**Mark Eatough**

**CSIS 2430 9:00 Class**

**Programming Project 2**

**Sorting Program**

**Assignment objective:**

Implement (3) sorting algorithms - quicksort, bubblesort and "roll-your-own".  You will generate 50 random numbers between 1 and 1000, place them in a data structure of your own choosing, implement all 3 sorting algorithms, and then you MUST time/compare them.  Bonus points if your "own" algorithm does better than the other two.

For you to get FULL points on this assignment, you will need to hand in ALL as follows:

* Cover page displaying your answers, the reason for these results written up in a paragraph.
* Code
* Screen CAP of your output.

Remember, NO ZIP files!!

**What Worked?:**

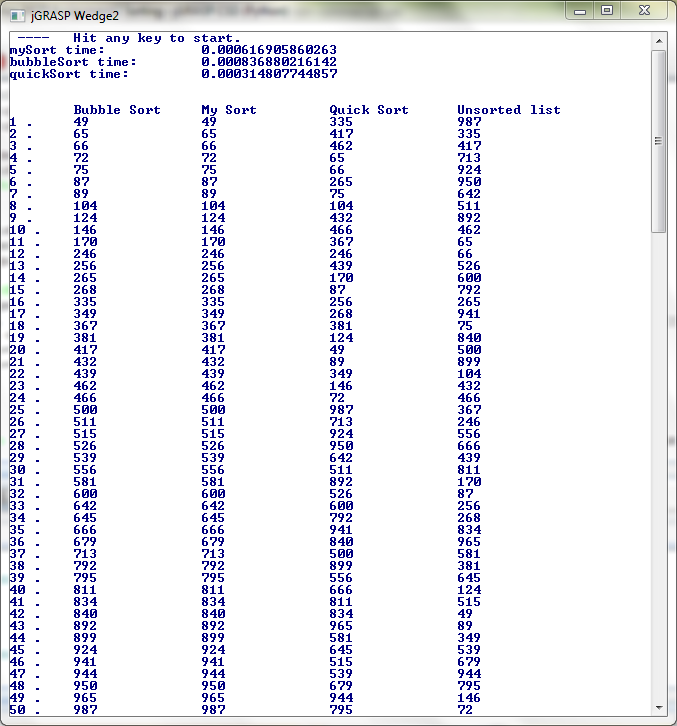
The bubble sort was pretty easy to implement. My sort started at both ends of the list, searching for the highest and lowest numbers respectively. After the highest and lowest numbers were found my markers both moved in one position from the ends of the list. Once they met at the middle the list was sorted.

**What did not work?:**

When it came to the quick sort I epically failed. My plan was to pick the pivot point similarly to how we did in class for the guess the number between one and a million exercise. I picked 500 at first, sorting that into two lists, one of elements more than 500 the other of elements that are less. Then use recursion to pass 250 and 750 as the pivot points for those two lists respectively and so on. I could only get the method to do the first split, so the quicksort method only sorts the numbers to below and above 500.

**Comments:**

I used the build in time.clock() method to time my sorting algorithms. The quicksort time was by far the fastest, but my sort was faster than the bubble sort. However my sort was more difficult to implement than the quicksort, and not as fast so it is probably not a practical sort. The bubble sort was by far the easiest to implement, but had the worst time complexity.



'''  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
 \* Discrete Structures  
 \* Sorting Program  
 \* Programmer: Mark Eatough  
 \* Course: CSIS 2430   
 \* Created September 1, 2013  
   
 \*This program takes a list of 50 unsorted integers that  
 \*were randomly selected between 1 and 1000. The program  
 \*sorts the list using a bubble sort, a quick sort, and a   
 \*sort that I came up with and them times how long the sorts   
 \*took  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 '''  
  
import time  
from random import randint  
#create a list to store my data   
randList = []  
pivot = 500  
  
#populate my list with 50 random integers between 1 and 1000  
for i in range(0, 50):  
 randList.extend([randint(1, 1000)])  
  
randList2 = randList[:]  
randList3 = randList[:]  
randList4 = randList[:]  
  
firstSum = 0  
secondSum = 0  
thirdSum = 0  
  
# quicksort method  
def quickSort(r, pivot):  
 quickStart = time.clock()  
 greater = []  
 less = []  
 for i in range(len(r)):  
 if(r[i] < pivot):  
 less.append(r[i])  
 if(r[i] >= pivot):  
 greater.append(r[i])  
   
 r = less[:] + greater[:]  
 print "quickSort time: \t", (time.clock() - quickStart)  
 return r  
   
#bubble sort method  
def bubbleSort(r):  
 start = time.clock()  
 k = 0  
 l = 0  
 for k in range(len(r)-1):  
 for l in range(len(r)-1):  
 if r[l] > r[l+1]:  
 temp = r[l]  
 r[l] = r[l+1]  
 r[l+1] = temp  
 print "bubbleSort time: \t", (time.clock() - start)  
   
#my sort method  
def mySort(r):  
 myStart = time.clock()  
 leftSide = 0  
 rightSide = len(r)-1  
   
 #nested while loop to sort list  
 while(leftSide < len(r)/2):  
 lCounter = leftSide  
 rCounter = rightSide  
 lowTemp = r[leftSide]  
 highTemp = r[rightSide]  
 lowIndex = leftSide  
 highIndex = rightSide  
 swap = 0;  
 #inner while loop  
 while(lCounter < len(r)/2):  
 if(lCounter < rCounter):  
 if(r[lCounter] < lowTemp):  
 lowTemp = r[lCounter]  
 lowIndex = lCounter  
 swap+=1  
 if(r[rCounter] < lowTemp):  
 lowTemp = r[rCounter]  
 if(r[lCounter] > highTemp):  
 highTemp = r[lCounter]  
 if(r[rCounter] > highTemp):  
 highTemp = r[rCounter]  
 lCounter+=1  
 rCounter-=1  
 if(leftSide >= rightSide-1):  
 if(r[leftSide] > r[rightSide]):  
 lowTemp = r[rightSide]  
 r[rightSide] = r[leftSide]  
 r[leftSide] = lowTemp  
 else:  
 lowIndex = r.index(lowTemp)   
 r[lowIndex] = r[leftSide]  
 r[leftSide] = lowTemp  
 highIndex = r.index(highTemp)  
 r[highIndex] = r[rightSide]  
 r[rightSide] = highTemp  
   
 leftSide+=1  
 rightSide-=1   
 print "mySort time: \t\t", (time.clock() - myStart)   
   
   
mySort(randList2)  
bubbleSort(randList)  
randList3 = quickSort(randList3, pivot)  
  
print "\n\n\tBubble Sort\tMy Sort\t\tQuick Sort\tUnsorted list"  
for j in range(len(randList2)):  
 print j+1, ".\t", randList[j], "\t\t", randList2[j], "\t\t", randList3[j], "\t\t", randList4[j]  
 secondSum += randList2[j]  
 firstSum += randList[j]   
 thirdSum += randList3[j]  
print "\n\n\n first sum = ", firstSum, "\n second sum = ", secondSum, "\n third sum = ", thirdSum