Time Spent: 2 hours

Collaborators and Resources: Spoke with Nathaniel MacArthur-Warner on methodology

---

Problem 3

Given that each database contains n numerical values which are distinct (that is, no two values are the same), we need to find the median- that is, we need to find the nth smallest value out of the combination of the two sets. The only way to access these values is through queries to one of the two databases. That is, in a single query, you can specify a value k to one (of the two) database(s) and the chosen database will return the kth smallest value which it contains. So we specify k and the database returns the kth smallest value. I realized that because of the nature of queries it doesn't really matter that each database is not sorted. We can simply make recursive calls, each time going halfway up (or down) the remaining range of our dataset when our median is above the two-set median (compared to the median of the other dataset) (or below the two-set median). Basically, we can design a recursive algorithm which makes one recursive call on a subsection of each original database which has half the cardinality as the previous one does . . .

1. Define the function medValue(d_1, d_2, k_1, k_2, lb_{d1}, ub_{d1}, lb_{d2}, ub_{d2}):
   (a) Query the k_1^{th} smallest value in d_1. Call it q_{d1}.
   (b) Query the k_2^{th} smallest value in d_2. Call it q_{d2}.
   (c) If $q_{d1} < q_{d2}$ # That is, if (our section of) d1's median is smallest then we redefine variables in order to look at the third quartile of that section, etc. for the recursive call:
      i. Set $lb_{d1} = k_1$,
      ii. $ub_{d2} = k_2$,
      iii. $k_1 = k_1 + \frac{|ub_{d1} - k_1|}{2}$, and
      iv. $k_2 = k_2 + \frac{|k_2 - lb_{d2}|}{2}$.

(d) Otherwise, if $q_{d1} > q_{d2}$ # That is, if (our portion of) d1's median is the greatest out of the two then the true median is below that value so we look at the 1st quartile of this portion for the recursive call.

   i. Set $ub_{d1} = k_1$,

   ii. $lb_{d2} = k_2$,

   iii. $k_1 = k_1$ - $\frac{|k_1 - lb_{d1}|}{2}$, and

   iv. $k_2 = k_2 + \frac{|ub_{d2} - k_2|}{2}$.

(e) Return medValue(d_1, d_2, k_1, k_2, lb_{d1}, ub_{d1}, lb_{d2}, ub_{d2})

(f) Break when the algorithm reaches the point at which $|q_{d1} - q_{d2}|$ is at a minimum (when they reach the maximum closeness). Return the lower value out of q_{d1} and q_{d2}.

2. Given two databases $d_1 = \{v_1, v_2, ..., v_n\}$ and $d_2 = \{w_1, w_2, ..., w_n\}$

3. Initialize $k_1 = k_2 = \frac{n}{2}$

4. Initialize ub_{d1} = n, $lb_{d1} = 0$

5. Initialize ub_{d2} = n, $lb_{d2} = 0$. # These are the upper and lower "bounds" for our two subsets of interest which progressively narrow

6. Find medValue(d_1, d_2, k_1, k_2, lb_{d1}, ub_{d1}, lb_{d2}, ub_{d2}) # That is, run the algorithm with these starting values

Claim 1. The algorithm finds the median value using at most log(n) queries, that is, it uses O(log n) queries.

Proof. Within each recursive sequence, we make two queries. Additionally, we have two if statements each of which indicates that four more redefinitions of variables (upper and lower bounds, indices) are made. As a result, all of these operations are done in linear time which implies that f(n) = 6. Thus, since we are making one recursive call (implies b = 1) and doing the same operation within each call (a = 1) we know that by the master theorem T(n) $\leq$ aT($\frac{n}{b}$) + f(n) = T(n) + 6 = θ(log$n$). So we have confirmed that the runtime essentially approaches log$n$. Secondly, it seems clear because we're only checking each element once and we're not checking all of them. Furthermore, it finds the median value because by definition of the median (center) value, it's always in the center of the large set.

Thus, because we keep jumping by half of the remaining space toward the true median on each side (that is, the true median is between the median of d1 and the median of d2), we will eventually find it. Furthermore, the algorithm finds the minimum of the last two queried values which means that it will find the left median. Essentially, our algorithm will work and it will do so in fairly efficient time. □