corresponding goal value is:

$$b = 111133$$

SUBSET-SUM (4a): Illustration

An example for a formula with n variables and m clauses:

	1	2	3	$\cdots$	n	$n+1$	$n+2$	$\cdots$	$n+m$
a_1^+	1	0	0	$\cdots$	0	1	0	$\cdots$	$\cdots$
a_1^-	1	0	0	$\cdots$	0	0	0	$\cdots$	$\cdots$
a_2^+	0	1	0	$\cdots$	0	0	1	$\cdots$	$\cdots$
a_2^-	0	1	0	$\cdots$	0	1	0	$\cdots$	$\cdots$
a_3^+	0	0	1	$\cdots$	0	1	1	$\cdots$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_n^+	0	0	0	$\cdots$	1	0	0	$\cdots$	$\cdots$
a_n^-	0	0	0	$\cdots$	1	0	1	$\cdots$	$\cdots$
d_1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
d_1'	0	0	0	$\cdots$	0	1	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
d_m	0	0	0	$\cdots$	0	0	0	$\cdots$	1
d_m'	0	0	0	$\cdots$	0	0	0	$\cdots$	1
b	1	1	1	$\cdots$	1	3	3	$\cdots$	3

SUBSET-SUM (4b): Illustration

- For every decimal position $i \in \{1, \ldots, n\}$ we have: Only two of the Var-numbers and Dummy-numbers have a digit 1 at this position; all other numbers have a digit 0 at this position.
- For every decimal position $i \in \{n+1, \ldots, n+m\}$ we have: Only five of the Var-numbers and Dummy-numbers have a digit 1 at this position; all other numbers have a digit 0 at this position.

SUBSET-SUM (4b): Illustration

- For every decimal position $i \in \{1, \ldots, n\}$ we have: Only two of the Var-numbers and Dummy-numbers have a digit 1 at this position; all other numbers have a digit 0 at this position.
- For every decimal position $i \in \{n+1, \ldots, n+m\}$ we have: Only five of the Var-numbers and Dummy-numbers have a digit 1 at this position; all other numbers have a digit 0 at this position.

Observation: No carry-overs

If we add up an arbitrary subset of Var-numbers and Dummy-numbers,
then there are no carry-overs from one decimal position to the next one.

SUBSET-SUM (5): Time Complexity of Reduction

- The SAT instance φ consists of n variables and m clauses.
  The input size is $\geq m + n$.
- The constructed SUBSET-SUM instance consists of $2n + 2m + 1$ decimal numbers each with $m + n$ decimal places.
- The reduction can be performed $O\left((m+n)^2\right)$ in polynomial time.

SUBSET-SUM (6a): Correctness

Lemma A: Formula φ satisfiable $\Rightarrow$ SUBSET-SUM instance solvable

Suppose there is a satisfying truth setting x^* for formula φ.

SUBSET-SUM (6a): Correctness

Lemma A: Formula φ satisfiable $\Rightarrow$ SUBSET-SUM instance solvable

Suppose there is a satisfying truth setting x^* for formula φ.

- If $x_i^* = 1$, then select a_i^+; otherwise select a_i^-
- The sum of the selected Var-numbers is denoted A
- As for every $i \in \{1, \ldots, n\}$ either a_i^+ or a_i^- has been selected, we have $A(i) = 1$

SUBSET-SUM (6a): Correctness

Lemma A: Formula φ satisfiable $\Rightarrow$ SUBSET-SUM instance solvable

Suppose there is a satisfying truth setting x^* for formula φ.

- If $x_i^* = 1$, then select a_i^+; otherwise select a_i^-
- The sum of the selected Var-numbers is denoted A
- As for every $i \in \{1, \dots, n\}$ either a_i^+ or a_i^- has been selected, we have $A(i) = 1$
- Furthermore $A(n+j) \in \{1, 2, 3\}$ for $1 \leq j \leq m$, as every clause contains one or two or three true literals.

