



Arrays

Part 02

Arrays

- Arrays are a collection of variables of the same type
- Foundational Data Structure
- Contiguous Block of Memory
 - The size of the Array must be specified initially
 - Arrays cannot be resized
- In Java, Arrays are considered a special kind of Object
 - Container Object
 - Identifiers contain only the reference to its contents
 - The reference *points* to contents
 - “==” Does not check the contents of the array

Creating an Array Syntax

```
//Declaring an Array
<<type>>[] <<id>>;
//Initializing an Array]
<<id>> = new <<type>>[<<size>>];
//or
<<type>>[] <<id>> = new <<type>>[size];
```

Example

```
//Creates an array of 5 integers
int[] array = new int[5];
```

Arrays

- If the values are known, it is possible to both construct the array and initialize the values at the same time.
- Values are put inside of curly braces (“{}”)
- Each value is separated by a comma (“,”)

Creating an Array Syntax

```
//Declaring an Array and Initializing its Values  
<<type>>[] <<id>> = {<<Value0>>,<<Value1>>,...};
```

Example

```
//Creates an array of 5 integers  
int[] array = {0,1,2,3,4};
```

Searching and Sorting

- Search and Sorting Arrays are fundamental to their function
- Searching involves *looking* through an array for some “target” value
 - Target values can be specific values
 - They can also be values with special properties like the minimum or maximum

Searching for a Target Value

```
int[] a = {0,1,2,3,4};
boolean found = false;
int target = keyboard.nextInt();
for(int i=0;i<a.length;i++)
{
    if(a[i] == target)
    {
        found = true;
    }
}
```

Searching and Sorting

- Search and Sorting Arrays are fundamental to their function
- Searching involves *looking* through an array for some “target” value
 - Target values can be specific values
 - They can also be values with special properties like the minimum or maximum

Searching for the MAX Value

```
int[] a = {0,1,2,3,4};
int max = a[0]; //Should not be arbitrary
for(int i=1;i<a.length;i++)
{
    if(a[i] > max)
    {
        max = a[i];
    }
}
```

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

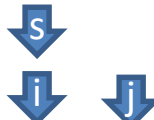
Selection Sort Example

Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram shows three blue arrows pointing downwards to a table. The top arrow is labeled 's' and points to index 0. The middle arrow is labeled 'i' and points to index 0. The bottom arrow is labeled 'j' and points to index 1.

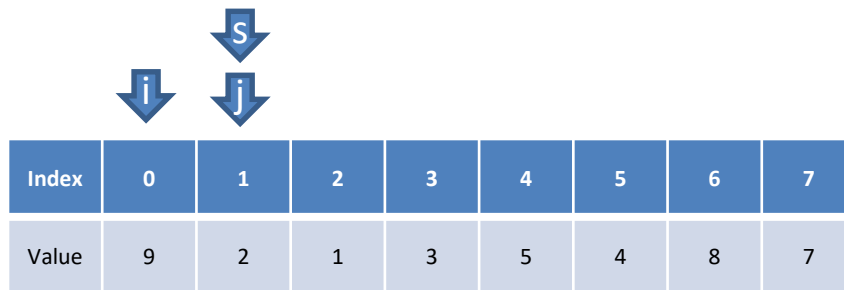
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows a table with two rows: 'Index' and 'Value'. The 'Index' row contains values 0 through 7. The 'Value' row contains values 9, 2, 1, 3, 5, 4, 8, and 7. Above the table, there are three blue arrows pointing downwards. The first arrow points to index 0 and is labeled 'i'. The second arrow points to index 1 and is labeled 'j'. The third arrow points to index 1 and is labeled 's', indicating that 's' is the index of the smallest value found in the current iteration.

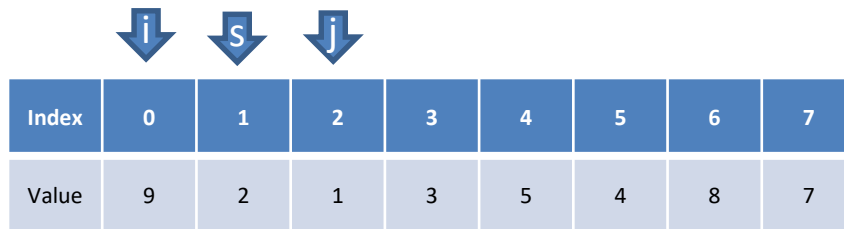
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows an array of values: 9, 2, 1, 3, 5, 4, 8, 7. The current index being processed is 0 (labeled 'i'). The smallest value in the array is 1, located at index 2 (labeled 'j'). The index of the current element is 0 (labeled 's').

