Sorting Arrays

- What?
  - Rearrange the array elements so that they are in sorted order.
- Why?
  - Think of a phone book
- How?
  - Review some array techniques
  - Use standard problem solving techniques

Moving the elements (review)

- What if we want to swap the first and last elements of an array?
- Here we don't need a for loop.
- Does this work?

```
lastIndex = num.length - 1;
num[lastIndex] = num[0];
num[0] = num[lastIndex];
```

Moving the elements

- Method to swap array elements

```
// swaps array elements whose
// indices are i and j
void swap(int [] num, int i, int j)
{
   int tmp = num[i]
   num[i] = num[j];
   num[j] = tmp;
}
```

Moving the elements

- Using this method, we can swap the first and last elements of an array as follows:

```
swap(num, 0, num.length-1);
```

Sorting

- Assume we're sorting in ascending order (i.e. smallest first: the numbers then ascend)

so

22	47	23	10	3	48	11

is rearranged to become

3	10	11	22	23	47	48

Sorting

- So does any subproblem come to mind? Some simpler problem that will bring you closer to the final solution.

22	47	23	10	3	48	11

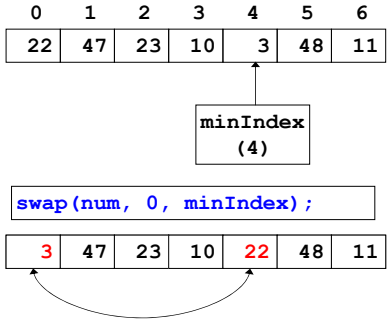
Sorting

One subproblem is to find the smallest element and move it to the front. Note that to move an element you need its position (or index).

22	47	23	10	3	48	11

The smallest element is `a[4]` (which contains 3), we could swap that with the first element. (The index of the smallest element is **4**.)

Sorting

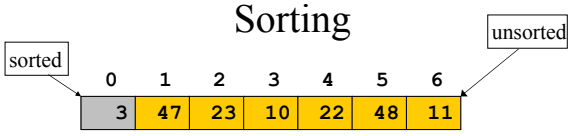


Sorting

0	1	2	3	4	5	6
3	47	23	10	22	48	11

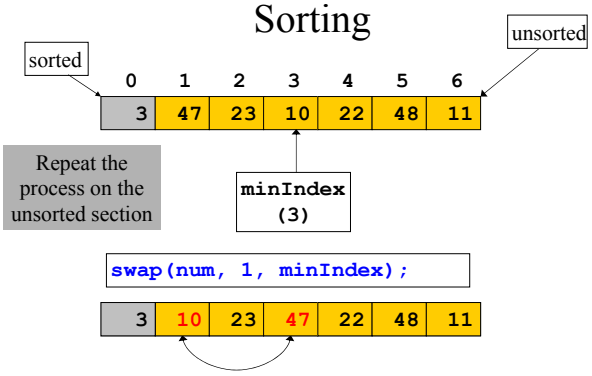
- The first element is now in the correct position.
- What's the next step?
- Can we apply the same technique again; finding the next smallest element and placing it correctly.

Sorting

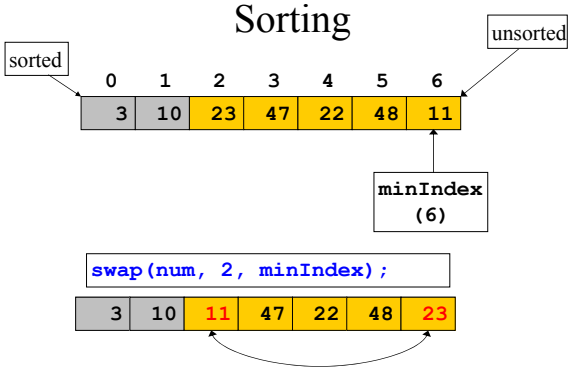


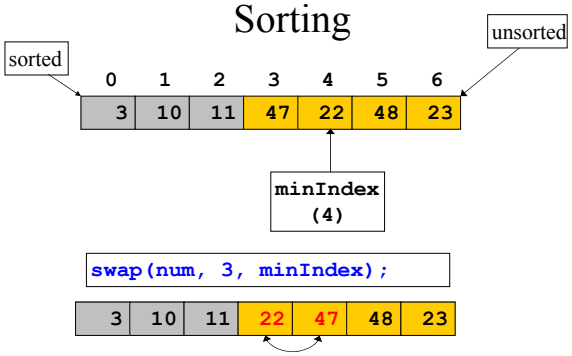
- It helps to divide the array into two sections: a sorted section and an unsorted section.
- We can concentrate on the unsorted section.
- Find the smallest element in the unsorted section and move it to the front.

