Sorting a List:
- bubble sort
- selection sort
- insertion sort

Sept. 22, 2017
## Sorting

<table>
<thead>
<tr>
<th>BEFORE</th>
<th>AFTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-5</td>
</tr>
<tr>
<td>17</td>
<td>-2</td>
</tr>
<tr>
<td>-5</td>
<td>3</td>
</tr>
<tr>
<td>-2</td>
<td>4</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
</tr>
</tbody>
</table>
Example: sorting exams by last name
Sorting Algorithms

• Bubble sort
• Selection sort \( \mathcal{O}(N^2) \) today
• Insertion sort

\[ \text{today \quad } \mathcal{O}(N^2) \]

• Mergesort
• Heapsort \( \mathcal{O}(N \log N) \) later
• Quicksort
Sorting Algorithms

Today we are concerned with algorithms, not data structures.

The following algorithms are independent of whether we use an array list or a linked list.
Bubble Sort

Repeatedly loop (iterate) through the list.
For each iteration,
    if two neighboring elements are in the wrong order,
    then swap them.
Reminder from 202: \( \text{swap}(x, y) \)

The following does not work:

\[
x = y  \\
y = x
\]

Rather, you need to use a temporary variable:

\[
tmp = y  \\
y = x  \\
x = tmp
\]
**Example:** first pass

```
if list[0] > list[1]
    swap( list[0], list[1] )
```
Example: first pass

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>17</td>
<td>-5</td>
<td>-2</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>17</td>
<td>-5</td>
<td>-2</td>
<td>23</td>
<td>4</td>
</tr>
</tbody>
</table>

If list[1] > list[2]
swap(list[1], list[2])

Indicates elements get swapped
Example: first pass

Indicates elements get swapped
What can we say at end of the first pass?

Q: Where is the largest element?
A:

Q: Where is the smallest element?
A:
What can we say at end of the first pass?

Q: Where is the largest element?

A: It must be at the end of the list (position N-1).

Q: Where is the smallest element?

A: Anywhere (except position N-1).
Bubble Sort Algorithm

repeat {
    continue = false
    for i = 0 to N – 2 // N-1 is the last index
        if list[ i ] > list[ i + 1 ]{
            swap( list[ i ], list[ i + 1 ] )
            continue = true
        }
} until continue == false
Bubble Sort Algorithm

repeat {
    ct = 0
    continue = false
    for i = 0 to N - 2 - ct {  // N-1 is the last index
        if list[i] > list[i + 1] {
            swap( list[i], list[i + 1] )
            continue = true
        }
    }
    ct = ct + 1  // now list[ N - ct, ... N-1] is sorted
}
} until continue == false
Selection Sort

Partition the list into two parts: (1) a sorted part and (2) a “rest” part, as follows:

The sorted part is initially empty.

Repeat N times {
find the smallest element in the rest part and swap it with the first element in the rest part
}
Example

\[
\begin{array}{c}
3 \\
17 \\
-5 \\
-2 \\
23 \\
4 \\
\end{array}
\]

sorted part is empty

rest
Example

\[
\begin{array}{c}
3 \\
17 \\
\text{(-5)} \\
-2 \\
23 \\
4 \\
\end{array}
\quad \begin{array}{c}
-5 \\
17 \\
3 \\
-2 \\
23 \\
4 \\
\end{array}
\]
Example

\[
\begin{array}{c}
  3 \\
  17 \\
  -5 \\
  -2 \\
  23 \\
  4 \\
\end{array}
\]

\[
\begin{array}{c}
  -5 \\
  17 \\
  3 \\
  -2 \\
  23 \\
  4 \\
\end{array}
\]

\[
\begin{array}{c}
  -5 \\
  -2 \\
  3 \\
  17 \\
  23 \\
  4 \\
\end{array}
\]

sorted

rest
Example

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>17</td>
<td>-5</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>-2</td>
<td>3</td>
<td>-2</td>
<td>17</td>
</tr>
<tr>
<td>-5</td>
<td>3</td>
<td>23</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>-5</td>
<td>-2</td>
<td>17</td>
<td>17</td>
<td>3</td>
</tr>
</tbody>
</table>

sorted

rest
### Example

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-5</td>
<td>17</td>
<td>17</td>
<td>-2</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>-2</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>-5</td>
<td>3</td>
<td>17</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