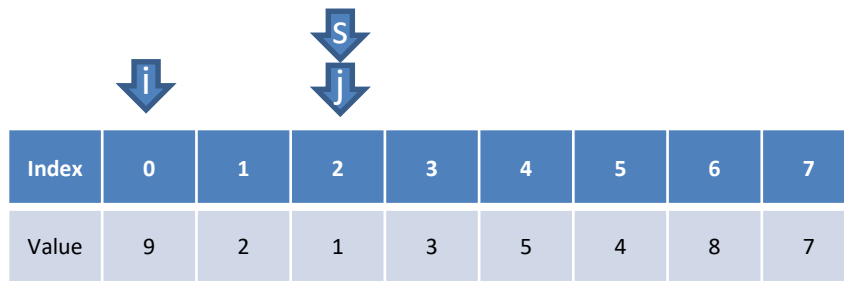
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows a table with two rows: 'Index' and 'Value'. The indices are 0 through 7, and the values are 9, 2, 1, 3, 5, 4, 8, and 7. A blue arrow labeled 'i' points to index 0. Another blue arrow labeled 's' points to index 2, and a blue arrow labeled 'j' points to index 2, indicating that 's' is the current index being compared and 'j' is the index of the smallest value found so far.

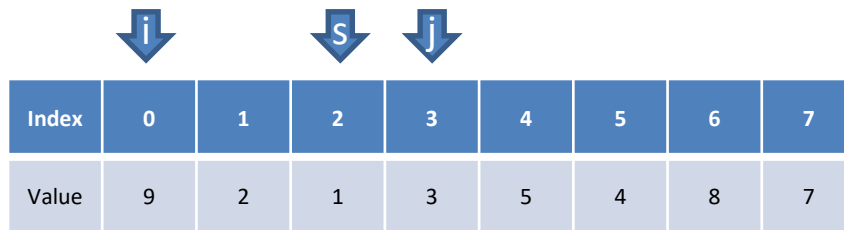
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows an array of values: 9, 2, 1, 3, 5, 4, 8, 7. Above the array, three blue arrows point to indices 0, 2, and 3, labeled 'i', 's', and 'j' respectively. This indicates that 'i' is the current index being checked, 's' is the index of the smallest value found so far, and 'j' is the index of the next element to be compared.

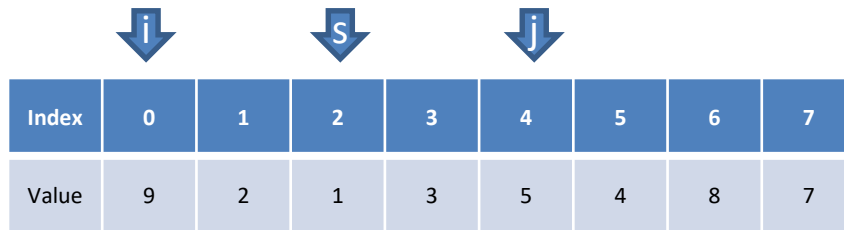
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows a horizontal array of 9 elements. Above the array, three blue arrows point downwards to specific indices: 'i' points to index 0, 's' points to index 2, and 'j' points to index 4. Below the array is a table with two rows: 'Index' and 'Value'. The 'Index' row contains values 0 through 7, and the 'Value' row contains values 9, 2, 1, 3, 5, 4, 8, 7. The values 9, 2, 1, 3, 5, 4, 8, 7 are arranged in a light blue grid.

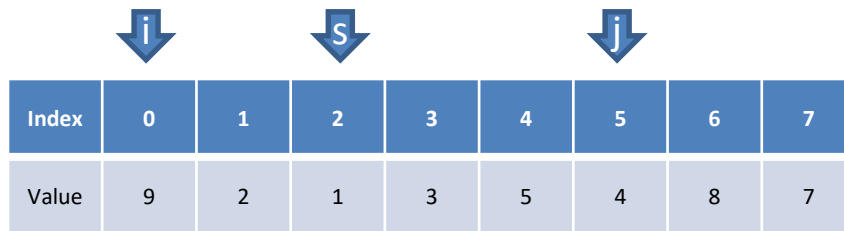
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows an array of values: 9, 2, 1, 3, 5, 4, 8, 7. The current index being checked is 0 (marked with 'i'). The smallest value found so far is at index 2 (marked with 's'). The next index to check is 5 (marked with 'j').

