AP Computer Science A

1. Log in
2. Unit 5
   - Array Review and Swapping Elements
   - Selection Sort
   - Insertion Sort
   - Bubble Sort
   - Merge Sort
   - Sequential Search
   - Binary Search

Oct 16-7:52 AM

Insertion Sort (sorting a list of n elements)...

1. Start at a[1] and compare to current "sorted list" ... a[0]
2. Place to make new "sorted list" ... a[0],a[1]
3. Move to a[2] and compare to list ... a[0],a[1]
4. Place to make new "sorted list" ... a[0],a[1],a[2]

   ...
5. Compare a[n-1] to list (a[0],a[1], ... , a[n-2])
6. Place where needed creating the final ordered list!

Example of Selection Sort

Original Array	8	2	5	7	3

1st Pass treats 8 like a sorted list, checks the next number a[1], and places it into the current sorted list (inserting where it belongs and shifting elements when needed).

Nov 13-12:26 PM

Insertion Sort (sorting a list of n elements)...

1. Start at a[1] and compare to current "sorted list" ... a[0]
2. Place to make new "sorted list" ... a[0],a[1]
3. Move to a[2] and compare to list ... a[0],a[1]
4. Place to make new "sorted list" ... a[0],a[1],a[2]

   ...
5. Compare a[n-1] to list (a[0],a[1], ... , a[n-2])
6. Place where needed creating the final ordered list!

Example of Selection Sort

Original Array	8	2	5	7	3
After 1st Pass	2	8	5	7	3

After the 1st Pass we have 2 elements in order but probably not in their final "resting place". We have a sorted list with 2 elements.

Nov 13-12:26 PM

Insertion Sort (sorting a list of n elements)...

1. Start at a[1] and compare to current "sorted list" ... a[0]
2. Place to make new "sorted list" ... a[0],a[1]
3. Move to a[2] and compare to list ... a[0],a[1]
4. Place to make new "sorted list" ... a[0],a[1],a[2]

   ...
5. Compare a[n-1] to list (a[0],a[1], ... , a[n-2])
6. Place where needed creating the final ordered list!

Example of Selection Sort

Original Array	8	2	5	7	3
After 1st Pass	2	8	5	7	3

The 2nd pass will check the next "unsorted item" and will insert it into the "sorted list" where it belongs, shifting elements that need to be shifted.

Nov 13-12:26 PM

Insertion Sort (sorting a list of n elements)...

1. Start at a[1] and compare to current "sorted list" ... a[0]
2. Place to make new "sorted list" ... a[0],a[1]
3. Move to a[2] and compare to list ... a[0],a[1]
4. Place to make new "sorted list" ... a[0],a[1],a[2]

   ...
5. Compare a[n-1] to list (a[0],a[1], ... , a[n-2])
6. Place where needed creating the final ordered list!

Example of Selection Sort

Original Array	8	2	5	7	3
After 1st Pass	2	8	5	7	3
After 2nd Pass	2	5	8	7	3

After the 2nd Pass we have 3 elements in order but probably not in their final "resting place". We have a sorted list with 3 elements.

Nov 13-12:26 PM

Insertion Sort (sorting a list of n elements)...

Example of Selection Sort

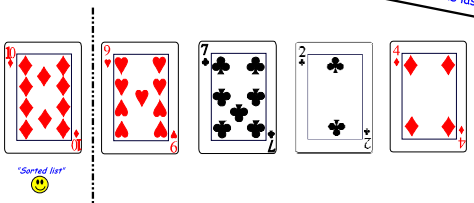
Original Array	8	2	5	7	3
After 1st Pass	2	8	5	7	3
After 2nd Pass	2	5	8	7	3
After 3rd Pass	2	5	7	8	3
After 4th Pass	2	3	5	7	8

It takes n-1 passes to sort the array using an insertion sort (we skip a[0] and check all the remaining elements)

Nov 13-12:26 PM

Insertion Sort Example with Cards ...

* Count the passes
* Notice the last pass?!?



Nov 13-12:26 PM

Things to note with Selection Sort ...

1. With each k^{th} pass we have $k+1$ elements guaranteed to be in ***an*** ordered list (2 passes = 3 elements)
2. If there are n elements then we only need $n-1$ passes
3. After the k^{th} pass on an array with n elements, we have $(n-1)-k$ elements to check.

Nov 13-12:26 PM

Now ... to code insertion sorting ...

1. Start with a[1] ...
   Compare to a[0] ...
   Make swap if needed ...

```
int[ ] theArray = {8,2,5,7,3};

if(theArray[1]<theArray[0])
{
      int temp = theArray[0];
      theArray[0]=theArray[1];
      theArray[1]=temp;
}
```

Nov 13-12:26 PM

Now ... to code selection sorting ...

```
public class insertionSortFull {
   public static void main(String[] args) {
      int[] theArray = {8, 2, 5, 7, 6, 1};

      for(int element: theArray)
         System.out.print(element+" ");

      System.out.println("");

      int j,i;
      for(i=1; i<theArray.length; i++)          //i is the element checked
      {
         int temp = theArray[i];                // store marked element
         j = i;                                 // start shifts at i
         while(j>0 && theArray[j-1] >= temp)    // until one is smaller,
         {
            theArray[j] = theArray[j-1];        // shift right
            --j;                                // look next left slot
         }
         theArray[j] = temp;                    // insert marked element
         for(int element: theArray)
            System.out.print(element+" ");
         System.out.println("");
      }
   }
}
```

Nov 13-12:26 PM

Things to do ...

1. Wrap Up Unit 6 WS 01-02
2. Work on Unit 6 WS03 Arrays and Insertion Sort

Oct 16-9:12 AM