CS 161 Winter 2020 Section 3

January 23-24, 2020

1 Light Bulbs and Sockets

You are given a collection of n differently sized light bulbs that have to be fit into n sockets in a dark room. You are guaranteed that there is exactly one appropriately-sized socket for each light bulb and vice versa; however, there is no way to compare two bulbs together or two sockets together as you are in the dark and can barely see! (You are, however, able to see where the sockets and light bulbs are.) You can try and fit a light bulb into a chosen socket, from which you can determine whether the light bulb's base is too large, too small, or is an exact fit for the socket. If the bulb fits exactly, it will flash once, in which case you have a correct match. (Note that the flashing light does not allow you to visually compare bulbs/sockets to other bulbs/sockets.)

Suggest a (possibly randomized) algorithm to match each light bulb to its matching socket. Your algorithm should run strictly faster than quadratic time in expectation. Give an upper bound on the worst-case runtime, then justify your algorithm's correctness and expected runtime.

2 Batch Statistics

Design an algorithm which takes as input array A consisting of n possibly very large integers as well as an array R that contains k ranks $r_0, \ldots, r_k$, which are integers in the range $\{1, \ldots, n\}$. (You may assume that $k < n$.) The algorithm should output an array B which contains the r_j-th smallest of the n integers, for every j in $\{1, \ldots, k\}$. So if an $r_j = 3$ in input array R, then we want to return the 3rd smallest element in the input array A as part of the output.

Input: A which is an unsorted array of n unbounded distinct integers; R which is an unsorted array of k distinct ranks.
Output: B which is an array containing the r-th (smallest) integer of A for every r in R.

Example:
Input: A = [11, 19, 13, 14, 16, 18, 17, 12, 15]; R = [3, 7]
Output: [17,13]
Explanation: 17 is the 7-th smallest element of A and 13 is the 3-rd smallest of A. [13,17] is also an acceptable output

We are expecting a clear yet thorough English description of your algorithm, pseudocode, and an analysis of runtime. Hint: we are looking for an $O(n \log(k))$-time algorithm.