DIVIDE AND CONQUER ALGORITHM

Divide and Conquer Algorithm is a problem-solving technique used to solve problems by dividing the main problem into subproblems, solving them individually and then merging them to find solution to the original problem.

Advantages of Divide and Conquer Algorithm:

1.Solving difficult problems:Divide and conquer technique is a tool for solving difficult problems conceptually.

2.Algorithm efficiency:The divide-and-conquer algorithm often helps in the discovery of efficient algorithms. It is the key to algorithms like Quick Sort and Merge Sort, and fast Fourier transforms.

Disadvantages of Divide and Conquer Algorithm:

1.Overhead: The process of dividing the problem into subproblems and then combining the solutions can require additional time and resources. This overhead can be significant for problems that are already relatively small or that have a simple solution.

2.Memory limitations: When working with large data sets, the memory requirements for storing the intermediate results of the subproblems can become a limiting factor.

DIVIDE AND CONQUER ALGORITHM IN ASCENDING ORDER

```
def divide_array(array):
    # Divide the array into halves
    midpoint = len(array) // 2
    left_half = array[:midpoint]
    right_half = array[midpoint:]
    return left_half, right_half

def iterate_divide(array):
    divided_array = [array]  # Start with the whole array as the only element in the list
    while any(len(subarray) > 1 for subarray in divided_array):
        new_divided_array = []
        for subarray in divided_array:
            if len(subarray) > 1:
                left_half, right_half = divide_array(subarray)
                new_divided_array.append(left_half)
                new_divided_array.append(right_half)
            else:
                new_divided_array.append(subarray)  # Already a single-element subarray
        divided_array = new_divided_array
    return divided_array

def join_array(array):
  new_joint_array=[]
  for [x] in array:
    new_joint_array.append(x)
  return new_joint_array

# Example usage
array = [1, 2, 3, 4, 5, 6, 7]
```

```
result = iterate_divide(array)
print(result)
joint_array = join_array(result)
print(joint_array)
```

```
[[1], [2], [3], [4], [5], [6], [7]]
[1, 2, 3, 4, 5, 6, 7]
```