Hilbert Sort

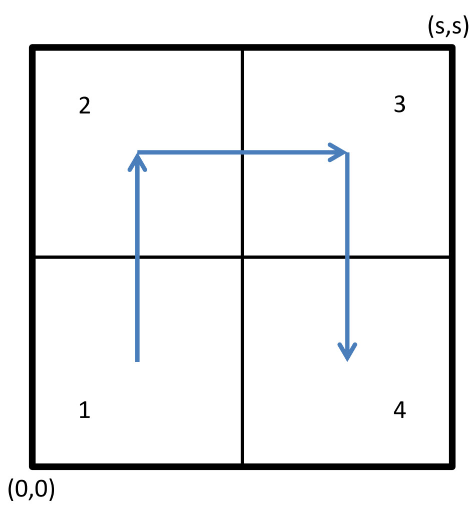
Sorting numerical data not only makes it easy to search for a particular item, but also makes better use of a CPU's cache: any segment of data that's contiguous in memory will describe a set of items that are similar in some sense. Things get more complicated if our data represents points on a 2D grid. If points (***x***,***y***) are sorted by ***x***, breaking ties by ***y***, then adjacent points will have similar ***x*** coordinates but not necessarily similar ***y***, potentially making them far apart. To better preserve distances, we can sort the data along a space-filling curve.

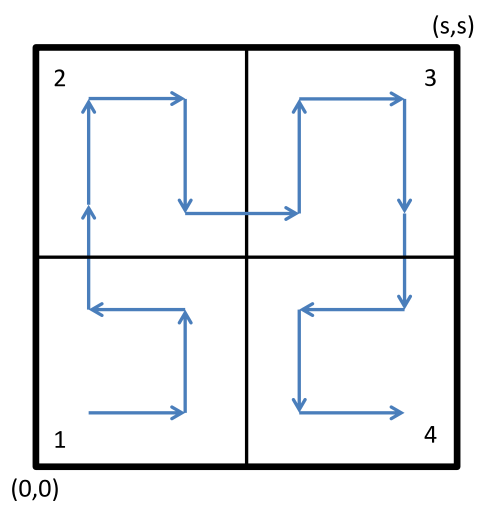
The Hilbert curve starts at the origin (**0**,**0**), finishes at (***s***,**0**), in the process traversing every point in axis-aligned square with corners at (**0**,**0**) and (***s***,***s***). It has the following recursive construction: split the square into four quadrants meeting at (***s***/**2**, ***s***/**2**). Number them 1 to 4, starting at the lower left and moving clockwise. Recursively fill each of them with a suitably rotated and scaled copy of the full Hilbert curve.

Start with a single point at (***s***/**2**,***s***/**2**). Then, repeat these steps:

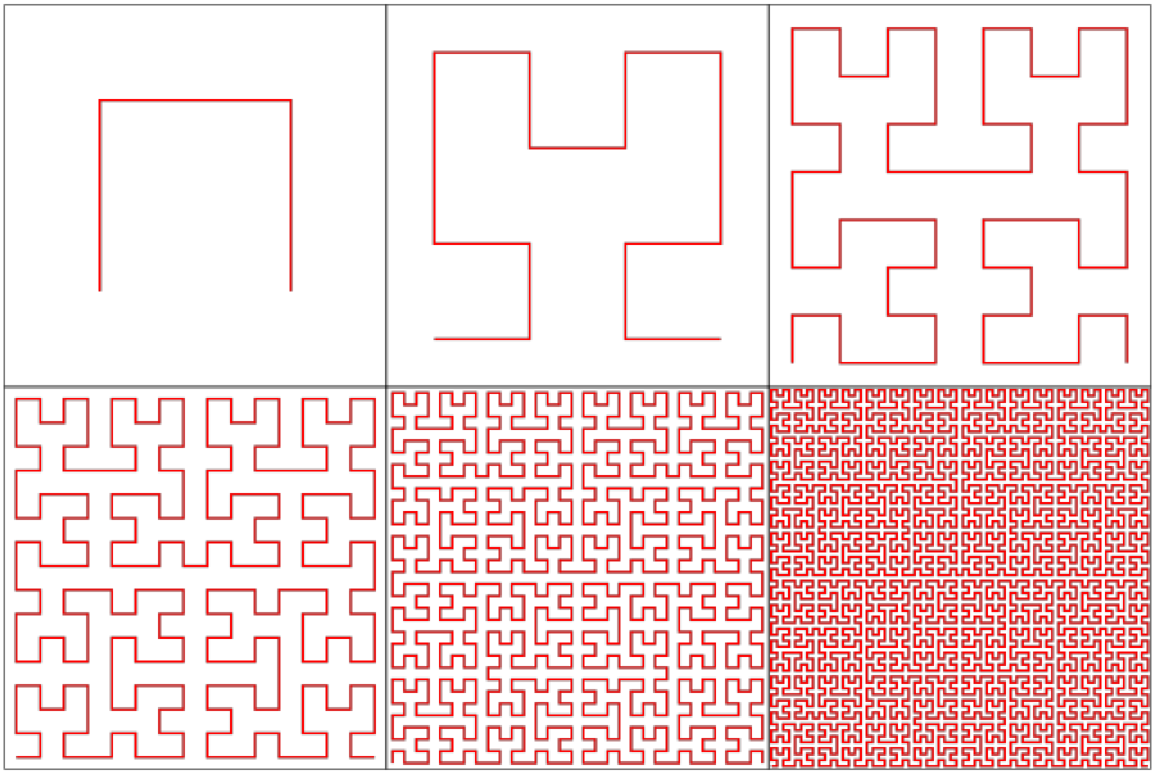
- Scale and copy the current construction into each of the 4 quadrants.
- Rotate quadrant 1 by -90 degrees and flip it vertically, so that the start of the curve is closest to the lower left corner (**0**,**0**).
- Rotate quadrant 4 by 90 degrees and flip it vertically, so that the end of the curve is closest to the lower right corner (***s***,**0**).
- Now, connect the end of the curve in quadrant 1 to the start of the curve in quadrant 2, connect the end of quadrant 2 to the start of quadrant 3, and the end of quadrant 3 to the start of quadrant 4.

Here are the first two iterations:





The Hilbert Curve is built by repeating this construction infinitely many times. The following diagram shows the first six steps of building the Hilbert Curve:



Given some places of interest inside of a square region, sort them according to when the Hilbert curve visits them, starting from (0,0). Without going into gory detail about Fractal theory, note that making ***s*** odd guarantees that all integer points are visited just once, so their visitation order in relation to each other is unambiguous.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input contains two space-separated integers ***n*** and ***s*** ($1 \le n \le 100{,}000$, $1 \le s < 10^9$, ***s*** is odd). The next ***n*** lines describe locations of interest by space-separated integers ***x*** and ***y*** ($0 \le x,y \le s$). No two locations will share the same position.

Output

Output the ***n*** ordered pairs, one per line, with ***x*** and ***y*** separated by a space, Hilbert-sorted according to their positions.

Sample Input	**Sample Output**
14 25 5 5 5 10 5 20 10 5 10 10 10 15 10 20 15 5 15 10 15 15 15 20 20 5 20 10 20 20	5 5 10 5 10 10 5 10 5 20 10 20 10 15 15 15 15 20 20 20 20 10 15 10 15 5 20 5