**Minimize the sum of product**

**Easy**

You are given two arrays,**A** and **B**, of equal size **N**. The task is to find the minimum value of **A[0] \* B[0] + A[1] \* B[1] + .... + A[N-1] \* B[N-1],** where shuffling of elements of arrays **A** and **B** is allowed.  
  
  
**Example 1:**

**Input:**

N = 3

A[] = {3, 1, 1}

B[] = {6, 5, 4}

**Output:**

23

**Explanation:**

1\*6+1\*5+3\*4 = 6+5+12

= 23 is the minimum sum

**Example 2:**

**Input:**

N = 5

A[] = {6, 1, 9, 5, 4}

B[] = {3, 4, 8, 2, 4}

**Output:**

80

**Explanation:**

2\*9+3\*6+4\*5+4\*4+8\*1

=18+18+20+16+8

= 80 is the minimum sum

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **minValue()** which takes the arrays **A[]**, **B[]** and its size **N**as inputs and returns the minimum sum.

**Expected Time Complexity:** O(N. log(N))  
**Expected Auxiliary Space:** O(1)  
  
  
**Constraints:**  
1 ≤ N ≤ 105  
1 ≤ A[] ≤ 106

//{ Driver Code Starts

//Initial Template for Java

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class GFG {

public static void main(String[] args) throws IOException

{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(br.readLine().trim());

while(t-->0)

{

int n = Integer.parseInt(br.readLine().trim());

long a[] = new long[(int)(n)];

long b[] = new long[(int)(n)];

String inputLine[] = br.readLine().trim().split(" ");

for (int i = 0; i < n; i++) {

a[i] = Long.parseLong(inputLine[i]);

}

String inputLine1[] = br.readLine().trim().split(" ");

for (int i = 0; i < n; i++) {

b[i] = Long.parseLong(inputLine1[i]);

}

Solution obj = new Solution();

System.out.println(obj.minValue(a, b, n));

}

}

}

// } Driver Code Ends

class Solution {

public long minValue(long a[], long b[], long n)

{

// Your code goes here

long sum=0;

Arrays.sort(a);

Arrays.sort(b);

for(int i=0;i<n;i++){

sum+=a[i]\*b[(int)(n-i-1)];

}

return sum;

}

}