ALGO Tutorial 06

1. Explain how main sorting algorithms can be performed using an appropriate array example.
    - **Bubble Sort:**
        - ✓ Bubble Sort repeatedly steps through the array, compares adjacent elements, and swaps them if they are in the wrong order.
        - ✓ The process continues until the array is sorted.
        - ✓ Example: [2, 5, 8, 3, 12]
    - **Selection Sort:**
        - ✓ Selection Sort divides the array into two portions: the sorted portion and the unsorted portion.
        - ✓ It repeatedly finds the minimum element from the unsorted portion and places it at the beginning of the sorted portion.
        - ✓ Example: [5, 2, 8, 12, 3]

2. Compare and contrast bubble sort and selection sort algorithms.
    - **Approach:**
    - ✓ Bubble Sort compares adjacent elements and swaps them if they are in the wrong order. It repeatedly passes through the array until it is fully sorted.
    - ✓ Selection Sort divides the array into two portions: the sorted portion and the unsorted portion. It finds the minimum element from the unsorted portion and places it at the beginning of the sorted portion. This process is repeated until the entire array is sorted.
    - **Complexity:**
    - ✓ Bubble Sort has an average and worst-case time complexity of O(n^2), where n is the number of elements in the array. This means it can be inefficient for large datasets.
    - ✓ Selection Sort also has an average and worst-case time complexity of O(n^2). It performs better than Bubble Sort in practice due to fewer swaps, but it is still inefficient for large datasets.
    - **Performance:**
    - ✓ Bubble Sort performs many unnecessary swaps since it compares adjacent elements and swaps them one by one. This makes it slower compared to other sorting algorithms.
    - ✓ Selection Sort performs fewer swaps since it only swaps the minimum element once in each iteration. This reduces the number of memory writes, making it slightly faster than Bubble Sort.
    - **Space Complexity:**
    - ✓ Both Bubble Sort and Selection Sort have a space complexity of O (1) since they sort the elements in-place without requiring additional memory.

3. What are the real-world examples of sorting?
    - Databases: Sorting is essential in database management systems for organizing and retrieving data efficiently. For example, when executing a query that requires sorted results based on certain criteria, the database engine employs sorting algorithms to arrange the data in the desired order.

    - E-commerce: Sorting plays a crucial role in e-commerce platforms. Product listings can be sorted by relevance, price, popularity, or customer ratings to provide users with personalized and easy-to-navigate shopping experiences.

    - Contact Lists: Sorting is commonly used in contact management applications or phonebooks to arrange contacts alphabetically by name or to allow users to sort by different criteria like date added, organization, or location.

4. Write function using pseudo or source codes to sort an integer array using bubble sort and selection sort.
    - **Bubble Sort:**

```
def bubble_sort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n-1):

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # Swap if the element found is greater than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

    return arr
```

- **Selection Sort:**

```
def selection_sort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):

        # Find the minimum element in the remaining unsorted array
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j

        # Swap the found minimum element with the first element
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

    return arr
```