Collect the Coins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

There are 10^9 cells arranged in a row, numbered from 1 to 10^9 from left to right. Two robots are patrolling in the cells. The maximum speed of each robot is v cells per second (v is an integer), indicating that if a robot is currently in cell p, it can move to any cell p' in the next second, as long as $1 \le p' \le 10^9$ and $|p' - p| \le v$.

There will be n coins appearing in the cells. The i-th coin will appear at the t_i-th second in cell c_i. If a robot is in the same cell at that time, it can pick up the coin. Otherwise the coin will instantly disappear.

More formally, during each second, the following steps will happen in order:

- Each robot can move to a position no more than v cells away (it is OK to just stay in its current cell).
- Coins appear in the cells.
- If at least one robot is in the same cell with a coin, that coin is collected.
- All uncollected coins disappear.

Your task is to determine the initial positions of two robots before the 1-st second, and move them wisely to collect all the coins with the smallest possible v. Output this optimal value of v.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^6$) indicating the number of coins appearing in the cells.

For the following n lines, the i-th line contains two integers t_i and c_i ($1 \le t_i, c_i \le 10^9$) indicating the time when the i-th coin appears and the position where the i-th coin appears. It's guaranteed that $t_i \le t_{i+1}$ for all $1 \le i < n$. It's also guaranteed that $t_i \ne t_j$ or $c_i \ne c_j$ for all $i \ne j$.

It's guaranteed that the sum of n of all test cases will not exceed 10^6.

Output

For each test case, output one line containing one integer, indicating the smallest possible maximum speed of the robots. If it's impossible to collect all the coins, output `-1` instead.

Example

standard input	standard output

3
5
1 1
3 7
3 4
4 3
5 10
1
10 100
3
10 100
10 1000
10 10000
	2
	0
	-1