1. Characterizing \( \text{NP} \)

**Definition.**

1. A relation \( R \subseteq \Sigma^* \times \Sigma^* \) is *polynomially decidable* if there is a deterministic TM deciding the language \( \{x,y \mid (x,y) \in R\} \) in polynomial time.
2. A relation \( R \) is *polynomially balanced* if \( (x,y) \in R \) implies \( |y| \leq |x|^k \) for some \( k \geq 1 \).

**Proposition.** Let \( L \subseteq \Sigma^* \) be a language.

Now \( L \in \text{NP} \) iff there is a polynomially balanced and polynomially decidable relation \( R \) such that \( L = \{x \in \Sigma^* \mid (x,y) \in R \) for some \( y \in \Sigma^*\} \).
**Boundary between NP and P**

- Most problems arising in computational practice are in NP.
- Computational complexity theory provides us tools to study which problems in NP belong to P and which do not.
- NP-completeness is a basic tool in this respect:
  - Showing that a problem is NP-complete implies that the problem is among the least likely to be in P.
  - (If an NP-complete problem is in P, then NP = P.)

**NP-completeness and algorithm design techniques**

When a problem is known to be NP-complete, further efforts are usually directed to:

(i) Attacking special cases

(ii) Approximation algorithms

(iii) Studying average case performance

(iv) Randomized algorithms

(v) (Exponential) algorithms that are practical for small instances

(vi) Local search methods

2. **Variants of Satisfiability**

- Many problems if generalized enough become NP-complete.
- Often it is important to find the dividing line between P and NP-completeness.
- One basic technique is to investigate the set of instances produced by a reduction R involved in the NP-completeness proof in order to capture another NP-complete problem.
- Next we consider variants of SAT such as 3SAT, 2SAT, MAX2SAT, and NAESA T and analyze their computational complexities.

3SAT problems

**Definition.** kSAT, where \( k \geq 1 \) is an integer, is the set of Boolean expressions \( \phi \in \text{SAT} \) (in CNF) whose all clauses have exactly \( k \) literals.

**Proposition.** 3SAT is NP-complete.

**Proof.**

- 3SAT is in NP as a special case of SAT which is in NP.
- CIRCUIT SAT was shown to be NP-complete and a reduction from CIRCUIT SAT to SAT has already been presented.
- Consider now the clauses in the reduction. They have all at most 3 literals. Each clause with one or two literals can be modified to an equivalent clause with exactly 3 literals by duplicating literals.
- Hence, we can reduce CIRCUIT SAT to 3SAT. □
Narrowing NP-complete languages

An NP-complete language can sometimes be narrowed down by transformations which eliminate certain features of the language but still preserve NP-completeness.

The following result is a typical example.

Proposition. 3SAT remains NP-complete even if each variable is restricted to appear at most three times in a Boolean expression \( \phi \in 3\text{SAT} \) and each literal at most twice in \( \phi \).

Proof. This is shown by a reduction where any instance \( \phi \) of 3SAT is rewritten to eliminate the forbidden features.

Proof

Consider a variable \( x \) appearing \( k > 3 \) times in \( \phi \).

(i) Replace the first occurrence of \( x \) in \( \phi \) by \( x_1 \), the second by \( x_2 \), and so on where \( x_1, \ldots, x_k \) are new variables.

(ii) Add clauses \( (\neg x_1 \lor x_2), (\neg x_2 \lor x_3), \ldots, (\neg x_k \lor x_1) \) to \( \phi \).

Let \( \phi' \) be the expression \( \phi \) modified systematically in this way.

It follows that \( \phi' \) has the desired syntactic properties.

Now \( \phi \) is satisfiable iff \( \phi' \) is satisfiable:

For each \( x \) appearing \( k > 3 \) times in \( \phi \), the truth values of \( x_1, \ldots, x_k \) are the same in each truth assignment satisfying \( \phi' \).

Boundary between P and NP-completeness

The boundary is between 2SAT and 3SAT.

