Math 382

Homework 2

Due Friday, February 12

1. For each of the recurrence relations below, use the tree method to generate a guess $f(n)$ of the running time $T(n)$ and then prove by induction that $T(n) \in O(f(n))$.

   (a) $T(n) = 4T(n/3) + n^2$
   $T(1) = 1$

   (b) $T(n) = 4T(n/3) + n$
   $T(1) = 1$

2. Below is pseudocode for a new sorting algorithm, NewSort. You should look over the code and make sure you understand how and why the algorithm works.

   (a) Prove that NewSort is indeed a correct sorting algorithm.

   (b) How long does NewSort take to run? Prove your answer.

```
Define NewSortHelper(A, start, end):
|  if start = end then
|  |  return
|  if start+1 = end then
|  |  if A[start] > A[end] then
|  |  |  swap A[start], A[end]
|  |  return
|  t = ⌊(end−start+1)/3⌋
|  NewSortHelper(A, start, end − t)
|  NewSortHelper(A, start + t, end)
|  NewSortHelper(A, start, end − t)
Define NewSort(A):
|  NewSortHelper(A, 1, A.length)
```

3. When discussing mergesort, we gave an algorithm MERGE which took two sorted lists and returned a sorted list that contained the elements of both input lists. This algorithm took $O(n)$ time, where n was the total number of elements in the two lists combined. Now consider the case of merging k separate sorted lists, again with n total elements in all lists combined. Find an algorithm that runs in $O(n \lg k)$ time. (Hint: One option is to use a heap to help.) Show that this really is the runtime of your algorithm.