- The elements are sorted.
- The rest of the elements are not sorted.
Selection Sort

for i = 0 to N-2  {
    index = i
    minValue = list[ i ]

    for k = i+1 to N-1  {
        if ( list[k] < minValue ){
            index = k
            minValue = list[k]
        }
    }
    if ( index != i )
        swap( list[i], list[ index ] )
}

// repeat N times
// Take the first element in the rest.
// It has the min value so far.

// For each other element in rest,
// if it is smaller than the min value,
// then remember its index.
// It is the new min value.

// Swap if necessary
Selection Sort

for i = 0 to N-2
  for k = i+1 to N-1
    ......

Q: how many passes through inner loop?
Selection Sort

for i = 0 to N-2
    for k = i+1 to N-1
        ......

Q: how many passes through inner loop?
A: N-1 + N-2 + N-3 + .... + 2 + 1
Selection Sort

for i = 0 to N-2
    for k = i+1 to N-1
        .......

Q: how many passes through inner loop?

A: \[ N-1 + N-2 + N-3 + \ldots + 2 + 1 \]
   \[ = \frac{N(N-1)}{2} \]
Comparison

Bubblesort

repeat {
    for i = 0 to N - 2 - ct
until continue == false

We can terminate outer loop if there are no swaps during a pass.

Best case

Worst case

Selection sort

for i = 0 to N-2
for k = i+1 to N-1

Best case

Worst case

Outer loop

Outer loop
Insertion Sort

for k = 1 to N-1 {
    Insert list element at index k into its correct position with respect to the elements at indices 0 to k – 1
}


Initial list

3
17
-5
-2
23
4
Suppose we have sorted elements 0 to k-1

e.g. k = 3
Suppose we have sorted elements 0 to k-1, e.g., k = 3. Insert element k into its correct position with respect to 0 to k-1.

Initial list:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>17</td>
<td>-5</td>
<td>-2</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Insert element k = 3:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>3</td>
<td>17</td>
<td>-2</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Final list:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>3</td>
<td>17</td>
<td>-2</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Mechanism is similar to inserting (adding) an element to an array list:

Shift all elements ahead by one position to make a hole, and then fill the hole.
Insertion Sort

for k = 1 to N - 1  { 
    elementK = list[k]
    i = k
    while (i > 0) and ( elementK < list[ i - 1]){
        list[i] = list[i - 1]         // copy to next
        i = i -1
    }
    list[i] = elementK        // paste elementK
}
Best case:

the list is already sorted, so it takes $O(N)$ time.

i.e. the while loop terminates immediately.

Worse case:

the list is sorted in backwards order.

$$1 + 2 + 3 + \ldots + N - 1 = \frac{N(N - 1)}{2}$$

which takes time $O(N^2)$. Lots of shifts!
## Comparison of 3 methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Best Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubblesort</td>
<td>repeat { for i = 0 to N – 2 – ct until continue == false</td>
<td>We can terminate outer loop if there are no swaps during a pass.</td>
</tr>
<tr>
<td>Selection sort</td>
<td>for i = 0 to N-2 for k = i+1 to N-1</td>
<td></td>
</tr>
<tr>
<td>Insertion sort</td>
<td>for k = 1 to N - 1 { while ....</td>
<td></td>
</tr>
</tbody>
</table>

Performance depends highly on initial data. Also, it depends on implementation (array vs. linked list), e.g. what is cost of swap and ‘shift’. 
Assignment 1 division question: hint

\[
\begin{array}{c}
723 \overline{)41672542996} \\
3615 \\
\hline
3124 \\
552 \ldots \text{ etc}
\end{array}
\]

You need to rethink what you are doing. Don’t just try to blindly code what you learned in grade school.
Quiz 1 on Monday on mycourses

8 AM to 8 PM

No discussion during that time.

Email me if there is a problem.

Solutions, grades, feedback will be posted after.