/\* Amir Yamini

\* Professor Schwartz

\* 09-26-2017

\* Project 1 - SortsTester

\*/

0

10

M:1,Q:19

Merge Average: 10655, Quick Average: 4726

The mean MergeSort runtime is: 320.7474603799719

The mean QuickSort runtime is: 142.2667759507975

1

100

M:4,Q:16

Merge Average: 75853, Quick Average: 36148

The mean MergeSort runtime is: 114.17014130549983

The mean QuickSort runtime is: 54.408161416307955

2

1000

M:0,Q:20

Merge Average: 481700, Quick Average: 135559

The mean MergeSort runtime is: 48.335382970446574

The mean QuickSort runtime is: 13.60244172740454

3

10000

M:0,Q:20

Merge Average: 3434938, Quick Average: 1296819

The mean MergeSort runtime is: 25.850484281151104

The mean QuickSort runtime is: 9.75953544867421

4

100000

M:0,Q:20

Merge Average: 18241042, Quick Average: 11955767

The mean MergeSort runtime is: 10.982201588332998

The mean QuickSort runtime is: 7.198088976339139

5

1000000

M:0,Q:20

Merge Average: 196226812, Quick Average: 131975376

The mean MergeSort runtime is: 9.845026060919476

The mean QuickSort runtime is: 6.621424477505382

6

2000000

M:0,Q:20

Merge Average: 413205879, Quick Average: 278906429

The mean MergeSort runtime is: 9.870399287838497

The mean QuickSort runtime is: 6.662339424689499

////////////////////////////////////////////////////////////////////////////////////////

//BOTH

long start;

long end;

long timeM;

long timeQ;

int trialCount = 0;

int quickCount = 0;

int mergeCount = 0;

long quickRun = 0;

long mergeRun = 0;

long qAverage = 0;

long mAverage = 0;

int arrayCount = 0;

int arraySize[] = {10,100,1000,10000,100000,1000000,2000000};

while(arrayCount < 7) {

int rand1[] = new int[arraySize[arrayCount]];

int rand2[] = new int[arraySize[arrayCount]];

System.out.println(arrayCount);

trialCount = 0;

mergeRun = 0;

quickRun = 0;

mergeCount = 0;

quickCount = 0;

while(trialCount < 20) {

//MERGESORT

for(int i = 0; i < arraySize[arrayCount]; i++) {

rand1[i] = (int) (Math.random() \* 1000000 + 1);

rand2[i] = rand1[i];

}

start = System.nanoTime();

soides.mergeSort(rand1, 0, arraySize[arrayCount]-1);

end = System.nanoTime();

timeM = end - start;

//System.out.println(timeM);

//QUICKSORT

start = System.nanoTime();

soides.quickSort(rand2, 0, arraySize[arrayCount]-1);

end = System.nanoTime();

timeQ = end - start;

//System.out.println(timeQ);

mergeRun += timeM;

quickRun += timeQ;

if(timeM > timeQ) {

//System.out.println("QuickSort Won!");

quickCount++;

//System.out.println(quickCount);

}else {

//System.out.println("MergeSort Won!");

mergeCount++;

//System.out.println(mergeCount);

}

trialCount++;

}// trials loop

mAverage = mergeRun/20;

qAverage = quickRun/20;

System.out.println(arraySize[arrayCount]);

System.out.println("M:" + mergeCount + "," + "Q:" + quickCount);

System.out.println("Merge Average: " + mAverage + ", " + "Quick Average: " + qAverage);

System.out.println("The mean MergeSort runtime is: " + mAverage / (arraySize[arrayCount] \* (Math.log(arraySize[arrayCount]) / Math.log(2))));

System.out.println("The mean QuickSort runtime is: " + qAverage / (arraySize[arrayCount] \* (Math.log(arraySize[arrayCount]) / Math.log(2))));

arrayCount++;

}