NICEQUAD - Nice Quadrangles

There are n points with integer coordinates. We can form different quadrangles out of them by taking four different points and connecting them with lines. Let's call a quadrangle ABCD nice if and only if:

1. $A_x > 0$ and $A_y > 0$;
2. $B_x > 0$ and $B_y < 0$;
3. $C_x < 0$ and $C_y < 0$;
4. $D_x < 0$ and $D_y > 0$;
5. ABCD has an integer area.

Your task is to count all different nice quadrangles that can be formed on the given points.

The first line of input file contains number t – the number of test cases. Then the description of each test case follows. The first line of each test case contains number n – the number of points. Then n lines follow each consisting of two integers x, y – the coordinates of a point. No two points in the same test case coincide.

$1 <= t <= 10$; $1 <= n <= 30000$; $-30000 <= x, y <= 30000$

For each test case print the number of nice quadrangles that can be formed using given points.

Explanation:

To detect whether the quadrangle has an integer area we will use the Pick theorem which states that the area of a grid polygon can be calculated in the following way S = W + I/2 - 1. Where W is the amount of lattice points strictly inside the polygon and I is the amount of lattice points on its edge. So the area S will be integer if the amount of lattice points on its edge will be even. Now how we will count the amount of lattice points on the edge of a given quadrangle.

Let us consider any quadrangle ABCD the edges of which don't intersect. Let's look at its edge AB. The amount of lattice points lying on that edge can be calculated as $gcd(|A_x - B_x|, |A_y - B_y|) + 1$, where gcd is the greatest common divisor. Or it will be $gcd(|A_x - B_x|, |A_y - B_y|) - 1$ if we don't count the points on the ends of segment AB.

So now if we count the same value for each edge we will have the following expression for-

$$I = \gcd(|A_x - B_x|, |A_y - B_y|) + \gcd(|B_x - C_x|, |B_y - C_y|) + \gcd(|C_x - D_x|, |C_y - D_y|) + \gcd(|D_x - A_x|, |D_y - A_y|) \ldots\ldots \quad (1).$$

But we only need to each if that value is odd or even. For two given non-negative integers gcd(a, b) is even iff a and b are both even. So $\gcd(|A_x - B_x|, |A_y - B_y|)$ is even iff $|A_x - B_x|$ and $|A_y - B_y|$ are both even. Now $|A_x - B_x|$ is even iff A_x and B_x are both even of both odd. So $\gcd(|A_x - B_x|, |A_y - B_y|)$ is even when the A_x and B_x have the same remainder modulo 2 and A_y and B_y have the same remainder modulo 2. That means that we don't need to know the exact value of A_x, A_y, B_x, B_y, but only their remainders modulo 2. That is true for all other points as well. After we find if any of the gcds in eq(1) are odd or even we can easily determine if I is odd or even.

Next is how to count all the quadrangles with the stated quality. First we need to split the given point into four groups: the first group with $x > 0$ and $y > 0$, the second – $x > 0$ and $y < 0$, the third $x < 0$ and $y < 0$ and the fourth – $x < 0$ and $y > 0$. Then in each group we need to split all points of the group into four groups again: the first group with x even and y even, the second – x even and y odd, the third – x odd and y even and the fourth – x odd and y odd. We will count the amount of points in each group and each subgroup as well. After that we can iterate over all possible variants of taking points from the group. Surely to form a triangle we have to take one point from each group of the first type. Then we try all possible assignment of points from the subgroups.

There are in total 4^4 different types of quadrangles we can form. For each type we know the amount of quadrangles of that type and we can determine if the area of quadrangles of such type is an integer. So we try all those types and if the area is an integer we add the amount of those quadrangles to the answer.
The complexity of the described algorithm is O(n).

Sample Code:

```
typedef long long ll;
int A[4], B[4], C[4], D[4];
int ptIdx(int x, int y)
{
    x = abs(x); y = abs(y);
    if((x%2==0) && (y%2==0))return 0;
    if((x%2==1) && (y%2==0))return 1;
    if((x%2==0) && (y%2==1))return 2;
    if((x%2==1) && (y%2==1))return 3;
}

bool isNiceQuad(int a, int b, int c, int d)
{
    int g1,g2,g3,g4;

    if(a==b) g1 = 0;
    else g1 = 1;

    if(b==c) g2 = 0;
    else g2 = 1;

    if(c==d) g3 = 0;
    else g3 = 1;

    if(d==a) g4 = 0; // GCD is even
    else g4 = 1;     // GCD is odd

    int I = g1^g2^g3^g4;

    if(I%2==0) return 1;
    else return 0;
}

int main()
{
    int t; scanf("%d",&t);
    for (int ks=1; ks <= t; ks++)
    {
        int n; scanf("%d",&n);

        memset(A, 0, sizeof(A));
        memset(B, 0, sizeof(B));
        memset(C, 0, sizeof(C));
        memset(D, 0, sizeof(D));
```

```
        for(int i=0; i < n; ++i)
        {
            int x,y; scanf("%d%d",&x,&y);

            if(x>0 && y>0) A[ptIdx(x,y)]++; // First Quadrant
            if(x>0 && y<0) B[ptIdx(x,y)]++; // Fourth Quadrant
            if(x<0 && y<0) C[ptIdx(x,y)]++; // Third Quadrant
            if(x<0 && y>0) D[ptIdx(x,y)]++; // Second Quadrant
        }

        ll numQuads = 0;

        for(int a=0; a<4; a++)
        for(int b=0; b<4; b++)
        for(int c=0; c<4; c++)
        for(int d=0; d<4; d++)
           if(isNiceQuad(a,b,c,d))
          {
               ll comb = (ll)(A[a]*B[b]) * (ll)(C[c]*D[d]);
               numQuads += comb;
          }

        printf("%lld\n",numQuads);
    }
    return 0;
}
```