Bubble-Sort on Sequences

Outline and Required Reading:

• Bubble-Sort on a Sequence  (§ 5.4)
Bubble-Sort Algorithm

**Sorting Problem** – reorder a sequence so that the elements are organized in non-decreasing order

- various criteria for element comparison can apply (A<T, 2<3, ...)

**Sort Algorithms** – well-researched area – best sort algorithms run in $O(n \cdot \log(n))$ time to sort $n$ elements

**Bubble Sort Algorithm** – a simple sort algorithm – runs in $O(n^2)$ time

- not very efficient – better algorithms exist
- however, works equally well on arrays and linked lists
Bubble-Sort Algorithm (cont.)

**Bubble Sort Algorithm** – performs a series of passes over the sequence

**Step 1:** Compare each element (except the last one) with its neighbour to the right – if they are out of order, swap them.

**Step 2:** Compare each element (except the last two) with its neighbour to the right – if they are out of order, swap them.

... 

Continue as above, until there are no unsorted elements on the left.

**NOTE:** At the end of $i^{th}$ pass, the right-most $i$ elements are in final position

**NOTE:** $i^{th}$ pass can be limited to the first $(n-i+1)$ elements
Bubble-Sort Algorithm (cont.)

Example 1 [ Bubble Sort ]

1st pass

2nd pass

3rd pass

4th pass

Largest element found.

2nd largest element found.

3rd largest element found.

4th largest element, out of 5, found – sorting DONE!
Bubble-Sort Algorithm (cont.)

Run-Time Complexity of Bubble Sort Algorithm

General assumption: access and swap method take $O(1)$!

for (int $j=0$; $j<a.length-1$; $j++$) {
    for (int $i=0$; $i<(a.length-1-j)$; $i++$) {
        if ($a[i] > a[i+1]$) {
            temp = $a[i]$;
            $a[i] = a[i+1]$;
            $a[i+1] = temp$;
        }
    }
}

$(n-1)$ passes in total !!!

$n$ – in textbook

$$RT(n) = \sum_{j=1}^{n-1} \sum_{i=1}^{n-j} 1 = ((n-1) + (n-2) + \ldots + 1) = \frac{(n-1)n}{2} = O(n^2)$$
boolean newSwaps = true;
while (newSwaps) {
    newSwaps = false;
    for (int i=0; i<a.length-1; i++) {
        if (a[i] > a[i+1]) {
            temp = a[i];
            a[i] = a[i+1];
            a[i+1] = temp;
            newSwaps = true;
        }
    }
}

Worst-case running time, still $O(n^2)$ – when elements in decreasing order.
Best-case running time $\Omega(n)$ – for an almost sorted sequence.
Bubble-Sort on Sequences

**Sequence ADT** – elements can be accessed as ranks or as positions both in list- and array- implementation

```java
void bubbleSort(Sequence S) {
    int n = S.size();
    for (int i = 1; i<n; i++) {
        for (int j = 0; j<n-i; j++)
            if (valueAtRank(S,j) > valueAtRank(S,j+1))
                S.swapElements(S.atRank(j-1), S.atRank(j));
    }
}

void bubbleSort(Sequence S) {
    int n = S.size();
    for (int i = 1; i<n; i++) {
        Position prec = S.first();
        for (int j = 0; j<n-i; j++) {
            Position succ;
            succ = S.after(prec);
            if (valueAtPos(prec) > valueAtPos(succ)) {
                S.swapElements(prec,succ);
            }
            prec = succ;
        }
    }
}
```
Bubble-Sort on Sequences (cont.)

Cost of Bubble Sort on Sequence – the real cost of access and swap methods must be taken into consideration!

<table>
<thead>
<tr>
<th>Sequence Method</th>
<th>Array</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>after</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>atRank</td>
<td>O(1)</td>
<td>O(n)</td>
</tr>
<tr>
<td>swapElements</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cost of Access + Swap</th>
<th>Array Sequence</th>
<th>List Sequence</th>
</tr>
</thead>
</table>
| access by rank        | atRank + swapElements  
                        | Cost: O(1)       | atRank + swapElements  
                        |                 | Cost: O(n)       |
| access by position    | after + swapElements  
                        | Cost: O(1)       | after + swapElements  
                        |                 | Cost: O(1)       |
### Overall Cost of Bubble Sort on Sequence

<table>
<thead>
<tr>
<th>Overall Cost of Bubble Sort</th>
<th>Array Sequence</th>
<th>List Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>access by rank</td>
<td>$O(n^2)$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>access by position</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
</tr>
</tbody>
</table>