Compression Algorithm:

Input: A string \( x = x_1 x_2 \cdots x_n \) of symbols over the alphabet \( A = \{ a_0, a_1, \ldots, a_m \} \). That is, \( x_i \in A \) for all \( i \).

Output: A sequence of nonnegative integers (encoded as bit strings).

For the sake of convenience, we assume that \( a_0 \) is a special, end-of-string symbol that appears as the last symbol in \( x \) but nowhere else in \( x \).

\[ T := \text{trie consisting of single node}; \]
FOR each \( i \) in 0..\( m - 1 \) LOOP
\( \quad \) from root of \( T \), draw edge labeled \( a_i \) to new node labeled \( i \);
END LOOP;

\( i := 1; k := m; \)
WHILE \( i \neq n + 1 \) LOOP
\( \quad u := \text{root node of } T; \)
\( \quad \) WHILE there exists an edge labeled \( x_i \) leaving \( u \) LOOP
\( \quad \quad u := \text{node reached by following edge labeled } x_i \text{ leaving } u; \)
\( \quad \quad i := i + 1; \)
END LOOP;
\( \quad \) emit label of \( u \); // using \( \lfloor \log k \rfloor \) bits
\( \quad \) from \( u \), draw an edge labeled \( x_i \) to a new node labeled \( k \);
\( \quad k := k + 1; \)
END LOOP;

The corresponding decompression algorithm is on the next page.
Decompression Algorithm:

**Input:** A bit string \( b = b_1b_2 \cdots b_m \) (representing a sequence of nonnegative integers) and an alphabet \( A = \{a_0, a_1, \ldots, a_{m-1}\} \), where, for each \( i \) satisfying \( 0 \leq i < m \), it is understood that the first occurrence of \( i \) in \( b \) (when \( b \) is viewed as a sequence of nonnegative integers) encodes \( a_i \).

**Output:** A string over the alphabet \( A \).

\[
T := \text{trie consisting of single node;}
\]
\[
\text{FOR each } i \text{ in } 0..m - 1 \text{ LOOP}
\]
\[
\text{from root of } T, \text{ draw edge labeled } a_i \text{ to new node labeled } i;
\]
\[
\text{END LOOP;}
\]
\[
i := 1; \quad k := m; \quad \text{prev} j := -1;
\]
\[
\text{WHILE } i \neq n + 1 \text{ LOOP}
\]
\[
j := \text{integer represented by } b_i b_{i+1} \cdots b_{i-1+\lceil g k \rceil};
\]
\[
\text{IF } j < k \text{ THEN}
\]
\[
s := \text{string spelled out on path in } T \text{ from root to node } j;
\]
\[
a := \text{first symbol of } s;
\]
\[
\text{emit } s;
\]
\[
\text{ELSE} \quad / / \text{ necessarily } j = k
\]
\[
s := \text{string spelled out on path in } T \text{ from root to node } \text{prev} j;
\]
\[
a := \text{first symbol of } s;
\]
\[
\text{emit } sa;
\]
\[
\text{END IF;}
\]
\[
\text{IF } \text{prev} j \neq -1 \text{ THEN} \quad / / \text{ if this is not 1st iteration}
\]
\[
\text{from node } \text{prev} j, \text{ draw edge labeled } a \text{ to new node labeled } k;
\]
\[
\text{END IF;}
\]
\[
\text{prev} j := j;
\]
\[
k := k + 1;
\]
\[
i := i + \lceil g k \rceil;
\]
\[
\text{END LOOP;}
\]