COP3503 1/9/24

~~Built In~~

(1) Custom Sorting - old way

Implement comparable interface

```
public int compareTo(<T> other)
// if this < other return a neg int
// if this > other return pos int
// if this == other return 0;
```

```
public int compareTo(Point other) {
    if (x != other.x) return x - other.x;
    return y - other.y;
}
```

p1 → [x 3, y 7]

p2 → [x 3, y 2]

p1.compareTo(p2)

```
ArrayList<Point> mylist;
```

↓

```
Collections.sort(mylist);
```

```
Point[] arr = new Point[n];
```

↓

```
Arrays.sort(arr);
```

Note: 2nd method

Create class that implements Comparable<Point> with compare function. See TestPoint.java

Java API → ✓ ArrayDeque

✓ PriorityQueue

~~HashSet~~ & ✓ HashSet

HashMap

TreeSet

TreeMap

btw, ArrayList insert is O(n)

n = # items in list.

ArrayDeque:

addFront
addBack
delFront
delBack
} O(1)

Stack
queue
double ended queue

PriorityQueue:

insert
deleteMin
] O(lg n)

can store repeats

Add All Algorithm

Could call min heap

1. Add each item in PQ ($n \lg n$)
2. While pq.size() > 1

$O(\lg n)$ → a. remove 2 smallest items a, b.

$O(1)$ b. res += (a+b)

$O(\lg n)$ → c. add (a+b) back into PQ

$O(n \lg n)$ ← n-1 times

Run-time $O(n \lg n)$

HashSet

add
remove
search

} $O(1)$ expected time

Did CD problem

NO DUPLICATES!