REVERSE PAIRS

☆ In this problem, similar to counting inversions we are supposed to count number of pairs where left element is greater than double of right element.

Brute force solution is to obviously check for all pairs by using 2 loops. Time Complexity is $O(N^2)$

Optimal solution is to use a solution similar to our counting pairs solution with slight modification When an element on the left fails to form a pair with an element on the right, we increment left but we let right stay as we know right will definitely form pairs

Pseudocode :

```
mergeSort(arr, low, high) {
    if (low >= high) return
    mid = (low + high) | 2
    mergeSort(arr, low, mid)
    mergeSort(arr, mid + 1, high)
    merge(arr, low, mid, high)
}
merge(arr, low, mid, high) {
    left = low
    right = mid + 1
    int ans []
    while (left <= mid && right < high) {
        if (arr [left] > 2 * arr [right]) {
            count + = mid - left + 1
            right ++
        } else {
            left ++
        }
    }
    left = low
    right = mid + 1
    while (left <= mid && right < high) {
        if (arr [left] > = arr [right]) {
            ans. append (arr [right])
            right ++ , count + = (mid - left + 1)
        } else {
            ans. append (arr [left])
            left ++
        }
    }
    while (left <= mid) {
        ans. append (arr [left])
        left ++
    }
    while (right < high) {
        ans. append (arr [right])
        right ++
    }
    return ans
}
count = 0
reverse Pairs (arr , N) {
    merge Sort (arr , 0 , N - 1)
    return count
}
```