CS 466/666, Spring 2012
Assignment 2 solutions sketch

Q1.i

When it searches for a successor in each set it will actually gallop to its duplicate in each set.

Q1.ii

Keep a counter of how many times a duplicate has been found. If it ever reaches the value of *k* output the element and move to the next item in any arbitrary set.

Q2.i

The maximum number is achieved when the list appears in reverse order. The first item has n-1 inversions, the second has n-2 inversions, and so on for a total number of n*(n-1)/2 inversions or $O(n^2)$

Q2.ii

First observe that during the i-th iteration, the array contains in positions 1 to *i* the first *i* elements of the original array in sorted order. Hence element *i+1* is compared with all the elements that appeared before it in the original list configuration, i.e. the number of inversions involving element a_{i+1} in the original list.

Q3

We modify the tree to add thread pointers from every rightmost (leftmost) node in a path to its leftmost (rightmost) sibling on the same level. We also add a *finger* pointer to the position of the last search in the tree. When searching for a new key rather than starting the search from the root, we start the search from the finger pointer position going up the path. If the key is within the current subtree we continue going up until we reach the proper key and descend down to the position. If it is in an adjacent subtree we use the threaded pointers to move to the sibling subtree and move up in that path.

Observe that if we reach height h from the previous finger position then the node we are looking for must be at a distance of at least 2^h from the previous position.

Q4.

A simply linear order only leads to an *n-1* lower bound on the number of comparisons. Hence we need an adversary argument for the lower bound. We use high and low sets as in the median proof seen in class to force n/2 useless comparisons. Observe that the median of the entire set is the min of set H and the max of set L. Therefore, within each of the sets H and L we run the max and min adversary seen in class which proves that min&max over a set of size N takes 3N/2 comparisons. Each of H and L have n/2 elements so each one of them requires 3n/4 comparisons. Hence the total number of comparisons is at least

n/2+3n/4+3/n4 = 8n/4= 2n

Q5.

To obtain 9n/2 we run the 3n algorithm and then the 3n/2 algorithm seen in class for a total cost of 9n/2 comparisons.

To obtain an algorithm below 9*n*/2 we run the 3*n* algorithm and keep track of all comparisons made. We know that any correct median algorithm produces a partial order which has *n*/2 elements above the median and *n*/2 below the median. Then we can run a linear time max (min) algorithm on each half respectively at a cost of *n*/2 comparisons in each half for a total number of 4*n* comparisons.

Q6.

Assume there are two elements a, b such that we do not know if $a < b$ or $a > b$ by the end of the execution of the algorithm. Then we can set $a=b$ which should not contradict any of the performed comparisons as there is no directed path through *a* and *b*.