CSCI 136
Data Structures & Advanced Programming
(Lecture 10)

Sorting 1

Jon Park
Feb 27, 2017
Announcements

- Lab 3 due today
- Lab 4 (Sorting) out tonight
Last Time

• Search
  • Linear Search
  • Binary Search
• Defining Sortable Classes
  • Comparable
  • Comparator
Today’s Outline

• Review
  • Comparable vs Comparator
• Sorting
  • Selection Sort
  • Bubble Sort
# 2 Types of Sortable Classes

- Classes with 1 “obvious” way to compare/sort (a) vs Classes with multiple ways to compare/sort (b)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implements Comparable&lt;E&gt; (i.e. contains compareTo(E otherObj))</td>
<td>☒</td>
<td>✗</td>
</tr>
<tr>
<td>Need Comparator&lt;E&gt; classes containing compare(E obj1, E obj2)</td>
<td>✗</td>
<td>☒</td>
</tr>
<tr>
<td>The class itself supports comparison</td>
<td>☒</td>
<td>✗</td>
</tr>
<tr>
<td>Can be compared/sorted in multiple ways</td>
<td>✗</td>
<td>☒</td>
</tr>
</tbody>
</table>
Today’s Outline

• Review
  • Comparable vs Comparator

• Sorting
  • Selection Sort
  • Bubble Sort
Sorting a Deck of Cards

• Come up with your own algorithm! (and let me know when one of the algorithms presented today is exactly like yours. ;) )
• Hint: If you’re stuck, think of it this way:
  • After 1st iteration, at least 1 item is sorted.
  • After ith iteration, at least i items are sorted.
  • After nth iteration, all items are sorted. Done!
  • What needs to happen during each iteration?
Today’s Outline

• Review
  • Comparable vs Comparator
• Sorting
  • Selection Sort
  • Bubble Sort
Sorting a Deck of Cards

Time Complexity:

A. $O(n)$
B. $O(n \log n)$
C. $O(n^2)$  ← worst, ave, best
D. $O(n^3)$
E. Not sure
Selection Sort

• \([11 \ 3 \ 27 \ 5 \ 16]\)
• \([11 \ 3 \ 16 \ 5 \ 27]\)
• \([11 \ 3 \ 5 \ 16 \ 27]\)
• \([5 \ 3 \ 11 \ 16 \ 27]\)
• \([3 \ 5 \ 11 \ 16 \ 27]\)
Aside: swap() method

```java
public static void swap(int[] data, int i, int j) {
    int temp = data[i];
    data[i] = data[j];
    data[j] = temp;
}
```
Selection Sort

class SelectionSort {
    public static void selectionSort(int[] data) {
        for (int curN = data.length - 1; curN > 0; curN--) {
            int maxIdx = 0;
            for (int i = 1; i <= curN; i++) {
                if (data[i] > data[maxIdx]) {
                    maxIdx = i;
                }
            }
            swap(data, maxIdx, curN);
        }
    }
}

Selection Sort (with Comparator)

```java
public static void selectionSort(int[] data){
    for (int curN = data.length - 1; curN > 0; curN-- ) {
        int maxIdx = 0;
        for (int i = 1; i <= curN; i++) {
            if (data[i] > data[maxIdx])
                maxIdx = i;
        }
        swap(data, maxIdx, curN);
    }
}
```

```java
public static void main(String[] args){
    Vector<Patient> patients;
    ...,
    selectionSort(patients, new NameComparator());
}
```
Selection Sort Summary

• Overview
  • After $ith$ iteration, at least $i$ items are sorted.
    $\Rightarrow$ the list is sorted at least after $n$ iterations.
  • During $ith$ iteration, select the max item in the unsorted portion of the list and move it to right-most location of the unsorted portion.

• Time complexity:
  • Best case: $O(n^2)$
  • Worst case: $O(n^2)$
  • Average case: $O(n^2)$
Today’s Outline

• Review
  • Comparable vs Comparator
• Sorting
  • Selection Sort
  • Bubble Sort
Sorting a Deck of Cards

Time Complexity:
A. $O(n)$ ← Best
B. $O(n \log n)$
C. $O(n^2)$ ← worst, but
D. $O(n^3)$
E. Not sure
Bubble Sort

- [ 5 1 3 2 9 ]
- First Pass:
  - [ 5 1 3 2 9 ]
  - [ 1 5 3 2 9 ]
  - [ 1 3 5 2 9 ]
  - [ 1 3 2 5 9 ]
- Second Pass:
  - [ 1 3 2 5 9 ]
  - [ 1 3 2 5 9 ]
  - [ 1 2 3 5 9 ]
  - [ 1 2 3 5 9 ]
- Third Pass:
  - [ 1 2 3 5 9 ]
  - [ 1 2 3 5 9 ]
  - [ 1 2 3 5 9 ]
  - [ 1 2 3 5 9 ]
public static void bubbleSort(int[] data) {
    for (int curN = data.length - 1; curN > 0; curN--)
        boolean swapped = false;
    for (int i = 1; i <= curN; i++)
        if (data[i - 1] > data[i]) {
            swap(data, i, i - 1);
            swapped = true;
        }
    if (!swapped) break;
}
Bubble Sort Summary

• Overview
  • After \textit{ith} iteration, at least \textit{i} items are sorted.
  • During \textit{ith} iteration, sweep through the unsorted portion of the list, swapping 2 adjacent elements if the right one is smaller.
    (End after iteration \textit{i} if no swapping happens!)

• Time complexity:
  • Best case: \(O(n)\)
  • Worst case: \(O(n^2)\)
  • Average case: \(O(n^2)\)