Kyle Knudson

CSCI 591

Cs301188

Section 1

04/10/18

Project 10

Project 10 - Design Document

Introduction:

In programming, sorting algorithms provide very important functions and in many cases it is important to minimize the time that the algorithm takes to run. This program builds heavily on the concept of optimizing code and using the sorting methods that are the most efficient. In general, the quick sort algorithm and the merge sort algorithm have a time which is O(n). The other algorithm that is looked at in this program is the insertion sort which in general has a time of O(n^2). This program looks to validate these theoretical values by providing a client with the ability to test various different arrays. This program generates three arrays with the same random values and uses each of the algorithms to sort the arrays. The number of steps that the sort runs is kept track of and reported back to the user to show the difference in the efficiency of the algorithms.

Data Structures:

This program utilizes arrays of integers as the primary data structure in the program. It also uses integers to set limits and keep track of steps in the algorithms. The swap function also uses pointers as a way of swapping the values of two items in the array.

Functions:

void insertion\_sort(); - This function sorts the array that is passed to it via the insertion sort method.

void merge\_sort(); - This function sorts the array that is passed to it via the merge sort method. This function uses the merge() function to bring each section together.

void merge(); - This function is the critical step in the merge sort algorithm and is used to bring the partitions of the array together so that the array can be sorted.

void quick\_sort(); - This function sorts the array that is passed to it via the quick sort method and utilization of the partition function.

void partition (); - This function divides the array into different partitions depending on each values relation to a pivot values.

void swap(); - This function is utilized to swap two values in an array by exchanging their addresses in memory via pointer values.

The Main program:

The main program in this project essentially acts as only the initializer and displaying of the information that is vital to this project. The main program prompts the user to enter the number of values that the user wants in the array that is to be sorted. The program then requests a seed value from the user so that the arrays can be populated with random integer values. The main function then uses this seed value and populates three different arrays via a loop, one for each type of sorting algorithm that is being examined. Once this is done, a sorting call is performed on each array that corresponds to the different sort, one array for insertion sort, one for merge sort and one for quick sort. A count variable is passed by reference in each of the sorting function calls so that the steps that the algorithm takes is available in the main function. Once the sorts are complete, the step counts are printed to the terminal for the user to compare.

.