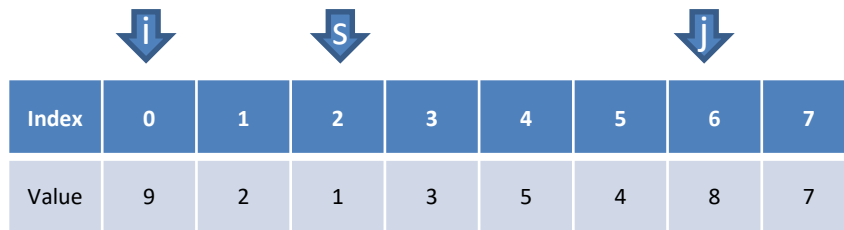
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows a table with two rows: 'Index' and 'Value'. The indices are 0 through 7, and the values are 9, 2, 1, 3, 5, 4, 8, 7. Three blue arrows point down to indices 0, 2, and 6, labeled 'i', 's', and 'j' respectively. This indicates that 'i' is the current index being checked, 's' is the index of the smallest value found so far, and 'j' is the index of the current element being compared.

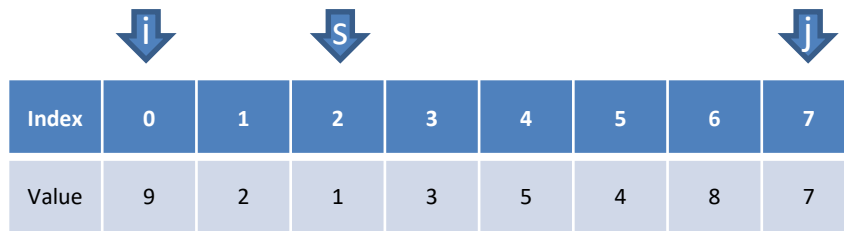
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the first pass of Selection Sort. It shows an array of 9 elements with indices 0 through 7. Above the array, three blue arrows point down to indices 0, 2, and 7. The arrow at index 0 is labeled 'i', the arrow at index 2 is labeled 's', and the arrow at index 7 is labeled 'j'. Below the array is a table with two rows: 'Index' and 'Value'. The 'Index' row contains values 0, 1, 2, 3, 4, 5, 6, 7. The 'Value' row contains values 9, 2, 1, 3, 5, 4, 8, 7.

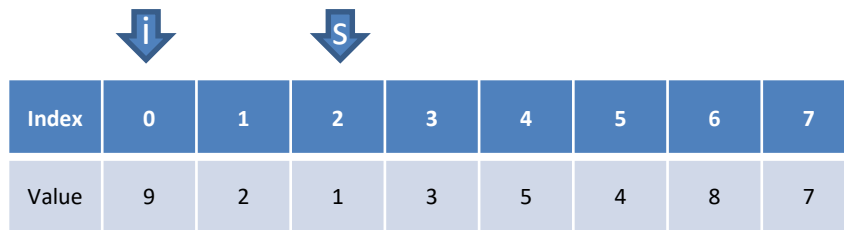
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows a table with two rows: 'Index' and 'Value'. The indices are 0 through 7, and the values are 9, 2, 1, 3, 5, 4, 8, and 7. A blue arrow labeled 'i' points to index 0, and another blue arrow labeled 's' points to index 2, indicating that 's' is the current smallest value found.

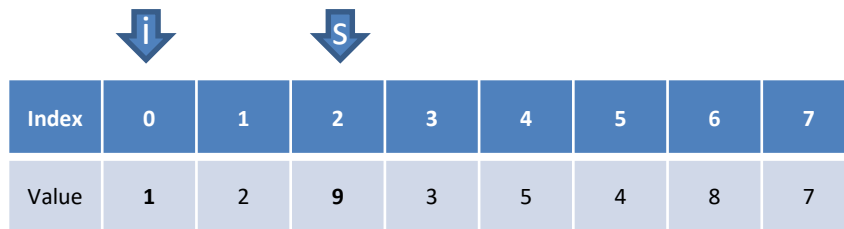
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows a table with two rows: 'Index' and 'Value'. The 'Index' row contains values 0 through 7. The 'Value' row contains values 1, 2, 9, 3, 5, 4, 8, 7. A blue arrow labeled 'i' points to index 0, and another blue arrow labeled 's' points to index 2. This indicates that the algorithm is currently comparing the value at index 0 (1) with the value at index 2 (9) to find the smallest value.

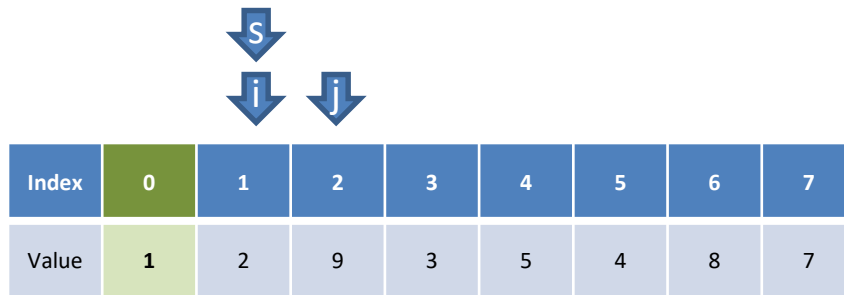
Index	0	1	2	3	4	5	6	7
Value	1	2	9	3	5	4	8	7

s = Index with
smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
 1. Start at the first index
 2. Assume this value is in the correct location
 3. Check all other values after this index
 4. If a smaller value is found, then mark that index
 5. If the assumed index does not contain smallest value, then swap that with the marked index
 6. Repeat step 2 for all indices

