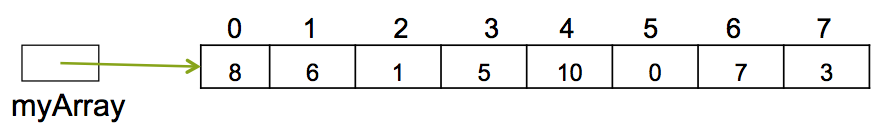
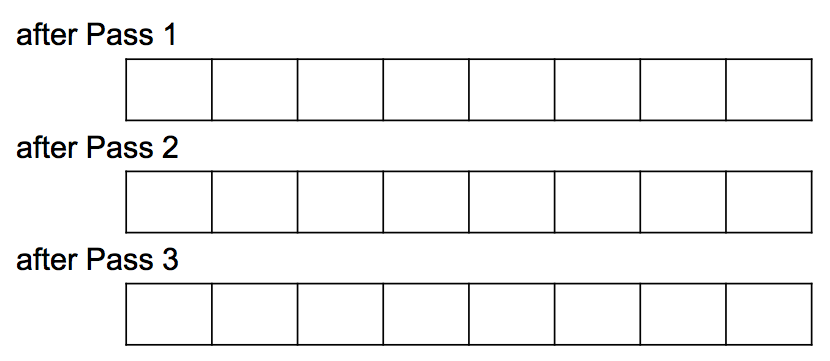
**Insertion Sort**

*Strategy:*





**Algorithm**

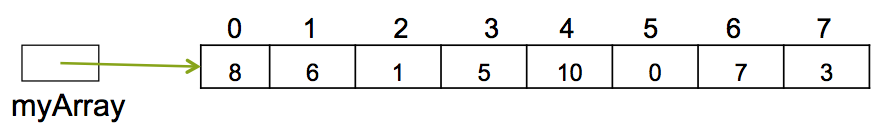
1. From index 1 to index size-1
   1. Store value at index in temp
   2. Set j equal to index
   3. While j is not zero and the value in index j-1 > temp
      1. Shift the value from index j-1 to index j
      2. J decreased by 1
   4. Store temp into index j

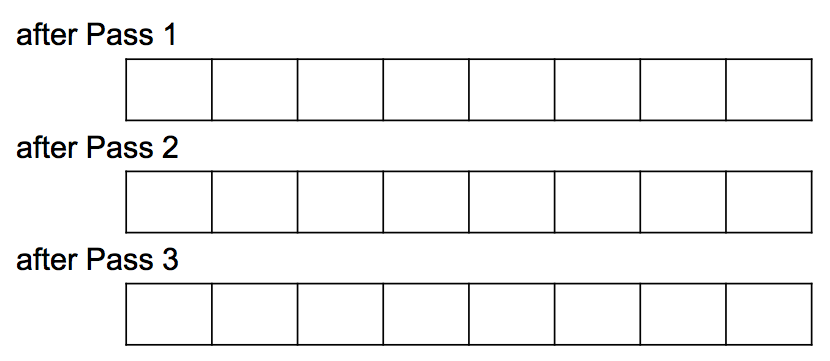
**Code**

public static void insertionSort( int [ ] array, int num)  
{  
 int temp;  
 for ( int i = 1; i < num; i++ )  
 {  
 temp = array[i];  
 int j = i;  
 while ( j != 0 && array[j - 1] > temp )  
 {  
 array[j] = array[j - 1]; //shift  
 j--;  
 }  
 array[j] = temp; //stuff  
 }  
}

**Selection Sort**

*Strategy:*





**Algorithm:**

1. From count 0 to length-1
   1. Find the maximum element in the unsorted array
   2. Swap the max element and the element in the rightmost unsorted spot

**Code:**

int [ ] myArray = { 8, 6, 1, 5, 10, 0 ,7, 3}; //my Array that needs to be sorted

int temp; // temporary location for swap  
int max; // index of maximum value in subarray

for ( int i = 0; i < myArray.length − 1; i++ ) // these are the passes. How many?  
{

// find index of largest value in subarray  
 max = indexOfLargestElement( myArray, myArray.length − i );

// swap elements at [max] and [array.length – i - 1]

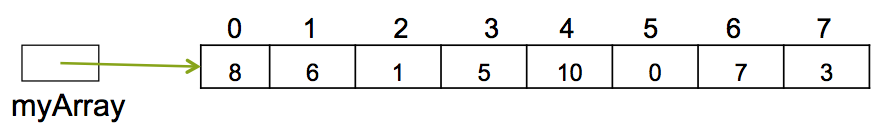
}

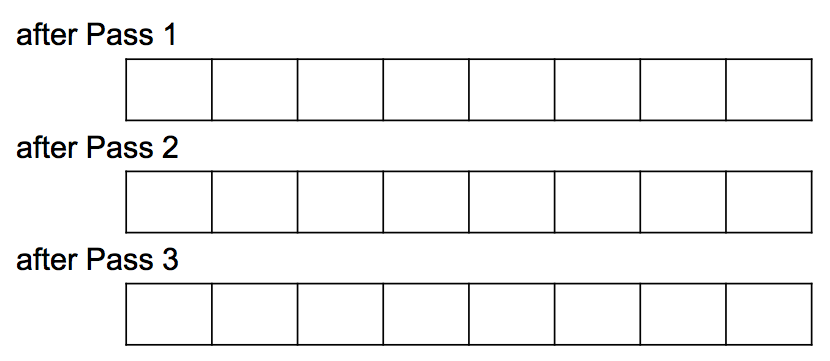
// finds and returns the largest element in myArray from index [0] to index [size – 1]

private **int indexOfLargestElement** ( int [] myArray, int size )  
{   
 int index = 0; // assume that the location of the largest element is  
 for( int i = 1; i < size; i++ )  
 if ( myArray[i] > myArray[index] )  
 index = i;  
 return index;  
 }

**Bubble Sort**

*Strategy:*





**Algorithm**

1. For each location from size-1 to 0, decreasing by 1 each time
   1. For each index from 0 to , increasing by 1 each time
      1. If value at current index is greater than
         1. Swap value at current index with value at index to

**Code**

int [ ] myArray = { 8, 6, 1, 5, 10, 0 ,7, 3};

int num = myArray.length; // # of elements stored in the array—at location 0..num - 1

for ( int loc = num-1; loc > 0; loc − − ) // loc indicates

for ( int i = 0; i < loc ; i ++) // compare the elements at i and

if ( myArray[i] **>**

{

int temp = // do a swap

}

What’s wrong with this shorter swap?

myArray[i] = myArray[i+1];

myArray[i+1] = myArray[i];

What changes do I need to make to sort in descending order?

**Smart Bubble Sort**

How might we want to make bubble sort “smarter” than it was?

//array has values stored at locations 0…num-1

public static void **bubbleSort**( int [ ] array, int num )   
 {  
 int temp = 0; // for ?  
 boolean arraySorted = false; // a "flag"

int loc = num - 1;   
 while ( loc > 0 && ! arraySorted) // note the "double" condition  
 { // array[loc] is being determined  
 arraySorted = true; // start a new iteration--maybe the array is sorted  
 for ( int j = 0; j < loc ; j++ ) // bubble down  
 if ( array [j] > array [j + 1] )  
 {  
 temp = array [j + 1]; // swap the adjacent elements  
 array [j + 1] = array [j];  
 array [j] = temp;  
 arraySorted = false; // and set arraySorted to false b/c

} //end of if

loc--;  
 } // end of while loop  
 } //end of method