

Beginning J A V A ...

Unit 6: Arrays



[Return to Unit Menu](#) | [Java Main Page](#) | [MathBits.com](#) | [Terms of Use](#) | [Java Resource CD](#)

Selection Sort



The **selection sort** is a combination of searching and sorting.

During each pass, the unsorted element with the smallest (or largest) value is moved to its proper position in the array.

The number of times the sort passes through the array is one less than the number of items in the array. In the selection sort, the inner loop finds the next smallest (or largest) value and the outer loop places that value into its proper location.

Let's look at our same table of elements using a selection sort for descending order. Remember, a "pass" is defined as one full trip through the array comparing and if necessary, swapping elements.


Array at beginning:	84	69	76	86	94	91
After Pass #1:	84	91	76	86	94	69
After Pass #2:	84	91	94	86	76	69
After Pass #3:	86	91	94	84	76	69
After Pass #4:	94	91	86	84	76	69
After Pass #5 (done):	94	91	86	84	76	69

While being an easy sort to program, the selection sort is one of the least efficient. The algorithm offers no way to end the sort early, even if it begins with an already sorted list.

// Selection Sort Method for Descending Order

```
public static void SelectionSort ( int [ ] num )
{
    int i, j, first, temp;
    for ( i = num.length - 1; i > 0; i -- )
    {
        first = 0; //initialize to subscript of first element
        for(j = 1; j <= i; j ++ ) //locate smallest element between positions 1 and i.
        {
            if( num[ j ] < num[ first ] )
                first = j;
        }
        temp = num[ first ]; //swap smallest found with element in position i.
        num[ first ] = num[ i ];
        num[ i ] = temp;
    }
}
```

```
}  
}
```



[Return to Unit Menu](#) | [Java Main Page](#) | [MathBits.com](#) | [Terms of Use](#) | [Java Resource CD](#)

Copyright 2000-2016 [MathBits.com](#)
All Rights Reserved