CS 278 **Algorithm Analysis**
Assignment # 7 Due Date: December 06, 2001

Note: Create a subdirectory in your home directory and call it `<your_last_name>.7`, where `<your_last_name>` is your real last name (for example, `bhatia` for me). Do all the programs for this assignment in that directory. After you are done, submit the code by typing the following command:

```
~sanjiv/bin/handin <your_last_name>.7 cs278 7
```

Again, do not forget to substitute for `<your_last_name>`. The command should be executed from your home directory.

Bonus Assignment

1. **Anagrams**

   Given a dictionary of English words, find all sets of anagrams. For instance, "pots," "stop," and "tops" are all anagrams of one another because each can be formed by permuting the letters of the others.

   To solve the problem, you should try to find all anagram classes. Thus, all anagrams belong to the same set.

   Many approaches to this problem are surprisingly ineffective and complicated. Any method that considers all permutations of letters for a word is doomed to failure. The word *cholecystoduodenostomy* (an anagram of *duodenocholecystostomy* has 22! permutations ($22! \approx 1.124 \times 10^{21}$). Even assuming the blazing speed of one picosecond per permutation, this will take 1.1×10^9 seconds (or a few decades). Any method that compares all pairs of words is doomed to at least an overnight run, considering about 230,000 words in a dictionary. We need to find an algorithm that will avoid the above pitfalls.

   Sign each word in the dictionary so that words in the same anagram class have the same *signature*, and then bring together words with equal signatures. This reduces the original anagram problem to two subproblems:

   (a) selecting a signature, and

   (b) collecting words with the same signature.

   For the first problem, we can use a signature based on sorting: order the letters in the word alphabetically. The signature of *pots* is *opst* which is also the signature for *stop*. To solve the second problem, we can sort the words in order of their signature (think of modifying the comparison function in `qsort(3)`).

   Create a data file with at least 500 words, with at least 100 anagram classes to test your code.

   Submit e-copy only by the due date. This assignment will be due at the beginning of class, and cannot be extended.