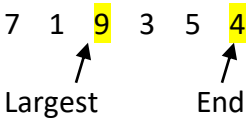
Sorting – Selection Sort

In using an insertion sort, values are constantly being moved. A sorting technique that reduces the amount of data movement is the **selection sort**. Again we perform a number of passes through the data and, after each pass place one more item into an ordered sequence.

The important difference in Selection Sort is that once an item has been placed in its position in the ordered sequence, **it is never moved again**.

We can begin a selection sort by scanning all the values, finding the largest one, and placing it at the end of the list, exchanging it with the item that was originally in this position.

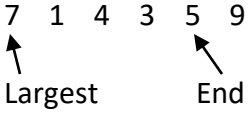
Example 1
To begin sorting the data



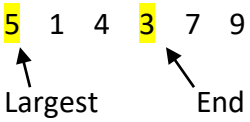
we find the largest element, 9, and swap it with 4, the element at the right end of the list. The result, after the first pass of a selection sort is

7 1 4 3 5 9 After first pass

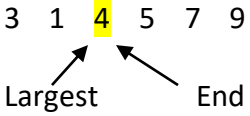
On the second pass, all the items except the last are examined to see which of these is the largest and this item is then placed at the right end of this sublist. This pattern continues on subsequent passes; on each one, the largest value among the unsorted items is placed at the end of the sublist.



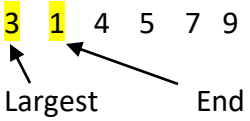
Still After first pass



After second pass



After third pass



After fourth pass

1 3 4 5 7 9

Sorting – Selection Sort

To code this algorithm for an array called list, we must successively find the largest item in sublists of sizes list.length, list.length-1, list.length-2, list.length-3, list.length-4 ..., 2. Since the upper bound of an array has index that is one less than the size of the array, the loop that controls the passes of the sort will have the form

```
for (int end = list.length - 1; end > 0; end--)
{
   // locate largest item and then
   // swap it with item at list[end]
 }
```

Look at these web sites for a visual demo of the selection sort.

`http://math.hws.edu/eck/js/sorting/xSortLab.html`

The method `selectionSort` uses a selection sort to arrange an array of double values in ascending order.

```
public static void selectionSort (double list[])
{
    int end;
    int largeLoc;
    int i;
    double temp;
    for (end = list.length - 1; end > 0; end--)
    {
        largeLoc = 0;
        for (i = 1; i <= end; i++)
        {
            if (list[i] > list[largeLoc])
            {
                largeLoc = i;
            }
        }
        temp = list[end];
        list[end] = list[largeLoc];
        list[largeLoc] = temp;

    }
}
```

Exercises

1. If a selection sort were to be used to sort the data shown below in alphabetical order, show the data after each pass of the sort.

   Robert Brian Victor David Scott

2. In the `selectSort` method, what would happen if the expression `list[i] > list[largeLoc]` were to be changed to `list[i] < list[largeLoc]`?

Sorting – Selection Sort

3. On each pass of our version of selection sort, the *largest* value among the remaining unsorted items was placed in its correct position. An alternate form of the algorithm uses each pass to place the *smallest* value among the remaining unsorted values in its correct position.

   (a) Given the set of data

   8 9 6 1 2 4

   show the data as they would appear after each pass of a selection sort that moves the *smallest* data.

4. Sometimes we are only interested in knowing the values that would occupy one end of the list if the list were sorted. For example, we may want to know the scores of only the top ten competitors in a contest. Overload our selection sort method so that it puts the *k* largest values in order in the last *k* positions of the array. The value of *k* should be a parameter of the method.

   ```
   public static void selectSort (double[] list, int k)
   ```

5. Write a program that will sort (using selection sort) 20 random numbers with values between 1 and 70. The program will **graphically** (with * *asterisks*) output to the screen the data as it is being sort after each pass of the sort. You might have to add a delay between each pass so that the sort is slow enough to see. Use `Thread.sleep(500)` to create a delay of 500 milliseconds.