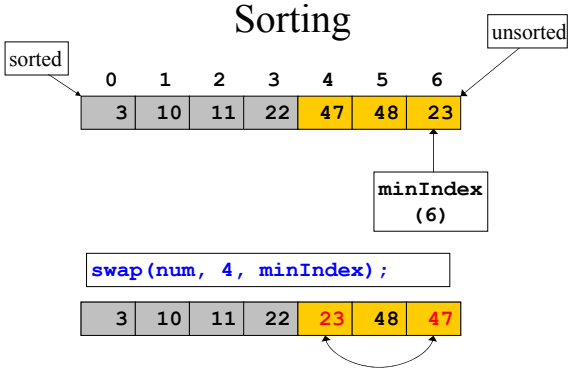
Sorting

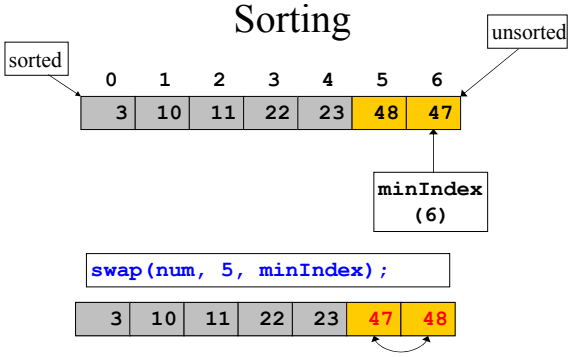


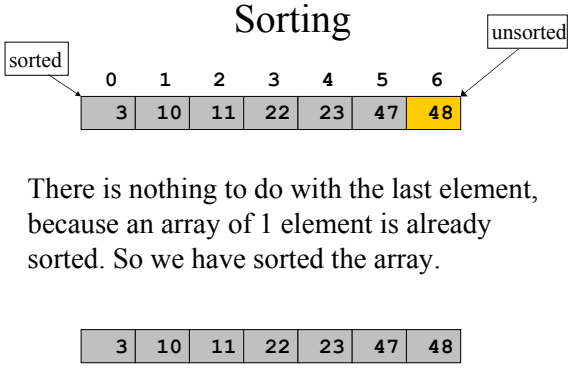
Sorting











There is nothing to do with the last element, because an array of 1 element is already sorted. So we have sorted the array.

So how do we turn this into code

- We have seen conceptually how we sort an array.
- Do it manually first, write down the swaps required to sort an array of 7 numbers.

So how do we turn this into code

- Here are all the swap method calls

```
swap(num, 0, minIndex);
swap(num, 1, minIndex);
swap(num, 2, minIndex);
swap(num, 3, minIndex);
swap(num, 4, minIndex);
swap(num, 5, minIndex);
```

Also note we don't go to the end of the array

However `minIndex` is different each time.

Question

- Could the minindex be the same?
- Can you think of a situation where all 6 times that minIndex is calculated it comes up with the same value?

So how do we turn this into code

- The pseudocode would look like

```
for(i = 0; i < num.length - 1; i++)
{
   find minIndex
   swap(num, i, minIndex);
}
```

start of the unsorted section of the array

How do we get minIndex?

- Remember the array idiom to find the smallest element in the array.

```
min = num[0];
for(i = 1; i < num.length; i++)
   if(num[i] < min)
      min = num[i];
```

- This only finds the smallest element, not the index of the smallest element.

How do we get minIndex?

- We could use two for loops, the first to find the min, the second to find the index

```
min = num[0];
for(i = 1; i < num.length; i++)
   if(num[i] < min)
      min = num[i];

for(i = 0; i < num.length; i++)
   if(num[i] == min)
      minIndex = i;
```

How do we get minIndex?

There is a better method; remember `minIndex` as we go along.

```
minIndex = 0;
for(i = 1; i < num.length; i++)
   if(num[i] < num[minIndex])
      minIndex = i;
```

`minIndex` initially points to the first element in the array. If we later find a smaller element, we update `minIndex`.

Bring it all together

Here's the pseudocode:

```
for(i = 0; i < num.length - 1; i++)
{
   calculate minIndex
   swap(num, i, minIndex);
}
```

Bringing it all together

```
for(i = 0; i < num.length - 1; i++)
{
   minIndex = i;
   for(j = i + 1; j < num.length; j++)
      if(num[j] < num[minIndex])
         minIndex = j;
   swap(num, i, minIndex);
}
```

Remember you have to calculate the minIndex for the unsorted array

Bringing it all together

The unsorted array starts at `i`

```
for(i = 0; i < num.length - 1; i++)
{
   minIndex = i;
   for(j = i + 1; j < num.length; j++)
      if(num[j] < num[minIndex])
         minIndex = j;
   swap(num, i, minIndex);
}
```

Remember you have to calculate the minIndex for the unsorted array

Summary

- How to sort an array:
  - Swap the smallest element with the first element.
  - Continue the process with the unsorted section of the array.
- This method of sorting an array is an example of an algorithm; a sequence of steps to perform a particular task.
- This particular algorithm is called selection sort; it *selects* the smallest element and swaps it with the first element.

To Think About

- When the array is sorted, is there a faster way to search for an element in the array? (The previous idiom searched every element in the array.)
- How do humans search a phone book for a number?
- Can you think of other methods to sort an array?