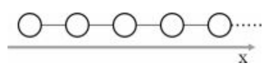
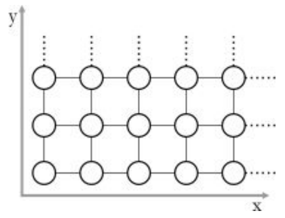
PAIRS

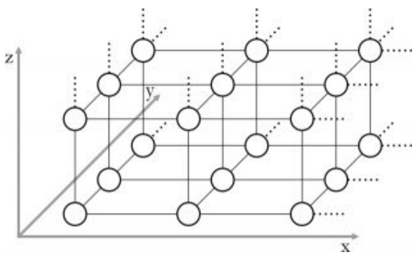
Mirko and Slavko are playing with toy animals. First, they choose one of three boards given in the figure below. Each board consists of cells (shown as circles in the figure) arranged into a one, two or three dimensional grid.



Board 1



Board 2



Board 3

Mirko then places N little **toy animals** into the cells.

The **distance** between two cells is the smallest number of moves that an animal would need in order to reach one cell from the other. In one move, the animal may step into one of the adjacent cells (connected by line segments in the figure).

Two animals can hear each other if the distance between their cells is **at most D**. Slavko's task is to calculate how many **pairs of animals** there are such that one animal can hear the other.

TASK

Write a program that, given the board type, the locations of all animals, and the number D, finds the desired number of pairs.

INPUT

The first line of input contains four integers in this order:

- The board type B ($1 \le B \le 3$);
- The number of animals N ($1 \le N \le 100\,000$);
- The largest distance D at which two animals can hear each other ($1 \le D \le 100\,000\,000$);
- The size of the board M (the largest coordinate allowed to appear in the input):
  - When B=1, M will be at most 75 000 000.
  - When B=2, M will be at most 75 000.
  - When B=3, M will be at most 75.

Each of the following N lines contains B integers separated by single spaces, the coordinates of one toy animal. Each coordinate will be between 1 and M (inclusive).

More than one animal may occupy the same cell.

OUTPUT

Output should consist of a single integer, the number of pairs of animals that can hear each other.

Note: use a 64-bit integer type to calculate and output the result (`long long` in C/C++, `int64` in Pascal).

GRADING

In test cases worth a total of 30 points, the number of animals N will be at most 1 000.

Furthermore, for each of the three board types, a solution that correctly solves all test cases of that type will be awarded at least 30 points.

EXAMPLES

input

```
1 6 5 100
25
50
50
10
20
23
```

output

```
4
```

input

```
2 5 4 10
5 2
7 2
8 4
6 5
4 4
```

output

```
8
```

input

```
3 8 10 20
10 10 10
10 10 20
10 20 10
10 20 20
20 10 10
20 10 20
20 20 10
20 20 20
```

output

```
12
```

Clarification for the leftmost example. Suppose the animals are numbered 1 through 6 in the order in which they are given. The four pairs are:

- 1-5 (distance 5)
- 1-6 (distance 2)
- 2-3 (distance 0)
- 5-6 (distance 3)

Clarification for the middle example. The eight pairs are:

- 1-2 (distance 2)
- 1-4 (distance 4)
- 1-5 (distance 3)
- 2-3 (distance 3)
- 2-4 (distance 4)
- 3-4 (distance 3)
- 3-5 (distance 4)
- 4-5 (distance 3)

SOLUTIONS

Let N be the number of animals, M the size of the board and D the largest distance.

Solution for 1D board

To solve the 1D board, we first sort the coordinates and then perform a sweep-line algorithm. We keep track of two pointers: **head** and **tail**. When the head is pointing to an element with coordinate x, the tail is pointing to the first element with coordinate greater than or equal to x−D.

For each position of head we add head−tail to the total result. As we advance head to the next element, we can easily adjust tail to point to the required element.

The time complexity of this algorithm is O(N log N) for sorting, and O(N) for sweeping.

Solution for 2D board

To solve the 2D version of the problem, we first consider the distance formula:

$$\text{dist}(P, Q) = |\ P.x - Q.x\ | + |\ P.y - Q.y\ |$$

The formula above will resolve to one of the following four formulas:

$$\text{dist}(P, Q) = P.x - Q.x + P.y - Q.y = (P.x + P.y) - (Q.x + Q.y)$$
$$\text{dist}(P, Q) = P.x - Q.x - P.y + Q.y = (P.x - P.y) - (Q.x - Q.y)$$
$$\text{dist}(P, Q) = -P.x + Q.x + P.y - Q.y = (Q.x - Q.y) - (P.x - P.y)$$
$$\text{dist}(P, Q) = -P.x + Q.x - P.y + Q.y = (Q.x + Q.y) - (P.x + P.y)$$

It is easy to see that the distance is always equal to the largest of these four values. As P.x + P.y and P.x − P.y represent the "diagonal coordinates" of point P we substitute:

$$P.d1 := P.x + P.y \text{ and } P.d2 := P.x - P.y.$$

Now, we can rewrite the distance formula in terms of d1 and d2:

$$\text{dist}(P, Q) = \max\{\ P.d1 - Q.d1, P.d2 - Q.d2, Q.d2 - P.d2, Q.d1 - P.d1\ \},$$

or shorter:

$$\text{dist}(P, Q) = \max\{\ |P.d1 - Q.d1|, |P.d2 - Q.d2|\ \}$$

After substitution, we sort all the points by the first coordinate (d1) increasingly and perform a sweep-line algorithm similar to the one-dimensional case. Since each point P between head and tail satisfies the inequality $head.d1 - P.d1 \le D$, we only need to find out for how many of them the inequality $|head.d2 - P.d2| \le D$ is satisfied as well. To calculate that value, we keep all points (their d2 coordinates) between head and tail in either an **interval tree** or a **binary indexed tree** [1] data structure.

The time complexity of the algorithm implemented with a binary indexed tree data structure is O(N log N) for sorting, and O(N log M) for sweeping, where M is the upper bound on the coordinates.

Solution for 3D board

Inspired by our 2D solution we start with the distance formula once again and obtain:

$$\text{dist}(P, Q) = \max\{\ |P.f1 - Q.f1|, |P.f2 - Q.f2|, |P.f3 - Q.f3|, |P.f4 - Q.f4|\ \}, \text{ where}$$
$$P.f1 := P.x + P.y + P.z$$
$$P.f2 := P.x + P.y - P.z$$
$$P.f3 := P.x - P.y + P.z$$

$P.f4 := P.x - P.y - P.z$

Again, we perform a sweep-line algorithm on the f1 coordinate while keeping all of the points between **head** and **tail** in a 3D binary indexed tree in order to count the number of points P satisfying inequalities $|head.f2 - P.f2| \le D$, $|head.f3 - P.f3| \le D$ and $|head.f4 - P.f4| \le D$.

The time complexity of the algorithm is $O(N \log N)$ for sorting, and $O(N \log^3 M)$ for sweeping.

It is worth mentioning that we can use this solution to solve all types of boards. We just assign any constant value (1 for example) to each of the missing coordinates and implement a 3D binary indexed tree using either dynamic memory allocation, or using a one-dimensional array and manually mapping the 3-dimensional space to elements of the array.

References

[1] P. M. Fenwick, A new data structure for cumulative frequency tables, **Software - Practice and Experience** 24, 3 (1994), 327-336, 1994.