Homework set #1
CSC 317 Data Structures And Algorithm Analysis

Dilip Sarkar

Feb 01, and Due on Feb 08 11:59 pm 2021

This problem set covers Sections 1.1-2, 2.1-2, and class notes. Please upload your answers on the Blackboard.

1. For sorting problem, there are two categories of algorithms: one sorts by reducing the problem size by one at each step, and the other category of algorithms divides the problem into two smaller problems, solves each smaller problem recursively, and combines the two sorted lists. The first category of algorithms are not optimal and time complexity of these algorithms are $O(n^2)$. Whereas, some of the algorithms in the second category have optimal time complexity. But *optimal algorithms are not always fastest.* In many cases, if input size is greater than a constant only then algorithm is faster. In this homework problem you have to find the constant. Suppose your are comparing implementations of `bubble sort` and `heap sort` algorithms on the same machine. For input size n, `bubble sort` runs in $8n^2$ steps, while the `heap sort` runs in $512\,n\log_2 n$ steps.
   (a) For which values of n `bubble sort` beats the `heap sort`? Note that $n \geq 1$.
   (b) Explain how got your answer and show your work.

2. To answer this question you will discuss role of data structure for implementation of an algorithm. Let us consider a search problem. Given a set of integers S_I and an integer i, the task is to search in the list to find i. If i is in the list, then the algorithm returns TRUE else it returns FALSE. Suppose SEARCH-A, SEARCH-B, and SEARCH-C are three algorithms available for the *search* **problem**. Implementations of SEARCH-A, SEARCH-B, and SEARCH-C used linked-list, queue, and array as data structures, respectively. This problem is for showing effect of data structures for implementation of selection algorithm.
   (a) What is the worst-case time complexity of the algorithm SEARCH-A? Explain your answer.
   (b) What is the worst-case time complexity of the algorithm SEARCH-B? Explain your answer.
   (c) What is the worst-case time complexity of the algorithm SEARCH-C? Explain your answer.

3. Let us consider the sorting **problem**.
   (a) What is the *lower bound* for the sorting **problem**. Briefly explain your answer.
   (b) List names of 5 sorting **algorithms** and write *worst-case* time complexity of each.
   (c) Suppose the list above has all *known* sorting algorithms in the world. What is the *upper bound* for the sorting **problem**.
   (d) In the list above, how many are *optimal algorithm sorting*? Write their names. Briefly explain your answer.

4. Rewrite the INSERTION SORT procedure (on page 18 of the textbook) to sort into non-increasing instead of non-decreasing order.

5. Using *loop invariant* show correctness of the procedure you wrote.