Problem 1. Assume your machine has 32 bit words. Assume you can multiply two n-word numbers in time $3n^2$ with a standard algorithm. Assume you can multiply two n-word numbers in time $20n^{\lg 3}$ with a "fancy" algorithm.

(a) Approximately how large does n have to be for the fancy algorithm to be better?

(b) Approximately how many bits is that?

(c) Approximately how many decimal digits is that?

Problem 2. Use the same assumptions as for problem (1), except assume you can multiply two n-word numbers in time only $10n^{\lg 3}$ with a "fancy" algorithm.

(a) Approximately how large does n have to be for the fancy algorithm to be better?

(b) Approximately how many bits is that?

(c) Approximately how many decimal digits is that?

Problem 3. Recall that in class we showed that a sorted list of size m and a sorted list of size n can be merged with $m + n - 1$ comparisons.

Consider the following "merge sort" algorithm for a list of size n. If the size of the list is at most three then Bubble Sort the list. Otherwise, remove the last three elements and recursively sort the remaining $n - 3$ elements. Sort the last three (removed) elements with Bubble Sort. Form the final sorted list by using the merge algorithm from class to merge the three sorted elements and the sorted list (of size $n - 3$).

(a) Write the algorithm in pseudocode using recursion. You can assume that you have available a Bubble Sort routine

```
BubbleSort(A,i,j)
```

and a Merge routine.

```
Merge(A,(i,j-1),(j,k))
```

(Your algorithm should look very similar to the Merge Sort algorithm we did in class.)

(b) Give a recurrence for the *exact* number of comparisons this algorithm uses.

(c) Use the tree method to solve the recurrence, where n is a multiple of 3. Simplify as much as possible.

(d) How does this compare with Bubble Sort.

(e) Use mathematical induction to verify your solution.