The problem

Given \( a[0], a[1], \ldots, a[n-1] \)
reorder entries so that
\[
a[0] \leq a[1] \leq \ldots \leq a[n-1]
\]
a very common operation
lots of application
(search in a database...)

First: Look at a simpler problem
Suppose
\[
a[0] \leq a[1] \leq \ldots \leq a[i-1]
\]
How to add \( a[i] \) so that
\[
a[0] \leq a[1] \leq \ldots \leq a[i]
\]
e.g.: \( 1 \ 4 \ 5 \ 8 \ 9 \ 6 \)
add 6 \[
1 \ 4 \ 5 \ 6 \ 8 \ 9
\]

Insertion algorithm

1. Store \( a[i] \) in a temporary variable \( \text{temp} \)
2. Shift all elements \( > \text{temp} \) to the right by one position
3. Insert \( \text{temp} \) in the space created

Requirement:
a is sorted up through \( a[i-1] \)
(if not, does NOT work)

Outcome:
after insertion, \( a \) is sorted up through \( a[i] \)

Insert as a method

```java
public void insert(int[] a, int i)
{
    int j, temp;
    temp = a[i];
    for(j=i; j>0 && a[j-1]>temp; j--)
        a[j] = a[j-1];
    a[j] = temp;
}
```

How does it work?

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Exit the loop: \( j \) is 3
\[
1 \ 4 \ 5 \ 6 \ 8 \ 9
\]
```

Insertion Sort

Use insert
```java
public void sort (int[] a)
{
    int i;
    for(i=1; i<a.length; i++)
        insert(a,i);
}
```

Example:
```
Example: \( \text{sort}(a) \)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>
```

initially
**Bubble Sort**

**Idea:**
- Compare $a[i]$ and $a[i+1]$
- If $a[i] > a[i+1]$, exchange $a[i]$ and $a[i+1]$

**Loop over:**
- $i = 0$ to $dim - 2$
- $i = 0$ to $dim - 3$
- $\vdots$
- $i = 0$ to $0$

---

**Example: first pass of bubble sort ($i=0$ to 3)**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>initially</td>
<td>10</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>15</td>
<td>5</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>15</td>
<td>5</td>
<td>3</td>
<td>20</td>
</tr>
</tbody>
</table>

20 drifts to the top (like a bubble in a liquid)

For all of the passes:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>initially</td>
<td>10</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>5</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

---

**Efficiency is the key**

Bubble Sort is less efficient than Insert Sort

Check it out

There are many sorting algorithms:

to find the best algorithm, theory + practice