Data Structures (IT205)
July-December, 2015
First Midsemester-semester Exam

10^{th} September, 2015

Time: 2 hours

marks: 80

This exam is open book and open notes. However, you may use only material brought in by yourself. Exchange of material (notes/ textbook) between students during the exam is not allowed.
This paper consists of four questions printed on 2 back-to-back pages on a single paper. Please cross-check that your question paper is complete. Each question is for 20 marks.

1. (a) Consider a sorted array with 31 elements, all distinct. Now suppose you were to rearrange this array into an array B according to the following condition: In B, the elements will appear in the increasing order of their search time by binary search in A. For elements that take the same number of steps to search for by binary search in A, they are ordered in increasing order of their value (i.e. the same order in which they appear in A). Compute the number of inversions in Array B.

   (b) Derive a general formula for an identical problem where the size of array is of the form $2^k - 1$ for any positive integer k. (Part (a) is for $k = 5$).

   (c) Suppose it is (wrongly) assumed that B is sorted, how many elements will go undetected by binary search on B (despite the fact that they are present in the array). Solve this for the sepcial case with 31 elements, and also generalise to any integer of the form $2^k - 1$.

2. (a) Consider three stacks A, B, C of capacity $3k$ elements each. Also suppose each of them initially has $2k$ elements on them, and that all elements are distinct. One can infer that there is vacancy for exactly k more elements in each stack. The total number of elements in the three stacks of total capacity $9k$ is $6k$. Show that, without using any extra space, and just a sequence of pops and pushes between these three stacks, any arrangement of these $6k$ elements into three stacks of $2k$ elements each, can be reached. (Capacity of the stacks should not be exceeded at any point of time).

   (b) Explain briefly why the choice of $3k$ capacity and $2k$ elements in each stack is critical (i.e if there is more than $2k$ elements in each and the capacity is $3k$, then some configurations cannot be reached.

3. Consider an array A of size 2^k containing a permutation of the elements $1, \dots, 2^k$. Suppose we run merge sort on such an input.

   (a) What is the fewest number of pairs of elements which are compared during the course of merge sort? Give an example configuration achieving this minimum.

   (b) What is the largest number of pairs of elements which are compared during the course of merge sort? Give a configuration achieving this maximum.

4. Suppose there is a binary max heap on $2^k - 1$ nodes with distinct keys.

   (a) Without reading key values explicitly, describe how you can find a sequence of k elements which are in decreasing order.

   (b) There are several such sequences of k elements, so describe one such that the residual set upon ignoring these elements should be $k-1$ heaps of sizes $2^{k-1}-1, 2^{k-2}-1, \dots, 1$. Each of these disjoint heaps can be split into monotonic sequences as computed in part (a). Write a recursive formula to compute the number of such monotonic sequences into which a heap can be split. Solve this to obtain a number of monotic decreasing blocks into which the elements of a heap can be broken.

   For example if we take a heap with three elements then we can break it into two blocks as Block 1: $A[1], A[2]$ and Block 2: $A[3]$. If we take a heap with seven elements then we can break it into four blocks as Block 1: $A[1], A[2], A[4]$, Block 2: $A[3], A[6]$, Block 3: $A[7]$ and Block 4: $A[5]$. Here, Block 1 is from the original monotonic sequence, Blocks 2 and 3 are from the recursive call to a 3 element heap and block 4 is from a recursive call to a 1 element heap.