Sorting

2/3/16
Announcements

• Reading: Read about Mergesort and Quicksort in Chapter 9 for Friday

• Makeup quizzes available tomorrow, Friday, and Monday. Details to follow by email.

• Quiz Friday: Writing a method for RecursiveBag (just the EmptyNode and DataNode methods)
Lab review

• Creating a test case
• Completing toString
• Implementing removeAll
A common problem: Sorting

- Have collection of objects (numbers, strings, dates, ...) and want to put them into a defined order

- How do we specify the order?

- How do we do this efficiently?
  - assume objects are in an array
  - assume you can compare any pair of objects
Generalizing comparisons

• Comparable<T> interface
  int compareTo(T other)
  negative: this < other
  positive: this > other
  zero: this == other

• Comparator<T> interface
  int compare(T o1, T o2)
  as above, with o1 taking role of “this”
Selection sort
(code adapted from Wikipedia)

• Grow sorted part of array by finding smallest value in the remaining

for(int i=0; i < A.length-1; i++) {
    int minIndex = i;  //index of min
    for(int j=i; j < A.length; j++) {  //find the min
        if(A[j] < A[minIndex])
            minIndex = j;
    }
    swap A[i] and A[minIndex];
}
Selection sort
(code adapted from Wikipedia)

• Grow sorted part of array by finding smallest value in the remaining

```java
for(int i=0; i < A.length-1; i++) {
    int minIndex = i;    //index of min
    for(int j=i; j < A.length; j++) {
        if(A[j] < A[minIndex])
            minIndex = j;
    }
    swap A[i] and A[minIndex];
}
```

What most accurately characterizes the running time of selection sort?
A. O(1)
B. O(n)
C. O(n²)
D. O(n³)
E. None of the above
Selection sort
(code adapted from Wikipedia)

• Grow sorted part of array by finding smallest value in the remaining

```java
for(int i=0; i < A.length-1; i++) {
    int minIndex = i;  //index of min
    for(int j=i; j < A.length; j++) {  //find the min
        if(A[j] < A[minIndex])
            minIndex = j;
    }
    swap A[i] and A[minIndex];
}
```

What most accurately characterizes the running time of selection sort?
A. $O(1)$
B. $O(n)$
C. $O(n^2)$
D. $O(n^3)$
E. None of the above
Insertion sort
(code adapted from Wikipedia)

- Keep sorted portion of the array and grow it by inserting new values into the proper place

```java
for(int i=0; i< A.length; i++) {
    //add value A[i]
    int j = i;  //iterator to seek possible places
    while((j > 0) && (A[j-1] > A[j])) {
        swap A[j] and A[j-1]
        j--;
    }
}
```
Insertion sort
(code adapted from Wikipedia)

• Keep sorted portion of the array and grow it by inserting new values into the proper place

```java
for(int i=0; i< A.length; i++) {
    //add value A[i]
    int j = i;  //iterator to seek possible places
    while((j > 0) && (A[j-1] > A[j])) {
        swap A[j] and A[j-1]
        j--;
    }
}
```

What most accurately characterizes the running time of insertion sort?

A. O(1)
B. O(n)
C. O(n^2)
D. O(n^3)
E. None of the above
Insertion sort
(code adapted from Wikipedia)

- Keep sorted portion of the array and grow it by inserting new values into the proper place

```java
for(int i=0; i< A.length; i++) {
    //add value A[i]
    int j = i; //iterator to seek possible places
    while((j > 0) && (A[j-1] > A[j])) {
        swap A[j] and A[j-1]
        j--;
    }
}
```

What most accurately characterizes the running time of insertion sort?

A. O(1)
B. O(n)
C. O(n^2)
D. O(n^3)
E. None of the above
Bubble sort
(code adapted from Wikipedia)

• Repeatedly find out-of-order pairs and swap them

```java
boolean swapped = true;
while(swapped) {
    swapped = false;
    for(int i=0; i < A.length-1; i++)
        if(A[i] > A[i+1]) {
            swap A[i] and A[i+1];
            swapped = true;
        }
}
```
Bubble sort
(code adapted from Wikipedia)

- Repeatedly find out-of-order pairs and swap them

```java
boolean swapped = true;
while(swapped) {
    swapped = false;
    for(int i=0; i < A.length-1; i++)
        if(A[i] > A[i+1]) {
            swap A[i] and A[i+1];
            swapped = true;
        }
}
```

What most accurately characterizes the running time of bubble sort?
A. $O(1)$
B. $O(n)$
C. $O(n^2)$
D. $O(n^3)$
E. None of the above
Bubble sort
(code adapted from Wikipedia)

• Repeatedly find out-of-order pairs and swap them

```java
boolean swapped = true;
while(swapped) {
    swapped = false;
    for(int i=0; i < A.length-1; i++)
        if(A[i] > A[i+1]) {
            swap A[i] and A[i+1];
            swapped = true;
        }
}
```

What most accurately characterizes the running time of bubble sort?

A. O(1)
B. O(n)
C. O(n²)
D. O(n³)
E. None of the above
Summary: Basic algorithms

• Selection sort
  – Grow sorted part of array by finding smallest value in the remaining

• Insertion sort
  – Grow sorted part of the array by inserting new values into the proper place

• Bubble sort
  – Repeatedly find out-of-order pairs and swap them