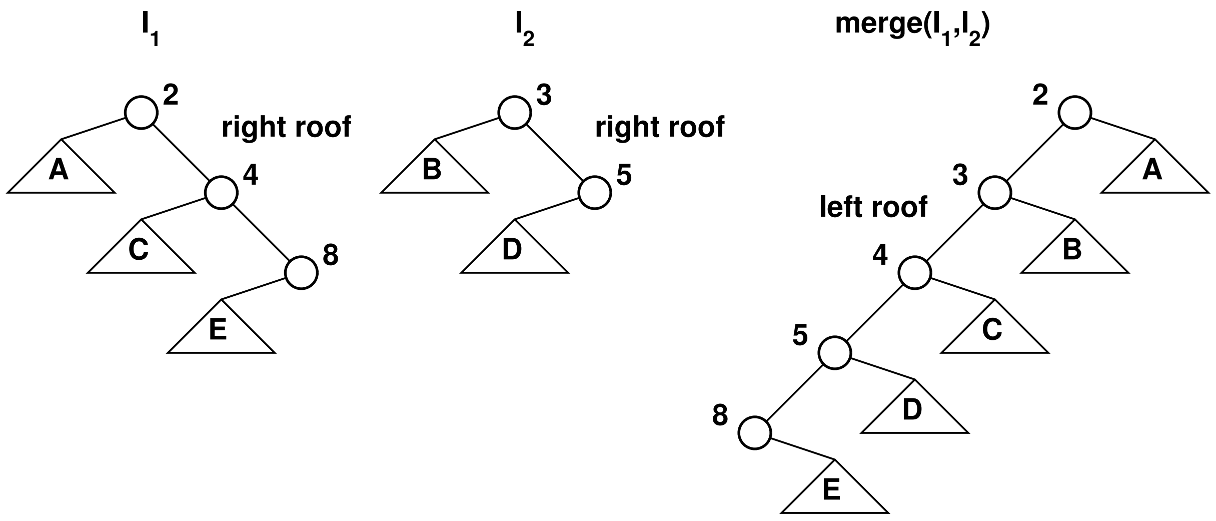
Assignment 6, COMP 252, Winter 2020. Apr 9, 2020, 10am. Due Apr 15, 2020, 10am.

Submit to myCourses.

Exercise 1. AMORTIZED ANALYSIS AND PRIORITY QUEUES. Consider the following data structure for handling the operations `insert`, `deletemin` and `merge`, which we will call an iceberg. The iceberg is nothing but a heap-ordered binary tree with the minimum on top. Subtree sizes are stored and maintained with each node. The `merge` operation is simple and atomic. We call the right roof of the binary tree the path from the root obtained by following right links. The left roof is defined in a similar fashion. Given two icebergs, the two right roofs are merged as in a standard merge of ordered lists. For every node on the new right roof, the left and right child pointers are exchanged—the right roof thus becomes the left roof. The figure below illustrates the `merge` operation. We write `I` to denote an iceberg: this means that `I` denotes a pointer to the root of the iceberg.



(i) How would you implement `insert (k,I)`, where `k` is a new key, and `I` is an iceberg? Similarly, how would you perform `deletemin (I)`?

(ii) We define a white node in an iceberg `I` as a node whose right subtree is greater (in size) than its left subtree. The other nodes are black. The potential Φ of a collection of icebergs is defined as the total number of white nodes. (So, consider white nodes as "bad".) With this definition, show that the amortized time of `merge (I1, I2)` is not more than twice the number of black nodes in the right roofs of `I1` and `I2`.

(iii) Prove that the number of black nodes in the right roof of an iceberg `I` does not exceed $1+\log_2(n)$, where n is the size of `I`.

(iv) Conclude that the amortized time for a `merge` operation is $O(\log_2(n))$, where n is the size of the merged iceberg.

(v) Show that if you start with an empty structure, and apply t operations `insert` and `deletemin` to an iceberg (in any order), such that the maximum size of `I` is never more than n, then the total time taken is not more than $O(t \log_2(n))$.

(vi) Show how to create an iceberg of n elements from an unordered list of n keys using not more than $O(n)$ time.

Exercise 2. COMPRESSION. We have a binary file of length n, where each bit independently is one with probability $1/3$ and zero else.

(i) What is the binary entropy of the input source?

(ii) Give Shannon's lower bound for the expected length of a compressed sequence.

(iii) Let us set ourselves the goal of compressing the input sequence such that the expected length of the compressed sequence is less than 10 percent more than Shannon's lower bound (for all n large enough). This can be done in multiple ways, but we would like something simple and do not want to use the Lempel-Ziv method. Develop a simple home-made method, and prove that you are within the 10 percent margin.

(iv) Repeat (iii) with a 3 percent margin. Include an algorithm and a proof.

Exercise 3. GRAPH ALGORITHMS. Consider an acyclic directed graph $G = (V, E)$ given in adjacency list format, where $V = \{1, 2, \ldots, n\}$. For two nodes $x, y \in V$ for which there is a path from x to y, we define the longest path distance $L(x, y)$ as the maximal path distance taken over all paths that lead from x to y. A node is said to be a root if its in-degree is zero, and is a leaf if its out-degree is zero. We define the span of the graph as $\max L(x, y)$, where the maximum is taken over all roots x and all leaves y for which there is a path from x to y. (To check your understanding, verify that the span of a graph without edges is zero.) Develop a simple linear time ($O(|E| + |V|)$ time) algorithm for computing the span of G. Points for exactness, elegance, and readability.