In this TP we consider a compression function h that takes an n-bit message block m and an index i, and returns an n-bit string:

$$h: \mathbb{N} \times \{0,1\}^n \to \{0,1\}^n \ .$$

The function h is defined in the file `tp4_code.py`, where $n := 48$. For compactness, both input and output values are encoded as python integers.

For a fixed integer k, we use h to define a k-block hash function as follows:

$$\begin{cases} H \ : \quad (\{0,1\}^n)^k & \to \quad \{0,1\}^n \\ \quad m_0, \ldots, m_{k-1} & \mapsto \quad \left(\bigoplus_{i=0}^{k-1} h(i, m_i)\right) \end{cases} \tag{1}$$

k-Collisions

Let c, m be two integers. Let L_i, L_j be two lists of pairs $(h(i,x), x)$ and $(h(j,y), y)$ respectively, which are sorted using lexicographic ordering of the bit-strings $h(i,x)$ and $h(j,y)$ (LSBs first). In the file `tp4_code.py` we provide several functions to facilitate the manipulation of integers instead of bit-strings:

- Of course, the ^ operator on integers takes the XOR of the bit-strings;
- The function `lower(x,y,c)` returns True iff the value of the c LSBs of x is strictly lower (in lexicographic ordering) than the one of y;
- The function `eq(x,y,c)` returns True iff they are equal;
- The function `sort(l)` sorts a list of integer tuples according to the lexicographic ordering of the bit-string of the first value.

If the c LSBs of a bit-string / integer x are zero, we write $x|_c = 0$.

Let $L_i \bowtie_c L_j$ be the set of tuples:

$$L_i \bowtie_c L_j = \{(h(i,x) \oplus h(j,y), x, y), x \in L_i, y \in L_j, (h(i,x) \oplus h(j,y))|_c = 0\} \ .$$

also represented as a *sorted list.* The list $L_i \bowtie_c L_j$ will be called "merged list". In other words, we are computing a list of *partial collision* pairs, where the c LSBs of the pairs collide.

Question 1. *Show that there exists an algorithm of complexity:* $\max(|L_i|, |L_j|, |L_i \bowtie_c L_j|)$ *that, on input L_i, L_j, returns $L_i \bowtie_c L_j$.*

Question 2. *Implement this algorithm in a general setting, when the two lists are lists of tuples of integers, lexicographically sorted according to the first value (see the description in* `tp4_code.py`*).*

Question 3. *Fix $i, j = 1, 2$ and $c = n/3$. Asymptotically, what is the expected size of $L_1 \bowtie_c L_2$?*

Question 4. *Let $k = 4$. Find an algorithm of complexity $\mathcal{O}(2^{n/3})$ that finds a preimage of 0 by H. Implement this attack.*

Question 5. *Can we improve this complexity? How does it depend on the value of k?*