Homework 3 Solutions:

1. A complete binary tree will have at most 3 neighbors (parent + children). A vertex is a local minimum if its value (label) is **lesser** than the values (labels) of its neighbors. As a result, we need **at most 4 probes** to determine this. The root has no parent hence, we start by probing the value of the current vertex and its children, if the value is smallest, return it, else probe the smaller value children node.
   This will select one out of the two children. The path from the root to a leaf (at furthest) will have to traverse to a **depth of d**. Since we do **at most 3 probes**, therefore, complexity is **3d**. As d = log n + 1, 3d is O(3log n) or O(log n).

2. The array (list) of distinct numbers is not sorted. Use any median finding algorithm, to obtain a median of an **unsorted list**, in **O(n) time**. Now, maintain a data structure of size k. Scan the list from the start till end. Insert elements into k while scanning. Keep a variable which calculates the maximum distance (by value) of elements in k, from the calculated median. Put a check and remove the element with the maximum distance from the median. This process takes O(n) time and the entire algorithm takes O(2n) or O(n) time, as required.

3. Sort any one of the two given sets. Permute the other to maintain the correspondence between the two endpoints. Sorting algorithm will take O(nlog n) time. Go over the set which is not sorted and for indices i < j, count the pair of indices such that Q[i] > Q[j]. This is because, for indices i < j, segments i and j intersect iff Q[i] > Q[j].

4. Let **mid** be the middle element of the sequence. Compare the value to the left of the mid and to the right of the mid, with mid. If the right value > mid > left value, then your new sequence starts from mid and ends at n. Calculate the mid of the new sequence again. And do the neighbor comparisons to determine the new sequence. You will eventually arrive at the peak value in O(log n) time.
   This is very similar to binary search.

5. The given formula computes the dot product between two vectors in 3n/2 multiplications. Now, consider two n*n matrices, A and B.
   For normal A * B operation, we select the first row of A and do a dot product with the first column of B. Let the first row of A be V_1, second row be V_2 and so on, and the first column of B be W_1, second row be W_2 and so on (V_1 and W_1 are two different vectors and not the first elements of vectors V and W respectively).
   The first element of the first row in the resulting matrix C, will be $V_1.W_1$. As stated earlier, this takes 3n/2 multiplications. The second element of the first row in C, will be $V_1.W_2$.

Notice that V_1 remains the same and the only thing that changes is W. Therefore, we do not need to calculate V_1 again, for this entire row. The number of multiplications required for this row is,
$3p + (n - 1)*2p$
{2p because, we do not calculate the middle summation component in the formula}

Now, for the second row, we do $V_2.W_1$. We already have the value for W_1, and do not need to calculate the third summation component in the formula. Therefore this takes 2p multiplications. For the second element in the second row, V_2 remains the same and we already have the value of W_2 from before. As a result, this takes p multiplications and so on. The number of multiplications required for this row is,
$2p + (n - 1) * p$

Now, this step is carried out for all the remaining rows in C. Therefore, the total number of multiplications required for this entire process is,

$3p + (n - 1) * 2p + (2p + (n - 1) * p) * (n – 1)$,
$3p + 2pn – 2p + (2p + pn – p) * (n - 1)$,
$p + 2pn + (p + pn) * (n – 1)$
$p + 2pn + pn – p + pn^2 – pn$,
$2pn + pn^2$

Now $p = n/2$. Therefore,

$n^2 + n^3/2$

Or,

$n^3/2 + n^2$