Day 80 coding Statement :

Alice and Bob went to a pet store. There are *N* animals in the store where the *ith* animal is of type *Ai*?.

Alice decides to buy some of these *N* animals. Bob decides that he will buy **all** the animals **left** in the store after Alice has made the purchase.

Find out whether it is possible that Alice and Bob end up with **exactly same multiset** of animals.

Input Format

- The first line of input will contain a single integer *T*, denoting the number of test cases.
- Each test case consists of multiple lines of input.
    - The first line of each test case contains an integer *N* — the number of animals in the store.
    - The next line contains *N* space separated integers, denoting the type of each animal.

Output Format

For each test case, output on a new line, YES, if it is possible that Alice and Bob end up with **exactly same** multiset of animals and NO otherwise.

You may print each character in uppercase or lowercase. For example, the strings YES, yes, Yes, and yES are considered identical.

Sample Input

4

3

4 4 4

4

2 3 3 2

4

1 2 2 3

6

5 5 1 5 1 5

Sample Output

NO

YES

NO

YES

```
import java.util.*;
import java.util.ArrayList;
import java.io.*;

class TestClass {
	static class FastReader {
		BufferedReader br;
		StringTokenizer st;

		public FastReader() {
			br = new BufferedReader(new InputStreamReader(System.in));
		}

		String next() {
			while (st == null || !st.hasMoreElements()) {
				try {
					st = new StringTokenizer(br.readLine());
				} catch (IOException e) {
					e.printStackTrace();
				}
			}
			return st.nextToken();
		}

		int nextInt() {
			return Integer.parseInt(next());
		}

		long nextLong() {
			return Long.parseLong(next());
		}

		double nextDouble() {
			return Double.parseDouble(next());
```

```
		}

		double nextFloat() {
			return Float.parseFloat(next());
		}

		String nextLine() {
			String str = "";
			try {
				str = br.readLine();
			} catch (IOException e) {
				e.printStackTrace();
			}
			return str;
		}
	}

	public static void main(String[] sadf) {
		FastReader fr = new FastReader();
		int t = fr.nextInt();
		while (t-- > 0) {
			solve(fr);
		}
	}

	public static void solve(FastReader fr) {
		int n = fr.nextInt();
		HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();
		for (int i = 0; i < n; i++) {
			int num = fr.nextInt();
			map.put(num, map.getOrDefault(num, 0) + 1);
		}
		for (Map.Entry<Integer, Integer> e : map.entrySet()) {
			if (e.getValue() % 2 != 0) {
				System.out.println("NO");
				return;
			}
		}
		System.out.println("YES");
	}

	private static int log(int N) {
		return 31 - Integer.numberOfLeadingZeros(N);
	}
}
```