CS21003 - Tutorial 5

August 31st, 2018

1. Let A be a sorted array of n distinct elements. An array B is formed by cyclically right-shifting the array A by some k cells. Given B (but not A or k), determine how the shift amount k can be computed in $O(logn)$ time.

2. Suppose you are given two *sorted* arrays A and B with n and m integer elements each. Provide an $O(log(max(m,n)))$ algorithm to find the median of these $n+m$ elements.

3. Assume you have an array $A[1,\ldots,n]$ of n elements. A majority element of A is any element occurring in more than $n/2$ positions. Assume that elements cannot be ordered or sorted, but can be compared for equality. Design a divide and conquer algorithm to find a majority element in A (or determine that no majority element exists) to run in $O(nlogn)$ time.

4. The maximum partial sum problem (MPS) is defined as follows. Given an array $A[1,\ldots,n]$ of integers, find values of i and j with $1 \leq i \leq j \leq n$ such that $\sum_{k=i}^{j} A[k]$ is maximized. Example: For the array $[4,-5,6,7,8,-10,5]$, the solution to MPS is $i=3$ and $j=5$ (sum 21). Can you think of a brute force solution in $O(n^2)$? How can you use divide and conquer to improve this to $O(nlogn)$?

5. Can you generalize your algorithm for the question 2 for $k-th$ order statistics of the two arrays combined, and improve the algorithm to work in $O(log(min(m,n)))$?

6. To solve problem 3, you can also use a divide and conquer strategy to work in $O(n)$ time. Describe the algorithm. **Hint:** Assume that n is even. Arbitrarily pair elements of A. In each pair (a_i, a_j), check whether $a_i = a_j$. If so, write this element in an array B. If not, discard the pair. The array B consists of at most $n/2$ elements after all pairs are considered. Recursively compute the majority of B.