CSC-421 Applied Algorithms and Structures
Spring 2017

Instructor: Iyad Kanj
Office: CDM 832
Phone: (312) 362-5558
Email: ikanj@cs.depaul.edu
Office Hours: Monday 4:40 - 5:40 PM & Wednesday 4:45 - 6:45 PM
Course Website: https://d2l.depaul.edu/

Assignment #1

(Due April 12)

Note. When asked to give an algorithm that meets a certain time bound, you need to give the algorithm (pseudocode/description) and analyze its running time to show that it meets the required bound; giving only the algorithm is not enough to receive full credit.

Please upload your submission as a single PDF file on D2L. If your submission consists of more than one file, convert all your files into a single PDF file and upload it.

1. Given a collection of n nuts and a collection of n bolts, arranged in an increasing order of size, give an $O(n)$ time algorithm to check if there is a nut and a bolt that have the same size. The sizes of the nuts and bolts are stored in the sorted arrays $NUTS[1..n]$ and $BOLTS[1..n]$, respectively. Your algorithm can stop as soon as it finds a single match (i.e, you do not need to report all matches).

2. Let $A[1..n]$ be an array of distinct positive integers, and let t be a positive integer.

   (a) Assuming that A is sorted, show that in $O(n)$ time it can be decided if A contains two distinct elements x and y such that $x + y = t$.

(b) Use part (a) to show that the following problem, referred to as the 3-SUM problem, can be solved in $O(n^2)$ time:

> 3-SUM
> Given an array $A[1..n]$ of distinct positive integers that is not (necessarily) sorted, and a positive integer t, determine whether or not there are three distinct elements x, y, z in A such that $x + y + z = t$.

3. Let $A[1..n]$ be an array of positive integers (A is not sorted). Pinocchio claims that there exists an $O(n)$-time algorithm that decides if there are two integers in A whose sum is 1000. Is Pinocchio right, or will his nose grow? If you say Pinocchio is right, explain how it can be done in $O(n)$ time; otherwise, argue why it is impossible.

4. Suppose that we are given an array $A[1..n]$ of integers such that $A[1] < A[2] < \cdots < A[n]$. Give an $O(\lg n)$ time algorithm to decide if there exists an index $1 \leq i \leq n$ such that $A[i] = i$.

5. Let $A[1..n]$ be an array of numbers. To find the largest number in A, one way is to divide A into two halves, recursively find the largest number in each half, and pick the maximum between the two.

   (a) Write a recursive algorithm to implement the above scheme. Write a recurrence relation describing the running time of the algorithm and solve it to give a tight bound on the running time of this algorithm.

   (b) Does this recursive algorithm makes fewer comparisons than an incremental algorithm that computes the largest element in A by iterating through the elements of A? Explain.