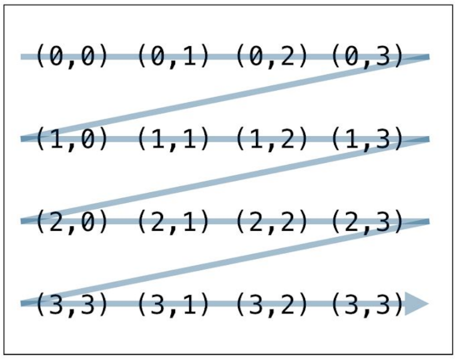
**COS**226 **Precept Exercises:** Week 2 **Spring** ‘19

### EXERCISE 1: A Grid Iterator

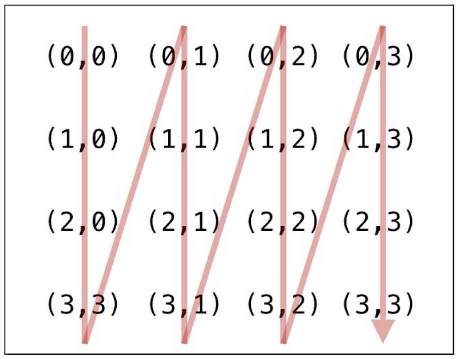
Download **`Grid.zip`** from the precepts page, unzip the project and open it using IntelliJ.

(a) Implement the **`GridIterator`** class in **`Grid.java`** to enable iterating over the elements in the grid in row-major order (as shown below). Test your program by running the given driver program.



```
(0,0) (0,1) (0,2) (0,3)
(1,0) (1,1) (1,2) (1,3)
(2,0) (2,1) (2,2) (2,3)
(3,3) (3,1) (3,2) (3,3)
```

Row–major



```
(0,0) (0,1) (0,2) (0,3)
(1,0) (1,1) (1,2) (1,3)
(2,0) (2,1) (2,2) (2,3)
(3,3) (3,1) (3,2) (3,3)
```

Column–major

(b) Create another iterator **`ColMajorIterator`** that returns elements in *column-major* order. Add code to **`main`** that prints the grid elements using this iterator.

(c) Convert **`Grid.java`** to an *Iterable*, where the default iteration is in row-major order. Test your code by converting the while loop in **`main`** to a for-each loop.

(d) Add code to **`main`** that prints “**`Distinct`**” if all the elements in the grid are distinct and “**`Not Distinct`**” if any element appears more than once.

### EXERCISE 2: Running Time Analysis (~40 minutes)

(a) What is the order of growth of the running time of the following piece of code?

```
for (int i = 0; i < n; i++)
    for (int j = n; j > i; j--)
        sum++;
```

(b) Consider an *organ-pipe* array that contains two copies of the integers $1$ through $n$, first in ascending order, then in descending order. For example, here is the array when $n = 8$:

1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1

Note that the length of the array is $2n$, not $n$.

How many compares does **Insertion sort** make to sort the array as a function of $n$? Use tilde notation to simplify your answer.

(c) What is the order of growth of the running time of the following piece of code?

```
for (int i = 1; i <= n; i++)
      for (int j = 1; j <= i; j++)
            for (int k = 1; k <= i; k++)
                  StdOut.print(k + “ “);
```