Code for Recursive approach:

import java.util.\*;

public class EXPT\_02\_Recursive {

    static int num;

    int recursive\_binary\_search(int a[], int s, int l, int h) {

        num++;

        int mid;

        mid = (l + h) / 2;

        if (s == a[mid]) {

            return mid;

        } else if (s < a[mid]) {

            return recursive\_binary\_search(a, s, l, mid - 1);

        } else if (s > a[mid]) {

            return recursive\_binary\_search(a, s, mid + 1, h);

        }

        return -1;

    }

    public static void main(String args[]) {

        Scanner in = new Scanner(System.in);

        int n, i, j, key, search, pos;

        Random randomnumber = new Random();

        System.out.print("Enter the number of elements: ");

        n = in.nextInt();

        System.out.println("");

        int arr[] = new int[n];

        System.out.println("The random-generated array is: ");

        for (i = 0; i < n; i++) {

            // System.out.print("For index "+(i+1)+", enter value: ");

            // arr[i] = in.nextInt();

            arr[i] = randomnumber.nextInt(10000);

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

        }

        System.out.println();

        System.out.println("Sorting using Insertion sort, ");

        for (i = 0; i < n; i++) {

            key = arr[i];

            j = i - 1;

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

                arr[j + 1] = arr[j];

                j = j - 1;

            }

            arr[j + 1] = key;

        }

        System.out.println("The sorted array is:\n ");

        for (i = 0; i < n; i++) {

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

        }

        System.out.println("");

        System.out.print("Enter the number which you want to search: ");

        search = in.nextInt();

        int low, high;

        low = 0;

        high = n;

        EXPT\_02\_Recursive obj = new EXPT\_02\_Recursive();

        long start = System.nanoTime();

        pos = obj.recursive\_binary\_search(arr, search, low, high);

        if (pos == -1)

            System.out.println("The element was not found.");

        else {

            System.out.println("The element " + search + " has been found after " + num + " function calls.");

            System.out.println("Position is " + pos);

        }

        long end = System.nanoTime();

        long time = end - start;

        System.out.println("The time taken for the binary search using RECURSIVE METHOD is " + time + " nanoseconds.");

    }

}

Code for Iterative approach:

import java.util.\*;

public class EXPT\_02\_Iterative {

    void iterative\_binary\_search(int a[], int s) {

        // System.out.println("The array involved in search operation is, ");

        // for(int i = 0; i < a.length; i++)

        // {

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

        // }

        // System.out.println("\nAnd, the element to search is: "+s);

        long start = System.nanoTime();

        int low, mid, high, it;

        it = 0;

        low = 0;

        high = a.length;

        mid = (low + high) / 2;

        while (low <= high) {

            if (s == a[mid]) {

                System.out.println("The element " + s + " has been found after " + it + " iterations.");

                break;

            } else if (s < a[mid]) {

                high = mid - 1;

            } else if (s > a[mid]) {

                low = mid + 1;

            }

            mid = (low + high) / 2;

            it++;

        }

        long end = System.nanoTime();

        long time = end - start;

        System.out.println("The time taken for the binary search using ITERATIVE METHOD is " + time + " nanoseconds.");

    }

    public static void main(String args[]) {

        Scanner in = new Scanner(System.in);

        Random randomnumber = new Random();

        int n, i, j, key, search;

        System.out.print("Enter the number of elements: ");

        n = in.nextInt();

        System.out.println("");

        int arr[] = new int[n];

        System.out.println("The random-generated array is: ");

        for (i = 0; i < n; i++) {

            // System.out.print("For index "+(i+1)+", enter value: ");

            // arr[i] = in.nextInt();

            arr[i] = randomnumber.nextInt(10000);

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

        }

        System.out.println();

        System.out.println("Sorting using Insertion sort, ");

        for (i = 0; i < n; i++) {

            key = arr[i];

            j = i - 1;

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

                arr[j + 1] = arr[j];

                j = j - 1;

            }

            arr[j + 1] = key;

        }

        System.out.println("The sorted array is:\n ");

        for (i = 0; i < n; i++) {

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

        }

        System.out.println("");

        System.out.print("Enter the number which you want to search: ");

        search = in.nextInt();

        EXPT\_02\_Iterative obj = new EXPT\_02\_Iterative();

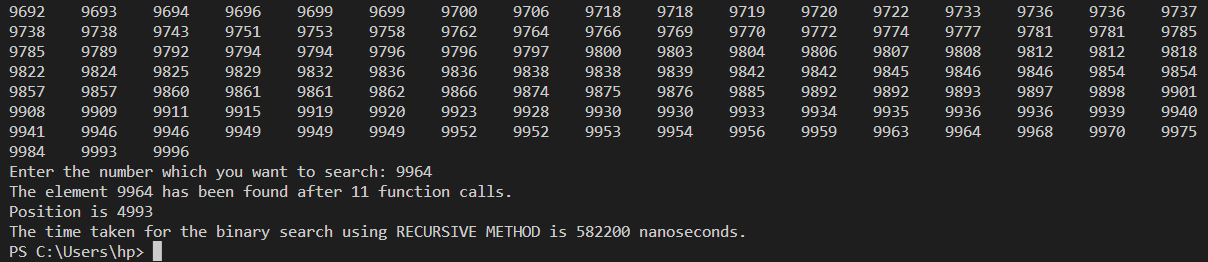
        obj.iterative\_binary\_search(arr, search);