SUBSET-SUM (6a): Correctness

Lemma A: Formula φ satisfiable $\Rightarrow$ SUBSET-SUM instance solvable

Suppose there is a satisfying truth setting x^* for formula φ.

- If $x_i^* = 1$, then select a_i^+; otherwise select a_i^-
- The sum of the selected Var-numbers is denoted A
- As for every $i \in \{1, \ldots, n\}$ either a_i^+ or a_i^- has been selected, we have $A(i) = 1$
- Furthermore $A(n+j) \in \{1, 2, 3\}$ for $1 \leq j \leq m$, as every clause contains one or two or three true literals.
- If $A(n+j) \in \{1, 2\}$, then we additionally select d_j and/or d_j', so that we have digit 3 at position $n+j$ of the sum.

Hence there exists a subset with the desired goal sum b.

SUBSET-SUM (6b): Correctness

Lemma B: SUBSET-SUM instance solvable $\Rightarrow$ formula φ satisfiable

Suppose that there exists a subset K_A of the Var-numbers (with sum A) and a subset K_D of the Dummy-numbers (with sum D), that sum up to the goal sum b; hence: $A + D = b$.

SUBSET-SUM (6b): Correctness

Lemma B: SUBSET-SUM instance solvable $\Rightarrow$ formula φ satisfiable

Suppose that there exists a subset K_A of the Var-numbers (with sum A) and a subset K_D of the Dummy-numbers (with sum D), that sum up to the goal sum b; hence: $A + D = b$.

- The set K_A contains for every $i \in \{1, \ldots, n\}$ exactly one of the two Var-numbers a_i^+ and a_i^-; otherwise we would have $A(i) \neq 1$.

SUBSET-SUM (6b): Correctness

Lemma B: SUBSET-SUM instance solvable $\Rightarrow$ formula φ satisfiable

Suppose that there exists a subset K_A of the Var-numbers (with sum A) and a subset K_D of the Dummy-numbers (with sum D), that sum up to the goal sum b; hence: $A + D = b$.

- The set K_A contains for every $i \in \{1, \ldots, n\}$ exactly one of the two Var-numbers a_i^+ and a_i^-; otherwise we would have $A(i) \neq 1$.
- We set $x_i = 1$ in case $a_i^+ \in K_A$, and otherwise $x_i = 0$.

SUBSET-SUM (6b): Correctness

Lemma B: SUBSET-SUM instance solvable $\Rightarrow$ formula φ satisfiable

Suppose that there exists a subset K_A of the Var-numbers (with sum A) and a subset K_D of the Dummy-numbers (with sum D), that sum up to the goal sum b; hence: $A + D = b$.

- The set K_A contains for every $i \in \{1, \ldots, n\}$ exactly one of the two Var-numbers a_i^+ and a_i^-; otherwise we would have $A(i) \neq 1$.
- We set $x_i = 1$ in case $a_i^+ \in K_A$, and otherwise $x_i = 0$.
- Then $A(n + j) \geq 1$ for $1 \leq j \leq m$.
  Otherwise we would have $A(n+j) + H(n+j) \leq A(n+j) + 2 < 3$.
- This guarantees that in each clause there is at least one literal with value 1.

Hence the formula φ is satisfiable.

PARTITION (1): Definition

Problem: PARTITION

Instance: Positive integers $a'_1, \ldots, a'_n$; with $\sum_{i=1}^{n} a'_i = 2A'$

Question: Does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a'_i = A'$?

PARTITION is the special case of SUBSET-SUM with $b := (\sum a_i)/2$

Theorem

PARTITION is NP-complete.

