Sergio E. Garcia Tapia
Algorithms by Sedgewick and Wayne (4th edition) [SW11]
September 14th, 2024

1.2: Data Abstraction

Exercise 1. Write a `Point2D` client that takes an integer value n from the command line, generates n random points in the unit square, and computes the distance separating the *closest pair* of points.

Solution. See the `com.segarciat.algs4.ch1.sec2.ex01.ClosestPointPair` class. The `Point2D` objects can simply by stored in an array of size n. We can use `StdRandom.uniformDouble(` to generate random x and y coordinates for each point, and then leverage the `distanceTo()` method available in the `Point2D` API. I employed a nested `for` loop to compute the closest pair.

Exercise 2. Write an `Interval1D` client that takes an `int` value n as command-line argument, reads n intervals (each defined by a pair of `double` values) from standard input, and prints all pairs that intersect.

Solution. See the `com.segarciat.algs4.ch1.sec2.ex02.IntervalIntersection` class. It's much the same as in Exercise 1, but instead of using `StdRandom.uniformDouble()` to generate the coordinates, I use `StdIn.readDouble()` to read coordinate from standard input. Also, instead of the `distanceTo()` method from the `Point2D` API, I leveraged the `intersects()` method from the `Interval1D` API.

Exercise 3. Write an `Interval2D` client that takes command-line arguments `n`, `min`, and `max` and generates `n` random 2D intervals whose width and height are uniformly distributed between `min` and `max` in the unit square. Draw them on `StdDraw` and print the number of pairs of intervals that intersect and the number of intervals that are contained in one another.

Solution. See the `com.segarciat.algs4.ch1.sec2.ex03.IntersectingRectangles` class.

One important consideration is that since the widths and heights are generated uniformly between `min` and `max`, we must ensure the bottom left corner of each point isn't so large that it would exceed the dimensions of the unit square. That is, given `width` and `height`, each of the x and y coordinates of the bottom left vertex of all rectangles must not exceed `1 - width` and `1 - height`, respectively.

Another important consideration is that, to check if rectangle A contains rectangle B, we must check that the bottom-left and top-right vertices of rectangle B are contained in rectangle A.Since the `Interval2D` API does not expose methods for obtaining these quantities, it's necessary to save them while doing the computations to necessary to create the rectangles.

References

[SW11] Robert Sedgewick and Kevin Wayne. *Algorithms*. 4th ed. Addison-Wesley, 2011. ISBN: 9780321573513.