# SLOa 2024/2025L - Homework #2

Author: Bc. Tomáš Milostný

---

## Task #1

### 1. MOD-PARTITION $\in$ NP

Given an instance $(S, v, k)$ and certificate $A \subseteq S$ the Turing machine needs to:

1. Compute given sums
$$X = \sum_{i \in A} v(i) \bmod k, \quad Y = \sum_{i \in S \setminus A} v(i) \bmod k.$$
2. Check whether $X = Y$.

This nondeterministic machine can guess $A$ and verify that all sums and modular operations in polynomial time. Therefore MOD-PARTITION $\in$ NP.

---

### 2. MOD-PARTITION is NP-hard

We can create a polynomial-time reduction from the known NP-complete PARTITION problem which is defined as:

> **PARTITION**
> **Input:** A finite set $S$, weights $v : S \to \mathbb{N}$.
> **Output:** Is there $A \subseteq S$ such that
> $\sum_{i \in A} v(i) = \sum_{i \in S \setminus A} v(i)$?

Let $(S, v)$ be an instance of PARTITION. Then we can create a variable $T = \sum_{i \in S} v(i)$ and define the MOD-PARTITION instance as $(S, v,\ k = T + 1)$.

**2.1. Correctness of the reduction** For any $A \subseteq S$, since $0 \leq \sum_{i \in A} v(i) \leq T$,

$$\sum_{i \in A} v(i) \bmod (T+1) = \sum_{i \in A} v(i), \quad \sum_{i \in S \setminus A} v(i) \bmod (T+1) = T - \sum_{i \in A} v(i).$$

Hence

$$\sum_{i \in A} v(i) \bmod k = \sum_{i \in S \setminus A} v(i) \bmod k \iff \sum_{i \in A} v(i) = T - \sum_{i \in A} v(i) \iff \sum_{i \in A} v(i) = \tfrac{T}{2},$$

which is exactly the PARTITION condition.

### 2.2. Polynomial-time computability

- Computing $T = \sum_i v(i)$ takes $O(|S|)$ additions on bit-strings of length at most $\max_i \log v(i)$.

- Writing down $k = T + 1$ likewise costs only polynomial time in the input size.
  Thus the mapping $(S, v) \mapsto (S, v, T + 1)$ is a polynomial-time reduction.

Because PARTITION is NP-complete, and we have shown

$$(S, v) \in \text{PARTITION} \quad \Longleftrightarrow \quad (S, v, T + 1) \in \text{MOD-PARTITION},$$

it follows that MOD-PARTITION is NP-hard.

---

### 3. Conclusion

In part 1, we have shown that the given problem MOD-PARTITION $\in$ NP and is also NP-hard via reduction from PARTITION in part 2. Therefore **MOD-PARTITION is NP-complete**.

---

## Task #2

### 1. $L_t \in NP$?

A nondeterministic machine can simply check "is the input exactly the one-bit string '0'?" in one step. So, yes $L_t \in NP$.

### 2. Assume $P = NP$

Then every $A \in NP$ has a deterministic polynomial-time decider $D_A(x)$ that accepts exactly those $x \in A$.

### 3. Construct the reduction

For an $A \in NP$, we must create a function computable in polynomial time with the following signature:

$$f : \{0,1\}^* \longrightarrow \{0,1\}^*$$

such that

$$x \in A \iff f(x) \in L_t = \{\,0\}.$$

This function can be defined as:

$$f(x) \;=\; \begin{cases} 0, & \text{if } D_A(x) \text{ accepts } (x \in A), \\ 1, & \text{if } D_A(x) \text{ rejects } (x \notin A). \end{cases}$$

Since $D_A$ runs in polynomial time based on the assumption that $P = NP$, computing $f(x)$ also concludes in polynomial time.

### 4. Conclusion

We have shown that $L_t \in NP$ and $\forall\, A \in NP :\; A \leq_p L_t$. So, $L_t$ is also NP-hard. Therefore under the assumption $P = NP$, $L_t$ **is NP-complete**.

---

## Task #3

Once we assume $P = NP$, *any* NP language can be decided in polynomial time and we can always create a decision procedure to map inputs to either "0" or "1," so that membership in the original language is exactly captured by landing in the $L_t = \{0\}$. It is therefore implied that we can do this for any language in NP, making it NP-hard. And fullfilling both of these conditions makes any such language also NP-complete.

---