CSE 321 - Homework 3
Sena Özbelen - 1901042601

1-)

a.This algorithm is based on DFS. It traverses vertices using DFS. To reach the vertices which are not on the main line, this algorithm has a recursive function. Before visiting the neighbors of the current vertex, it checks all vertices in the graph to find out if there are vertices which are not the main line and their neighbor is the current vertex such as CSE 211 in the example. By doing this recursively and storing the vertices not on main line, we can reach the furthest vertex.

b.This algorithm is based on BFS. The only difference between these algorithms is that this one traverses vertices using BFS.

Time Complexity: O(V*V) since it checks all graph elements to find out if it is a previous vertex of the current vertex.

2-) This algorithm is based on binary search (decrease-by-a-constant-factor) since the required time complexity is logn. If the power is even, then it divides the multiplication sequence into 2 and calls itself recursively with halves. After that, it multiplies the results coming from both halves. If it is odd, then it performs the same process but also, it multiplies the results with the number itself since the sequence cannot be divided into two halves exactly.

2^6 = 2^3 * 2^3 -> result of 2^3 * result of 2^3
2^3 = 2^1 * 2 * 2^1 -> result of 2^1 * the number * result of 2^1

Time Complexity: O(logn) since this is a binary search based algorithm.

3-) This algorithm is based on recursive validations. “solve” function checks if the current cell is empty/zero. If so, “find_num” function tries to find the correct number from 1 to 9 and calls “valid_location” to find out if the current number is already in the given row/column/3x3 square. If it is valid, then calls “solve” function for new number and this algorithm recursively checks if this new number is the correct number to solve the sudoku. If not, then continues with the next number and tries again.

Time Complexity: Size is N*N. Since for every empty cell, 9 numbers are the possibilities,

T(N*N) = 9*T(N*N-1) + c = O(9^(N*N))

4-) array = {6,8,9,8,3,3,12}

-Insertion Sort:

current: 6 -> list: 6,8,9,8,3,3,12
current: 8 -> list: 6,8,9,8,3,3,12
current: 9 -> list: 6,8,9,8,3,3,12
current: 8 -> list: 6,8,8,9,3,3,12
current: 3 -> list: 3,6,8,8,9,3,12
current: 3 -> list: 3,3,6,8,8,9,12
current: 12 -> list: 3,3,6,8,8,9,12

Insertion sort is stable because if current element is smaller than the corresponding element in while loop, then it shifts right the larger element but if they are the same element, it does not satisfy the condition and skips the current element.

-Quick Sort:

pivot = 6 -> list: 6, 3,3,8,9,8,12 -> 3,3,6,8,9,8,12
pivot = 3 -> list: 3, 3, 6, 8, 9,8,12 -> 3,3,6,8,9,8,12
pivot = 8 -> list: 3, 3, 6, 8, 8,9,12 -> 3,3,6,8,8,9,12
pivot = 8 -> list: 3, 3, 6, 8, 8, 9,12 -> 3,3,6,8,8,9,12
pivot = 9 -> list: 3, 3, 6, 8, 8, 9, 12 -> 3,3,6,8,8,9,12

Quick sort is not stable. This algorithm has a pivot and it checks if the elements in left part is smaller than the pivot and the elements in right part is larger than the pivot. If it is not, then it swaps elements. For example, when pivot is 6, the first 8 and the last 3 should be swapped in order to satisfy the order but after this swap operation, the first 8 comes after the second 8. Similarly, the last 3 comes before the first 3. Thus, we can say this algorithm is not stable.

-Bubble Sort:

pair: 6,8 -> list: 6,8,9,8,3,3,12
pair: 8,9 -> list: 6,8,9,8,3,3,12
pair: 9,8 -> list: 6,8,8,9,3,3,12
pair: 9,3 -> list: 6,8,8,3,9,3,12
pair: 9,3 -> list: 6,8,8,3,3,9,12
pair: 9,12 -> list: 6,8,8,3,3,9,12
pair: 6,8 -> list: 6,8,8,3,3,9,12
pair: 8,8 -> list: 6,8,8,3,3,9,12
pair: 8,3 -> list: 6,8,3,8,3,9,12
pair: 8,3 -> list: 6,8,3,3,8,9,12
pair: 8,9 -> list: 6,8,3,3,8,9,12

pair: 6,8 -> list: 6,8,3,3,8,9,12
pair: 8,3 -> list: 6,3,8,3,8,9,12
pair: 8,3 -> list: 6,3,3,8,8,9,12

pair: 8,8 -> list: 6,3,3,8,8,9,12

pair: 6,3 -> list: 3,6,3,8,8,9,12
pair: 6,3 -> list: 3,3,6,8,8,9,12
pair: 6,8 -> list: 3,3,6,8,8,9,12

pair: 3,3 -> list: 3,3,6,8,8,9,12
pair: 3,6 -> list: 3,3,6,8,8,9,12

pair: 3,3 -> list: 3,3,6,8,8,9,12

Bubble sort is stable because this algorithm only swaps the element when the second one is smaller than the first one.

5-)

a. Brute force is an algorithm. Exhaustive search is a special case of brute force. The aim of brute force is to find a straightforward, simple way to solve a problem. Meanwhile, exhaustive search tries to find all possible combinations to satisfy the given condition. Since the simplest and basic way to solve a problem without considering efficiency is to evaluate all possible outcomes, brute force and exhaustive search are very related cases.

b. Caesar Cipher is a type of substitution cipher. It is based on replacing a letter with a shifted version with respect to its corresponding number. This is vulnerable to brute force attacks because there is a shifting limit which is 25. After 25, it returns to the original letter. Since the possible shifting amounts are so limited, it can be found by using brute force.

AES is a specification for the encryption. It has 4 steps to encrypt a text: SubBytes, ShiftRows, MixColumns, Add Round Key. AES performs these steps in rounds according to key size. AES is more powerful option since it has very complicated steps for encryption. It is not easy to find the proper combination by using brute force.

c. Primality testing is exponential in the size of the input. The input size is typically measured in bits. Since we have the input n which is represented by bits, the size of the input becomes 2^b where b is the number of bits. Since time complexity is O(n) in the value of input, time complexity in naive solution is O(2^b).