Sample Answers: Quiz 1

Q1: We are given a function $y_i = f(x_i)$ such that the function monotonically decreases between x_1 to x_k for $k \leq n$ and then monotonically increases from x_k to x_n. This is a unimodal function with minimum at x_k. Design an efficient algorithm to find x_k using as few function evaluations as possible. Assume that the function is available as a subroutine.

Ans: Wlog we assume that all function values in the given interval are positive. We also assume that querying a function value takes $O(1)$ time. Given two points x_1 and x_n, we will replace one of these terminal points with an intermediate point x_{p1} in the interval between x_1 and x_n. Let us denote the other terminal point by $x_{p2} \in \{x_1, x_n\}$ such that the following remains true.

- x_k lies in the interval between x_{p1} and x_{p2}.
- The function retains its monotonicity property between x_{p1} to x_k and x_k to x_{p2}.

Our candidate for x_{p1} is the mid point in the given interval, $\frac{x_1+x_n}{2}$. Depending on which side x_{p1} falls on with respect to x_k, we calculate the other candidate x_{p2}. We can show that finding x_{p1} and x_{p2} in each iteration requires constant time. The main idea is that we want to pick one point along the descending slope and one point along the ascending slope. Given the mid point, we can find in constant time, which slope it belongs to. Then, x_{p2} will be the representative point along the other slope. Since, in each iteration we are reducing the length of the interval by half, we would need at most $\log n$ iterations.

Marking scheme: 10 marks if they give $\log n$ solution. $(4+3+3)$- 4 marks for idea, 3 marks for correct algorithm, 3 marks for recurrence and time complexity. 2 marks if somebody gives any other $O(n)$ solution.

Q2: Analyse the space complexity of the median-of-median selection algorithm. The input is given as a read-only array and any other space utilized must be accounted for. For your analysis, you must describe exactly how the algorithm is using space at each stage of the recursion.

Ans:

Algorithm:

$S(n)$: space complexity of the algorithm.

- Copy the elements in a $5 * n/5$ array. (wlog assume n is a multiple of 5) Space: n $S(n) = n + ..$
- Compute the median of each column, can be done using inplace sorting. Space: No extra space $S(n) = n + ..$
- Compute the median of the 3^{rd} row (i.e middle row). This is done by a recursive call. Space: Account for the space in recursive call $S(n) = n + S(n/5) + ...$
- Once the median of the medians is computed, then partition the middle row on the median and move the length 5 columns around accordingly. Space: No extra space, done inplace. $S(n) = n + S(n/5) + ...$
- We would like to form two sets L and H, comprising of elements greater and lesser than the median of median respectively. After the previous step, the North-West (NW) portion of the 2d array is lesser than the median of median, and the South-East (SE) is greater. The NE and SW remains to be checked, we check them and put them accordingly in sets L or H. Space: Atmost n extra space $S(n) = n + S(n/5) + n + ...$
- Note that in the worst case the sets L and H can be 3/4 Vs 1/4 lopsided, since we have 1/2 Vs 1/2 guarantee for the NW and SE portions. Thus we may have to recurse on the $(3/4)^{th}$ side. Space: Account for the space in the recursive call $S(n) = n + S(n/5) + n + S(3n/4)$

The final reccurrence becomes $S(n) = n + S(n/5) + n + S(3n/4)$. The solution can be verified to be $O(n)$ by induction.

Marking Scheme: Accounting for correct space in each of the intermediate steps. The solution must atleast have the correct reccurrence if not proved rigorously through induction (or any other method like the recursion tree).

Induction proof for the recurrence
Guess $S(n) = O(n)$ and verify by inducting on n.
I.H: Assume $S(k) \leq ck$ (c=constant) for $k < n$
Induction:

$$\begin{aligned} S(n) &= 2n + S(n/5) + S(3n/4) \\ &\leq 2n + cn/5 + 3cn/4 \cdots \text{by I.H} \\ &= 2n + 19cn/20 \\ &= cn - (cn/20 - 2n) \\ &\leq cn \text{ if } cn/20 - 2n \geq 0 \text{ i.e } c \geq 40 \end{aligned}$$

Base Case: We must assert $S(1) = O(1)$. So we require $S(1) = O(1) \leq c$, choosing c to be sufficiently large suffices.