Lab Assignment 10
Arrays and recursion
COL100

May 23 2021

1. (**Minimum element**) Implement a recursive function that when given an integer array containing distinct numbers, outputs the index of minimum element. Your program should not use any loops. The idea is to find the minimum element of the left half of the array and the right half and then comparing them to locate the minimum. Here is an example input and output. The first line gives the size of the array followed by the array elements.

   Example input:
   5
   23 12 56 67 11

   Example output:
   4

   Analysis question: What is the total number of calls to the recursive function that is made during the entire execution of your program as a function of the array size?

2. (**Minimum and second minimum**) Implement a recursive function as in the previous question with the difference that in this question the output should be the indices of the minimum and the second minimum element. Again, you are not allowed to use loops. You may assume that the input array is of size at least 2.

   Example input:
   7
   23 12 56 67 11 99 97

   Example output:
   4 1

3. (**Binary search variant**) Implement the following variant of the binary search algorithm that returns 1 if the given element *x* is present in a sorted array *A*, otherwise it returns -1:
   - Check if *x* is the middle element of the array A. If so, return 1. Otherwise, copy the relevant half of the array into another array B and make a recursive call to the function to check if x is present in B.

   The first line of the input is the size of the array, followed by the array elements in the next line. The third line gives the element x to be searched.

Example input:
7
1 4 9 12 34 45 77
12

Example output:
1

Analysis question: Analyse the running time of this program and compare it with the running time of the version discussed in class where the relevant indices are passed. Give reasons in case you observe a significant difference.

4. (**Ternary search**) In this problem implement a search algorithm similar to binary search. Instead of splitting the array into two parts, split the array into three parts, find which part the element may belong to and then make a recursive call for searching the element in that part. The details of the function are left for you to figure out. The input/output format is as in the previous question.

5. (**Partitioning**) Given an array *A* containing distinct integers and an integer *x* present in the array *A*, design and implement an algorithm that rearranges the elements of the array such that the following properties hold (after rearrangement):
- All the elements that are smaller than x are placed before x which is followed by all the elements that are larger than x. Note that the final array need not be sorted.

You are not allowed to define any additional arrays (other than the given array) for this question.

The first line of the input is the size of the array, followed by the array elements in the next line. The third line gives the element x in the array.

Example input:
7
77 4 34 12 9 45 1
12

Example output:
1 9 4 12 77 45 34

Please note that the solution may not be unique. There may be various correct outputs.

6. (**Quicksort**) You can use your program from Q5 to implement a sorting algorithm known as Quicksort. This can be described in the following recursive manner:
- Pick a random element x from the array A
- Use your program from Q5 to partition the array A.
- Let A[0]A[1]…A[L] denote the elements that are smaller than x and A[L+2]A[L+3]…A[n-1] denote the elements that are larger. Make two recursive calls, one to sort subarray A[0]…A[L] and another to sort subarray A[L+2]…A[n-1]. Note that once both these recursive calls return, the entire array would be sorted.

The first line of the input gives the size of the array followed by the array elements in the second line.

Example input:
7
23 12 56 67 11 99 97

Example output:
11 12 23 56 67 97 99

7. (**Merging sorted arrays**) Given two sorted integer arrays A and B of size n and m respectively, the goal is to compute an array of size (n+m) that containing the numbers from A and B merged in sorted order. Design and implement an algorithm for this problem. The resulting array should be printed by your program.

   The first line of the input gives n followed by the array A in the second line. The third line gives m followed by the array B in the fourth line.

   Example input:
   4
   1 3 5 8
   5
   2 3 6 8 9

   Example output:
   1 2 3 3 5 6 8 8 9

8. (**Merge Sort**) You can use your program for Q7 to implement the following recursive sorting function (called Merge Sort):
   - Split the given array A into the left half AL and the right half AR.
   - Make recursive calls to sort AL and AR. Let BL and BR denote the sorted arrays returned by the recursive calls.
   - Use your program for Q7 to merge the arrays BL and BR to compute the final sorted array.

   The first line of the input gives the size of the array followed by the array elements in the second line.

   Example input:
   7
   23 12 56 67 11 99 97

   Example output:
   11 12 23 56 67 97 99

Submission and other logistics

Submit at least 4 solutions (from the first 8 questions) as a zip file on Gradescope (to your respective group's course). There is no need to submit solutions to the challenge problems. Submit only one .c file for each question. Use separate .c files for each new question. Please name your .c files as per the question number (q1.c, q2.c, ... etc). Following this naming convention will help TAs to figure out where to look for the answers easily. Also add the corresponding screenshots, showing output of your code on certain input examples. You can also submit more than 4 or all questions to increase your chances of full-marks.

Example: To zip folder 'a10' as 'a10.zip': zip −r a10.zip a10

It is highly recommended that you name the code files and variables in those code files with proper names as per the question to easily identify them. Comments in your codes are also highly encouraged and makes life easier for everyone.

You can check 2nd Chapter in NASA's C style guide for styling recommendations.

You can work either individually or with another student of your group for the assignment.only one submission on gradescope is enough for a team but you need to add your teammate on gradescope after submission.

Follow these steps for adding your team member