题目描述：

某部门计划通过结队编程来进行项目开发，已知该部门有 N 名员工，每个员工有独一无二的职级，每三个员工形成一个小组进行结队编程，结队分组规则如下：

从部门中选出序号分别为 i、j、k 的 3 名员工，他们的职级分别为 level[i]、level[j]、level [k]

结队小组需满足： level [i] < level [j] < level [k] 或者 level [i] > level [j] > level [k] ，**其中 0 <= i < j < k < n**

请你按上述条件计算可能组合的小组数量。同一员工可以参加多个小组。

输入描述：
第一行输入：员工总数 n
第二行输入：按序号依次排列的员工的职级 level，中间用空格隔开

限制：

1 <= n <= 6000

1 <= level[i] <= 10^5

输出描述：
可能组合的小组数量
示例 1
输入：
4
1 2 3 4
输出：
4
说明：
可能结队成的组合 (1,2,3)、(1,2,4)、(1,3,4)、(2,3,4)
示例 2
输入：
3
5 4 7
输出：
0
说明：
根据结队条件，我们无法为该部门组建小组

```
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
```

```
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        int[] levels = new int[n];
        for (int i = 0; i < n; i++) {
            levels[i] = in.nextInt();
        }

        int result = 0;
        result += escResult(levels);
        result += descResult(levels);

        System.out.println(result);

    }

    // 暴力求解，升序
    static int escResult(int[] levels) {
        int result = 0;
        int len = levels.length;
        for (int i = 0; i < len - 2; i++) {
            int a = levels[i];
            for (int j = i + 1; j < len - 1; j++) {
                int b = levels[j];
                if (a >= b) continue;
                for (int k = j + 1; k < len; k++) {
                    int c = levels[k];
                    if (b >= c) continue;;
                    result++;
                }
            }
        }

        return result;
    }


    static int descResult(int[] levels) {
        int result = 0;
        int len = levels.length;
        for (int i = 0; i < len - 2; i++) {
            int a = levels[i];
```

```
                for (int j = i + 1; j < len - 1; j++) {
                    int b = levels[j];
                    if (a <= b) continue;
                    for (int k = j + 1; k < len; k++) {
                        int c = levels[k];
                        if (b <= c) continue;;
                        result++;
                    }
                }
            }

            return result;
        }
    }
```