UVa Online Judge

11918 Traveler of Gridland

Girdland is a square shaped country of size 2000000000 × 2000000000 units. We will identify any point in this land using Cartesian coordinate system. Using this system the coordinate of the center of Gridland is (0, 0), the coordinate of the lower-left corner is (-1000000000, -1000000000) and upper-right corner is (1000000000, 1000000000). So the x-axis is a horizontal line which divides the land into two equal rectangles and y-axis is a vertical line which divides the land into two equal rectangles. There are roads in the country which goes through the grid lines only i.e. along $x = -1000000000, -999999999, \ldots, -1, 0, 1, \ldots, 999999999, 1000000000$ and $y = -1000000000, -999999999, \ldots, -1, 0, 1, \ldots, 999999999, 1000000000$ So all roads are either a horizontal line or a vertical line having integer distance from axis.

Travelers like this Gridland because of the simplicity of its road network. Each of the roads is so straight and axis parallel.

From any position (a, b), in unit time any traveler can move to

i. $(a + 1, b)$

ii. $(a - 1, b)$

iii. $(a, b + 1)$

iv. $(a, b - 1)$

A traveler cannot move outside the Gridland. She also cannot move to any position which is occupied by a monster. There may be some monsters in Gridland. There are three types of monsters

i. Point monster

ii. Line monster

iii. Rectangle monster

A point monster can occupy only a single point, a line monster can occupy a straight line, and a rectangle monster can occupy a rectangular region.

The position of the point monster can be specified by a pair of integers (u, v), which indicates that the monster occupies the coordinate position (u, v).

The position of a line monster can be specified by two pair of integers (u_1, v_1) and (u_2, v_2). (u_1, v_1) is the one end of the line monster and (u_2, v_2) is the other end of the monster. It is guaranteed that this line will always be axis parallel. The monster occupies the whole line region inclusively.

The position of rectangle monster can be specified by two pairs of integers (u_1, v_1) and (u_2, v_2). (u_1, v_1) is the coordinate of the lower left corner of monster and (u_2, v_2) is the upper right corner of the monster. It is guaranteed that the edges of monster will always be axis parallel. The monster occupies the whole rectangular region inclusively.

Initially a traveler is in a position of coordinate $(sourceX, sourceY)$ in Gridland. She needs to reach the position of coordinate $(destinationX, destinationY)$. You have to calculate the minimum unit time required for her to reach the destination. If it is not possible to reach the destination, you have to output '`Impossible`' (quotes for clarity).

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each test case starts with four integer $sourceX$, $sourceY$, $destinationX$, $destinationY$ ($-1000000000 \leq sourceX, sourceY, destinationX, destinationY \leq 1000000000$). Next line contains three integers M, N and Q, the number of point monsters, the number of line monsters and the number of rectangle monsters. Total number of monsters can be at most 100, i.e. ($0 \leq M + N + Q \leq 100$). Each of the next M lines describe a point monster by two integers u and v ($-1000000000 \leq u, v \leq 1000000000$). Each of the next N lines will describes a line monster by four integers u_1, v_1, u_2 and v_2 ($-1000000000 \leq u_1, v_1, u_2, v_2 \leq 1000000000$). Each of the next Q lines will describes a rectangle monster by four integers u_1, v_1, u_2 and v_2 ($-1000000000 \leq u_1, v_1, u_2, v_2 \leq 1000000000$). Note that one monster can overlap with another. You can safely assume that source and destination is not occupied by any monster.

Output

For each test case, output a single line in the format '`Case` C`:` N', where C will be replaced by the case number and N will be replaced by the shortest path distance from source to destination. If it is not possible to reach source to destination replace 'N' by '`Impossible`' without quotes.

Sample Input

```
1
1 2 4 3
1 1 1
4 6
4 1 7 1
2 1 3 5
```

Sample Output

```
Case 1: 14
```