CSE 321 Homework 3

1.

a.

The logic of DFS-based topological sort is like Depth First Search. The only difference is caused by the vertex with no incoming edges. To avoid that, all vertices must be considered after Depth First Search.

The DFS-based topological sort works in this way:

- Start from first node.
- Depth First Search is applied to first node, and all vertices which are visited add to visited set.
- After that, if any of the vertices is not visited, same procedure applied on that node.
- Then print the stack which is filled with elements in topological order.

The DFS-based topological sort has O(V+E) time complexity. Because all vertex and edges are visited once. There is no repetition.

b.

The logic of BFS-based topological sort is like Breadth First Search. The only difference is caused by the vertex with no incoming edges. To avoid that, all vertices must be considered after Breadth First Search.

The BFS-based topological sort works in this way:

- Start from first node.
- Breadth First Search is applied to first node, and all vertices which are visited add to visited set.
- After that, if any of the vertices is not visited, same procedure applied on that node.
- Then print the stack which is filled with elements in topological order.

The BFS-based topological sort has O(V+E) time complexity. Because all vertex and edges are visited once. There is no repetition.

2.

As it is known that the time complexity of calculating “a^n” is O(n) with brute force approach. So, to reduce the time complexity of this function, a smarter designed algorithm must be used. One way to find “a^n” in a shorter time is finding a^1, a^2, a^4, …, a^(n/2) respectively.

The algorithm I designed works in this way:

- Check whether “n” is negative, if “n” is negative, convert “a” into “1/a” and recursively call the function.
- Check whether “n” is 1, if “n” is 1, return “a”.
- Check whether “n” is even, if “n” is even, recursively call function with dividing “n” by two and return square of the result of the recursive call.
- Check whether “n” is odd, if “n” is odd, recursively call function with dividing “n” – 1 by 2 and return square of the result of the recursive call.

As it is indicated in the homework pdf, the worst-case time complexity of this algorithm is O(logn). The logic of this algorithm can be thought like binary search algorithm. Because in every step half of the power is dealt with.

3.

The logic of algorithm which solve 9*9 by using exhaustive search is that starting from first empty grid, all possible numbers are tried until the first appropriate number is found. All grids are filled in this way. If there is a grid which cannot be filled with any of the numbers from 1 to 9, the last move is taken back and fills with next appropriate number. If this step also is not resulted in sudoku complete, one more step is taken back. This approach goes until beginning of the sudoku. After every grid is filled with appropriate number, the sudoku will be solved.

The worst-case time complexity of this algorithm is O(9^(n*n)). Because there are 9 possibilities for every grid and there are at most n*n grid.

4.

a. Insertion Sort

6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	8	9	3	3	12
6	8	8	9	3	3	12
6	8	8	9	3	3	12
6	8	8	9	3	3	12
6	8	8	3	9	3	12
6	8	3	8	9	3	12
6	3	8	8	9	3	12
3	6	8	8	9	3	12
3	6	8	8	9	3	12
3	6	8	8	9	3	12
3	6	8	8	3	9	12
3	6	8	3	8	9	12
3	6	3	8	8	9	12
3	3	6	8	9	9	12
3	3	6	8	8	9	12
3	6	8	8	9	9	12
3	6	8	8	9	9	12
3	6	8	8	9	9	12
3	6	8	8	9	9	12

Red: ordered, blue: compared with, green: current, black: unordered

As it is seen by the table, in insertion sort order of equivalent elements is preserved. Because two items are swapped only if the right one is less than the left one.

