1.What is the divide and conquer strategy?
ANS:

Divide And Conquer

This technique can be divided into the following three parts:

- **Divide:** This involves dividing the problem into smaller sub-problems.
- **Conquer:** Solve sub-problems by calling recursively until solved.
- **Combine:** Combine the sub-problems to get the final solution of the whole problem.

The following are some standard algorithms that follow Divide and Conquer algorithm.

- **Quicksort** is a sorting algorithm. The algorithm picks a pivot element and rearranges the array elements so that all elements smaller than the picked pivot element move to the left side of the pivot, and all greater elements move to the right side. Finally, the algorithm recursively sorts the subarrays on the left and right of the pivot element.

- **Merge Sort** is also a sorting algorithm. The algorithm divides the array into two halves, recursively sorts them, and finally merges the two sorted halves.

- **Closest Pair of Points** The problem is to find the closest pair of points in a set of points in the x-y plane. The problem can be solved in O(n^2) time by calculating the distances of every pair of points and comparing the distances to find the minimum. The Divide and Conquer algorithm solves the problem in O(N log N) time.

- **Strassen’s Algorithm** is an efficient algorithm to multiply two matrices. A simple method to multiply two matrices needs 3 nested loops and is O(n^3). Strassen’s algorithm multiplies two matrices in O(n^2.8974) time.

- **Cooley–Tukey Fast Fourier Transform (FFT) algorithm** is the most common algorithm for FFT. It is a divide and conquer algorithm which works in O(N log N) time.

- **Karatsuba algorithm for fast multiplication** does the multiplication of two *n*-digit numbers in at most

$3n^{log_2^3} \approx 3n^{1.585}$

- single-digit multiplications in general (and exactly

$n^{\log_2 3}$

when *n* is a power of 2). It is, therefore, faster than the classical algorithm, which requires *n*2 single-digit products. If *n* = 210 = 1024, in particular, the exact counts are 310 = 59, 049 and (210)2 = 1, 048, 576, respectively.

What does not qualifies as Divide and Conquer:

Binary Search is a searching algorithm. In each step, the algorithm compares the input element x with the value of the middle element in the array. If the values match, return the index of the middle. Otherwise, if x is less than the middle element, then the algorithm recurs for the left side of the middle element, else recurs for the right side of the middle element. Contrary to popular belief, this is not an example of Divide and Conquer because there is only one sub-problem in each step (Divide and conquer requires that there must be two or more sub-problems) and hence this is a case of Decrease and Conquer.

2.What is binary search and how does it work?

ANS:

Binary Search

Problem: Given a sorted array **arr[]** of **n** elements, write a function to search a given element **x** in **arr[]**.

Examples:

Input: arr[] = {10, 20, 30, 50, 60, 80, 110,

130, 140, 170}, x = 110
Output: 6
Explanation: Element x is present at index 6

Input: arr[] = {10, 20, 30, 40, 60, 110, 120, 130, 170}, x = 175
Output: -1
Explanation: Element x is not present in arr[].

Linear Search Approach: A simple approach is to do a **linear search.** The time complexity of the Linear search is O(n). Another approach to perform the same task is using *Binary Search*.

Binary Search Approach:

Binary Search is a searching algorithm used in a sorted array by **repeatedly dividing the search interval in half**. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

Binary Search Algorithm: The basic steps to perform Binary Search are:

- Begin with an interval covering the whole array.
- If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
- Otherwise, narrow it to the upper half.
- Repeatedly check until the value is found or the interval is empty.

Step-by-step Binary Search Algorithm: We basically ignore half of the elements just after one comparison.

- Compare x with the middle element.

- If x matches with the middle element, we return the mid index.

- Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element. So we recur for the right half.

- Else (x is smaller) recur for the left half.

3.Explain the distinction between a list and tuple.
ANS:

Difference Between List and Tuple

List and Tuple in Python are the class of data structure. The list is dynamic, whereas the tuple has static characteristics.

List is just like the arrays, declared in other languages. Lists need not be homogeneous always which makes it the most powerful tool in Python. In Python, the list is a type of container in Data Structures, which is used to store multiple data at the same time. Lists are a useful tool for preserving a sequence of data and further iterating over it.

Syntax:

```
list_data = ['an', 'example', 'of', 'a', 'list']
```

Tuple is also a sequence data type that can contain elements of different data types, but these are immutable in nature. In other words, a tuple is a collection of Python objects separated by commas. The tuple is faster than the list because of static in nature.

Syntax:

```
tuple_data = ('this', 'is', 'an', 'example', 'of
' ,'tuple', )
```

4.Can you explain how Python manages memory?

ANS:

As we know, Python **uses the dynamic memory allocation which is managed by the Heap data structure**. Memory Heap holds the objects and other data structures that will be used in the program. Python memory manager manages the allocation or de-allocation of the heap memory space through the API functions.

5.What is the difference between pickling and unpickling?

ANS:

The pickle module is used for implementing binary protocols for serializing and de-serializing a Python object structure.

- Pickling: It is a process where a Python object hierarchy is converted into a byte stream.

- Unpickling: It is the inverse of Pickling process where a byte stream is converted into an object hierarchy.

6.What are the different types of search algorithms?
ANS:

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored. Based on the type of search operation, these algorithms are generally classified into two categories:

- **Sequential Search**: In this, the list or array is traversed sequentially and every element is checked. For example: Linear Search.

- **Interval Search**: These algorithms are specifically designed for searching in sorted data-structures. These type of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half. For Example: Binary Search.