Selection Sort Example



The diagram illustrates the Selection Sort algorithm. It shows a horizontal array of 9 elements. The first element (index 0) is highlighted in green and labeled 's'. The second element (index 1) is labeled 'i'. The third element (index 2) is labeled 'j'. Blue arrows point down from 's', 'i', and 'j' to their respective indices in the array. The array contains the values [1, 2, 9, 3, 5, 4, 8, 7].

Index	0	1	2	3	4	5	6	7
Value	1	2	9	3	5	4	8	7

s = Index with
smallest value

Example

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2


Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	2	9	1	3	5	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	2	9	1	3	5	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	2	1	9	3	5	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	2	1	9	3	5	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	2	1	3	9	5	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	2	1	3	9	5	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example




Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	9	4	8	7

$j = i+1$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



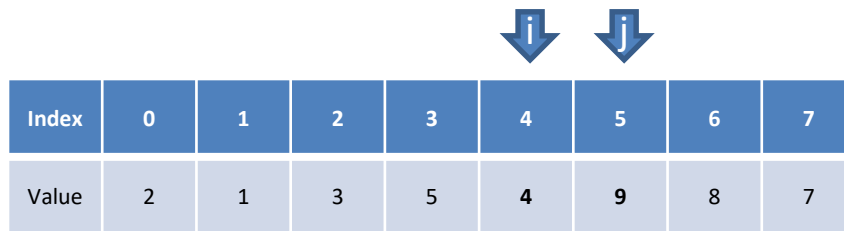
Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	9	4	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



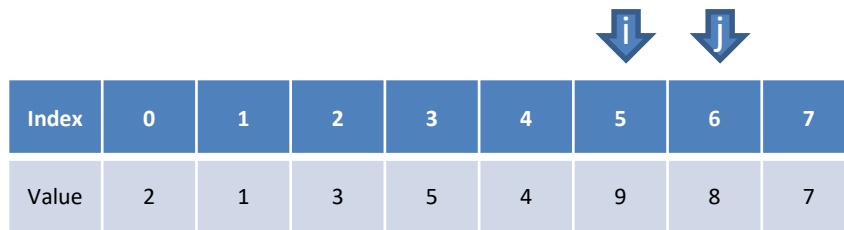
Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	9	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



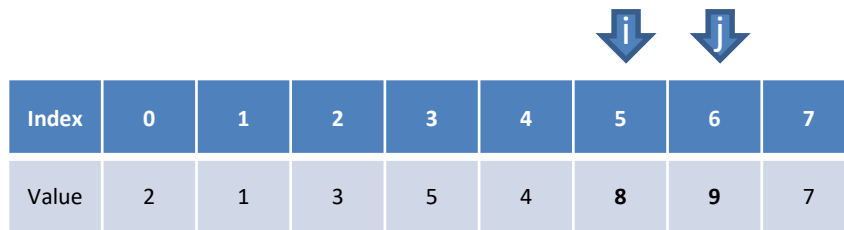
Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	9	8	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example





Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	8	9	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



								
Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	8	9	7

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example


								
Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	8	7	9

$$j = i+1$$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



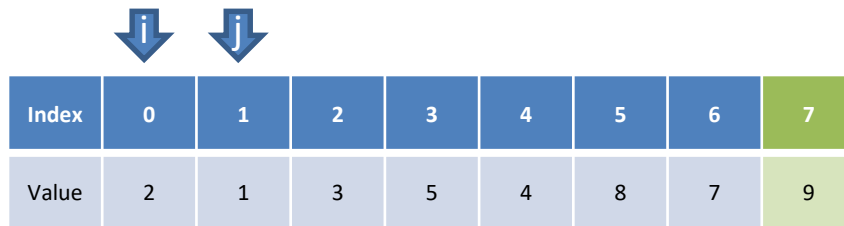
Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	8	7	9

$j = i+1$

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
 1. Start at the first index
 2. Examine that index's *neighbor*
 3. If the neighbor has a smaller value, then swap values
 4. Move to the next index
 5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



The diagram shows an array of 9 elements. The first eight elements are in blue boxes, and the last element is in a green box. Two blue arrows labeled 'i' and 'j' point to the first and second elements respectively.

Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	8	7	9

$$j = i+1$$

Example