Merge Sort

Assignment Solutions

PW SKILLS

1. Given an array of integers, sort it in descending order using merge sort algorithm.

```
def merge_sort_descending(arr):
    if len(arr) > 1:
        mid = len(arr) // 2  # Finding the mid of the array
        left_half = arr[:mid]  # Dividing the elements into 2 halves
        right_half = arr[mid:]

        merge_sort_descending(left_half)  # Sorting the first half
        merge_sort_descending(right_half)  # Sorting the second half

        i = j = k = 0

        # Copy data to temp arrays left_half[] and right_half[]
        while i < len(left_half) and j < len(right_half):
            if left_half[i] > right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

def main():
    arr = [12, 11, 13, 5, 6, 7]
    print("Given array is")
    print(arr)
    merge_sort_descending(arr)
    print("Sorted array in descending order is")
    print(arr)

if __name__ == "__main__":
    main()
```

2. Reverse Pairs (Leetcode Problem) : Given an integer array nums, return the number of reverse pairs in the array.

A reverse pair is a pair (i, j) where:
0 <= i < j < nums.length and
nums[i] > 2 * nums[j].

```
class Solution:
    def reversePairs(self, nums: List[int]) → int:
        self.count = 0
        def merge(left_arr, right_arr):
            # Step 1: Count the index satisfied condition first
            i, j = 0, 0
            # note that i is index of left_arr, j is index of right_arr .
            while i < len(left_arr) and j < len(right_arr):
                if left_arr[i] > 2 * right_arr[j]:
                    self.count += len(left_arr) - i
                    j += 1
                else:
                    i += 1
            # Step 2: Merge sort left and right
            l, r = 0, 0
            res = []
            while l < len(left_arr) and r < len(right_arr):
                if left_arr[l] < right_arr[r]:
                    res.append(left_arr[l])
                    l += 1
                else:
                    res.append(right_arr[r])
                    r += 1
            return res + left_arr[l:] + right_arr[r:]
        def divide(arr):
            if len(arr) ≤ 1: return arr
            # divide the input array into 2 parts
            mid = len(arr)//2
            left_arr = divide(arr[:mid])
            right_arr = divide(arr[mid:])
            return merge(left_arr, right_arr)
        nums = divide(nums)
        return self.count
```