1. The diameter of a graph is the maximum of the shortest paths’ lengths between all pairs of nodes in graph . Design an algorithm which computes the diameter of a connected, undirected, unweighted graph in time, and explain why it has that runtime.
2. Given a directed graph where every edge has weight as either 1 or 2, find the shortest path from a given source vertex ‘s’ to a given destination vertex ‘t’. Expected time complexity is O(V+E).
3. Consider a divide-and-conquer algorithm that splits the problem of size n into 4 sub-problems of size n/2. Assume that the divide step takes O(n^2) to run and the combine step takes O(n^2 log n) to run on problem of size n. Use any method that you know of to come up with an upper bound (as tight as possible) on the cost of this algorithm.
4. Input: An array A = [1...n] of integers (can be either positive or negative).  
   Output: indices i\*, j\* (with i\*≤ j\*) such that the sum over the subarray A[i\*... j\*] is maximized.

Example

Input: A = [3, -4, 5, -2, -2, 6, -3, 5, -3, 2]

Output: i\*= 3, j\*= 8, sum = 5-2-2+6-3+5 = 9

Give a divide and conquer algorithm to solve this problem.

1. Suppose you have a two-dimensional array A[1::n; 1::n] of integers (or other comparable item). The array is given to you in such a way that each row is sorted in ascending order and each column is also sorted in ascending order. Our goal is to determine if a given value x exists in the array.
2. One way to do this is to call binary search on each row (alternately, on each column). What is the running time of this approach?
3. Give another divide-and-conquer way to do this, and state the runtime of your algorithm. For credit, your algorithm should take strictly less than O(n^2) time, and should make use of the fact that each row and each column is in sorted order (i.e., don't just call binary search on each row).