1. Sort the following set of functions in increasing order of their asymptotic growth rates. State clearly the partial order, as some subsets of these functions may have identical asymptotic growth rates.

<table>
<thead>
<tr>
<th>log(log* n)</th>
<th>2log*n</th>
<th>(√2)log n</th>
<th>n^2</th>
<th>n!</th>
<th>(log n)!</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3/7)^n</td>
<td>n^3</td>
<td>log^2 n</td>
<td>log (n!)</td>
<td>2^n</td>
<td>n log n</td>
</tr>
<tr>
<td>ln ln n</td>
<td>log* n</td>
<td>n^2^n</td>
<td>n log log n</td>
<td>ln n</td>
<td>1</td>
</tr>
<tr>
<td>2log n</td>
<td>(log n)^log n</td>
<td>e^n</td>
<td>4 log n</td>
<td>(n + 1)!</td>
<td>√log n</td>
</tr>
<tr>
<td>log* log n</td>
<td>2√2log n</td>
<td>n</td>
<td>2^n</td>
<td>n log n</td>
<td>2^{n+1}</td>
</tr>
</tbody>
</table>

2. Solve the following recurrence equations:

(a) $T(n) = 3T\left(\frac{3n}{4}\right) + \frac{1}{n}$
(b) $T(n) = T(\sqrt{n}) + \log n$
(c) $T(n) = T(n - 1) + 19$
(d) $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \sqrt{n}$
3. A binary search tree is a data structure consisting of an underlying binary tree and each node has a key value from a totally ordered set. The key values are such that the key of any node is greater than or equal to the key of any node in its left subtree and less than or equal to the key of any node in its right subtree. A binary heap is a binary tree with each node having an associated key value from a totally ordered set. In this case the key of any node is greater than or equal to the key of its children. These data structures are useful, because they improve efficiency of standard operations, as compared to the simple array representations. Consider a set of data records with two different fields. These records need to be stored, one each in the nodes of a binary tree, such that when the records are projected to the first field, it is a binary search tree and when projected to the second field, it is a binary heap.

(a) Prove that there exists such a binary tree for any set of $n$ records and describe the way you might construct the data structure given a set of $n$ such records.

(b) Write an algorithm to construct a treap given a set of pairs of values from a totally ordered set.

(c) Implement a C-program to construct a treap given such an input.
4. (a) Name any optimal (in the worst-case) sorting algorithm and state its asymptotic running time.

(b) Now consider a set of \( n \) numbers stored as a sequence in an array \( A \), which needs to be sorted. One possibility is to sort the first \( f \) fraction of the sequence, and then sort the last \( f \) fraction of the sequence and then to sort the first \( f \) fraction again, in each case using the algorithm you stated above as a subroutine to sort the selected fractional subarray. What is the smallest value of \( f \) for which this procedure will correctly sort any input sequence? What is the running time of this modified sorting procedure in terms of \( f \) and the running time expression you wrote earlier (for the subroutine)? Solve it and express your answer in closed form asymptotic notation.

(c) Consider an extended version of the above procedure, where you repeat the alternating procedures of sorting the first and last \( f \) fraction of the array until the sequence is sorted. What is the smallest value of \( f \), for which this procedure will eventually sort the sequence. Compute the number of iterations required by your algorithm as a function of \( f \).

(d) Is the running time of these sorting algorithms smaller or larger, asymptotically, than the standard algorithms? If it is larger, then suggest some reason why this algorithm might be useful.

---

1. The photocopier machine in faculty block 4 has been malfunctioning for a while now. Anyone displaying extra-ordinary prowess in copying, during the course of this assignment, will be recommended for a full time position in that block! You may discuss, but finally write down the solutions yourselves. The credit for this (and future) assignments will be on the basis of a viva also, to be held immediately after the submission of the assignment is due and not at the end of the semester.