b. Quick Sort

6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	12	3	3	8
6	8	9	12	3	3	8
6	8	9	12	3	3	8
6	8	9	12	3	3	8
6	3	9	12	3	8	8
6	3	9	12	3	8	8
6	3	9	12	3	8	8
6	3	9	12	3	8	8
6	3	3	12	9	8	8
6	3	3	12	9	8	8
6	3	3	12	9	8	8
6	3	3	12	9	8	8
6	3	3	8	9	8	12
6	3	3	8	9	8	12
6	3	3	8	9	8	12
6	3	3				
6	3	3				
6	3	3				
6	3	3				
3	3	6				
3	3	6				
3	3	6				
	3	6				
	3	6				
	6	3				
	6	3				
	3	6				
	3	6				
	3	6				
				9	8	12
				9	12	8
				9	12	8
				8	12	9
				8	12	9
				8	12	9
					12	9
					12	9
					9	12
					9	12
3	3	6	8	8	9	12

Red: big or equal, blue: pivot, green: less

Normally, quick sort can be stable but in the version of quick sort algorithm which is applied to above array is not a stable sorting algorithm.

c. Bubble Sort

6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	9	8	3	3	12
6	8	8	9	3	3	12
6	8	8	9	3	3	12
6	8	8	3	9	3	12
6	8	8	3	9	3	12
6	8	8	3	3	9	12
6	8	8	3	3	9	12
6	8	8	3	3	9	12
6	8	8	3	3	9	12
6	8	8	3	3	9	12
6	8	3	8	3	9	12
6	8	3	8	3	9	12
6	8	3	3	8	9	12
6	8	3	3	8	9	12
6	8	3	3	8	9	12
6	8	3	3	8	9	12
6	3	8	3	8	9	12
6	3	8	3	8	9	12
6	3	3	8	8	9	12
6	3	3	8	8	9	12
6	3	3	8	8	9	12
3	6	3	8	8	9	12
3	6	3	8	8	9	12
3	3	6	8	8	9	12
3	3	6	8	8	9	12
3	3	6	8	8	9	12
3	3	6	8	8	9	12
3	3	6	8	8	9	12
3	3	6	8	8	9	12
3	3	6	8	8	9	12
3	3	6	8	8	9	12

Red: ordered, blue: compered, green: swapped

As it is seen by the above table, bubble sort is one of the stable sorting algorithms. Because two items are swapped only if the right one is less than the left one. Otherwise, their order is preserved like insertion sort algorithm.

5.

a.

Brute Force is the one of the basic approaches to solve a problem. In general, it is the first solution that comes to mind, also it does not require the problem is understood as much as other problem-solving techniques. The main goal of Brute Force approach is trying all possible solutions for a given problem until the correct solution is found. For example, selection sort is one of the algorithms which uses Brute Force approach. In selection sort, the ith minimum element is found on the array and placed it to the ith position of the array. It is easy to apply but one of the sorting algorithms with the greatest time complexity.

Exhaustive Search use brute force approach to deal with combinatorial problems (combinations, permutations). It is applied in some problems such as travelling salesman, Knapsack, and assignment.

b.

Caesars's Cipher is kind of an encryption techniques which replace each letter in plaintext by a letter some fixed number of positions down the alphabet. For this reason, there are possible 25 keys which is one minus than number of letters in standard English alphabet. So, finding 25 possible keys may be hard for a world which there are no computers. Because computer can easily find 25 possible keys with brute force approach. So, Caesars's Cipher is vulnerable to brute force attacks. On the other hand, AES is not vulnerable to brute force attacks not because all possible keys cannot be found with brute force attacks but because it is takes very long. To explain, there are 2^128 different keys for AES128, and assume that a computer can try 2^26 different keys per second (It is better than average computer performance), the number of keys which can be tried by this computer in a year is 2,117 trillion. So, trying all 2^128 keys takes 10^23 years (The age of universe is 10^17).

c.

At the first glance at the problem, if it is assumed that checking if n is prime T(n), it may seem like T(n) = O(n). But in this solution, n is the numeric value of the number, not input size. So, to represent time complexity right, the function T must be defined according to input size. The input size of number n is log2(n). Let's assume that k = log2(n) so, T(log2(n)) = O(n). In other words, T(k) = O(2^k).