Problem 5 - Intersecting Triangles (100 pts)

Time Limit	:	3 s
Memory Limit	:	1024 MB

Problem Description

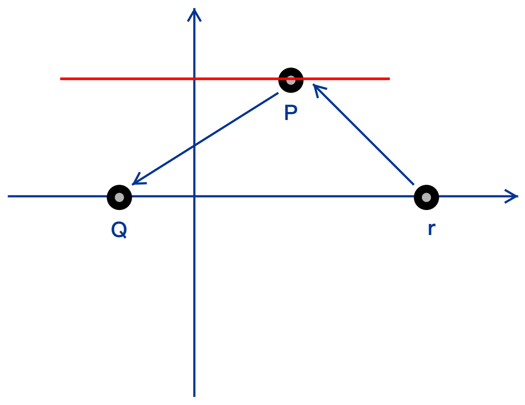
There are N ants who live on a 2D plan. The i-th ant rests on a point p_i in the line of $y = 1$ during the night. Then, at some starting time, the ant first walks by a straight line to another point q_i in the line of $y = 0$ to start working. After going to q_i, the ant starts working by going straight from point q_i to r_i, both in the line of $y = 0$. Then, at some ending time, the ant walks back to p_i from r_i by a straight line. That is, the path of each ant follows a triangle (p_i, q_i, r_i). Each different ant can have a different walking speed, a different starting time, and/or a different end time.

The Queen Ant recently received some complaints from the ants, saying that some ants bump into each other in their daily triangular path. To understand how serious the situation is, the Queen Ant asks the Scientist Ant to list, from the paths of all ants, which ants may bump into each other—at the vertex of the triangular path, at one other point of the path, on a segment of the path. Please help the scientist determine the number of **pairs** of ants that may bump into each other on their triangular paths.

Input

The input can be read through the provided header file `generator.h`. Please first call `generator.init()` to initialize the generator, then calling `generator.getT()` will return the number of test cases T. For each test case, calling `generator.getData(&N, &P, &Q, &R)` will let you get the number of triangles N, and their vertices. Note that the type of `N` should be `int`, and the type of `P, Q, R` should be `int*`. After calling, `P[0..N-1]` would store $\{p_1, p_2, \ldots, p_N\}$, and `Q[0..N-1]`, `R[0..N-1]` will store the other points, respectively. Below is an example usage, the program will run and get the data correctly; however, you will get WA if you submit it directly:

```
#include <stdio.h>
#include "generator.h"
int main() {
    generator.init();
    int t = generator.getT();
    while (t--) {
        int n, *p, *q, *r;
```



```
        generator.getData(&n, &p, &q, &r);
        /* do something
        int ans = 0;
        for (int i = 0; i < n; i++) ans += p[i] * q[i] * r[i];
        printf("%d\n", ans);
        */
    }
}
```

Note that you should not read anything from standard input in your program; otherwise, the behavior is undefined.
You can download the generator header "generator.h" here.

Output

For each test case, print the number of pairs of intersecting triangles.

Subtasks

In all subtasks, $-2^{20} \leq p_i, q_i, r_i \leq 2^{20} - 1$ holds for each $1 \leq i \leq N$.

Subtask 1 (20 pts)

- $1 \leq N \leq 3000$, $1 \leq T \leq 10$

Subtask 2 (20 pts)

- $1 \leq N \leq 10^5$, $1 \leq T \leq 10$
- all p_i are distinct.
- all q_i and r_i are distinct.

Subtask 3 (10 pts)

- $1 \leq N \leq 10^5$, $1 \leq T \leq 10$
- all p_i are distinct.

Subtask 4 (10 pts)

- $1 \leq N \leq 10^5$, $1 \leq T \leq 10$

Subtask 5 (40 pts)

- $1 \le N \le 3 \times 10^6$, $T = 1$

Sample Cases

Sample Input 1

```
1 1 5 16
538724387836423741 325591348600219474 187178394057222755 353408734984306357
```

Actual Sample Input 1

```
T = 1
N = 5
P = {12, 7, -10, 12, 9}
Q = {11, 5, 5, -16, 5}
R = {-2, -13, -8, 8, 10}
```

Sample Output 1

```
10
```

Sample Input 2

```
2 1 5 16
51414933668525662 550301789874357166 622479167726386043 650347521267739593
```

Actual Sample Input 2

```
T = 1
N = 5
P = {2, -3, 6, -16, 0}
Q = {-7, -14, -5, 1, 6}
R = {-12, 5, 9, 13, -8}
```

Sample Output 2

```
9
```

Sample Input 3

```
3 1 5 16
203739077647024131 805985539835675567 140205801930598907 908190957489194415
```

Actual Sample Input 3

```
T = 1
N = 5
P = {2, 14, -16, -7, 13}
Q = {7, 14, 0, 7, -5}
R = {11, -12, 8, -12, -1}
```

Sample Output 3

```
10
```

Sample Input 4

```
1 2 10 16
869534322540300934 837268419296844257 456729939812480767 541019751318820673
```

Actual Sample Input 4

```
T = 2
N = 10
P = {-5, 15, 15, -8, -11, -5, -13, -14, 15, 10}
Q = {-7, -12, -4, -13, 5, 1, 14, -16, -16, -14}
R = {2, -13, 1, 9, 9, 1, 1, 12, -15, 6}
N = 10
P = {7, -4, -7, 12, 9, -13, -11, 5, 3, 8}
Q = {8, 5, 3, -4, -10, -10, 2, 2, -7, -9}
R = {-11, -1, -1, -12, 4, 13, -16, -3, -7, 4}
```

Sample Output 4

```
45
44
```