

2. Comparators Algorithms Textbook 2.5

- (a) If a Java class implements the Comparable interface, what method must be provided in the class code?
- (b) If a Java class implements the Comparator interface, what method must be provided in the class code?
- (c) Consider the algs4 Point2D class provided in the appendix (see precept page for complete code). Suppose we want to create a Comparator that compares two Point2D objects based on their distance from some third point. Fill in the code below. The compare must return 1, 0 or -1 if point 1 is more, same or less distance from point 2. You may find the code in the appendix helpful.

```
public static class DistanceComparator implements Comparator<_____> {
    Point2D _____;

    public DistanceComparator(_____) {
        _____ = _____;
    }

    public int compare(_____ p, _____ q) {
        double distToP = p.distanceTo(_____);
        double distToQ = q.distanceTo(_____);
        if (distToP < distToQ) return ____;
        if (distToP > distToQ) return ____;
        return ____;
    }
}
```

- (d) Now suppose we want to use our comparator to sort an array of Point2D objects by their distance from the origin. Fill in the code below to accomplish this task.

```
Point2D[] points = getRandomPoints();
Point2D origin = new Point2D(____,____);
Comparator<Point2D> originDistanceComparator = _____;
Arrays.sort(points, _____);
```

- (e) When Arrays.sort() is called, which sorting algorithm does Java use to sort the array?

3. Sorting Invariants

The column on the left is the original input of strings to be sorted; the column on the right are the strings in sorted order; the other columns are the contents at some intermediate step during one of the 6 sorting algorithms listed below. Match up each algorithm by writing its number under the corresponding column. Use each number exactly once.

nite	deni	deni	deni	deni	dint	dine	deni
rein	dent	dent	dent	dent	dine	deni	dent
deni	nite	ding	ding	diet	deni	dent	diet
dent	rein	grin	nite	dine	dent	edit	dine
rent	ding	nite	rein	ding	edit	ding	ding
ding	grin	rein	rent	rent	ding	grin	dint
grin	rent	rent	grin	dint	grin	dreg	dire
ride	ride	ride	ride	ride	dreg	dire	dreg
rind	diet	diet	rind	rind	dire	diet	edit
diet	dint	dint	diet	nite	diet	dint	grin
dint	rind	rind	dint	grin	nite	nite	nite
ring	ring	ring	ring	ring	ring	ring	rein
dire	dine	dine	dire	dire	rind	rind	rent
dreg	dire	dire	dreg	dreg	ride	ride	ride
edit	dreg	dreg	edit	edit	rent	rent	rind
dine	edit	edit	dine	rein	rein	rein	ring

- (0) Original input
- (1) Sorted
- (2) Selection sort
- (3) Insertion sort

- (4) Mergesort
(*top-down*)
- (5) Mergesort
(*bottom-up*)

- (6) Quicksort
(*standard, no shuffle*)
- (7) Quicksort
(*3-way, no shuffle*)

4. 3-way Merge sort (Bonus Question)

3-way merge sort is a modification of the merge sort algorithm that considers 3 “equal” sub arrays instead of 2 sub arrays.

- (a) Given 3 sorted sub arrays of size $N/3$, how many comparisons are needed to merge them to a sorted array of size N . Provide your answer in tilde notation.
- (b) Argue that number of compares to sort an array of size N using 3-way merge sort is still linearithmic.
- (c) Given a choice, would you choose 3-way or 2-way merge sort? Justify your answer.

1 Appendix

Below is the syntax highlighted version of Point2D.java from Algorithms.

```
/******  
 * Compilation: javac Point2D.java  
 *  
 * Immutable point data type for points in the plane.  
 *  
 *****/  
  
import java.util.Arrays;  
import java.util.Comparator;  
  
public class Point2D implements Comparable<Point2D> {  
    public static final Comparator<Point2D> X_ORDER = new XOrder();  
  
    .....  
    // see precept page for full code  
    .....  
  
    // compare by y-coordinate, breaking ties by x-coordinate  
    public int compareTo(Point2D that) {  
        if (this.y < that.y) return -1;  
        if (this.y > that.y) return +1;  
        if (this.x < that.x) return -1;  
        if (this.x > that.x) return +1;  
        return 0;  
    }  
  
    // compare points according to their x-coordinate  
    private static class XOrder implements Comparator<Point2D> {  
        public int compare(Point2D p, Point2D q) {  
            if (p.x < q.x) return -1;  
            if (p.x > q.x) return +1;  
            return 0;  
        }  
    }  
  
    ....  
    // see precept page for full code  
    ....  
  
    // does this point equal y?  
    public boolean equals(Object other) {  
        if (other == this) return true;  
        if (other == null) return false;  
        if (other.getClass() != this.getClass()) return false;  
        Point2D that = (Point2D) other;  
        return this.x == that.x && this.y == that.y;  
    }  
  
    // see precept page for full code  
  
    public static void main(String[] args) {  
        // code not given  
    }  
}
```