Assignment 2

Recursion, Divide and Conquer Approach

Implement the following programs in Java.

1. An array `A` of `N` elements is *symmetric* if for every index `i ≤ N/2, A[i] = A[N-1-i]`. Implement a recursive algorithm that returns TRUE if the input array is symmetric and FALSE otherwise. What is the running time of your recursive algorithm?
   Complete method `symmetric()` in file symm.java. Submit only file symm.java.
   Use the main program symm_main.java and input data file symm.in to test your program. Check file symm.out for the correct output.

2. Given an array `A` of `N` distinct integers sorted in increasing order, we seek an algorithm to determine if there exist two integers in `A` that sum to `k`. The algorithm returns TRUE if such a pair exists, and FALSE otherwise.
   a. Give an *exhaustive* algorithm and obtain its running time in the worst case. Complete method `sum_exh()` in file sum.java.
   b. Give a more efficient *recursive* algorithm and obtain its running time in the worst case. Complete method `sum_rec()` in file sum.java.

   Submit only file sum.java.
   Use the main program sum_main.java and input data file sum.in to test your program. Check file sum.out for the correct output.

3. Given an array `A` of `N` distinct integers sorted in increasing order, we seek an algorithm to determine if there exists an index `i` such that `A[i] = i`. The algorithm returns `i` if such an index `i` exists, and -1 otherwise.
   a. Give an *exhaustive* algorithm and obtain its running time in the worst case. Complete method `match_exh()` in file match.java.
   b. Give a *divide-and-conquer* algorithm and obtain its running time in the worst case. Complete method `match_dac()` in file match.java.

   Submit only file match.java.
   Use the main program match_main.java and input data file match.in to test your program. Check file match.out for the correct output.

4. Given an (unsorted) array `A` of `N` distinct integers, implement a *divide-and-conquer* algorithm to find the Kth smallest element (K ≤ N) in the array (it would be the overall smallest if K=1). The algorithm returns the value of the Kth smallest element in the array. Your algorithm should run in O(N) time in the average case.
   Complete method `find_kth_smallest()` in file kthsmallest.java. Submit only file kthsmallest.java.
   Use the main program kthsmallest_main.java and input data file kthsmallest.in to test your program. Check file kthsmallest.out for the correct output.

Hints:

1. Borrow the idea from the problem of reversing an array.
2. Begin with the first and last elements. Move one step left or right depending on the current value of their sum.
3. Borrow the idea from the binary search algorithm. There may exist more than one index `i` such that `A[i] = i`. Your program can return any one of those indices.
4. Borrow the idea from the partitioning algorithm of quick sort. Suppose it led to a partition of L and (N-1-L) elements. If L < K, you need to recur on only one side (which side?). Otherwise, recur on the other side. **Do not sort the whole array** and then return the Kth element. If you do that, your program will get zero point.

Notes:

- Do not modify the given class and method definitions.
- Do not add I/O statements (e.g., scanner, print) to the submitted Java files. I/O statements will mess up our automatic grading programs and produce incorrect outputs. Your program will get zero in such cases.
- **Indicate the running time** of each algorithm in the same Java file implementing it, right above the method definition (see the given templates). To indicate exponentiations, use the symbol `^`. For example, `n * n * n = n^3`.
- The input data given in the above main programs are only examples for your testing. We may use different data sets to mark your programs.
- Assume that all inputs are valid and `N < MAXSIZE` in all programs.
- To compile and run the programs, use the following commands and the appropriate files:

```
javac symm.java
javac symm_main.java
java symm_main < symm.in
```

Posted Jan. 29, 2020
Last updated Jan. 29, 2020