Assignment 5

Grades: each assignment is 5% in the final score and there is a 2% bonus in each assignment for bonus problems. We use the following scale for the simplicity of grading: the total score is 50 (additional 20 for bonus problems) for each assignment. Late Assignment is Accepted until March 14th (Monday) at noon (12:00pm)! Solution will be posted then!

Problem 1 [Full Score: 10]. R-4.15. (on page 252)

Problem 2 [Full Score: 10]. C-4.14. (on page 253)

Problem 3 [Full Score: 30]. We say that an array A[1 · · · n] is k-sorted if it can be divided into k blocks, each of size n/k, such that the elements in each block are larger than the elements in earlier blocks, and smaller than elements in later blocks. The elements within each block need not be sorted. For example, the following array is 4-sorted:

\[ [1, 2, 4, 3]|7, 6, 8, 5]|10, 11, 9, 12]|15, 13, 16, 14] \]

(1)[15 ] Describe an algorithm that k-sorts an arbitrary array in \(O(n \log k)\) time. (You should provide an algorithm and analyze its complexity. HINT: think about the quick-sort discussed in the lecture.)

(2)[15 ] Use the decision tree method to prove that any comparison-based k-sorting algorithm requires \(\Omega(n \log k)\) comparisons in the worst case. (HINT: count the number of possible outcomes of any k-sorting algorithm. Use the Stirling’s approximation \(\log(n!) \sim n \log n - n\). )

Problem 4 [Bonus Score: 20]. Given a size-n array where every element occurs twice, except one element which occurs only once. Find the element that occurs once. For example, if the array is

\[ [12, 1, 12, 15, 6, 7, 7, 1, 3, 6], \]

then the output should be 3.

(1)[10 ] Use the adversarial method to prove any deterministic algorithm for the above task needs \(\Omega(n)\) time.

(2)[10 ] Provide an \(O(n)\) algorithm. (HINT: consider the binary form of each element and bitwise operations.)