CS 367 - Introduction to Data Structures  
Thursday, July 30, 2015, Lecture 27

Course website: http://pages.cs.wisc.edu/~cstapleton/367/
Piazza: https://piazza.com/wisc/summer2015/cs367/

Instructor
Cea Stapleton
cstapleton@cs.wisc.edu ; Use Piazza for course related questions.

TA
Haseeb Tariq
haseeb@cs.wisc.edu ; Use Piazza for course related questions.

Last Time
- Sorting (smarter)

Today
- Sorting (finish)
- Radix Sort
- Stable sorts
- Sorting in Java
- Graphs

Next Time
- Last week in review
- Graphs (cont'd)
- Using Edge Representations
- Searches/Traversals: Depth-First Search
Radix Sort

Strategy:

Analysis:
# Stable Sorts

<table>
<thead>
<tr>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>New York</td>
<td>NY</td>
</tr>
<tr>
<td>Chicago</td>
<td>IL</td>
</tr>
<tr>
<td>Detroit</td>
<td>MI</td>
</tr>
<tr>
<td>Buffalo</td>
<td>NY</td>
</tr>
<tr>
<td>Milwaukee</td>
<td>WI</td>
</tr>
<tr>
<td>Peoria</td>
<td>IL</td>
</tr>
</tbody>
</table>

Sorted by state:

<table>
<thead>
<tr>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>IL</td>
</tr>
<tr>
<td>Peoria</td>
<td>IL</td>
</tr>
<tr>
<td>Detroit</td>
<td>MI</td>
</tr>
<tr>
<td>New York</td>
<td>NY</td>
</tr>
<tr>
<td>Buffalo</td>
<td>NY</td>
</tr>
<tr>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>Milwaukee</td>
<td>WI</td>
</tr>
</tbody>
</table>

What's the difference?
**Sorting in Java** (java.util)

`Arrays.sort(array_to_sort):

    String[] names = {"Tim", "Alison", "Nate"};
    Arrays.sort(names);
    System.out.println(Arrays.toString(names));

Collections.sort(List):

Use Comparators to define your own order:

    public interface Comparator<T> {
        int compare(T o1, T o2);
    }

    public class myCmp<myClass> implements Comparator<myClass> {
        public int compare(myClass o1, myClass o2) {
            ...
        }
    }
Types of Data Structures
(in terms of #predecessors and #successors)

Linear

Hierarchical

Graphical
Graph Terminology

- Nodes

- Edges: Weighted and unweighted

- Directed vs. undirected

- Degree of a node

- Self-edge
- Cyclic vs. acyclic

- Adjacency

- Predecessor, successor

- Path, path length

- Search/Traversal
- Complete graphs

- Directed Acyclic Graphs (DAG)

- Connected: Weakly, Strongly
Implementing Graphs

GraphNode

Graph

Representing Edges

Adjacency Matrix:

Adjacency matrix for these graphs:
Representing Edges (cont'd)

Adjacency Lists:

Adjacency list for these graphs: