CS 171: Introduction to Computer Science II

Simple Sorting

Li Xiong
Elementary Sorting Algorithms

- Comparable interface
- Bubble sort
- Selection sort
- Insertion sort
Sorting problem

**Ex.** Student records in a university.

<table>
<thead>
<tr>
<th>Item</th>
<th>Key</th>
<th>Code</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen</td>
<td>3</td>
<td>A 991-878-4944</td>
<td>308 Blair</td>
</tr>
<tr>
<td>Rohde</td>
<td>2</td>
<td>A 232-343-5555</td>
<td>343 Forbes</td>
</tr>
<tr>
<td>Gazsi</td>
<td>4</td>
<td>B 766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Furia</td>
<td>1</td>
<td>A 766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Kanaga</td>
<td>3</td>
<td>B 898-122-9643</td>
<td>22 Brown</td>
</tr>
<tr>
<td>Andrews</td>
<td>3</td>
<td>A 664-480-0023</td>
<td>097 Little</td>
</tr>
<tr>
<td>Battle</td>
<td>4</td>
<td>C 874-088-1212</td>
<td>121 Whitman</td>
</tr>
</tbody>
</table>

**Sort.** Rearrange array of $N$ items into ascending order.

<table>
<thead>
<tr>
<th>Item</th>
<th>Key</th>
<th>Code</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrews</td>
<td>3</td>
<td>A 664-480-0023</td>
<td>097 Little</td>
</tr>
<tr>
<td>Battle</td>
<td>4</td>
<td>C 874-088-1212</td>
<td>121 Whitman</td>
</tr>
<tr>
<td>Chen</td>
<td>3</td>
<td>A 991-878-4944</td>
<td>308 Blair</td>
</tr>
<tr>
<td>Furia</td>
<td>1</td>
<td>A 766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Gazsi</td>
<td>4</td>
<td>B 766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Kanaga</td>
<td>3</td>
<td>B 898-122-9643</td>
<td>22 Brown</td>
</tr>
<tr>
<td>Rohde</td>
<td>2</td>
<td>A 232-343-5555</td>
<td>343 Forbes</td>
</tr>
</tbody>
</table>
Two useful sorting abstractions

**Helper functions.** Refer to data through compares and exchanges.

**Less.** Is item \( v \) less than \( w \)?

```java
private static boolean less(Comparable v, Comparable w)
{   return v.compareTo(w) < 0;
}
```

**Exchange.** Swap item in array \( a[] \) at index \( i \) with the one at index \( j \).

```java
private static void exch(Comparable[] a, int i, int j)
{   Comparable swap = a[i];
    a[i] = a[j];
    a[j] = swap;
}
```
Bubble Sort

• Intuition:
  – Find the biggest number.
  – Find the second biggest number.
  – Find the third biggest number.
  – ...

• Bubble sort achieves this by repeatedly swapping two adjacent numbers.
Analysis of Bubble Sort

- Number of comparisons?
  \[ \frac{N(N-1)}{2} = O(N^2) \]

- Number of swaps?
  best case: \( O(1) \)
  worst cast: \[ \frac{N(N-1)}{2} = O(N^2) \]
  average: \[ \frac{N(N-1)}{4} = O(N^2) \]
Selection Sort

1. Keep track of the index of the smallest number in each round.
2. Swap the smallest number towards the beginning of the array.
3. Repeat the above two steps.
Selection Sort

Algorithm. \( \uparrow \) scans from left to right.

Invariants.
- Entries the left of \( \uparrow \) (including \( \uparrow \)) fixed and in ascending order.
- No entry to right of \( \uparrow \) is smaller than any entry to the left of \( \uparrow \).
Selection Sort

- Move the pointer to the right.
  
  \[
  \text{i}++; 
  \]

- Identify index of minimum entry on right.
  
  \[
  \text{int } \text{min} = \text{i}; \\
  \text{for (int } j = \text{i+1}; j < N; j++) \\
  \text{if (less(a[j], a[\text{min}]))} \\
  \text{min} = j; 
  \]

- Exchange into position.
  
  \[
  \text{exch(a, i, min);} 
  \]
Selection Sort Implementation

```java
public class Selection {
    public static void sort(Comparable[] a) {
        int N = a.length;
        for (int i = 0; i < N; i++) {
            int min = i;
            for (int j = i+1; j < N; j++)
                if (less(a[j], a[min]))
                    min = j;
            exch(a, i, min);
        }
    }

    private static boolean less(Comparable v, Comparable w) {
        /* as before */
    }

    private static void exch(Comparable[] a, int i, int j) {
        /* as before */
    }
}
```
Selection Sort

• Online demo

• Gypsy dance demo
  – http://www.youtube.com/watch?v=Ns4TPTC8whw
Selection Sort

• Number of comparisons?

• Number of swaps?
Selection Sort

• Number of comparisons?
  \( O(N^2) \)

• Number of swaps?
  Best case: \( O(1) \)
  Worst case: \( O(N) \)
Insertion Sort

• How do you sort a hand of poker cards?
Insertion Sort

• Idea
  – Assume the left portion of the array is *partially sorted* (however, unlike selection sort, the elements are not necessarily in their final positions)
  – For each remaining element on the right portion, insert it to the left portion (similar to insertion in an ordered array).
  – Repeat until done.
Algorithm. \( \uparrow \) scans from left to right.

**Invariants.**

- Entries to the left of \( \uparrow \) (including \( \uparrow \)) are in ascending order.
- Entries to the right of \( \uparrow \) have not yet been seen.
• Move the pointer to the right.

    i++;

• Moving from right to left, exchange $a[i]$ with each larger entry to its left.

    for (int j = i; j > 0; j--)
      if (less(a[j], a[j-1]))
        exch(a, j, j-1);
      else break;
public class Insertion
{
    public static void sort(Comparable[] a)
    {
        int N = a.length;
        for (int i = 0; i < N; i++)
            for (int j = i; j > 0; j--)
                if (less(a[j], a[j-1]))
                    exch(a, j, j-1);
                else break;
    }

    private static boolean less(Comparable v, Comparable w)
    { /* as before */ }

    private static void exch(Comparable[] a, int i, int j)
    { /* as before */ }
}
Insertion Sort

• Online demo
  – http://www.sorting-algorithms.com/insertion-sort

• Romanian dance demo
  – http://www.youtube.com/watch?v=ROalU379l3U
Insertion Sort

• Number of comparisons?

• Number of exchanges?
Insertion sort

- **Best case**
  - \( N-1 \) comparisons
  - 0 exchanges

- **Worst case**
  - \( \sim N^2/2 \) comparisons
  - \( \sim N^2/2 \) exchanges
Summary

• **Bubble sort** uses repeated comparisons and swaps to find the biggest element in each pass

• **Selection sort** selects the smallest element in each pass and reduces the cost of exchanges

• **Insertion sort** inserts each element in the left assorted portion and reduces the cost of comparisons

• All have average comparison cost of $O(N^2)$