3.Hard\11.Reverse_pairs.cpp

```cpp
/*
QUESTION:
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].

Example:
Input: nums = [1,3,2,3,1]
Output: 2
Explanation: The reverse pairs are:
(1, 4) --> nums[1] = 3, nums[4] = 1, 3 > 2 * 1
(3, 4) --> nums[3] = 3, nums[4] = 1, 3 > 2 * 1

APPROACH:
To solve this problem, we can use the merge sort algorithm. While merging the two sorted subarrays, we can count the number of reverse pairs.

1. Define a variable 'rev_pair' to store the count of reverse pairs.
2. Implement the 'merge' function to merge two subarrays and count the reverse pairs.
3. Implement the 'mergesort' function to recursively divide the array into subarrays and perform merge sort.
4. Initialize 'rev_pair' to 0 and call the 'mergesort' function on the given array.
5. Return the 'rev_pair' as the result.

CODE:
*/

int rev_pair = 0;

void merge(int start, int mid, int end, vector<int>& nums){
    int left_size = mid - start + 1;
    int right_size = end - mid;
    vector<int> left(left_size);
    vector<int> right(right_size);

    for(int i = 0; i < left_size; i++){
        left[i] = nums[start + i];
    }
    for(int i = 0; i < right_size; i++){
        right[i] = nums[mid + 1 + i];
    }

    // main logic resides here
    int m = 0;
    for(int i = 0; i < left_size; i++){
        while(m < right_size && left[i] > (long long)2 * right[m]){
            m++;
        }
        rev_pair += m;
    }

```

```cpp
    int i = 0, j = 0, k = start;
    while(i < left_size && j < right_size){
        if(left[i] > right[j]){
            nums[k++] = right[j++];
        }
        else{
            nums[k++] = left[i++];
        }
    }
    while(i < left_size){
        nums[k++] = left[i++];
    }
    while(j < right_size){
        nums[k++] = right[j++];
    }
}

void mergesort(int start, int end, vector<int>& nums){
    if(start >= end)
        return;
    int mid = start + (end - start) / 2;
    mergesort(start, mid, nums);
    mergesort(mid + 1, end, nums);
    merge(start, mid, end, nums);
}

int reversePairs(vector<int>& nums) {
    rev_pair = 0;
    mergesort(0, nums.size() - 1, nums);
    return rev_pair;
}

/*
TIME COMPLEXITY: O(n log n), where n is the size of the array.
SPACE COMPLEXITY: O(n), where n is the size of the array.
*/

```