Robotique
CRC
Robotics

CRC

PROGRAMMING COMPETITION

TAK TIK 2025

PRELIMINARY PROBLEM 4

A FEW NOTES

- The complete rules are in section 4 of the rulebook.
- You have until **Sunday, January 26th, 11:59 pm** to submit your code.
- Feel free to use the programming forum on the CRC discord to ask questions and discuss the problem with other teams. It is there for that purpose!
- **We are giving you quick and easy-to-use template files for your code and the tests. You are required to use them.**

USING THE TEMPLATE FILE

- The template tests call the function to test, take the output and allow you to quickly check if your code works as intended. **All your code, except additional functions you create, should be written in the function of the part of the problem you are solving.**
- Points given in the document are indications of how difficult the section is and how many points you will get if you complete it. This preliminary problem is going to be 4% of the main challenge towards the global score of the programming competition and for more points related information consult the rulebook.

STRUCTURE

Every problem contains a small introduction like this about the basics of the problem and what is required to solve it.

Input and output specification:

Contains the inputs and their format, and which outputs the code is required to produce and in what format they shall be.

Sample input and output:

Contains a sample input, sometimes containing sub examples in the sample input, and what your program should return as an output.

Explanation of the first output:

Explains briefly the logic that was used to reach the first output, given the first input.

Fire risk must be reduced!

When cities are developed, it is essential to ensure that there are enough essential services to cover everything. Naturally, emergency services need to be both sufficient in number and well-positioned. However, you can't just move a city! So, you must decide how to assign services to neighboring cities.

In this problem, you will be assigning cities to fire stations to ensure the risk score is minimized!

Part 1: Calculate the distance (5 points)

To ensure risk is minimized, the fire station must be close to the city it serves. You will need to calculate the distance between the fire station and the chosen city. Since a fire truck must travel on roads rather than taking the shortest direct path, you will calculate the distance between two points using the Manhattan distance. (https://en.wikipedia.org/wiki/Taxicab_geometry)

Input and output specification:

As input you will receive:

- **city_a:** a *tuple* containing 2 *int* that are the x and y coordinates of the city a
- **city_b:** a *tuple* containing 2 *int* that are the x and y coordinates of the city b

As output you need to return:

- a *int* variable corresponding to the Manhattan distance between the 2 cities.

Sample input:

```
[0, 0], [1, 5]
[20, 18], [14, 15]
[2, 3], [4, 5]
```

Sample output:

```
6
9
4
```

Note: There will be no further explanations, as you have three weeks to solve this problem! There are pytest tests for the first 3 parts of the problem.

Part 2: Where are the fire stations? (5 points)

You will receive the complete problem as a 2D map, and it will be your task to locate the positions of the fire stations. You will need to find the coordinates of the fire stations on the given map. The fire stations are represented by the **value 1** on the map. You must return these coordinates sorted in ascending order by the **x** coordinate, and in case of a tie, sorted in ascending order by the **y** coordinate as well.

To better visualize the map and the stations, you can open the file **benchmark.py** and adjust the value on **line 15** to change the map size (values between 10 and 25 work well for visualization). The red points represent the fire stations, while the blue points represent the cities to be served. When you close the city visualization, your algorithm will be called to solve the problem.

Input and output specification:

For input you will receive:

- map: a 2D *array* of *int* representing the map
- n: an *int* representing the size of one side of the map

As an output, you will need to return an *array* of *tuples* of two *integers* representing the **x** and **y** coordinates of the fire stations.

Sample input:

```
[[0, 0, 2, 0, 0],
[0, 0, 0, 0, 0],
[0, 1, 0, 0, 0],
[0, 0, 2, 2, 0],
[0, 2, 0, 2, 0]]

5
```

Sample output:

```
[(2, 1)]
```

Note: There will be no further explanations, as you have three weeks to solve this problem! There are pytest tests for the first 3 parts of the problem.

