3.1 Elementary Sorts

Reference: Chapter 6, Algorithms in Java, 3rd Edition, Robert Sedgewick.

Basic Terms

Ex: student record in a University.

file → record → key →

Fox	1	A	243-456-9091	101 Brown
Quilici	1	C	343-987-5642	32 McCosh
Chen	2	A	884-232-5341	11 Dickinson
Furia	3	A	766-093-9873	22 Brown
Kanaga	3	B	898-122-9643	343 Forbes
Andrews	3	A	874-088-1212	121 Whitman
Rohde	3	A	232-343-5555	115 Holder
Battle	4	C	991-878-4944	308 Blair
Aaron	4	A	664-480-0023	097 Little
Gazsi	4	B	665-303-0266	113 Walker

Sort: rearrange sequence of objects into ascending order.

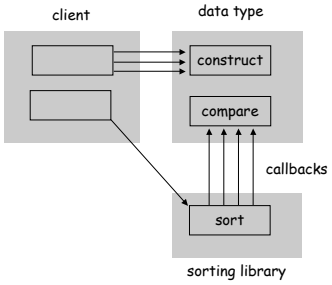
Aaron	4	A	664-480-0023	097 Little
Andrews	3	A	874-088-1212	121 Whitman
Battle	4	C	991-878-4944	308 Blair
Chen	2	A	884-232-5341	11 Dickinson
Fox	1	A	243-456-9091	101 Brown
Furia	3	A	766-093-9873	22 Brown
Gazsi	4	B	665-303-0266	113 Walker
Kanaga	3	B	898-122-9643	343 Forbes
Rohde	3	A	232-343-5555	115 Holder
Quilici	1	C	343-987-5642	32 McCosh

Rules of the Game

Goal. Write robust sorting library that can sort any type of data into sorted order using the data type's natural order.

Callbacks.
- Client passes array of objects to sorting routine.
- Sorting routine calls back object's comparison function as needed.

Implementing callbacks.
- Java: interfaces.
- C: function pointers.
- C++: functors.
- C#: delegates.
- Lisp: first class functions.



Comparable Interface

Comparable interface. Class is required to provide a method `compareTo` so that `v.compareTo(w)` returns:
- A negative integer if `v` is less than `w`.
- A positive integer if `v` is greater than `w`.
- Zero if `v` is equal to `w`.

Consistency. It is the programmer's responsibility to ensure that `compareTo` specifies a total order.
- If a < b and b < c, then a < c. [transitivity]
- Exactly one holds: a < b, b < a, a = b. [trichotomy]

Built-in comparable types: `String`, `Double`, `Integer`, `Date`, `File`.

User-defined types: easy to implement `Comparable` interface.

Implementing the Comparable Interface: Date

```
public class Date implements Comparable<Date> {
   private int month, day, year;

   public Date(int m, int d, int y) {
      month = m;
      day   = d;
      year  = y;
   }

   public int compareTo(Date b) {
      Date a = this;
      if (a.year  < b.year ) return -1;
      if (a.year  > b.year ) return +1;
      if (a.month < b.month) return -1;
      if (a.month > b.month) return +1;
      if (a.day   < b.day  ) return -1;
      if (a.day   > b.day  ) return +1;
      return 0;
   }
}
```

only compare dates to other dates

Two Array Sorting Abstractions

Helper functions. Refer to data only through two operations.

- Less. Is `v` less than `w` ?

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

- Exchange. Swap objects in array at index i with the one at index j.

```
private static void exch(Comparable[] a, int i, int j) {
      Comparable t = a[i];
      a[i] = a[j];
      a[j] = t;
   }
```


Check if Sorted

Example usage. Is the input sorted?

```
public static boolean isSorted(Comparable[] a) {
   for (int i = 1; i < a.length; i++)
      if (less(a[i], a[i-1]))
         return false;
   return true;
}
```

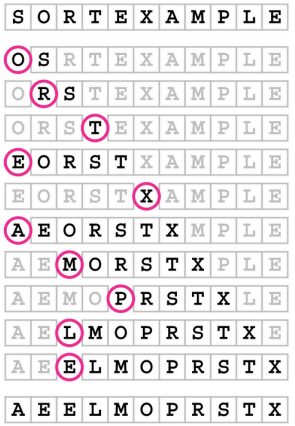

Insertion Sort

Insertion Sort

Insertion sort.