Proof:

- PARTITION is in NP
- We show SUBSET-SUM $\leq_p$ PARTITION

PARTITION (2): Reduction

- Let $a_1, \ldots, a_n \in \mathbb{N}$ and $b \in \mathbb{N}$ be an aribtrary instance of SUBSET-SUM
- Let $S := \sum_{i=1}^{n} a_i$, and w.l.o.g. assume $S \geq b$

We map this SUBSET-SUM instance into a PARTITION instance, that consists of the following $n + 2$ numbers $a'_1, \ldots, a'_{n+2}$:

PARTITION (2): Reduction

- Let $a_1, \ldots, a_n \in \mathbb{N}$ and $b \in \mathbb{N}$ be an aribtrary instance of SUBSET-SUM
- Let $S := \sum_{i=1}^{n} a_i$, and w.l.o.g. assume $S \geq b$

We map this SUBSET-SUM instance into a PARTITION instance, that consists of the following $n + 2$ numbers $a'_1, \ldots, a'_{n+2}$:

- $a'_i = a_i$ for $1 \leq i \leq n$
- $a'_{n+1} = 2S - b$ and $a'_{n+2} = S + b$

PARTITION (2): Reduction

- Let $a_1, \ldots, a_n \in \mathbb{N}$ and $b \in \mathbb{N}$ be an aribtrary instance of SUBSET-SUM
- Let $S := \sum_{i=1}^{n} a_i$, and w.l.o.g. assume $S \geq b$

We map this SUBSET-SUM instance into a PARTITION instance, that consists of the following $n+2$ numbers $a'_1, \ldots, a'_{n+2}$:

- $a'_i = a_i$ for $1 \leq i \leq n$
- $a'_{n+1} = 2S - b$ and $a'_{n+2} = S + b$

The sum of these $n+2$ numbers is $\sum_{i=1}^{n+2} a'_i = 4S$.
Therefore $A' := 2S$ holds in the PARTITION instance.

The reduction can be done in polynomial time.

PARTITION (3a): Correctness

Lemma A: SUBSET-SUM instance solvable $\Rightarrow$ PARTITION instance solvable

- If the SUBSET-SUM instance contains a subset of the numbers $a_1, \ldots, a_n$ with sum b, then the corresponding numbers $a'_1, \ldots, a'_n$ in the PARTITION instance also have sum b.
- We add the number $a'_{n+1} = 2S - b$ to this subset, and we get a subset with the desired goal sum $A' = 2S$.

PARTITION (3b): Correctness

Lemma B: PARTITION instance solvable $\Rightarrow$ SUBSET-SUM instance solvable

- In any solution of the PARTITION instance, the two numbers $a'_{n+1} = 2S - b$ and $a'_{n+2} = S + b$ cannot be both in the same part, as $a'_{n+1} + a'_{n+2} = 3S > A'$ holds.
- Hence one of the two parts consists of $a'_{n+1} = 2S - b$ and a subset of the numbers $a'_1, \ldots, a'_n$ with total sum $A' = 2S$.
- The corresponding numbers in the SUBSET-SUM instance then have sum b.

Coding Length (1)

- Let X be an algorithmic problem
- We measure the running time of an algorithm A for problem X in terms of the **coding length** (or **size**) of the instances I of X
- The coding length $|I|$ is the number of symbols in a "reasonable" description of instance I
- Small (polynomial) changes in such descriptions are irrelevant for our definitions / theorems / proofs / results

Coding Length (2)

Example: Undirected graphs

Reasonable descriptions of undirected graphs $G = (V, E)$ are

- adjacency lists of length $\ell_1(G) = O(|E| \log |V|)$
- adjacency matrices of length $\ell_2(G) = O(|V|^2)$

We have:

- $\ell_1(G)$ is polynomially bounded in $\ell_2(G)$
- $\ell_2(G)$ is polynomially bounded in $\ell_1(G)$

Coding Length (3)

Example: Natural numbers

Reasonable descriptions of natural numbers n are

- Decimal representation of length $\approx \log_{10} n$
- Binary representation of length $\approx \log_2 n$
- Octal representation of length $\approx \log_8 n$
- Hexadecimal representation of length $\approx \log_{16} n$

For all real numbers $a, b > 1$ we have: $\log_a n = \log_a b \cdot \log_b n$
Hence the various coding lengths only differ by a constant factor.

