CT102 Algorithms

Assignment 2

Date: Monday 14th March 2022

Due: on or before Friday 1st April 2022 via Blackboard

Total Marks: 15 (6+3+6)

QUESTION 1 (6 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  }
```

(i) Briefly explain what is meant by a well-defined recursive function, using the above functions to aid your explanation.

(2 marks)

(ii) 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 input size. Explain your approach, clearly showing how the timesteps are calculated.

(4 marks)

QUESTION 2 (3 MARKS)

The following two functions, `mergeSort()` and `merge()`, sort integer values in the array `arrA[]` with associated size (`size`). (Line numbers are included).

```
L1   // must be called initially with lb = 0 and ub = size - 1
L2   void mergeSort(int arrA[], int lb, int ub)
L3   {
L4       int mid;
L5       if (lb < ub) {
L6           mid = int((lb + ub) /2);
L7           mergeSort(arrA, lb, mid);
L8           mergeSort(arrA, mid + 1, ub);
L9           merge(arrA, lb, mid, ub);
L10      }
L11  }
L12
L13  void merge (int arrA[], int lb, int mid, int ub)
L14  {
L15      int i, j, k;
L16      int *arrC;
L17      int size = ub - lb + 1;
L18      arrC = (int*) malloc(size * sizeof(int));
L19
L20      for (i = lb, j = mid + 1, k = 0; i <= mid && j <= ub; k++) {
L21         if (arrA[i] <= arrA[j])
L22             arrC[k] = arrA[i++];
L23         else
L24             arrC[k] = arrA[j++];
L25      }
L26      while (i <= mid)
L27         arrC[k++] = arrA[i++];
L28      while (j <= ub)
L29         arrC[k++] = arrA[j++];
L30      for (i = lb, k = 0; i <= ub; i++, k++)
L31         arrA[i] = arrC[k];
L32  }
```

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

QUESTION 3 (6 MARKS)

Given three sorting functions, Count sort, Merge Sort, and Quick Sort (available on Blackboard), and one sample input file, containing 5000 integers:

5000Ints.txt

Answer the following questions

(a) Output the time taken when you run each function with the file 5000Ints.txt file. Summarise your results in terms of the relative advantages and disadvantages of each function given the results you receive. (3 marks)

(b) Output the number of recursive function calls, swaps/data moves and number of comparisons when you run Merge Sort and Quick Sort with the file 5000Ints.txt. Summarise your results in terms of the relative advantages and disadvantages of each function with this file, including information on where you put the lines of code to do the counting. (3 marks)

QUESTION 4

***** Please include the following plagiarism declaration in your solution if applicable:

Plagiarism Declaration:

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

Please note that any suspected cases of plagiarism, or absence of a plagiarism declaration, will not receive a mark until assurances can be given in person as to the origins of the solution.