For an instance \( \phi \) of 2SAT, there is a polynomial time algorithm which is based on reachability in a graph associated with \( \phi \).

Definition. Let \( \phi \) be an instance of 2SAT.

Define a graph \( G(\phi) \) as follows:

– The variables of \( \phi \) and their negations form the vertices of \( G(\phi) \).

– There is an arc \((\alpha, \beta)\) iff there is a clause \( \alpha \lor \beta \) or \( \beta \lor \alpha \) in \( \phi \), i.e., if \( (\alpha, \beta) \) is an arc, so is \( (\beta, \alpha) \) where \( \overline{\beta} \) is the complement of \( \beta \).

Theorem. Let \( \phi \) be an instance of 2SAT.

Then \( \phi \) is unsatisfiable iff there is a variable \( x \) such that there are paths from \( x \) to \( \neg x \) and from \( \neg x \) to \( x \) in \( G(\phi) \).

The complexity of 2SAT—cont’d

Corollary. 2SAT is in NL (\( \subseteq P \)).

Proof. Since NL is closed under complement, it is sufficient to show that 2SAT COMPLEMENT is in NL.

The reachability condition of the preceding theorem can be tested in logarithmic space non-deterministically by guessing a variable \( x \) and paths from \( x \) to \( \neg x \) and back.

MAX2SAT is a generalization of 2SAT:

INSTANCE: a Boolean expression \( \phi \) in CNF (having at most two literals per clause) and an integer bound \( K \).

QUESTION: Is there a truth assignment satisfying at least \( K \) clauses?

Theorem. MAX2SAT is NP-complete. (See tutorials.)
In this section, we will consider only undirected graphs $G = (V,E)$ and their properties.

For instance, consider the problem of finding an independent subset $I$ of $V$, i.e., a set $I$ such that for all $i, j \in I$, $[i, j] \notin E$.

**INDEPENDENT SET:**

**INSTANCE:** An undirected graph $G = (V,E)$ and an integer $K$.

**QUESTION:** Is there an independent set $I \subseteq V$ with $|I| = K$.

**Theorem.** INDEPENDENT SET is NP-complete. (See tutorials.)

The subclass of graphs needed in the reduction implies the following:

**Corollary.** 4-DEGREE INDEPENDENT SET is NP-complete.

---

**The case of not-all-equal SAT (NAESAT)**

For each $\phi \in$ NAESAT $\subseteq$ 3SAT, there is a truth assignment so that the three literals in each clause of $\phi$ do not have the same truth value.

**Theorem.** NAESAT is NP-complete.

Proof.

- CIRCUIT SAT was shown to be NP-complete and a reduction $R$ from CIRCUIT SAT to SAT has already been presented such that for a circuit $C$, $C \in$ CIRCUIT SAT iff $R(C) \in$ SAT.
- For all one- and two-liter clauses in the resulting set of clauses $R(C)$, add the same literal, say $z$, to make them 3-literal clauses.

Claim: it holds for the resulting Boolean expression $R_z(C)$ in 3CNF:

$R_z(C) \in$ NAESAT iff $C \in$ CIRCUIT SAT.

($\Rightarrow$) If a truth assignment $T$ satisfies $R_z(C)$ in the sense of NAESAT, so does the complementary truth assignment $\overline{T}$.

Thus, $z$ is false in either $T$ or $\overline{T}$ which implies that $R(C)$ is satisfied by $T$ or $\overline{T}$. Thus $C$ is satisfiable.

($\Leftarrow$) If $C$ is satisfiable, then there is a truth assignment $T$ satisfying $R(C)$. Let us then extend $T$ for $R_z(C)$ by assigning $T(z) = false$.

In no clause of $R_z(C)$ all literals are true (they cannot be all false):

(i) Clauses for true/false/NOT/variable gates contain $z$ that is false.

