Code for Min Max Algorithm using Recursive Approach:

import java.util.\*;

class MaxMin

{

    public int max, min;

    public MaxMin(int max, int min)

    {

        this.max = max;

        this.min = min;

    }

}

public class EXPT\_02\_Max\_Min\_Recursive

{

    static void Max\_Min\_Recursive(int arr[], int i, int j, MaxMin mm)

    {

        if(i == j)

        {

            if(mm.max < arr[i])

            {

                mm.max = arr[i];

            }

            if(mm.min > arr[j])

            {

                mm.min = arr[j];

            }

            return;

        }

        if(j - i == 1)

        {

            if(arr[i] < arr[j])

            {

                if(mm.min > arr[i])

                {

                    mm.min = arr[i];

                }

                if(mm.max < arr[j])

                {

                    mm.max = arr[j];

                }

            }

            else

            {

                if(mm.min > arr[j])

                {

                    mm.min = arr[j];

                }

                if(mm.max < arr[i])

                {

                    mm.max = arr[i];

                }

            }

            return;

        }

        int mid = (i+j)/2;

        Max\_Min\_Recursive(arr, i, mid, mm);

        Max\_Min\_Recursive(arr, mid+1, j, mm);

    }

    public static void main(String args[])

    {

        Scanner in = new Scanner(System.in);

        int n = 1000;

        int arr[] = new int[n];

        Random randomnumber = new Random();

        for(int i = 0; i < n; i++)

        {

            arr[i] = randomnumber.nextInt(1000);

            System.out.print("\t"+arr[i]);

        }

        System.out.println("");

        MaxMin mm = new MaxMin(0, 1000);

        long start = System.nanoTime();

        Max\_Min\_Recursive(arr, 0, n-1, mm);

        long end = System.nanoTime();

        long time = end - start;

        System.out.print("The number of elements in the array is 1000.");

        System.out.print("\nThe minimum is "+mm.min);

        System.out.print("\nThe maximum is "+mm.max);

        System.out.print("\nThe time taken for Min Max Algorithm using Recursive approach is "+time);

    }

}

Code for Min Max Algorithm using Iterative Approach:

import java.util.\*;

public class EXPT\_02\_Max\_Min\_Iterative

{

    public static void main(String args[])

    {

        Scanner in = new Scanner(System.in);

        int n = 1000;

        int arr[] = new int[n];

        Random randomnumber = new Random();

        System.out.print("The randomly-generated array is\n");

        for(int i = 0; i < n; i++)

        {

            arr[i] = randomnumber.nextInt(1000);

            System.out.print("\t"+arr[i]);

        }

        int max = arr[0];

        int min = arr[0];

        long start = System.nanoTime();

        for(int i = 0; i < n; i++)

        {

            if(max < arr[i])

            {

                max = arr[i];

            }

            if(min > arr[i])

            {

                min = arr[i];

            }

        }

        long end = System.nanoTime();

        long time = end - start;

        System.out.print("\nThe number of elements in the array is 1000.");

        System.out.print("\nThe minimum is "+min);

        System.out.print("\nThe maximum is "+max);

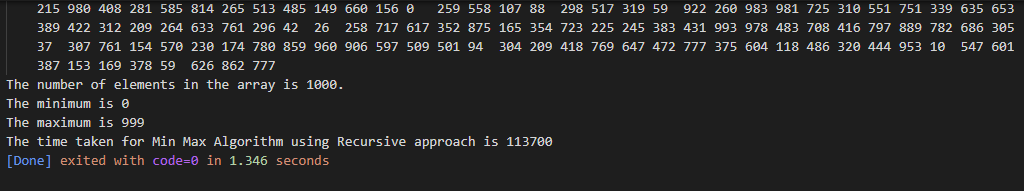
        System.out.print("\nThe time taken for Min Max Algorithm using Iterative approach is "+time);

