CS 473: Fundamental Algorithms, Spring 2011

HW 3

Homework is due by **Monday, 23:59:59, February 14**

Problem 1 is due by **Sunday, 23:59:59, February 13**

This homework contains four problems. **Read the instructions for submitting homework on the course webpage**.

Collaboration Policy: For this homework, Problems 2–4 can be worked in groups of up to three students.

Problem 1 should be answered in Compass as part of the assessment HW3-Online and should be done individually.

1. (30 pts.) Short questions to be answered on compass individually.

2. (30 pts.) You are given an array A with n distinct numbers in it, and another array B of ranks $i_1 < i_2 < \ldots < i_k$. An element x of A has rank u if there are exactly $u-1$ numbers in A smaller than it. Design an algorithm that outputs the k elements in A that have the ranks $i_1, i_2, \ldots, i_k$.

   (A) (5 pts.) As a warm-up exercise describe how to solve this problem in $O(nk)$ time.

   (B) (20 pts.) Describe a $O(n \log k)$ recursive algorithm for this problem. Prove the bound on the running time of the algorithm.

   (C) (5 pts.) Show, that if this problem can be solved in $T(n,k)$ time, then one can sort n numbers in $O(n + T(n,n))$ time (i.e., give a reduction). Provide a strong intuitive reason why the above problem can not be solved in time faster than $O(n \log k)$.

3. (20 pts.) Nowadays, one of the main techniques in handling huge computational tasks, is breaking them down, and handling each chunk separately in a different computer in a cluster. As an example, consider merge sort, where you split the array into k equal size sub-arrays, sort each of them recursively (or by sending them over to some other computer to handle this chunk). Finally, you need to merge these k sorted arrays into a single sorted array. We will refer to this sorting algorithm as k-merge sort.

   (A) (5 pts.) Describe two different algorithms that merge k sorted arrays (of total size n) in $O(n \log k)$ time. (Hint: One of them should be recursive, and the other one should use a data-structure.)

   (B) (10 pts.) What is the running time of k-merge sort (in the regular unparallel settings) on an array of n elements, for $k = 2$, $k = \log \log n$, $k = \log n$, $k = \sqrt{n}$, and $k = n^{\varepsilon}$, where $\varepsilon \in (0,1)$ is some constant. For each of these cases, state the recurrence on the running time, the number of nodes in the recurrence tree at level i, the depth of the recurrent tree, the total contribution to the running time of nodes at level i.

   (C) (5 pts.) Assume that there are infinitely many processors, and each recursive call is processed on a distinct single processor. We ignore the communication cost between processors. For example, in the parallel implementation of merge sort, both recursive calls are being performed simultaneously in parallel. Then in this case, what is the running time of the recursive k-merge sort algorithm that you designed in (A)?

4. (20 pts.)

   A French mathematician decides to invent a new variant of the Towers of Hanoi game, known in the US literature as the "Freedom Towers of Hanoi" game. Here, there are n disks placed on the first peg, and there are $k \geq 3$ pegs, numbered from 1 to k. You are allowed to move disks only on adjacent pegs (i.e., for peg i to peg $i + 1$, or vice versa). Naturally, you are not allowed to put a bigger disk on a smaller disk. Your mission is to move all the disks from the first peg to the last peg.

   (A) (7 pts.) Describe a recursive algorithm for the case $k = 3$. How many moves does your algorithm do?

   (B) (7 pts.) Describe a recursive algorithm for the case $k = n + 1$. How many moves does your algorithm do? (The fewer, the better, naturally.)

   (C) (6 pts.) Describe a recursive algorithm for the general case. How many moves does your algorithm do? (The fewer, the better, naturally.) In particular, how many moves does your algorithm do for $n = k^2$?

   (A good solution should probably yield the same solutions offered in (A) and (B).)

   (This question is somewhat more open ended - we have no specific solution at mind.)