(ii) For AND gates the clauses are: $(\neg g \lor h \lor z)$, $(\neg g \lor h' \lor z)$, $(g \lor \neg h' \lor \neg h')$ where in the first two $z$ is false, and in the third all three cannot be true as then the first two would be not true.

(iii) The case of OR gates is similar. □

**Graph problems: CLIQUE and NODE COVER**

- The problems in graph theory are often closely related; suggesting even trivial reductions between problems.

**Example.** Consider the following two graph theoretic problems:

**CLIQUE:**

**INSTANCE:** An undirected graph $G = (V,E)$ and an integer $K$.

**QUESTION:** Is there a clique $C \subseteq V$ with $|C| = K$?

(A set $C \subseteq V$ is clique iff for any two vertices $i, j \in I$, $[i, j] \in E$)

**NODE COVER:**

**INSTANCE:** An undirected graph $G = (V,E)$ and an integer $B$.

**QUESTION:** Is there a set $C \subseteq V$ with $|C| \leq B$ such that for all $[i, j] \in E$, $i \in C$ or $j \in C$?
Trivial reductions for CLIQUE and NODE COVER

> Independent sets are closely related to cliques and node covers:
> A set $I \subseteq V$ of vertices is
> 1. an independent set of $G$ iff it is a clique of the complement graph $\overline{G}$, and
> 2. an independent set of $G$ iff $V - I$ is a node cover of $G$.

Thus an instance $G;K$ of INDEPENDENT SET can be reduced to
— an instance $G;K$ of CLIQUE, and
— an instance $G;|V| - K$ of NODE COVER.

**Corollary.** CLIQUE and NODE COVER are NP-complete.

---

Graph problems: MIN CUT and MAX CUT

> A cut in an undirected graph $G = (V,E)$ is a partition of the nodes into two nonempty sets $S$ and $V - S$.

> The size of a cut is the number of edges between $S$ and $V - S$.

> The problem of finding a cut with the smallest size is in P: 
  (i) The smallest cut that separates two given nodes $s$ and $t$ equals to the maximum flow from $s$ to $t$.
  (ii) Minimum cut: find the maximum flow between a fixed $s$ and all other nodes and choose the smallest value found.

> However, the problem of deciding whether there is a cut of a size greater than or equal to $K$ (MAX CUT) is much harder:

**Theorem.** MAX CUT is NP-complete.

---

Reduction from NAESAT to MAX CUT

The NP-completeness of MAX CUT is shown for graphs with multiple edges between nodes by a reduction from NAESAT.

> For a conjunction of clauses $\phi = C_1 \land \ldots \land C_m$, we construct a graph $G = (V,E)$ so that

  $G$ has a cut of size $5m$ iff $\phi$ is satisfied in the sense of NAESAT.

> The nodes of $G$ are $x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$ where $x_1, \ldots, x_n$ are the variables in $\phi$.

> The edges in $G$ include a triangle $[\alpha, \beta, \gamma]$ for each clause $\alpha \lor \beta \lor \gamma$ and $n_i$ copies of the edge $[x_i, \neg x_i]$ where $n_i$ is the number of occurrences of $x_i$ or $\neg x_i$ in the clauses.

Correctness of the reduction

> Suppose there is a cut $(S, V - S)$ of size $5m$ or more.

> All variables can be assumed separate from their negations: 
  If both $x_i, \neg x_i$ are on the same side, they contribute at most $2n_i$ edges to the cut and, hence, changing the one with fewer neighbors does not decrease the size of the cut.

> Let $S$ be the set of true literals and $V - S$ those false.

> The total number of edges in the cut joining opposite literals is $3m$. The remaining $2m$ are coming from triangles meaning that all $m$ triangles are cut, i.e. $\phi$ is satisfied in the sense of NAESAT.

> Conversely, a satisfying truth assignment (in the sense of NAESAT) gives rise to a cut of size $5m$. □
Graph problems: MAX BISECTION

➤ In applications of graph partitioning, the sizes of $S$ and $V - S$ cannot be arbitrarily small or large. MAX BISECTION is the problem of determining whether there is a cut $(S, V - S)$ with size of $K$ or more such that $|S| = |V - S|$.

