Reverse Pairs Problem (Merge Sort Approach)

The Reverse Pairs problem requires counting pairs (i, j) such that i < j and arr[i] > 2 * arr[j]. We solve it efficiently using a modified Merge Sort approach: 1. **Divide** the array into two halves recursively. 2. **Count** reverse pairs across left and right halves before merging. - For each element in the left half, find how many elements in the right half satisfy arr[i] > 2*arr[j]. 3. **Merge** the two halves in sorted order (like normal merge sort). 4. **Return** the total count. ### Time Complexity: - O(n log n), because each level of recursion counts pairs in O(n) and merge sort has log n levels. ### Space Complexity: - O(n), due to temporary array used in merging.

```
class Solution:
    def countPairs(self, arr, low, mid, high):
        count = 0
        right = mid + 1
        for i in range(low, mid + 1):
            while right <= high and arr[i] > 2 * arr[right]:
                right += 1
            count += (right - (mid + 1))
        return count

    def merge(self, arr, low, mid, high):
        temp = []
        left = low
        right = mid + 1

        while left <= mid and right <= high:
            if arr[left] <= arr[right]:
                temp.append(arr[left])
                left += 1
            else:
                temp.append(arr[right])
                right += 1

        while left <= mid:
            temp.append(arr[left])
            left += 1

        while right <= high:
            temp.append(arr[right])
            right += 1

        for i in range(len(temp)):
            arr[low + i] = temp[i]

    def mergeSort(self, arr, low, high):
        count = 0
        if low < high:
            mid = (low + high) // 2
            count += self.mergeSort(arr, low, mid)
            count += self.mergeSort(arr, mid + 1, high)
            count += self.countPairs(arr, low, mid, high)
            self.merge(arr, low, mid, high)
        return count

    def countRevPairs(self, arr):
        return self.mergeSort(arr, 0, len(arr) - 1)
```