CODE for recursive Binary Search:

// package Codes.Binary;

import java.time.\*;

import java.util.\*;

import java.lang.Math;

class BinarySearchRecursive {

static void selectionSort(int arr[], int n)

{

int i, j, min\_idx;

for (i = 0; i < n - 1; i++)

{

// Find the minimum element in unsorted array

min\_idx = i;

for (j = i + 1; j < n; j++)

if (arr[j] < arr[min\_idx])

min\_idx = j;

// Swap the found minimum element with the first element

int temp = arr[min\_idx];

arr[min\_idx]= arr[i];

arr[i] = temp;

}

int k;

for (k = 0; k < n; k++) {

System.out.print(arr[k]+ " ");

}

System.out.println();

}

int binarySearch(int arr[], int l, int r, int x){

if(r>=l){

int mid= l+(r-l)/2;

if(arr[mid]==x)

{

return mid;

}

if(arr[mid]>x){

return binarySearch(arr, l, mid-1, x);

}

return binarySearch(arr, mid+1, r, x);

}

return -1;

}

public static void main(String args[])

{

BinarySearchRecursive object = new BinarySearchRecursive();

Scanner sc = new Scanner(System.in);

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

int N=sc.nextInt();

int arr[];

arr=new int[N];

for(int i=0;i<N;i++)

{

double x=Math.random()\*300;

int num= (int)x;

arr[i]= num;

}

System.out.println("Enter element to search: ");

int x=sc.nextInt();

long start = System.nanoTime();

selectionSort(arr, N);

int result = object.binarySearch(arr,0,N-1,x);

if(result ==-1){

System.out.println("Element not present :)");

}

else{

System.out.println("Element is at "+result+ " position");

}

long end = System.nanoTime();

long execution = end - start;

System.out.println("Execution time: " + execution + " nanoseconds");

}

}

CODE for iterative Binary Search:

// package Codes.Binary;

import java.time.\*;

import java.util.\*;

import java.lang.Math;

class Binary {

// Returns index of x if it is present in arr[],

// else return -1

int binarySearch(int arr[], int x)

{

int l = 0, r = arr.length - 1;

while (l <= r) {

int m = l + (r - l) / 2;

if (arr[m] == x)

return m;

if (arr[m] < x)

l = m + 1;

else

r = m - 1;

}

for (int k = 0; k < arr.length; k++) {

System.out.print(arr[k]+ " ");

}

System.out.println();

// if we reach here, then element was

// not present

return -1;

}

// Driver method to test above

public static void main(String args[])

{

Binary ob = new Binary();

Scanner sc = new Scanner(System.in);

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

int N=sc.nextInt();

int arr[];

arr=new int[N];

for(int i=0;i<N;i++)

{

double x=Math.random()\*300;

int num= (int)x;

arr[i]= num;

}

// int arr[] = { 2, 3, 4, 10, 40 };

// int n = arr.length;

System.out.println("Enter element to be searched: ");

int x = sc.nextInt();

long start = System.nanoTime();

int result = ob.binarySearch(arr, x);

if (result == -1){

System.out.println("Element not present");

}

else{

System.out.println("Element found at "+ "index " + result);

}

long end = System.nanoTime();

long execution = end - start;

System.out.println("Execution time: " + execution + "nanoseconds");

}

}

**CODE FOR ITERATIVE MIN-MAX:**

// package Minmax;

import java.util.\*;

// import java.time.\*;

import java.math.\*;

public class MinMaxIterative {

static int[] sort(int arr[], int n)

{

int i, key, j;

for(i=1; i<n; i++)

{

key=arr[i];

j=i-1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

for (int k = 0; k < n; k++) {

System.out.print(arr[k] + " ");

}

System.out.println("\n");

return arr;

}

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

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

int N=sc.nextInt();

int arr[];

arr=new int[N];

for(int i=0; i<N; i++)

{

double x =Math.random()\*300;

int num=(int)x;

arr[i]=num;

}

int array[]=sort(arr,N);

System.out.println("Min element is: "+array[0]+"\n");

System.out.println("Max elemet is: "+array[N-1]);

}

}

**CODE for recursive min-max:**

// package Minmax;

import java.util.\*;

import java.time.\*;

public class MinMaxRecursive {

public static int findMinRec(int arr[], int n)

{

if(n == 1)

return arr[0];

return Math.min(arr[n-1], findMinRec(arr, n-1));

}

public static int findMaxRec(int arr[], int n)

{

// if size = 0 means whole array

// has been traversed

if(n == 1)

return arr[0];

return Math.max(arr[n-1], findMaxRec(arr, n-1));

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

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

int N=sc.nextInt();

int arr[];

arr=new int[N];

for(int i=0; i<N; i++)

{

double x =Math.random()\*300;

int num=(int)x;

arr[i]=num;

}

for (int k = 0; k < N; k++) {

System.out.print(arr[k] + " ");

}

System.out.println("\n");

long start = System.nanoTime();

int min=findMinRec(arr, N);

int max=findMaxRec(arr, N);

System.out.println();

System.out.println("The min is: "+min+" and the max is: "+max);

long end = System.nanoTime();

long execution = end - start;

System.out.println();

System.out.println("Execution time: " + execution + " nanoseconds");

}

}