➤ Is MAX BISECTION easier than MAX CUT?

Lemma. MAX BISECTION is $\mathbf{NP}$-complete.

Proof. Reducing MAX CUT to MAX BISECTION by modifying input: Add $|V|$ disconnected new nodes to $G$. Now every cut of $G$ can be made a bisection by appropriately splitting the new nodes. Now $G = (V,E)$ has a cut $(S, V - S)$ with size of $K$ or more iff the modified graph has a cut with size of $K$ or more with $|S| = |V - S|$. $\square$

Graph problems: BISECTION WIDTH

➤ The respective minimization problem, i.e. MIN CUT with the bisection requirement, is $\mathbf{NP}$-complete, too. (Remember that MIN CUT $\in \mathbf{P}$).

➤ BISECTION WIDTH: is there a bisection of size $K$ or less?

Theorem. BISECTION WIDTH is $\mathbf{NP}$-complete.

Proof. A reduction from MAX BISECTION:

A graph $G = (V,E)$ where $|V| = 2n$ for some $n$ has a bisection of size $K$ or more iff the complement $\overline{G}$ has a bisection of size $n^2 - K$ or less. $\square$

Graph problems: HAMILTON PATH

Theorem. HAMILTON PATH is $\mathbf{NP}$-complete.

Proof.

➤ Reduction from 3SAT to HAMILTON PATH:

given a formula $\phi$ in CNF with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ each with three literals, we construct a graph $R(\phi)$ that has a Hamilton path iff $\phi$ is satisfiable.

➤ Choice gadgets select a truth assignment for variables $x_i$.

➤ Consistency gadgets (XOR) enforce that all occurrences of $x_i$ have the same truth value and all occurrences of $\neg x_i$ the opposite.

➤ Constraint gadgets guarantee that all clauses are satisfied.
Gadgets [Papadimitriou, 1994]

Reduction from 3SAT to HAMILTON PATH

The graph $R(\phi)$ is constructed as follows:

- The choice gadgets of variables $x_i$ are connected in series.
- A constraint gadget (triangle) for each clause with an edge identified with each literal $l$ in the clause.
  - If $l$ is $x_i$, then XOR to true edge of choice gadget of $x_i$.
  - If it is $\neg x_i$, then XOR to false edge of choice gadget of $x_i$.
- All nodes of the triangles, the end node of choice gadgets and a new node 3 form a clique. Add a node 2 connected to 3.

Basic idea: each side of the constraint gadget is traversed by the Hamilton path iff the corresponding literal is false. Hence, at least one literal in any clause is true since otherwise all sides for its triangle should be traversed which is impossible (implying no Hamilton path).
**Correctness of the reduction**

- If there is a Hamilton path, $\phi$ is satisfiable:
  The path starts at 1, makes a truth assignment, traverses the triangles in some order and ends up in 2. The truth assignment satisfies $\phi$ as there are no triangle where all sides are traversed, i.e., where all literals are false.

- If $\phi$ is satisfiable, there is a Hamilton path:
  From a satisfying truth assignment, we construct a Hamilton path by starting at 1, traversing choice gadgets according to the truth assignment, the rest is a big clique for which a trivial path can be found leading to 3 and then to 2.

---

**4. Coloring Problems**

Consider the following problem:

$k$-COLORING:

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

QUESTION: Is there an assignment of one of $k$ colors to each of the nodes in $V$ such that any two nodes $i, j$ connected by an edge $[i, j] \in E$ do not have the same color?

Coloring with $k = 2$ colors is easy (in $\mathbf{P}$) but not when $k = 3$.

---

**TSP (D) revisited**

**Corollary.** TSP(D) is NP-complete.

**Proof:** A reduction from HAMILTON PATH to TSP(D). Given a graph $G$ with $n$ nodes, construct a distance matrix $d_{ij}$ and a budget $B$ so that there is a tour of length $B$ or less iff $G$ has a Hamilton path.