    }

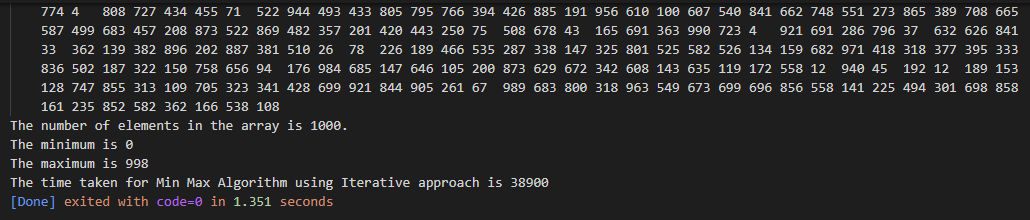
}

Case 1: When the number of elements is 1000

Recursive approach: 113,700 nanoseconds

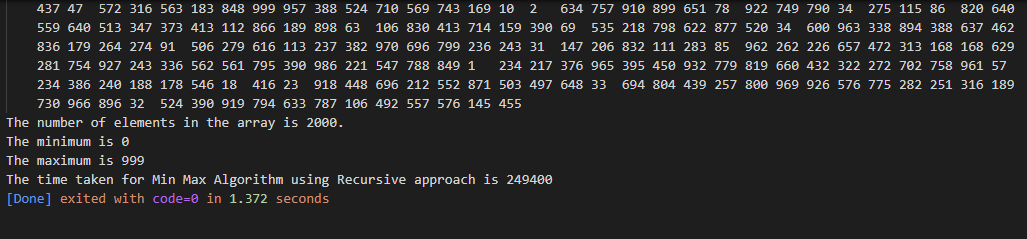


Iterative approach: 38,900 nanoseconds

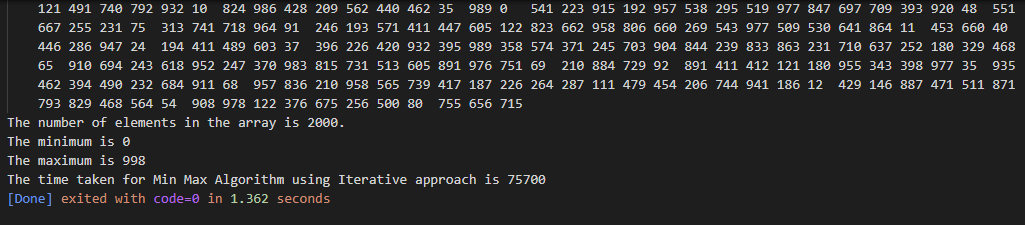


Case 1: When the number of elements is 2000

Recursive approach: 249,400 nanoseconds

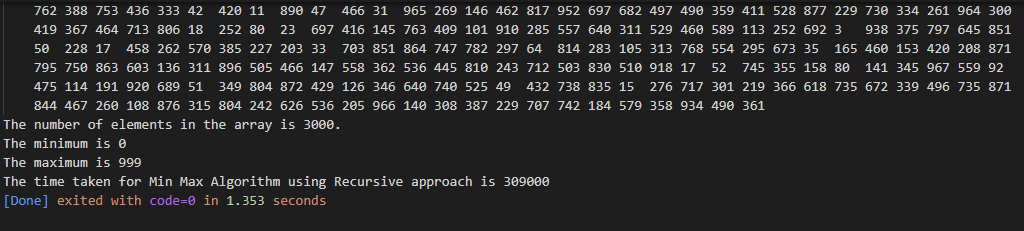


Iterative approach: 75,700 nanoseconds

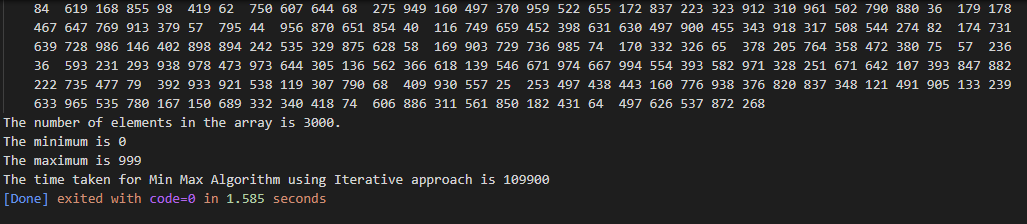


