CSCI 270 Homework #6

Due Date: Friday, March 28th, 11:59pm

1. Suppose you are choosing between the following four algorithms:

   - Algorithm A solves problems by dividing them into five subproblems each of size $\frac{n}{2}$, recursively solving each subproblem, and then combining the solutions in linear time.
   - Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
   - Algorithm C solves problems of size n by dividing them into nine subproblems of size $\frac{n}{3}$, recursively solving each subproblem, and then combining the solutions in $\Theta(n^2)$ time.
   - Algorithm D recursively divides an input of size n into four subsets each of size $\frac{n}{2}$ and combines them in $\Theta(n^2 \log n)$ time.

   What are the running times of each of these algorithms (in big-Θ notation), and which would you choose?

2. Suppose we have a sorted array of distinct integers A[1, . . . , n] and we want to decide whether there is an index i for which A[i] = i. Describe a divide-and-conquer algorithm that solves this problem faster than $\Theta(n)$, and analyze the runtime.

3. You are given an $n \times n$ matrix A where every row is in sorted order, and every column is in sorted order. Design an efficient (better than $\Theta(n^2)$) algorithm to search for an element x in the matrix, and analyze its runtime complexity with respect to n.

4. You are given a directed graph $G = (V, E)$ with exactly $\frac{n(n-1)}{2}$ edges. For every pair of nodes $u, v \in V$, one of the edges (u, v) and (v, u) is in the graph, and the other is not. You want to find a simple directed path which includes every node in the graph exactly once. It turns out that there is **always** such a path, which you will realize once you find the algorithm. Give an efficient divide-and-conquer algorithm to find such a path, and analyze the running time by setting up and solving a recurrence relation.