Problem A. Add All (version 2025)

Source file name:	Addall.c, Addall.cpp, Addall.java, Addall.py
Input:	Standard
Output:	Standard
Time / Memory limit:	4/9/6 (C++/Java/Python) second(s) / 128 megabytes
Author(s):	Hugo Humberto Morales & Gabriel Gutiérrez - UTP Colombia

Professor Humbertov Moralov, while organizing the papers in his office, found the following solution that he had proposed approximately 10 years ago for the programming challenge **UVa - 10954 - Add All:**

```c
#include <stdio.h>
#define MAXT 5000

int ExtractMin(int A[], int *n)
{
    int temp, i, min;
    for(i = *n - 1; i >= 1; i--)
    {
        if(A[*n] > A[i])
        {
            temp = A[i];
            A[i] = A[*n];
            A[*n] = temp;
        }
    }
    min = A[*n];
    *n = *n - 1;
    return min;
}

void Insert(int A[], int *n, int element)
{
    *n = *n + 1;
    A[*n] = element;
}

int AddAll(int A[], int n)
{
    int result = 0, value1, value2, value3;
    while(n >= 2)
    {
        value1 = ExtractMin(A, &n);
        value2 = ExtractMin(A, &n);
        value3 = value1 + value2;
        result = result + value3;
        Insert(A, &n, value3);
    }
    return result;
}

int main()
{
    int n, A[MAXT + 1], index;
    while(scanf("%d", &n) && (n > 0))
    {
        for(index = 1; index <= n; index++)
            scanf("%d", &A[index]);
        printf("%d\n", AddAll(A, n));
    }
    return 0;
```

```
}
```

In the previous solution, we work with an array $A[\ ]$ in which n elements are initially stored. In the function `AddAll`, within the `while` repetition loop, the two smallest values are extracted (and removed) from the array, added together, and this value is inserted (stored) in the array. This value is also used to update the total sum in the variable `result`, which stores the result of adding all pairs of smallest values in the array as long as it contains 2 or more values. The computational cost of the `AddAll` function is $O(n^2)$, being the number n the amount of elements stored in the array.

The `while` loop is executed $n-1$ times, because for each iteration there is one number less stored on the array. Each iteration has a total cost of $O(n)$, because the array must be traversed to determine the minimum value in other words: $O((n-1)\cdot n) = O(n^2)$.

But $O(n^2)$ is a computationally very expensive solution, and the only reason why the UVa Online Judge currently gives an `accepted` verdict is because it is a very old programming challenge (from 2005), and at that time a size of n=5,000n = 5,000 n=5,000 was sufficient for a time of 1 or 2 seconds with the computing power of the servers of that time.

Nowdays (year 2025) a solution on the order of 10^8 operations runs in 1 or 2 seconds on online judges. For this reason, you are asked as a student of a Computer Science academic program to propose a solution that in the worst case runs in a complexity of $O(n \cdot \log n)$ per test case, where the function `AddAll` needs to be efficiently programmed for the new size of variable n which is now 10^6.

Input

The input begins with a positive integer t $(1 \leq t \leq 5)$, denoting the number of test cases. Each test case consists of two lines. The first line contains a positive integer n $\left(2 \leq n \leq 10^6\right)$, which represents the number of elements to store in the array $A[\ ]$. The second line contains n positive integers (each greater than or equal to 1 and less than or equal to 10^6).

It is guaranteed that the sum of all n values across the t test cases is at most 10^6.

Output

For each test case, print a single line with the result produced by the `AddAll` function.

Example

Input	Output
5 3 1 2 3 4 1 2 3 4 5 5 4 3 2 1 5 3 2 5 1 4 5 1 1 1 1 1	9 19 33 33 12

Use fast I/O methods