HONR 278J
Solutions to Homework 3

Solution 1. Sorting in time $O(n+k)$ where k is the number of inversions and n is the number of elements. The algorithm is the same as Insertion Sort! We assume we are given n numbers, $A[i]$, $1 \leq i \leq n$. We will assume that we have already sorted the first p numbers in increasing order. We will show how to extend this sorted order to the first $p+1$ numbers. Initially, $p = 1$ (note that $A[1]$ is sorted). We now consider $A[p+1]$. If $A[p+1] > A[p]$ then there is nothing else to do, and now the first $p+1$ numbers are sorted. Otherwise we have to find the right place to "insert" $A[p+1]$. Note that we will scan the sorted group from right to left to find the correct place to insert the new element $A[p+1]$. Each time we compare $A[p+1]$ to $A[j]$ with $j < p+1$ and $A[p+1] < A[j]$, this is an inversion. Note that the number of comparisons needed is exactly one more than the number of elements that are greater than $A[p+1]$. But each of these elements is an inversion! Imagine that we have to pay one dollar for each comparison made. If this causes an inversion we pay the "inversion account". Note that while inserting $A[p+1]$ to the correct position, we charge all but one comparison to the "inversion account". For the last comparison we pay the element itself. Note that the inversion account has k dollars, and each element has one dollar. Thus the total number of comparisons is $n+k$.

Solution 2. Add the degree of each node. In doing this sum note that we count each edge exactly twice. Thus the total sum of degrees is exactly twice the number of edges. Another way to prove it is by induction on the number of edges. Consider a graph $G = (V, E)$. Suppose we order the edges $e_1, e_2 \ldots$. Suppose we have inserted i edges and have a graph G_i. Note that the claim is clearly true for $i = 1$ (base case). We have one edge, so the RHS is 2. Exactly two nodes have degree 1. Suppose this claim is true for the first p edges. When we add e_{p+1} then the RHS increases by 2, and the LHS also increases by exactly 2, since the degrees of two nodes is increased (end points of e_{p+1}), all the others stay the same. The RHS is $2(p+1) = 2+2p$. Note that by the induction hypothesis, $2p$ is exactly the sum of the degrees of all nodes before edge e_{p+1} was added. Thus the claim is true.

Solution 3. In both cases, once the heap has been built, the time to run $HEAPSORT(A)$ will take $O(n \log n)$. The only benefit of having the array sorted in decreasing order is that the time spent on building the heap is faster since the elements satisfy the heap order property for a Max Heap. However, each maximum element is moved to the end, and replaced by a small element. Thus it will still take $O(\log n)$ time to restore the heap order property.

Solution 4. Not done...

Solution 5.

1. A table in increasing order does represent a valid Min-Heap. This is because the children of element i are $2i$ and $2i+1$. If the table is in sorted order, then the

element values stored at $2i$ and $2i+1$ are both larger than the value stored at i. Thus it satisfies the requirement for a min-heap.

2. After inserting the 8 into the Max-heap, the table is 10,8,6,7,2,4,3,5. Essentially, 8 gets added at the end of the table, and since its value is greater than its parentvalue 5, we exchange the two values. Its new parent has value 7, and we exchange them again once more.