1. Consider two asymptotically positive functions $f_1(n)$ and $f_2(n)$. Assume that $f_1(n) = o(f_2(n))$, or equivalently, $f_2(n) = \omega(f_1(n))$. Compare the functions $f_1(f_2(n))$ and $f_2(f_1(n))$ asymptotically. Can you make a universal statement on which grows faster, or whether they are asymptotically equivalent? If not, then give examples of all the three possibilities. Generalise your examples by a single statement summarising the observations.

2. (a) Analyse the running time of your algorithm for computing a treap (in the earlier assignment) from a given set of input data, each with an ordered pair of key elements.
(b) Try and modify the algorithm to improve the running time.
(c) Also incorporate functionality to insert or delete records dynamically into the data-structure. Give efficient algorithms for these requirements and analyse the performance.

3. (a) Binary Search is an algorithm which takes as input an array of $n$ elements, in sorted order and a value, and returns the index of the array whose key value matches the input value, if one exists. It works by comparing the input value with the element in the middle of the array, and if it is not a match, then if the key value is larger than the array entry, it searches in the right half of the array, otherwise it searches in the left half, recursively. Write the recurrence relation for the running time of this algorithm, and solve the recurrence.
(b) Now consider a 2-dimensional $n \times n$ array. In each row the elements are in increasing order, and in each column also the ele-
ments are in increasing order. Develop an algorithm similar to binary search, and argue that it is correct. Write an expression for its running time as a recurrence relation and solve the recurrence to get an estimate on the time required for this search algorithm.

(c) Generalise this idea to a k-dimensional array, with size n on each dimension.

(d) How long does it take given a set of n numbers, in arbitrary (unsorted) order to place in these data-structures (2 or higher dimensional arrays) satisfying the requirements of relative values? Is this better or worse than sorting the elements into a long single dimensional array, asymptotically and in absolute terms?

4. A majority element of a multiset of m elements is any element which occurs at least \( \lceil \frac{m}{2} \rceil \) times. Does every multiset have a majority element? How many majority elements can a multiset have? If a given multiset has a majority element, show that it can be found in \( O(m) \) time.