- There are $n$ cities and the distance $d_{ij} = 1$ if there is $[i, j] \in G$ and $d_{ij} = 2$ otherwise. The budget $B = n + 1$.
- If there is a tour of length $n + 1$ or less, then there is at most one pair $(\pi(i), \pi(i + 1))$ in it with cost 2, i.e., a pair for which $[\pi(i), \pi(i + 1)]$ is not an edge. Removing it gives a Hamilton path.
- If $G$ has a Hamilton path, then its cost is $n − 1$ and it can be made a tour with additional cost of 2.

---

**Determining the complexity of 3-COLORING**

**Theorem.** 3-COLORING is NP-complete.

**Proof.** A reduction from NAESAT to 3-COLORING.

- For a conjunction clauses $\phi = C_1 \land \ldots \land C_m$ with variables $x_1, \ldots, x_n$, construct a graph $G(\phi)$ that can be colored with $\{0, 1, 2\}$ iff there is a truth assignment satisfying all clauses in the way of NAESAT.

- **Choice gadgets:** For each variable $x_i$, we introduce a triangle $[a, x_i, \neg x_i]$, i.e., all triangles share a node $a$.

- **Constraints:** For each clause $C_i$: a triangle $[C_{i1}, C_{i2}, C_{i3}]$ where each $C_{ij}$ is further connected to the node with the $j$th literal in $C_i$. 

© 2007 TKK, Laboratory for Theoretical Computer Science
\( \Rightarrow \) Suppose \( G \) can be colored with \( \{0, 1, 2\} \) and \( a \) has color 2. This induces a truth assignment \( T \) via the colors of the nodes \( x_i \): if the color is 1, then \( T(x_i) = \text{true} \) else \( T(x_i) = \text{false} \).

If we assume that \( T \) assigns all literals of some clause \( C_i \) to true/false, then color 1/0 cannot be used for coloring \( \{C_{i1}, C_{i2}, C_{i3}\} \), a contradiction. Thus \( \phi \) is satisfied in the sense of NAESA T.

\( \Leftarrow \) Assume that \( \phi \) is satisfied by \( T \) in the sense of NAESA T. Then we can extract a coloring for \( G \) from \( T \) as follows:

1. Node \( a \) is colored with color 2.
2. If \( T(x_i) = \text{true} \), then color \( x_i \) with 1 and \( \neg x_i \) with 0 else vice versa.
3. From each \( \{C_{i1}, C_{i2}, C_{i3}\} \), color two literals having opposite truth values with 0 (true) and 1 (false). Color the third with 2. □

**5. Sets and Numbers**

**TRIPARTITE MATCHING:**

**INSTANCE:** Three sets \( B \) (boys), \( G \) (girls), and \( H \) (homes) each containing \( n \) elements and a ternary relation \( T \subseteq B \times G \times H \).

**QUESTION:** Is there a set of \( n \) triples in \( T \) no two of which have a component in common?

**Theorem.** TRIPARTITE MATCHING is \( \text{NP-complete} \).

**Proof.** By a reduction from 3SAT. Each variable \( x \) has a combined choice and consistency gadget and each clause \( c \) a dedicated pair of a boy \( b_c \) and a girl \( g_c \) and three triples \( (b_c, g_c, h_l) \) where \( h_l \) ranges over the three homes corresponding to the occurrences of literals in the clause (appearing in the combined gadgets).
**Correctness of the reduction**

- Note that \( T(x) = \text{true} \) matching leaves the homes for \( x \) unoccupied and \( T(x) = \text{false} \) those for \( \neg x \) unoccupied.
- For a clause \( c \), the dedicated \( h_c \) and \( g_c \) are matched to a home that is left unoccupied when the variables are assigned truth values meaning that it corresponds to a true literal satisfying \( c \).
- One more detail needs to be settled: there are more homes \( H \) than boys \( B \) and girls \( G \) (but \( |B| = |G| \)).
- Add \( l = |H| - |B| \) new boys and \( l \) new girls. The \( i \)th such girl participates in \( |H| \) triples with the \( i \)th boy and every home.
- Now a tripartite matching exists iff the set of clauses is satisfiable.

