Charlson So

CS130

1.

**Code:**

**Used with methods in same folder.**

public class runtime{

public static void main(String[] args){

int arr[] = {5,4,3,2,1};

int ar[] = {13,12,11,10,9};

int b[]={10,8,5,2,1};

int c[]={12,11,3,2,0};

int d[] = {18,14,12,11,10};

//selection sort

long startTime = System.nanoTime();

SelectionSort k = new SelectionSort(arr);

k.sort();

long duration = System.nanoTime() - startTime;

System.out.println ("Selection Sort ran for " + duration + " nanoseconds" );

//insertion sort

startTime = System.nanoTime();

InsertionSort j = new InsertionSort(ar);

j.sort();

duration = System.nanoTime() - startTime;

System.out.println ("Insertion Sort ran for " + duration + " nanoseconds" );

//bubblesort

startTime = System.nanoTime();

BubbleSort i = new BubbleSort(b);

i.sort();

duration = System.nanoTime() - startTime;

System.out.println ("Bubble Sort ran for " + duration + " nanoseconds" );

//QuickSort

startTime = System.nanoTime();

QuickSort l = new QuickSort(c);

l.sort();

duration = System.nanoTime() - startTime;

System.out.println ("Quick Sort ran for " + duration + " nanoseconds" );

//mergesort

startTime = System.nanoTime();

MergeSort m = new MergeSort(d);

m.sort();

duration = System.nanoTime() - startTime;

System.out.println ("Merge Sort ran for " + duration + " nanoseconds" );

}

}

**Output:**

Selection Sort ran for 316459 nanoseconds

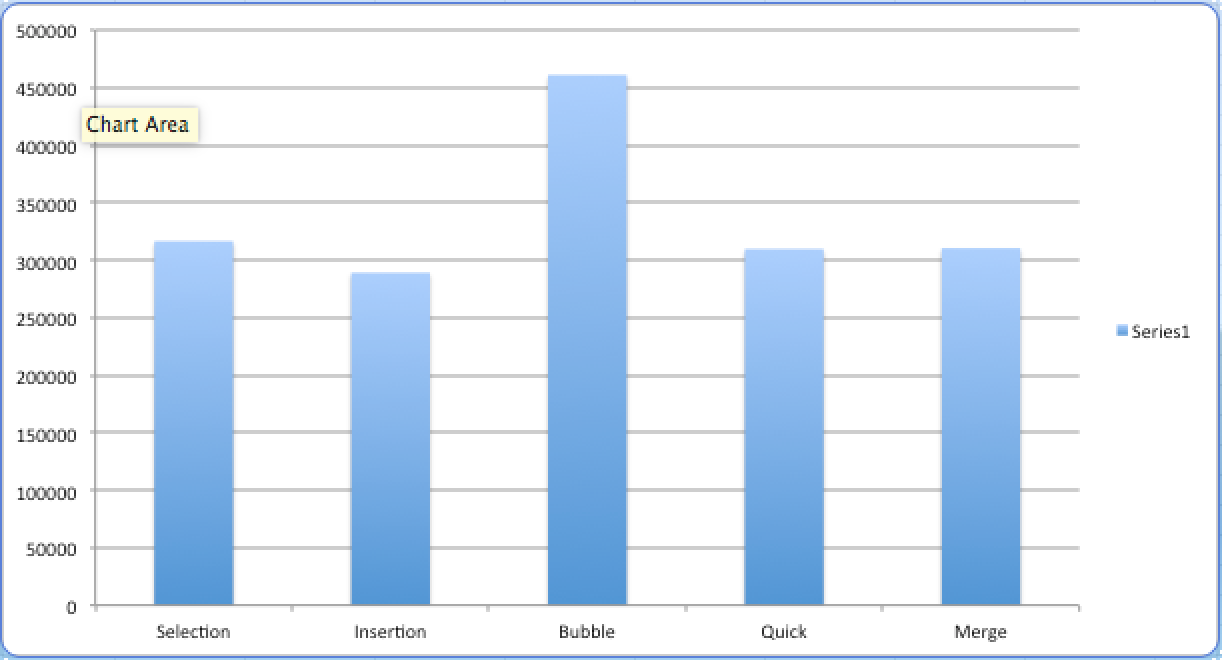
Insertion Sort ran for 289012 nanoseconds

Bubble Sort ran for 460915 nanoseconds

Quick Sort ran for 309651 nanoseconds

Merge Sort ran for 310339 nanoseconds

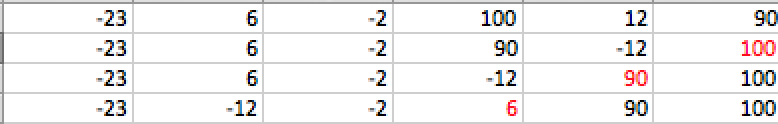
**Graph:**

****

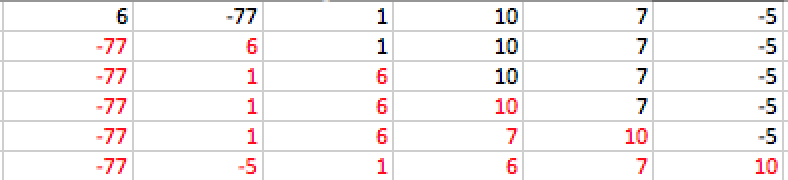
Insertion Sort seems to be the fastest for a descending array.

2. Selection Sort

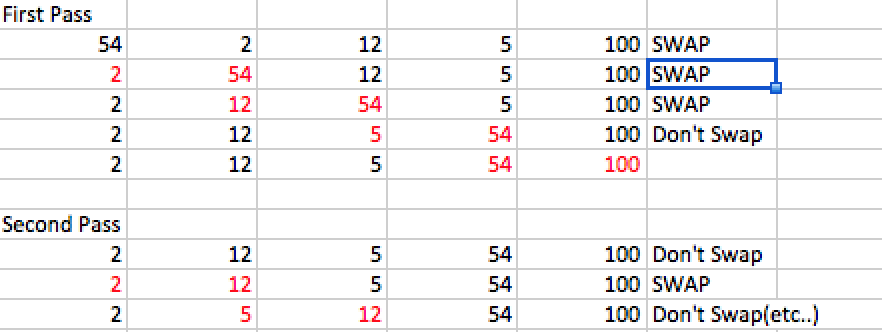
Takes the largest value and puts it at the end.



3. Insertion Sort

****

4. bubble Sort

****