Insertion sort

Insertion Sort: Idea

1. We have two group of items:
   - sorted group, and
   - unsorted group
2. Initially, all items in the unsorted group and the sorted group is empty.
   - We assume that items in the unsorted group unsorted.
   - We have to keep items in the sorted group sorted.
3. Pick any item from, then insert the item at the right position in the sorted group to maintain sorted property.
4. Repeat the process until the unsorted group becomes empty.

- It still executes in O(N2) time, but it's about twice as fast as the bubble sort and somewhat faster than the selection sort in normal situations. It's also not too complex, although it's slightly more involved than the bubble and selection sorts. It's often used as the final stage of more sophisticated sorts, such as quicksort.

Insertion Sort Efficiency

- Best case running time is O(n) and occurs when the array is already sorted.
- The worst and average case running times are $O(n^2)$.
- The insertion sort is very efficient with the array is "almost sorted".
- The insertion sort is the best of the quadratic sorting algorithms.

```
package javaapplication1;

public class Main {

    public static void main(String[] args) {

int in, out;
int a []={5,7,3,8,6,4};
for(out=1; out<a.length; out++) // out is dividing line
{
int temp = a[out]; // remove marked item
in = out; // start shifts at out
while(in>0 && a[in-1] >= temp) // until one is smaller,
{
a[in] = a[in-1]; // shift item right,
--in; // go left one position
}
a[in] = temp; // insert marked item
} // end for
}
```

Selection Sort: Idea

1. We have two group of items:
   - sorted group, and
   - unsorted group
2. Initially, all items are in the unsorted group. The sorted group is empty.
   - We assume that items in the unsorted group unsorted.
   - We have to keep items in the sorted group sorted.
3. Select the "best" (eg. smallest) item from the unsorted group, then put the "best" item at the end of the sorted group.
4. Repeat the process until the unsorted group becomes empty.

```
public void selectionSort()
{
int out, in, min;
for(out=0; out<nElems-1; out++) // outer loop
{
min = out; // minimum
for(in=out+1; in<nElems; in++) {

if(a[in] < a[min] ) // if min greater,
min = in; } // we have a new min
swap(out, min); // swap them
} // end for(outer)
} // end selectionSort()
```

A Lower Bound

- Bubble Sort, Selection Sort, Insertion Sort all have worst case of $O(N^2)$.
- Turns out, for any algorithm that exchanges adjacent items, this is the best worst case: $\Omega(N^2)$
- In other words, this is a lower bound!