**Classifications obtained by generalization**

**Corollary.** SET COVERING, SET PACKING, and EXACT COVER BY 3-SETS are all \( \mathbf{NP} \)-complete.

- TRIPARTITE MATCHING can be reduced to EXACT COVER BY 3-SETS by noticing that it is a special case where \( U \) is partitioned in three sets \( B, G, H \) with \( |B| = |G| = |H| \) and \( F = \{ (b, g, h) \mid (b, g, h) \in T \} \).
- EXACT COVER BY 3-SETS can be reduced to SET COVERING as a special case where the universe has \( 3m \) elements, all sets in \( F \) have 3 elements and the budget \( B = m \).
- EXACT COVER BY 3-SETS can be reduced to SET PACKING as a special case where the universe has \( 3m \) elements, all sets in \( F \) have 3 elements and limit \( K = m \).

**Other problems involving sets**

1. **SET COVERING:**  
   Instance: A family \( F = \{ S_1, \ldots, S_n \} \) of subsets of a finite set \( U \) and an integer \( B \).  
   Question: Is there a set of \( B \) sets in \( F \) whose union is \( U \)?

2. **SET PACKING:**  
   Instance: A family \( F = \{ S_1, \ldots, S_n \} \) of subsets of a finite set \( U \) and an integer \( K \).  
   Question: Is there a set of \( K \) pairwise disjoint sets in \( F \)?

3. **EXACT COVER BY 3-SETS:**  
   Instance: A family \( F = \{ S_1, \ldots, S_n \} \) of subsets of a finite set \( U \) such that \( |U| = 3m \) for some integer \( m \) and for all \( i \) \( |S_i| = 3 \).  
   Question: Is there a set of \( m \) sets in \( F \) that are disjoint and have \( U \) as their union?

**A number problem: INTEGER PROGRAMMING**

**Instance:** a system of linear inequalities with integer coefficients.  
**Question:** Is there an integer solution of the system?

**Corollary.** INTEGER PROGRAMMING is \( \mathbf{NP} \)-complete.

Proof. SET COVERING reducible to INTEGER PROGRAMMING:

Given a family \( F = \{ S_1, \ldots, S_n \} \) of subsets of a finite set \( U = \{ u_1, \ldots, u_n \} \) and an integer \( B \), construct a system:

\[
\begin{align*}
0 \leq x_1 & \leq 1, \ldots, 0 \leq x_n & \leq 1 \\
0 & \leq a_{11} x_1 + \cdots + a_{1n} x_n & \geq 1 \\
& \vdots \\
0 & \leq a_{m1} x_1 + \cdots + a_{mn} x_n & \geq 1
\end{align*}
\]

where \( a_{ij} = 1 \) if \( j \)th element of \( U \) is in the set \( S_j \), otherwise \( a_{ij} = 0 \). (The idea: \( x_i = 1 \) if \( S_i \) in the cover and otherwise \( x_i = 0 \).)
Further problems involving numbers

1. LINEAR PROGRAMMING (i.e. INTEGER PROGRAMMING where non-integer solutions are allowed) is in P.

2. KNAPSACK:
   INSTANCE: A set of $n$ items with each item $i$ having a value $v_i$ and a weight $w_i$ (both positive integers) and integers $W$ and $K$.
   QUESTION: Is there a subset $S$ of the items such that $\sum_{i \in S} w_i \leq W$ but $\sum_{i \in S} v_i \geq K$?

Theorem. KNAPSACK is NP-complete.

Proof. We show that a simple special case of KNAPSACK is NP-complete. Set $v_i = w_i$ for all $i$ and $W = K$:

INSTANCE: A set of integers $w_1, \ldots, w_n$ and an integer $K$.

