A4

#include <iostream>  
#include <vector>  
  
// IN-PLACE MERGE SORT  
  
int merge(std::vector<int>& arr, size\_t left, size\_t mid, size\_t right) {  
 // Индекс начала второй "половины" для слияния  
 size\_t left\_2 = mid + 1;  
  
 int cnt = 0;  
 while (left <= mid && left\_2 <= right) {  
 // Первые элементы в правильном порядке  
 if (arr[left] <= arr[left\_2] || arr[left] <= 2 \* arr[left\_2]) left++;  
 else {  
 // Вставляем на правильное место left элемент arr[left\_2]  
 // Запоминаем его и его индекс  
 int value = arr[left\_2];  
 size\_t ind = left\_2;  
  
 // Сдвигаем все элементы между left и left\_2 вправо  
 // и записываем на место left элемент arr[left\_2]  
 while (ind != left) {  
 arr[ind] = arr[ind - 1];  
 ind--;  
 if (2 \* value < arr[ind]) {  
 ++cnt;  
 }  
 }  
 arr[left] = value;  
  
 // Передвигаем все рабочие индексы  
 left++;  
 mid++;  
 left\_2++;  
 }  
 }  
 return cnt;  
}  
  
// Основной метод сортировки  
int mergeSort(std::vector<int>& arr, size\_t left, size\_t right) {  
 int cnt = 0;  
 if (left < right) {  
 size\_t mid = left + (right - left) / 2;  
 cnt = mergeSort(arr, left, mid) + mergeSort(arr, mid + 1, right);  
 cnt += merge(arr, left, mid, right);  
 }  
 return cnt;  
}  
  
int main() {  
 int n;  
 std::cin >> n;  
 std::vector<int> v(n);  
 for (auto &i : v) {  
 std::cin >> i;  
 }  
 std::cout << "Count: " << mergeSort(v, 0, v.size() - 1);  
 return 0;  
}

Divide: Разделить последовательность из n элементов на две по n/2 элементов в каждой

Conquer: Рекурсивно отсортировать две полученные подпоследовательсноти

Combine: Объединить две отсортированные последовательности и совершить подсчет перестановок.

T(n) = 2T() + Θ(n)