Sorting Objects in Java

Cody Henrichsen

2019

3/22/21

Sorting Algorithms

Loop based	Recursive	#Extra
Bubble	Mergesort	Radix
Selection	Quicksort	BoGo
Insertion		

3/22/21

Primitives

- All the numeric primitive data types support comparisons using the standard relational operators you know and love from elementary
  - <, >, <=, >=, ==, !=
- Thus the results in CS land match the results your are comfortable with from math land
  - Except for floating point math
  - Then you need to compare the difference

Expression	Result
`3 < 7`	`true`
`7 > 3245`	`false`
`'a' < 'z'`	`true`
`Math.PI < Math.E`	`false`
`.1 * 3 == .1+.1+.1`	`false`

3/22/21

Simple Object Comparison

- Implement the Comparable interface
  - `myThing.compareTo(otherThing)`
- Pro
  - As complex of a single comparison as you need/want it to be
  - Direct support in Collections methods
- Con
  - Only 1 comparison level

3/22/21

Implementation example

```
@Override
public boolean equals(Object other)
{
    if (other == this)
    {
        return true;
    }
    else if (other instanceof DebugDuck && ((DebugDuck) other).getQuestionCount() == this.questionCount)
    {
        return true;
    }
    return false;
}

@Override
public int compareTo(DebugDuck otherDuck)
{
    if (otherDuck.equals(this))
    {
        return 0;
    }
    else if (this.questionCount < otherDuck.getQuestionCount())
    {
        return Integer.MIN_VALUE / 2;
    }

    return 123;
}
```

3/22/21

Bubble Sort

- Compare each individual item to each other item over and over swapping each time there is a difference
- Almost always the worst possible option
  - $O(n^2)$
- Nested for loop over the structure with a swap every time the adjacent values are out of order
- Always does a final iteration with no swaps

3/22/21

Selection Sort

- Compare each item in the structure against the current (min/max) value and finds the index of the next min/max
- Average/Worst case is $O(n^2)$
- Nested for loops over the structure
- Update the index of the current min/max inside the inner loop
- Call swap every outer loop with current and index if not the same

3/22/21

Selection Sort Code – Array

```
//SelectionSort Algorithm
for (int outerLoop = 0; outerLoop < myDucks.length; outerLoop++)
{
    int minIndex = outerLoop;
    for (int inner = outerLoop + 1; inner < myDucks.length; inner++)
    {
        if (myDucks[inner].compareTo(myDucks[minIndex]) < 0)
        {
            minIndex = inner;
        }
    }

    if (minIndex != outerLoop)
    {
        swapItems(minIndex, outerLoop, myDucks);
    }
}
```

3/22/21

Insertion Sort

- Compare the current value in the search against each sorted value down to the start and insert the current value at the sorted index. Builds a sorted structure as it executes
- Average/Worst case is $O(n^2)$
- Nested loops over the structure
  - Outer loop is forward
  - Inner loop is backward
- Get a reference to the value at the outer loops variable
- Move values down in the inner loop
- Replace the tested (outer) value

3/22/21

Insertion Sort Code - List

```
//Insertion Algorithm

for (int outer = 1; outer < randomList.size(); outer++)
{
    DebugDuck tested = randomList.get(outer);
    int inner = outer - 1;

    while ( inner >= 0 && tested.compareTo(randomList.get(inner)) < 0 )
    {
        randomList.set(inner + 1, randomList.get(inner));
        inner--;
    }
    randomList.set(inner + 1, tested);
}
```

3/22/21

Divide and Conquer!

- The next two sorts use recursion to make the problem easier to solve AKA split the problem up so that it is ridiculously easy to solve
  - Everyone can sort a list of 1 or 2
- Need to know when to stop (base case) or when to call the same method with different info (recursive case)
- Structure is a method with parameters that has an if block
  - If you are done stop
  - Else break the problem down and send the parts to the same method!

3/22/21

Mergesort

- Check if the structure is one or two in size if it isn't split the structure in half and call the mergesort with both smaller structures. Merge the results
- O(n log n)
- **Never** use on sorted data
  - Goes to O(n^2)

3/22/21

Quicksort

- Pick the pivot index, move elements around the pivot (<,>), recursively call the method again to pick new pivot(s) as long as size is greater than two…
- Also never use on sorted data

3/22/21

#Extra Sorts

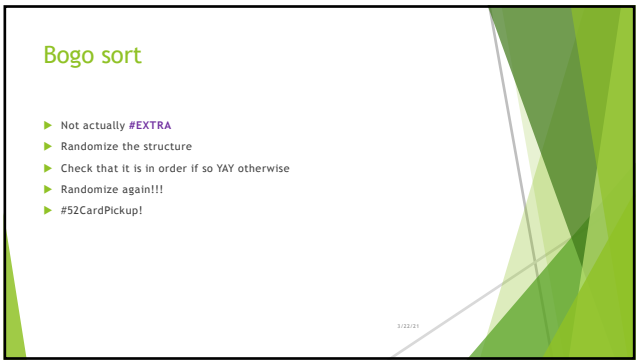
- There are many other ways of sorting data
- Here is one cool and one horrible example

3/22/21

Radix

- Put stuff in piles! Then break the piles down using the same logic
- It is something you have been doing already for a while (further proof CS is for All)
  - Laundry
- Data
  - Least Significant Digit to Most Sig
  - 0's, 1's, 2's, 3's, … then 10's, 20's…

3/22/21


Bogo sort
Not actually #EXTRA
Randomize the structure
Check that it is in order if so YAY otherwise
Randomize again!!!
#52CardPickup!
3/22/21