Threefold Problem Sheet – Winter Games

This sheet presents three problems based on a common storyline. The three problems share several characteristics and as a result they appear quite similar in nature. But this does not necessarily mean that these problems can be solved using the same strategies and techniques. In fact, the whole idea of these threefold sheets is to highlight how possibly subtle differences and changes in a problem formulation can have a great impact on how the resulting problem can be solved. We believe that studying these sheets is an excellent practice of how to approach the modeling aspects of solving a problem.

Read through the problems and try to work out by yourself possible lines of modeling and then algorithmically solving them. Give reasons of why or why not certain approaches appear plausible. Ideally, substantiate your ideas by sketching corresponding proof ideas.

This sheet was handed out during the tutorial on November 23, 2016, and solved and discussed there as well.

Main Story

The Winter Games will start soon and preparations enter the final stage. One of the tasks that still have to be completed is the preparation of the race tracks for the alpine skiing events.

To make sure that there is enough snow, the organizers use "snow cannons". The model T60 from algoALPINE consumes only 18.5 kW and uniformly distributes the snow in a disk around its position. The radius of operation can be set to an arbitrary positive value. However, due to an unfortunate bug in the software, it must be the same for all T60s used.

The organizers also have a bunch of snowcats (PistenBully 600, with 400 hp and fuel consumption starting at 20 l/h) which can be used to move the generated snow around and to flatten the track. This is especially useful when the operating ranges of two snow cannons overlap, as there is too much snow in such a region.

The huge power consumption that is incurred by the snow production upsets environmental friendly people around the world. In particular it has been criticized that the snow cannons produce too much snow in overlapping areas that must then be removed with the snowcats. To settle down the uprising, the organizers have promised that they will operate the snow cannons only in such a way that *no overlap occurs*. Two ranges *overlap* if they have a common interior point. That is, two disks may touch but not properly intersect. The remaining holes will then be filled up with snow that has been stored from the last winter season.

Skiing requires steep slopes. But for simplicity we choose a flat-folded model, in which the slope appears as a region in $\mathbb{R}^2$.

Exercise – *Downhill course*

One of the most important skiing events is the Downhill. The organizers have placed all available snow cannons along and around the race course. An external consultant was hired to determine the optimal radius of operation r for the cannons. (Recall that r must be the same for all T60 snow cannons.)

The organizers have taken care to place the snow cannons in such a way that the operation range of each snow cannon overlaps with the operation range of at most two other cannons. However, this may not be enough to fulfil the promises that were made to the environmental organizations. It is your job to determine the maximum number of snow cannons that can be used simultaneously.

Input The first line contains an integer $1 \leq t \leq 30$, denoting the number of test cases. Each of the following t testcases is described as follows: It starts with a line that contains two integers n, r, denoting the number of snow cannons ($1 \leq n \leq 10,000$) and the operation range ($0 < r < 2^{24}$). The following n lines each contain two integers x_i, y_i, denoting the position (x_i, y_i) of the i-th snow cannon ($|x_i|, |y_i| \leq 2^{24}$).

Output For each test case output a single line containing one number: the maximum number of snow cannons that can be used simultaneously.

Sample Input

```
2
2 1
0 0
2 0
2 2
0 0
2 0
```

Sample Output

```
2
1
```

Exercise – *Use them all!*

As with many events of this scale, costs are higher than expected (and promised). The organizers have to ask the hosting city for more money, already for the second (and probably not the last) time. They expect some serious questioning from the politicians and do not want to leave the impression that the money is not well spent.

The need to invest into thousands of snow cannons does not appear so convincing anymore, given that so many of them are not used. To prevent this from developing into a PR desaster, the organizers decided that it must be possible to operate all snow cannons simultaneously. It is your job to determine the largest integral operating radius for the snow cannons such that no overlap occurs. Recall that this radius must be the same for all T60 cannons.

Input The first line contains an integer $1 \leq t \leq 30$, denoting the number of test cases. Each of the following t testcases is described as follows: It starts with a line that contains one integer n, denoting the number of snow cannons ($2 \leq n \leq 90,000$). The following n lines each contain two integers x_i, y_i, denoting the position (x_i, y_i) of the i-th snow cannon ($|x_i|, |y_i| \leq 2^{24}$).

Output For each test case output a single line containing one number: the largest integral operating radius for the snow cannons such that there are no two cannons whose ranges of operation overlap.

Sample Input

```
2
2
0 0
1 0
3
0 0
2 0
1 -2
```

Sample Output

```
0
1
```

Exercise – *Software update*

The situation is quite unsatisfactory still. Due to the small operation radius of the snow cannons, a considerable part of the race course is not reached by the cannons. As a result, many working hours and huge amounts of energy have to be spent using snow around with snowcats. So the organizers asked algoALPINE for help. Luckily, they were able to fix the software bug so that now for each of the updated T60+ cannons one can set an individual radius of operation.

The hope is that by using different radii the region that is not reached by the cannons can be reduced significantly. For comparison, you have been asked to compute the maximal improvement that can be achieved with the new T60+ cannons. The cannons are currently operating at the range that you determined in the previous exercise, namely the largest radius such that no overlap occurs. Now you are allowed to increase this radius individually for each of the cannons as much as you like as long as no overlap occurs. The improvement is measured as the sum of all the differences of the new radii compared to the old ones.

Input The first line contains an integer $1 \leq t \leq 30$, denoting the number of test cases. Each of the following t testcases is described as follows: It starts with a line that contains one integer n, denoting the number of snow cannons ($2 \leq n \leq 50$). The following n lines each contain two integers x_i, y_i, denoting the position (x_i, y_i) of the i-th snow cannon ($|x_i|, |y_i| \leq 2^{24}$).

Output For each test case output a single line containing one number: the maximal improvement—rounded up to the next larger integer—that you can achieve such that there are no two cannons whose ranges of operation overlap.

Sample Input

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

Sample Output

```
1
4
```

Briefly sketch your approach for each of the three exercises in keywords.

Exercise 1 – *Downhill course*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Reasoning:

Exercise 2 – *Use them all!*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Reasoning:

Exercise 3 – *Software update*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Reasoning: