(2 points) 1. Consider a generalization of the RMQ problem in which, given an array $A[1..n]$ consisting of distinct integers, we want to construct a structure capable of finding the position of the minimum and the maximum in any range $A[i..j]$ without accessing the original array. Show an encoding consisting of $3n + o(n)$ bits that is capable of answering any such query (note that we don't care about the query time, but there is a way to keep it constant).

(2 points) 2. Recall that in the lecture we have seen a structure for answering RMQ queries in constant time that uses $2n + o(n)$ **additional** bits. Unfortunately, the structure needs to access the original array $A[1..n]$. This can be overcome by replacing Cartesian Tree with 2D Min-Heaps. We define $PSV(i) = \max\{j < i : A[j] < A[i]\}$ (assume that $A[0] = -\infty$).

   (a) Prove that PSV has a "crossing-free" property, meaning that for any $i < j$ we have $PSV(j) \notin [PSV(i) + 1, i)$.

   (b) We define a tree on nodes labeled with $0, 1, \ldots, n$ corresponding to the entries of A. The parent of node $i > 0$ is the node $PSV(i)$, and the children of every node are arranged to be increasing from left to right. Draw the tree for the following array:

$$A = [20, 4, 15, 16, 11, 3, 7, 18, 9, 5, 13, 8, 6, 12, 2, 10, 1, 17, 14, 19].$$

   (c) Show that labels of nodes correspond to their preorder numbers, the entry corresponding to a node is larger than the entry corresponding to its parent and right sibling.

   (d) Explain (with a proof) how to translate RMQ on A query into LCA query on its 2d Min-Heap.

(extra) (e) Do you see why a 2D Min-Heap is easier to use for designing a succinct RMQ structure than a Cartesian Tree?

3. Recall the definition of zero$^{\text{th}}$ order empirical entropy H_0 of an array $S[1..n]$. Show that Huffman encoding results in a code of total length from $[nH_0, n(H_0 + 1))$.

4. Consider an increasing list of n numbers from $\{1, 2, \ldots, U\}$. Show how to encode such a list in $\mathcal{O}(n \log(U/n))$ bits, and that this is asymptotically optimal.

5. We change the definition of the Burrows-Wheeler transform of s (in fact, this is the original definition): instead of appending the special character, we sort all cyclic rotations of the original s and output the last column denoted $bwt(s)$ together with a single number denoting the position of s in the sorted array. Observe that $bwt(\mathtt{babaabaaaa}) = \mathtt{babaabaaaa}$. Characterize all binary words s containing at most two letters b such that $bwt(s) = s$.