Part 3: Where are the cities? (5 points)

You will receive the complete problem as a 2D map, and it will be your task to locate the positions of the cities. You will need to find the coordinates of the cities on the given map. The cities are represented by the **value 2** on the map. You must return these coordinates sorted in ascending order by the **x-coordinate**, and in case of a tie, sorted in ascending order by the **y-coordinate** as well.

To better visualize the map and the stations, you can open the file **benchmark.py** and adjust the value on **line 15** to change the map size (values between 10 and 25 work well for visualization). The red points represent the fire stations, while the blue points represent the cities to be served. When you close the city visualization, your algorithm will be called to solve the problem.

Input and output specification:

For input you will receive:

- map: a 2D *array* of *int* representing the map
- n: an *int* representing the size of one side of the map

As an output, you will need to return an array of *tuples* containing two *int* representing the **x** and **y** coordinates of the cities.

Sample input:

```
[[0, 0, 2, 0, 0],
[0, 0, 0, 0, 0],
[0, 1, 0, 0, 0],
[0, 0, 2, 2, 0],
[0, 2, 0, 2, 0]]

5
```

Sample output:

```
[(0, 2), (3, 2), (3, 3), (4, 1), (4, 3)]
```

Note: There will be no further explanations, as you have three weeks to solve this problem! There are pytest tests for the first 3 parts of the problem.

Part 4: City Distribution (20 points)

Now that you know the positions of both the cities and the fire stations, you can proceed with assigning cities to fire stations! Here are a series of important rules to follow to minimize your risk score

This rule is essential for a valid solution: **all cities must be assigned to exactly one fire station**. However, fire stations can remain unused if needed—remember to include an empty array for stations that do not serve any city.

The second rule is that the number of cities assigned per fire station should be as close as possible to 5. For example, with n = 20, there will be 20 cities and 4 fire stations, meaning an average of 5 cities per station. You can assign a different number than 5, but the risk increases exponentially as you deviate from this target. For example, a station handling **3 extra cities** (8 total) would have a risk score of $3^2 = 9$. The risk score increases gradually at first when the city count is imbalanced but can escalate rapidly if a station serves too many cities.

The last factor contributing to the risk score is the distance between the station and the cities it covers. Fire stations should be close to the cities they serve for rapid response in case of a fire. A higher distance results in a higher risk score.

Your objective is to minimize the risk score across various map sizes: [10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000]. You will score **1 point** for each valid solution provided for these sizes, as long as the solution is computed in under **1 minute**. Teams will be compared based on their risk scores, with the best-performing team receiving **1 additional point** per map size. Other teams will receive a proportional fraction of that point based on how close they are to the best score, scaling down to 0 points for the highest risk score.

Successfully solving all map sizes with valid solutions will earn you 10 out of 20 points directly.

Input and output specification:

For input you will receive:

- map: a 2D *array* of *int* representing the map
- n: an *int* representing the size of one side of the map

As an output, you will need to return an *array* of *arrays* of *tuples* containing two *int* representing the **x** and **y** coordinates of the cities assigned to each fire station.

Sample input:

```
[[0, 0, 0, 0, 2, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 2, 0, 0, 0, 0, 0, 2, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 2, 0, 2, 0, 0, 0],
[0, 0, 0, 0, 0, 2, 0, 0, 0, 0],
[0, 0, 2, 0, 0, 0, 0, 2, 0, 0],
[0, 0, 1, 2, 0, 0, 0, 0, 0, 0]]

10
```

Sample output:

```
[[(0, 4), (1, 9), (4, 2)],
[(4, 8), (6, 4), (6, 6), (7, 5), (8, 2), (8, 7), (9, 3)]]
```

Explanation of the first output:

We took the first 3 cities and assigned them to fire station 0, and the remaining 7 cities were assigned to station 1. You can find better algorithms, good luck!!

risk score for imbalance: 8
risk score for distance: 5.5

total risk score: 13.5