CS 161: Homework 3

Due by October 21, 2016 at noon

Instructions: Please answer the following questions to the best of your ability. Provide full and rigorous proofs and include all relevant calculations. When writing proofs, please strive for clarity and brevity (in that order). Please see the course website for submission instructions and collaboration policy. Cite any sources that you use.

Remember that when you design an algorithm, in addition to an appropriate description of your algorithm, you must provide sufficient explanation for its correctness and analyze its running time. See the Homework Advice document on the course website for details.

Question 1 (25 points)

Imagine that we are given two distinct arrays: one of n distinct *Goldilocks* $G[1], \ldots, G[n]$, and one of n distinct *soups* $S[1], \ldots, S[n]$. Each soup has a different temperature which is measured once objectively and does not change (all the Goldilocks have the same perception of temperature). For each Goldilocks, there is exactly one soup that is neither too hot nor too cold, but just right. Similarly, each soup is just right for exactly one Goldilocks.

The computational task is to match each Goldilocks with their correct soup. However, the only comparison allowed is "Is soup j too hot, too cold, or just right for Goldilocks i?", so we cannot compare a Goldilocks to a different Goldilocks, and we cannot compare a soup to a different soup. Design an efficient randomized algorithm to solve this problem.

Question 2 (25 points)

In this problem, we will modify the QuickSort algorithm from class to try and select a better pivot. To do this, we randomly select 3 elements from the subarray and choose the pivot to be the median of this set of 3 elements. (Recall that the original QuickSort chooses the pivot uniformly at random.)

Let A be an input array of *odd* length n where $n \geq 3$. For the purpose of this problem, we will use the same assumption that we made in class that the elements of the array A are distinct.

For $1 \leq i \leq n$, an $(i-1, n-i)$ split is the result of choosing the i^{th} order statistic as the pivot: $i-1$ elements are less than the pivot, and $n-i$ elements are greater than the pivot. A split no worse than $(i-1, n-i)$ is any split in $\{(i-1, n-i), \ldots, (n-i, i-1)\}$.

(a) (10 points) What is the probability of selecting the median element as the pivot in the original algorithm? How about in the modified algorithm? How do they compare?

(b) (5 points) What is the probability of getting a split no worse than $(i-1, n-i)$ in the original QuickSort algorithm?

(c) (10 points) What is the probability of getting a split no worse than $(i-1, n-i)$ split in our modified QuickSort algorithm?

Question 3 (25 points)

In this problem, we will work with n x n matrices of numbers in which both columns and rows are nondecreasing. For example:

$$\begin{matrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{matrix} \quad \text{and} \quad \begin{matrix} 6 & 8 & 9 \\ 2 & 4 & 7 \\ 1 & 3 & 5 \end{matrix} \quad \text{and} \quad \begin{matrix} 4 & 7 & 9 \\ 2 & 5 & 8 \\ 1 & 3 & 6 \end{matrix}$$

are all valid examples of such a matrix. Call these matrices "ordered matrices".

Suppose we are given such an ordered matrix and need to sort all its n^2 elements into a single list. Prove that the number of comparisons needed for this problem has a lower bound of $\Omega(n^2 \log n)$. (Note that this is no better than the solution that simply sorts all n^2 numbers without reference to the ordered matrix).

Hint: Consider the initial ordering of upper-left to bottom-right diagonals in ordered matrices.

Question 4 (25 points)

We define r-group sorting as follows: For an input array of distinct numbers $a_1, ..., a_n$ the output is a partition of $\{1, 2, \ldots, n\}$ into disjoint sets $B_1, ...B_r$ such that the following hold: (For simplicity, assume r divides n.)

1. $|B_i| = \frac{n}{r}$
2. $\forall k \in B_i, \forall \ell \in B_{i+1}, a_k < a_\ell$

An r-group sorting algorithm is an algorithm that takes a sequence of distinct numbers as input and produces a correct r-group sorting. We will prove lower bounds on the running time of comparison-based algorithms for r-group sorting.

(a) (5 points) Consider inputs that are permutations of n distinct numbers. One particular r-grouping solution could be correct for multiple such inputs. How many input arrays map to one fixed r-grouping solution?

(b) (10 points) Given any n distinct numbers, show that the number of such input arrays that map to a single correct $\sqrt{n}$-grouping is $O(n^{cn})$, for some $c < 1$.

(c) (10 points) Prove that any comparison-based $\sqrt{n}$-group sorting algorithm must have worst case time complexity $\Omega(n \log n)$.

Note: You are allowed to use Stirling's approximation (as it appears in CLRS) in this question.