[CS 11 25.1] Lab 2e – Square

Problem Statement

One of the most interesting ideas in computer science is how you can represent large amounts of information with very little data.

For example, consider an infinite grid of cells. Each cell has position (i, j), where the i represents the vertical location and increases as you go down the grid, and the j represents the horizontal location and increases as you go right (...why does this seem familiar?).

Using two positions (i_1, j_1) and (i_2, j_2) such that $|i_1 - i_2| = |j_1 - j_2|$, you can represent a square! Specifically, the square being represented is the square whose sides are aligned with the x and y axes and where (i_1, j_1) and (i_2, j_2) are opposite corners. Now if, say, $|i_1 - i_2| = 10^{200} - 1$, you can represent an area of 10^{400} squares using just two positions! Isn't that cool?

For this problem, we want to know about the *boundary* of a square. Consider the following:

```
B--A
|..|
|..|
A--B
```

Here, the `A` and `B` characters denote the corners of the square, and the `-`, `|`, and `+` characters denote other cells that are also on the boundary of the square.

Given the two opposite corners of a square, can you identify all cells at its boundary?

Task Details

Your task is to implement a function named `trace_square`, which should have the following *signature*:

```
def trace_square(c1, c2):
```

The above says that it has two arguments c_1 and c_2. Both of them are pairs (`tuple`s of length 2) of integers (`int`s) denoting two diagonally opposite corners of the square whose boundary cells you want to get.

The function must return a tuple of pairs denoting the positions of the cells along the boundary of the square. The positions must be sorted in increasing order of i-value, breaking ties in increasing order of j-value.

Restrictions

- The following symbols can now be used: `min`, `max`, `sum`, `range`, `all`, `any`.
- recursion is *disallowed.*
- comprehensions are allowed.
- at most 6 functions can be defined.
- Your source code must have at most 2000 bytes.

Examples

Example 1 Function Call

```
trace_square((1, 1), (3, 3))
```

Example 1 Return Value

```
((1, 1), (1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2), (3,
```

Constraints

- The function `trace_square` will be called at most 5 times.
- $|i_1|, |j_1|, |i_2|, |j_2| \le 100{,}000$
- $|i_1 - i_2| = |j_1 - j_2| \ge 1$

Scoring

Note: New tests may be added and all submissions may be rejudged at a later time. (All future tests will satisfy the constraints.)

- You get 50 ❤️ points if you solve all test cases where:
  - $i_1 < i_2$ and $j_1 < j_2$.
  - $|i_1|, |j_1|, |i_2|, |j_2| \le 1$
- You get 25 ❤️ points if you solve all test cases where:
  - $|i_1|, |j_1|, |i_2|, |j_2| \le 1$
- You get 50 🔴 points if you solve all test cases where:
  - $|i_1|, |j_1|, |i_2|, |j_2| \le 60$
- You get 25 💔 points if you solve all test cases.

Submit solution

My submissions

Points: 150 (partial)
Time limit: 18.0s
Memory limit: 1G

Problem type

Allowed languages
py3

Clarifications

Report an issue

No clarifications have been made at this time.