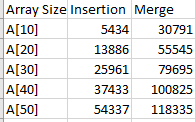
**Graph, Outputs and Code**

**Graph for array sizes 10, 20, 30, 40, 50:**



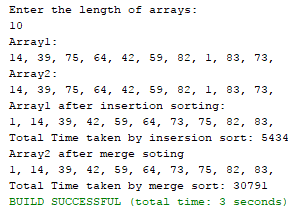
**Graph for array sizes 10, 20, 30:**

Time in nano seconds on y axis

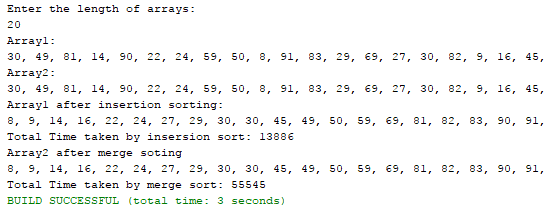
Size of arrays on x axis

**Some Outputs:**

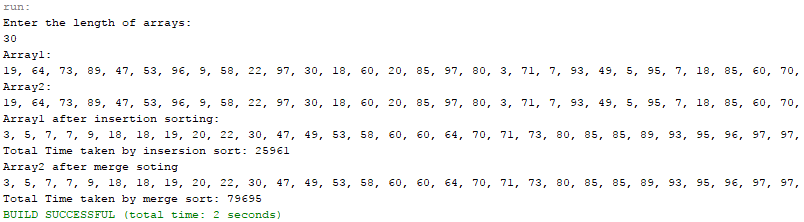
**For Array size 10**



**For array size 20**



**For array size 30**



**Code:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sorting;

import java.util.Random;

import java.util.Scanner;

/\*\*

\*

\* @author Speed Computers

\*/

public class sort {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

Scanner in= new Scanner(System.in);

System.out.println("Enter the length of arrays: ");

int s = in.nextInt();

int [] A1 = new int [s];

int [] A2 = new int [s];

for(int i=0; i<s; i++){

int a = gen();

A1[i] = a;

A2[i] = a;

}

System.out.println("Array1:");

for(int i=0; i<A1.length; i++){

System.out.print(A1[i]+", ");

}

System.out.println();

System.out.println("Array2:");

for(int i=0; i<A2.length; i++){

System.out.print(A2[i]+", ");

}

//insertion sort of A1

final long startTime = System.nanoTime();

int j;

for(int i=1; i<A1.length; i++){

j=i;

while(j>0 && A1[j-1]>A1[j]){

int c= A1[j-1];

A1[j-1] = A1[j];

A1[j] = c;

j = j-1;

}

}

final long duration = System.nanoTime() - startTime;

System.out.println();

System.out.println("Array1 after insertion sorting:");

for(int i=0; i<A1.length; i++){

System.out.print(A1[i]+", ");

}

System.out.println();

System.out.println("Total Time taken by insersion sort: "+duration);

//merge sort of A2

final long stime = System.nanoTime();

A2=mergesort(A2);

final long durationn = System.nanoTime() - stime;

System.out.println("Array2 after merge soting");

for(int i=0; i<A2.length; i++){

System.out.print(A2[i]+", ");

}

System.out.println();

System.out.println("Total Time taken by merge sort: "+durationn);

}

public static int gen() {

Random r = new Random();

return 0 + r.nextInt(100);

}

public static int []mergesort(int []A2){

if (A2.length<=1){

return A2;

}

int mid = A2.length/2;

int []left = new int[mid];

int []right;

if (A2.length%2==0){

right = new int[mid];

}

else

right = new int[mid+1];

int []result = new int [A2.length];

for(int i=0; i<mid; i++){

left[i] = A2[i];

}

int x=0;

for(int i=mid; i<A2.length; i++){

if(x<right.length){

right[x] = A2[i];

x++;

}

}

left = mergesort(left);

right = mergesort(right);

return result = merge(left, right);

}

public static int [] merge(int []left, int []right){

int combine = left.length + right.length;

int []result = new int[combine];

int L=0;

int R=0;

int Res=0;

while (L<left.length || R<right.length){

if(L<left.length && R<right.length){

if(left[L]<= right[R]){

result [Res] = left[L];

L++;

Res++;

}

else{

result[Res] = right[R];

R++;

Res++;

}

}

else if(L<left.length){

result[Res] = left[L];

L++;

Res++;

}

else if(R<right.length){

result[Res] = right[R];

R++;

Res++;

}

}

return result;

}

}