



## Challenge 16 - ÑAPA

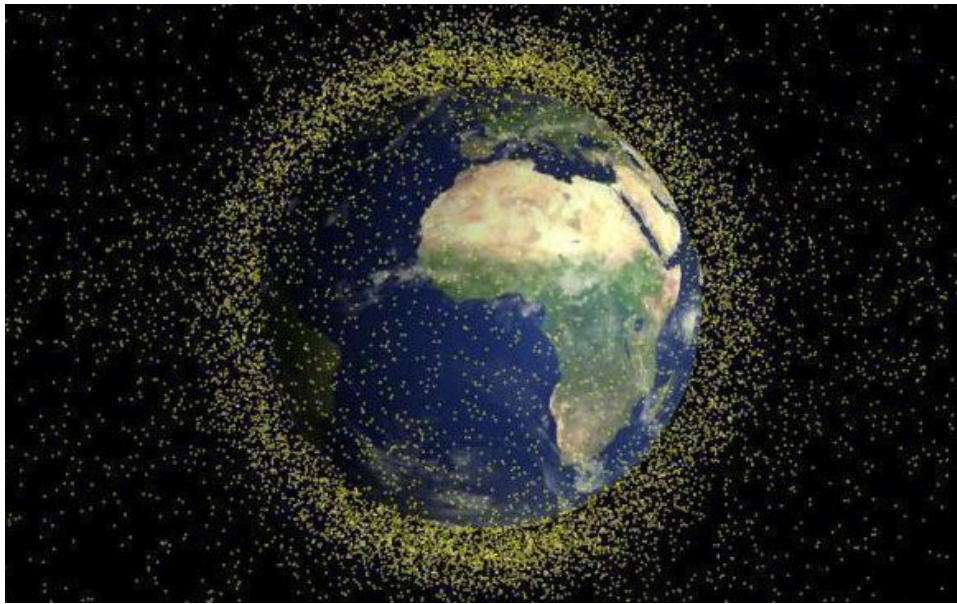
The Spanish government is in charge of developing the software to support the new ÑASA Atomic Particle Analyzer (ÑAPA) of the T250 (Javalambre Survey Telescope).



*T250 Javalambre Survey Telescope at night*

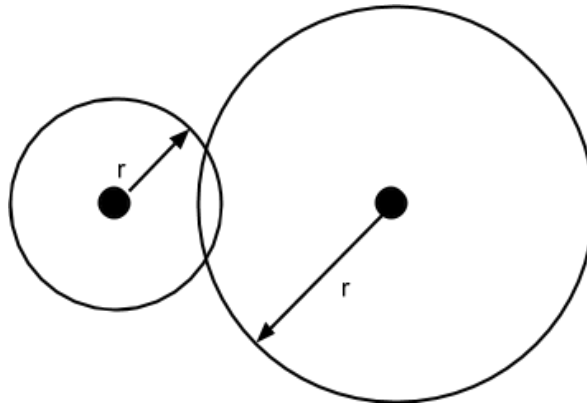
This telescope analyzes a small portion of the sky and reports  $N$  points mapped on a two-dimensional map with coordinates  $X$  and  $Y$  in which  $0 \leq X$  and  $Y < 100.000$ , for space dust, trash and small rocks that are on the same orbit.

The telescope also reports that points are circular with an estimated radius of  $R$  ( $1 \leq R \leq 500$ )



*Dust and trash detected, full sky example, coarse grained detection*

As the only capable software engineer in the team responsible for development, you must implement a program that, given the point report from the telescope, return the number of collisions.



*A collision between a pair of points occurs when the distance is less than the sum of their radius.*

The T250 team provides you with the first radar inspection (containing 3 million points) in the attached [file](#).

For each line in the file, the first number is the X coordinate, the second is the Y coordinate, and the third is the R radius, separated by spaces and tabs.

### Input

The input of the algorithm will be two positive integers separated by a comma. The first number is the nth point to read, and the second number will be the number of points to read.

### Output

A positive integer with the number of collisions between pairs of points from the subset in the range specified in the input.

### Example

For the following set of 12 points:

54791

92148

43

35138	75417	94
87668	20721	454
64455	33358	291
40423	35057	15
2467	41977	784
87438	28193	198
20680	76562	278
20930	75950	428
56698	14029	492
58959	3668	270
60306	70806	268

With this input:

6, 4

Will return this output:

1

This means that we will evaluate from the 6th point to the 9th point and there is one collision between the 8th and the 9th point.

## Submit & test your code

To test and submit code we provide a set of tools to help you. Download [contest tools](#) if you haven't already done that. You will then be able to test and submit your solution to this challenge with the challenge token.

Challenge token: A0iK95U7HJyiWDvLRbwS

### To test your program

```
./test_challenge A0iK95U7HJyiWDvLRbwS path/program
```

A nice output will tell you if your program got the right solution or not. You can try as many times as you need.

### To submit your program to the challenge

```
./submit_challenge A0iK95U7HJyiWDvLRbwS path/source_pkg.tgz
path/program
```

Note that you first need to solve the test phase before submitting the code. During the submit phase, in some problems, we might give your program harder questions, so try to make your program failsafe.

**Important:** In this phase, you must provide the source code used to solve the challenge and, if

necessary, a brief explanation of how you solved it.

Remember **you can only submit once!** Once your solution is submitted you won't be able to amend it to fix issues or make it faster, so please be sure your solution is finished before submitting it.

If you have any doubts, please check the [info](#) section.

## Go ahead

### I'm done! :)

Once you have submitted your code, hit refresh and continue to next challenge.

### I'm stuck! :(

Be sure you follow the [Tuenti Engineering](#) twitter for updates and possible hints during the contest.

If this challenge is too hard and you are blocked, you will be able to skip it after two hours. Note that **you won't be able to complete it later**, and you have a limited number of challenges to skip.

Finally, if you run out of skips but are still really stuck with one problem, you will be able to skip it after 24 hours.

### Challenge status:

Test case	Not done
Solution submitted	Not done
Skip	You can't skip this one

Refresh status

Tweet about this! [#TuentiChallenge4](#)



Follow [@Tuentieng](#)