Signature compression. Recall that the first step in Signature Sort was to reduce each \( w = \lg 2^{1+\varepsilon} n \) bit input word to a \( O(\lg^{1+\varepsilon} n) \)-bit signature. We achieved this by dividing each word into \( k = \lg^{1+\varepsilon} n \) chunks of \( w/k = \lg^2 n \) bits each and applying a hash function to each chunk, reducing it to \( \lg n \) bits. After hashing, our word was of the form
\[
0^{w/k-\lg n} h_1 0^{w/k-\lg n} h_2 \cdots 0^{w/k-\lg n} h_k,
\]
where each chunk hash \( h_i \) is \( \lg n \) bits long. Describe how to compress this so that all of the hashed chunks are adjacent in the least significant bits of the word, so that it has the form needed by the algorithm:
\[
0^{w-k\lg n} h_1 h_2 \cdots h_k.
\]
Your algorithm should take \( O(1) \) time using the word-RAM operations (\(+, -, *, /, \%, &, \|, \sim, ^\wedge, <<, >>\)). You may assume that \( \varepsilon < 1 \).

Level ancestors construction. In class we presented a data structure for answering level ancestor queries in \( O(1) \) time using \( O(n) \) space. Develop and analyze an algorithm for constructing this data structure in \( O(n) \) time.