CT102 Algorithms Assignment 2

Date: Monday 11th March 2024

Due: on or before Thursday 28th March* 2024 via Canvas – SINGLE document including Plagiarism

Total Marks: 15

*unless you have a LENS report

Plagiarism Declaration:

"I am aware of what plagiarism is and include this here to confirm that this work is my own"

QUESTION 1 (4 MARKS)

Given the following two functions `find1()` and `find2()` which are both passed an integer array, `arrA[]` , and its associated size, `size`, and return an integer. In addition, the function `find1()` is passed the value size - 1 for `curr` initially. (Line numbers are included):

```
L1   // curr should be size - 1 for the first call
L2   int find1 (int arrA[], int size, int curr)
L3   {
L4       if (size == 1) {
L5           return (curr);
L6       }
L7       else if ( arrA[curr] < arrA[size - 2] ) {
L8           return (find1 (arrA, size - 1, size - 2));
L9       }
L10      else {
L11          return (find1 (arrA, size - 1, curr));
L12      }
L13  }
L14
L15  int find2 (int arrA[], int size)
L16  {
L17      int curr = 0;
L18      for (int i = 1; i < size; i++) {
L19          if (arrA[i] > arrA[curr]) {
L20              curr = i;
L21          }
L22      }
L23      return (curr);
L24  }
```

With respect to time complexity analysis, and assuming a worst-case scenario, calculate the number of timesteps of each function as a function of the array size *N*. Explain your approach and any assumptions, clearly showing how the timesteps are calculated.

QUESTION 2 (3 MARKS)

The following function `merge()`, merges two sorted portions (from `lb` to `mid` and from `mid+1` to `ub`) of an array `arrA[]`. (Line numbers are included).

```
L1   void merge (int arrA[], int lb, int mid, int ub)
L2   {
L3
L4       int i, j, k;
L5       int size = ub - lb + 1;
L6       int *arrC = (int*) calloc(size, sizeof(int));
L7
L8
L9       for (i = lb, j = mid + 1, k = 0; i <= mid && j <= ub; k++) {
L10         if (arrA[i] <= arrA[j]) {
L11             arrC[k] = arrA[i++];
L12         }
L13         else {
L14             arrC[k] = arrA[j++];
L15         }
L16      } // end for loop
L17
L18      while (i <= mid) {
L19         arrC[k++] = arrA[i++];
L20      }
L21      while (j <= ub) {
L22         arrC[k++] = arrA[j++];
L23      }
L24      for (i = lb, k = 0; i <= ub; i++, k++) {
L25         arrA[i] = arrC[k];
L26      }
L27  }
```

Using some sample data, and with reference to the code line numbers, explain, in your own words, how the function `merge()` works.

QUESTION 3 (8 MARKS)

Given the test file, file1.txt (with 10,000 integers), perform a comparison of the two sorting techniques Merge Sort and Quick Sort considering different values of N (array size) in terms of:

a. time taken
b. number of swaps/data moves
c. number of comparisons
d. number of function calls (recursive if using, and non-recursive)

Present your results in a meaningful and clear way so that it is easy to see differences between the algorithms for the same value of N. You may re-use your code from assignment 1.

Note the following:

- You should read in different portions of the file for each test of N, for example the first 1000 integers, then the first 2000 integers, etc. The last test should use all, or nearly all, of the 10,000 integers. Make sure not to read in partially sorted data (i.e. from a previous run) to ensure fair comparisons.
- You may use a recursive or non-recursive version of the code – indicate which you use in your answer.
- Ensure to modify the `quickSort()` code to pick a better pivot value than the f isrt value and to include a call to Insertion sort when smaller sub-portions should be sorted.
- Ensure all functions have the correct code to count time, data movements, data moves and function calls. Please note that "data movement and comparisons" should only refer to the data in the array we are sorting – no other comparisons or data assignments should be counted (e.g. do not count "i < size" as a comparison). The count of function calls should only include `quickSort()`, `partition()`, `mergeSort()` and `merge()`.

In your answer:

- Explain your testing approach (values of N, what is counted and where, etc.)
- Present the results in a clear and tidy way (comparisons) and comment on/explain the trends that you see. Also, include some sample output (screenshots) to prove the results were obtained from your code.
- Include your `main()` code (screenshot) or wherever you have written the code to read in the data and call the appropriate functions. Ensure it can be easily followed/understood (include comments as needed). This will not be considered self-plagiarism if it taken from assignment 1 – it will make correcting easier if I do not have to refer back to assignment 1 for this code.
- Include any code which has been modified (e.g., `quickSort()` function with the modifications for choosing pivot and calling Insertion Sort).