Case 3: When the number of elements is 3000

Recursive approach: 309,000 nanoseconds

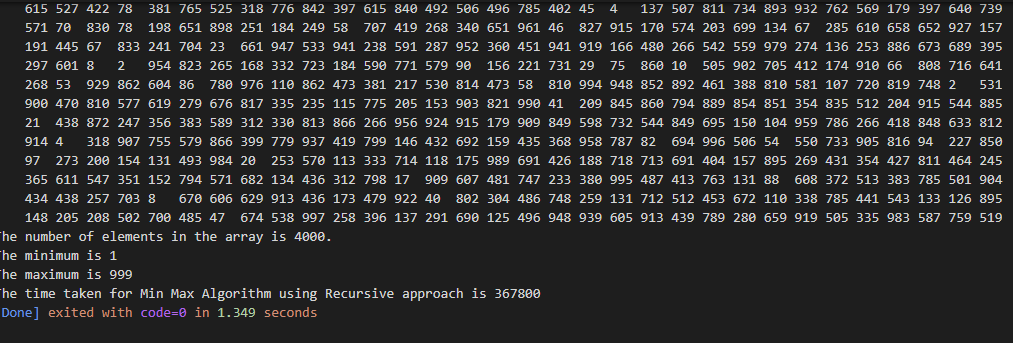


Iterative approach: 109,900 nanoseconds

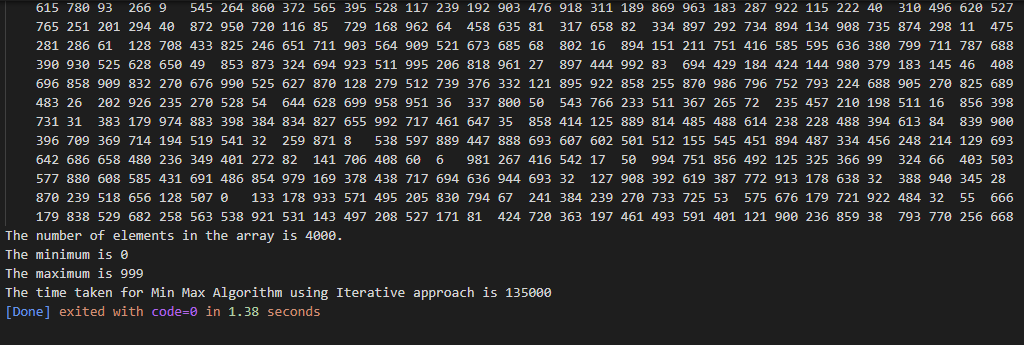


Case 4: When the number of elements is 4000

Recursive approach: 367,800 nanoseconds

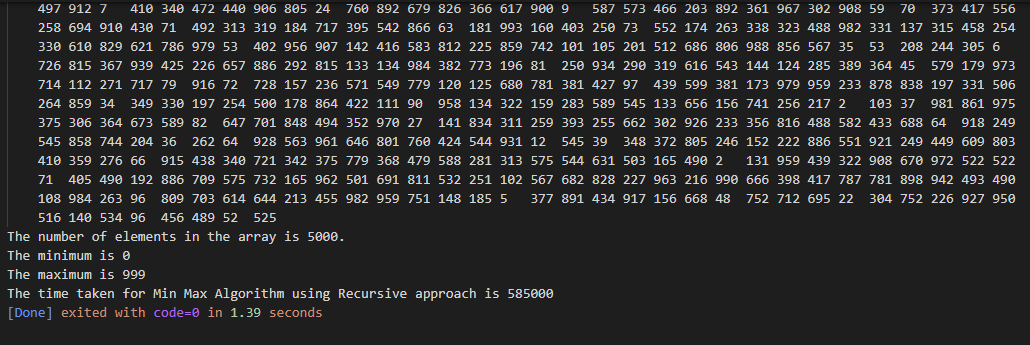


Iterative approach: 135,000 nanoseconds



Case 5: When the number of elements is 5000

Recursive approach: 585,000 nanoseconds



Iterative approach: 166,100 nanoseconds

