

**Czech ACM Student Chapter**

Charles University in Prague
Slovak University of Technology
Masaryk University

**Czech Technical University in Prague**

Technical University of Ostrava
University of Žilina
University of West Bohemia



**CTU Open Contest 2017**

# Punching Power

`power.c`, `power.cpp`, `power.c11`, `Power.java`, `power.py`

The park management finally decided to install some popular boxing machines at various strategic places in the park. In fact, to compensate for the previous lack of machines, they decided to install as many machines as possible. Surprisingly enough, the park is not going to be choked with new machines because there are some quite serious legal limitations regarding the locations of the machines. The management has marked all possible boxing machine locations and their respective coordinates on the park plan. Additionally, they also have to respect manufacturer security rule: The distance between any two boxing machines has to be at least 1.3 meters.

Help the management to establish the maximum possible number of boxing machines which can be installed in the park.

## Input Specification

There are several test cases. Each case starts with a line containing one integer $N$ which specifies the number of possible boxing machine locations in the park ($1 \leq N \leq 2000$). Next, there are $N$ lines representing the location coordinates, each line describes one location by a pair of integer coordinates in meters. All locations in one test case are unique. Each coordinate is non-negative and less than or equal to $10^9$.

You are guaranteed that all locations form a single connected group, that is, it is possible to start in any location and reach any other location by a sequence of steps, each of which changes exactly one coordinate by 1, without leaving the area suitable for placing boxing machines.

## Output Specification

For each test case, print a single line with one integer representing the maximum number of boxing machines which can be installed in the park.

### Sample Input

```
4
0 0
0 1
1 0
1 1
6
0 1
1 0
1 1
1 2
1 3
2 2
```

### Output for Sample Input

```
2
4
```