- Scans from left to right.
- Element to right of ↑ are not touched.
- Invariant: elements to the left of ↑ are in ascending order.
- Inner loop: repeatedly swap element ↑ with element to its left.

Insertion Sort Example



Insertion Sort: Java Implementation

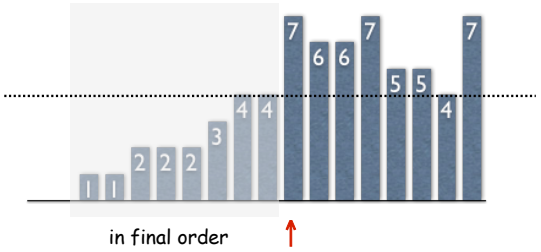
```
public static void sort(Comparable[] a) {
   int N = a.length;
   for (int i = 0; i < N; i++)
      for (int j = i; j > 0; j--)
         if (less(a[j], a[j-1])) exch(a, j, j-1);
         else break;
}
```


Selection Sort

Selection Sort

Selection sort.

- ↑ scans from left to right.
- Elements to the left of ↑ are fixed and in ascending order.
- No element to left of ↑ is larger than any element to its right.



Selection Sort Example

S	O	R	T	E	X	A	M	P	L	E	
S	O	R	T	E	X	A	M	P	L	E	
A	O	R	T	E	X	S	M	P	L	E	
A	E	R	T	O	X	S	M	P	L	E	
A	E	E	T	O	X	S	M	P	L	R	
A	E	E	L	O	X	S	M	P	T	R	
A	E	E	L	M	X	S	O	P	T	R	
A	E	E	L	M	O	S	X	P	T	R	
A	E	E	L	M	O	P	X	S	T	R	
A	E	E	L	M	O	P	R	S	T	X	
A	E	E	L	M	O	P	R	S	T	X	
A	E	E	L	M	O	P	R	S	T	X	

Selection Sort Inner Loop: Maintaining the Invariant

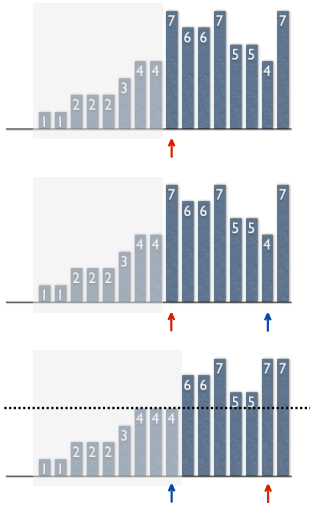
Selection sort inner loop.

- Identify index of minimum item.

```
int min = i;
for (int j = i+1; j < N; j++)
   if (less(a[j], a[min]))
      min = j;
```

- Exchange into position.

```
exch(a, i, min);
```



Selection Sort: Java Implementation

```
public class Selection {

   private static boolean less(Comparable v, Comparable w) {
      return v.compareTo(w) < 0;
   }                                                  is v less than w?

   private static void exch(Comparable[] a, int i, int j) {
      Comparable swap = a[i];
      a[i] = a[j];
      a[j] = swap;
   }                                          swap references a[i] and a[j]

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

Selection Sort: Sample Application

List files. List the files in the current directory, sorted by file name.

```
import java.io.File;

public class Files {
   public static void main(String[] args) {
      File directory = new File(args[0]);
      File[] files = directory.listFiles();
      Selection.sort(files);
      for (int i = 0; i < files.length; i++)
         System.out.println(files[i]);
   }
}
```