Remark

The number n represents the **value** n with **coding length** $O(\log n)$.
Note that the value depends exponentially on the coding length.

Value versus Coding Length

Definition: Number

For an instance I of some algorithmic problem,
we denote by **Number(I)** the value of the larget number in I.

Value versus Coding Length

Definition: Number

For an instance I of some algorithmic problem,
we denote by **Number(I)** the value of the larget number in I.

Example

- For a TSP instance I,
  $Number(I)$ equals the largest inter-city distance $max_{i,j} d(i,j)$.
- For a SUBSET-SUM instance I,
  $Number(I)$ equals the maximum of the numbers $a_1, \ldots, a_n$ and b.
- For a SAT instance I,
  $Number(I)$ equals the maximum of the numbers n and m.
  (Ergo: $Number(I) \leq |I|$.)

The parameter $Number(I)$ is only relevant for problems that deal with distances, costs, weights, lengths, profits, time intervals, etc

Pseudo-Polynomial Time (1): Definition

Definition: Pseudo-polynomial time

An algorithm A solves a problem X in **pseudo-polynomial** time,
if the run time of A on instances I of X
is polynomially bounded in $|I|$ and $Number(I)$.

Pseudo-Polynomial Time (1): Definition

Definition: Pseudo-polynomial time

An algorithm A solves a problem X in **pseudo-polynomial** time,
if the run time of A on instances I of X
is polynomially bounded in $|I|$ and $Number(I)$.

Theorem

The problems SUBSET-SUM and PARTITION are pseudo-polynomially solvable.

Pseudo-Polynomial Time (2): Example

SUBSET-SUM

Instance: Positive integers $a_1, \ldots, a_n$; a bound b

Question: Does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = b$?

Theorem

SUBSET-SUM is solvable in pseudo-polynomial time $O(n \cdot b)$.

Pseudo-Polynomial Time (2): Example

SUBSET-SUM

Instance: Positive integers $a_1, \ldots, a_n$; a bound b

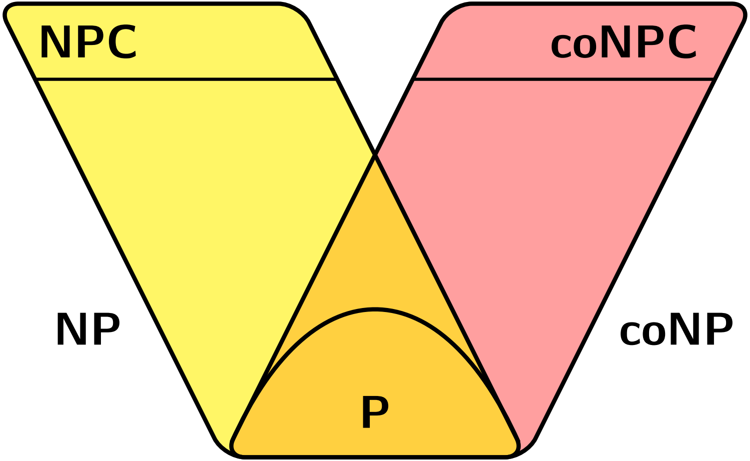
Question: Does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = b$?

Theorem

SUBSET-SUM is solvable in pseudo-polynomial time $O(n \cdot b)$.

Proof:

- Dynamic programming: For $k = 0, \ldots, n$ and $c = 0, \ldots, b$ we set $F[k, c] = TRUE$ if and only if there exists an index set $I \subseteq \{1, \ldots, k\}$ with $\sum_{i \in I} a_i = c$.
- $F[0, c] = (c == 0)$ for $c = 0, \ldots, b$
  $F[k, c] = F[k-1, c - a_k] \vee F[k-1, c]$
- In the end, the answer is found in $F[n, b]$ □


NPC
coNPC
NP
coNP
P