0697

import java.util.Arrays;

import java.util.Random;

public class QuickSort {

public static void quickSort(int[] A, int L, int H ){

if(A == null || A.length == 0){

return;

}

if (L >= H ){

return;

}

int M = L + (H - L)/ 2 ;

int pivot = A[M];

int i = L;

int j = H;

while(i <= j){

while(A[i] < pivot){

i++;

}

while(A[j] > pivot){

j--;

}

if(i <= j){

int temp = A[i];

A[i]= A[j];

A[j]= temp;

i++;

j--;

}

}

if(L < j){

quickSort(A, L, j);

}

if (H > i){

quickSort(A,L, i);

}

}

public static void main(String[] args){

int[]B = null ;

int A[]= new int[1000];

int k;

QuickSort q= new QuickSort();

QuickSort1 q1= new QuickSort1();

QuickSort2 q2= new QuickSort2();

Random rand = new Random();

for(int i= 0; i <1000; i++){

A[i]= rand.nextInt();

}

int low = 0;

int high= A.length -1;

for(k= 1; k < 1000; k= k+ 100){

//System.out.println("quickSort");

long time= System.currentTimeMillis();

for(int i= 1; i < 100; i++){

long t= System.currentTimeMillis();

for(int j= 1; j< 1000; j++){

B = Arrays.copyOf(A, A.length);

}

long overhead= System.currentTimeMillis()- t;

t= System.currentTimeMillis();

for(int j= 1; j< 1000; j++){

B = Arrays.copyOf(A, A.length);

q.quickSort(B, low, high);

}

t= System.currentTimeMillis()- t- overhead;

System.out.println(t);

}

//System.out.println("quickSort1");

for(int i= 1; i < 100; i++){

long t= System.currentTimeMillis();

for(int j= 1; j< 1000; j++){

B = Arrays.copyOf(A, A.length);

}

long overhead= System.currentTimeMillis()- t;

t= System.currentTimeMillis();

for(int j= 1; j< 1000; j++){

B = Arrays.copyOf(A, A.length);

q1.quickSort(B, low, high, k);

}

t= System.currentTimeMillis()- t- overhead;

System.out.println(t);

}

//System.out.println("quickSort2");

for(int i= 1; i < 100; i++){

long t= System.currentTimeMillis();

for(int j= 1; j< 1000; j++){

B = Arrays.copyOf(A, A.length);

}

long overhead= System.currentTimeMillis()- t;

t= System.currentTimeMillis();

for(int j= 1; j< 1000; j++){

B = Arrays.copyOf(A, A.length);

q2.quickSort(B, low, high, k);

q2.insertionSort(B);

}

t= System.currentTimeMillis()- t- overhead;

//System.out.println(t);

}

System.out.println(time);

}

}

}

public class QuickSort1 {

public static void quickSort(int[] A, int L, int H, int k ){

if(A == null || A.length == 0){

return;

}else if (L >= H ){

return;

} else if(A.length <= k){

insertionSort(A);

}

int M = L + (H - L)/ 2 ;

int pivot = A[M];

int i = L;

int j = H;

while(i <= j){

while(A[i] < pivot){

i++;

}

while(A[j] > pivot){

j--;

}

if(i <= j){

int temp = A[i];

A[i]= A[j];

A[j]= temp;

i++;

j--;

}

}

if(L < j){

quickSort(A, L, j, k);

}

if (H > i){

quickSort(A,L, i, k);

}

}

/\*public static void main(String[] args){

int[]A = null ;

Random rand = new Random();

for(int i= 0; i <1000; i++){

A[i]= rand.nextInt();

}

int low = 0;

int high= A.length -1;

quickSort(A, low, high);

} \*/

public static void insertionSort(int[] num){

int key;

int i;

for(int j= 1; j< num.length; j++){

key = num[j];

for(i= j-1; (i >=0)&& (num[i]< key); i--){

num[i+1]= num[i];

}

num[i+1]= key;

}

}

}

public class QuickSort2 {

public static void quickSort(int[] A, int L, int H, int k ){

if(A == null || A.length == 0){

return;

}else if (L >= H ){

return;

} else if(A.length <= k){

return;

}

int M = L + (H - L)/ 2 ;

int pivot = A[M];

int i = L;

int j = H;

while(i <= j){

while(A[i] < pivot){

i++;

}

while(A[j] > pivot){

j--;

}

if(i <= j){

int temp = A[i];

A[i]= A[j];

A[j]= temp;

i++;

j--;

}

}

if(L < j){

quickSort(A, L, j, k);

}

if (H > i){

quickSort(A,L, i, k);

}

}

/\*public static void main(String[] args){

int[]A = null ;

Random rand = new Random();

for(int i= 0; i <1000; i++){

A[i]= rand.nextInt();

}

;

int low = 0;

int high= A.length -1;

quickSort(A, low, high, k);

insertionSort(A);

} \*/

public static void insertionSort(int[] num){

int key;

int i;

for(int j= 1; j< num.length; j++){

key = num[j];

for(i= j-1; (i >=0)&& (num[i]< key); i--){

num[i+1]= num[i];

}

num[i+1]= key;

}

}

}