CSE331 – Project 2: QuickSort

In this project you will complete an implementation of the quicksort algorithm. The project must be submitted to Handin no later than 11:59 pm Wednesday January 23, 2013. Email the TA with any questions. Good Luck!

Project Description

You are being provided with sample code and test input for a program that reads a data file containing the filmographies for actors in the Internet Movie Database. The program takes two command line arguments: (1) the number of actors to be read in and (2) the datafile to read from. The program reads in the movies from the specified number of actors, sorts the movies in alphabetical order, outputs them, then exits.

Your assignment is to complete the implementation of quicksort found in QuickSort.h. You must also complete the implementations of VerifySort.h and DuplicatesRemove.h. VerifySort contains a method which will return true if and only if the elements in the provided vector are sorted. RemoveDuplicates should remove any duplicate keys from the vector. Both VerifySort and RemoveDuplicates should run in $O(n)$time.

Make no other changes to the files other than completing the assigned methods. You must provide your own code for quicksort, you may not copy another's code. You may not make any other explicit function calls from the assigned methods, except to functions you define yourself. You may not use any of the tools provided in the STL other than those in STL vector (i.e. you may not call STL sort, or any other STL algorithm, but you may use vector::size() and vector::resize()).

The code and a small example file may be found in ~/CSE331/Projects/Project2/. The file containing the entire Internet Movie Database may be found in ~/cse331/Projects/Data/.

Programming Notes

The CQuickSort class is template, so that it may perform the sort without specific knowledge of the exact types being sorted. You may assume that any class for which this insertion sort is instantiated will provide the following operators: =,==, <. These are the only operators you are guaranteed to have.

Project Deliverables

The following files must be submitted via Handin no later than 11:59 pm Wednesday January 23, 2013:

- QuickSort.h – contains your implementation of Quicksort
- VerifySort.h – contains your implementation of VerifySort
- DuplicatesRemove.h – contains your implementation of DuplicatesRemove
- project2.pdf – files containing your answers to the written questions.

Written Questions

Submit the answers to these questions in a PDF file, called project2.pdf, along with your source code files. The versions of MS Office and OpenOffice in the CSE labs both support exporting to PDF format.

1. Run your program for data sizes in 10 steps. Choose these sizes so you have an even distribution and the maximum running time does not exceed 10 minutes (yes, I said minutes). For the number of elements you are sorting (not the number of items), indicate:
    a. How long did the sort take?
    b. How many actors, movies, and unique movies?
    c. compare the maximum number of elements sorted to your results from Project 1.

    NOTE: to implement this in the code, simply change the following lines of code in Main.cpp:
    - Line 40: unsigned increment = 10;
    - Comment out lines 74 and 76.

2. The running time for Quicksort depends upon both the data being used and the partition rule used to select the pivot. Although Quicksort is $O(nlogn)$ on average, certain partition rules cause Quicksort to take $\Theta(n^2)$ time if the array is already sorted. For each of the following, justify your answer with a short paragraph.
    a. Suppose we always pick the pivot element to be the key from the last position of the subarray. On a sorted array does Quicksort now take $\Theta(n)$, $\Theta(nlogn)$ or $\Theta(n^2)$.
    b. Suppose we always pick the pivot element to be the key from the middle position of the subarray. On a sorted array does Quicksort now take $\Theta(n)$, $\Theta(nlogn)$ or $\Theta(n^2)$.
3. A stable sort is a sort that outputs equal keys in the same order that they were input. Prove or disprove the following: quicksort is a stable sort. Justify your answer in a few short paragraphs.