QUESTION: Is there a subset $S$ of the integers with $\sum_{i \in S} w_i = K$?

Reduction from Exact Cover by 3-sets

The reduction is based on the set $U = \{1, 2, \ldots, 3m\}$ and the sets $S_1, \ldots, S_n$ given as bit vectors $\{0, 1\}^m$ and $K = 2^{3m} - 1$. Then the task is to find a subset of bit vectors that sum to $K$.

$$\begin{array}{cccccc}
0 & 0 & \ldots & 0 & 0 \\
1 & 0 & \ldots & 0 & 0 \\
\vdots & & & & \\
0 & 0 & \ldots & 1 & 1 \\
1 & 1 & \ldots & 1 & 1 \\
\end{array}$$

- This does not quite work because of the carry bit, but the problem can be circumvented by using $n+1$ as the base rather than 2.
- Now $S_i$ corresponds to $w_i = \sum_{j \in S_i} (n+1)^{3m-j}$.
- Then a set of these integers $w_i$ adds up to $K = \sum_{j=0}^{3m-1} (n+1)^j$ iff there is an exact cover among $\{S_1, S_2, \ldots, S_n\}$. \(\square\)

6. Pseudopolynomial Algorithms

Proposition. Any instance of KNAPSACK can be solved in $O(nW)$ time where $n$ is the number of items and $W$ is the weight limit.

Proof.

- Define $V(w,i)$: the largest value attainable by selecting some among the first $i$ items so that their total weight is exactly $w$.
- Each $V(w,i)$ with $w = 1, \ldots, W$ and $i = 1, \ldots, n$ can be computed by $V(w,i+1) = \max\{V(w,i), v_i + V(w-w_i, i)\}$ where $V(w,0) = 0$ for all $w$ and $V(w,i) = -\infty$ if $w \leq 0$.
- For each entry this can be done in constant number of steps and there are $nW$ entries. Hence, the algorithm runs in $O(nW)$ time.
- An instance is answered “yes” iff there is an entry $V(w,i) \geq K$.

Strong NP-completeness

- The preceding algorithm is not polynomial w.r.t. the length of the input (which is $O(n \log W)$) but exponential ($W = 2^{\log W}$).
- An algorithm where the time bound is polynomial in the integers in the input (not their logarithms) is called pseudopolynomial.
- A problem is called strongly NP-complete if the problem remains NP-complete even if any instance of length $n$ is restricted to contain integers of size at most $p(n)$, for a polynomial $p$.
- Strongly NP-complete problems cannot have pseudopolynomial algorithms (unless $P = NP$).
- SAT, MAX CUT, TSP(D), HAMILTON PATH, … are strongly NP-complete but KNAPSACK is not.
**Yet another number problem: BIN PACKING**

**INSTANCE:** $N$ positive integers $a_1, \ldots, a_N$ (items) and integers $C$ (capacity) and $B$ (number of bins).

**QUESTION:** Is there a partition of the numbers into $B$ subsets such that for each subset $S$, $\Sigma_{i \in S} a_i \leq C$?

- BIN PACKING is strongly \textbf{NP}-complete:
  
  Even if the integers are restricted to have polynomial values (w.r.t. the length of input), BIN PACKING remains \textbf{NP}-complete.

- Any pseudopolynomial algorithm for BIN PACKING would yield a polynomial algorithm for all problems in \textbf{NP} implying $P = \text{NP}$.

---

**Learning Objectives**

- The concept of \textbf{NP}-completeness and its characterizations in terms of succinct certificates.

- You should know basic techniques to prove problems \textbf{NP}-complete and be able to construct such proofs on your own.

- A basic repertoire of \textbf{NP}-complete problems (related with satisfiability, graphs, sets, and numbers) to be used in further \textbf{NP}-completeness proofs.

- The definition of strong \textbf{NP}-completeness and awareness of number problems which are (not) strongly \textbf{NP}-complete.