```
% java Files .
Insertion.class
Insertion.java
InsertionX.class
InsertionX.java
Selection.class
Selection.java
Shell.class
Shell.java
ShellX.class
ShellX.java
index.html
```


Analysis

Performance for Randomly Ordered Files

Selection.

- Always search through right part.
- (1 + 2 + ... + N) ≈ N^2 / 2 compares.
  ≈ N exchanges.

Insertion.

- Each element moves halfway back.
- (1 + 2 + ... + N) / 2 ≈ N^2 / 4 compares.
  ≈ N^2 / 4 exchanges.

Bottom line: insertion, selection similar.

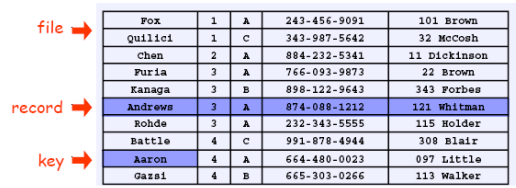
Sorting Challenges

Sorting Challenge 1

Problem: sort a file of huge records with tiny keys.
Ex: reorganizing your MP3 files.

Which sorting method to use?

1. system sort
2. insertion sort
3. selection sort



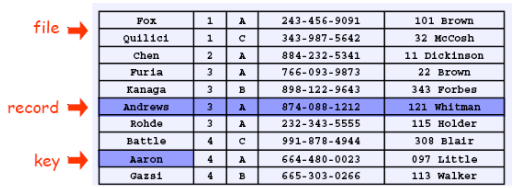
Fox	1	A	243-456-9091	101 Brown
Quilici	1	C	343-987-5642	32 McCosh
Chen	2	A	884-232-5341	11 Dickinson
Furia	3	A	766-093-9873	22 Brown
Kanaga	3	B	898-122-9643	343 Forbes
Andrews	3	A	874-088-1212	121 Whitman
Rohde	3	A	232-343-5555	115 Holder
Battle	4	C	991-878-4944	308 Blair
Aaron	4	A	664-480-0023	097 Little
Gazsi	4	B	665-303-0266	113 Walker

Sorting Challenge 2

Problem: sort a huge randomly-ordered file of small records.
Ex: process transaction records for a phone company.

Which sorting method to use?

1. system sort
2. insertion sort
3. selection sort



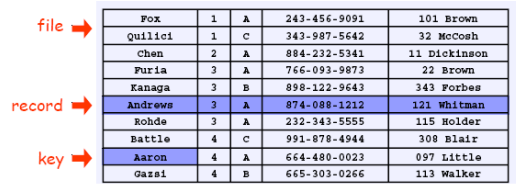
Fox	1	A	243-456-9091	101 Brown
Quilici	1	C	343-987-5642	32 McCosh
Chen	2	A	884-232-5341	11 Dickinson
Furia	3	A	766-093-9873	22 Brown
Kanaga	3	B	898-122-9643	343 Forbes
Andrews	3	A	874-088-1212	121 Whitman
Rohde	3	A	232-343-5555	115 Holder
Battle	4	C	991-878-4944	308 Blair
Aaron	4	A	664-480-0023	097 Little
Gazsi	4	B	665-303-0266	113 Walker

Sorting Challenge 3

Problem: sort a huge number of tiny files (each file is independent)
Ex: daily customer transaction records.

Which sorting method to use?

1. system sort
2. insertion sort
3. selection sort



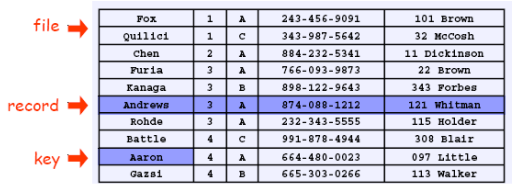
Fox	1	A	243-456-9091	101 Brown
Quilici	1	C	343-987-5642	32 McCosh
Chen	2	A	884-232-5341	11 Dickinson
Furia	3	A	766-093-9873	22 Brown
Kanaga	3	B	898-122-9643	343 Forbes
Andrews	3	A	874-088-1212	121 Whitman
Rohde	3	A	232-343-5555	115 Holder
Battle	4	C	991-878-4944	308 Blair
Aaron	4	A	664-480-0023	097 Little
Gazsi	4	B	665-303-0266	113 Walker

Sorting Challenge 4

Problem: sort a huge file that is already almost in order.
Ex: re-sort a huge database after a few changes.

Which sorting method to use?

1. system sort
2. insertion sort
3. selection sort



Fox	1	A	243-456-9091	101 Brown
Quilici	1	C	343-987-5642	32 McCosh
Chen	2	A	884-232-5341	11 Dickinson
Furia	3	A	766-093-9873	22 Brown
Kanaga	3	B	898-122-9643	343 Forbes
Andrews	3	A	874-088-1212	121 Whitman
Rohde	3	A	232-343-5555	115 Holder
Battle	4	C	991-878-4944	308 Blair
Aaron	4	A	664-480-0023	097 Little
Gazsi	4	B	665-303-0266	113 Walker

Visual Sorting Puzzle

1. Insertion sort.
2. Selection sort.
3. Bubble sort.