    }

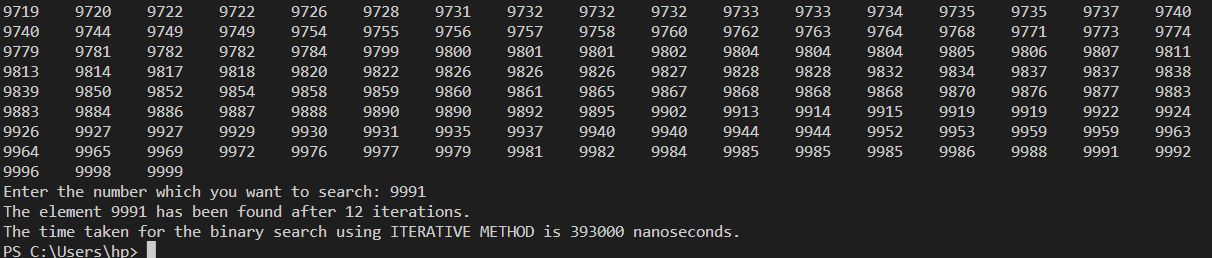
}

Case 1: When the number of elements is 5,000

Recursive approach: 582,200 nanoseconds

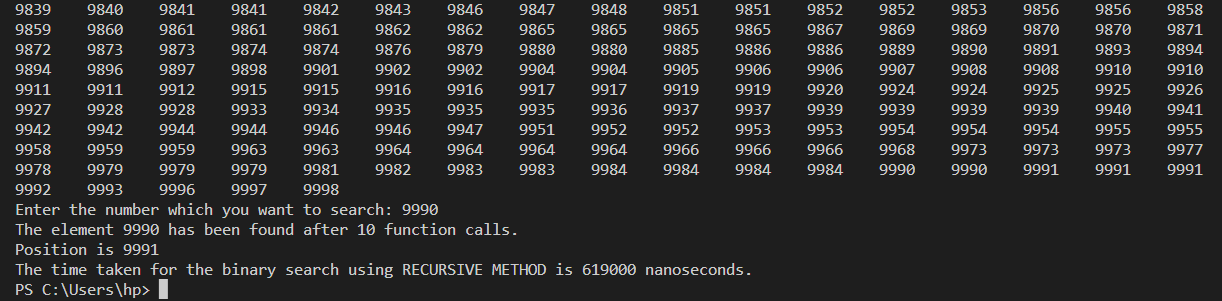


Iterative approach: 393,000 nanoseconds

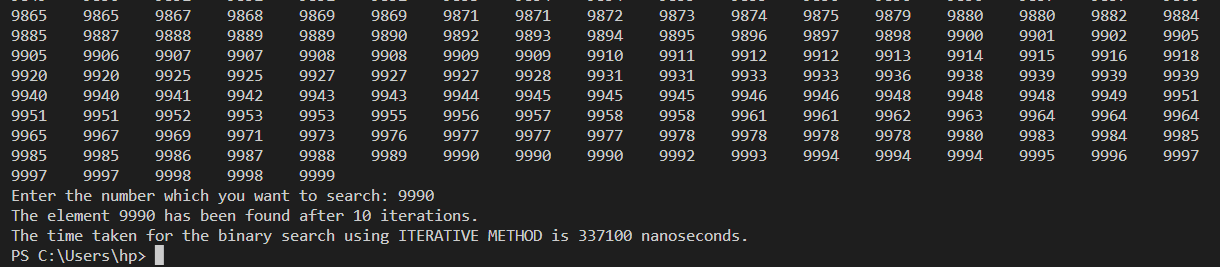


Case 2: When the number of elements is 10,000

Recursive approach: 619,000 nanoseconds

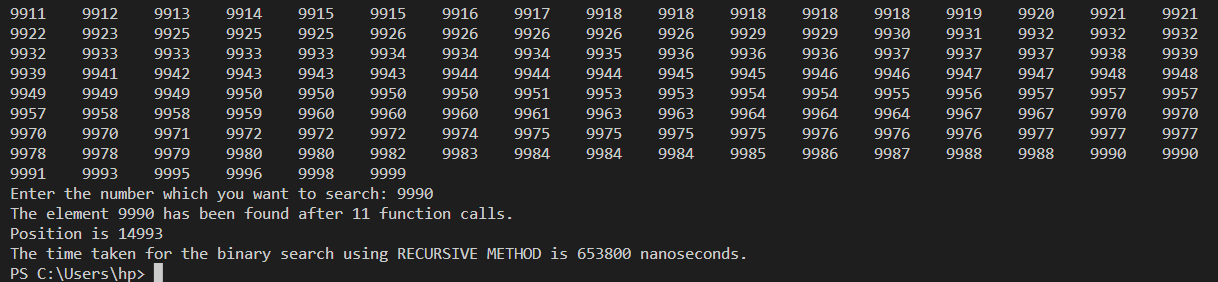


Iterative approach: 337,100 nanoseconds

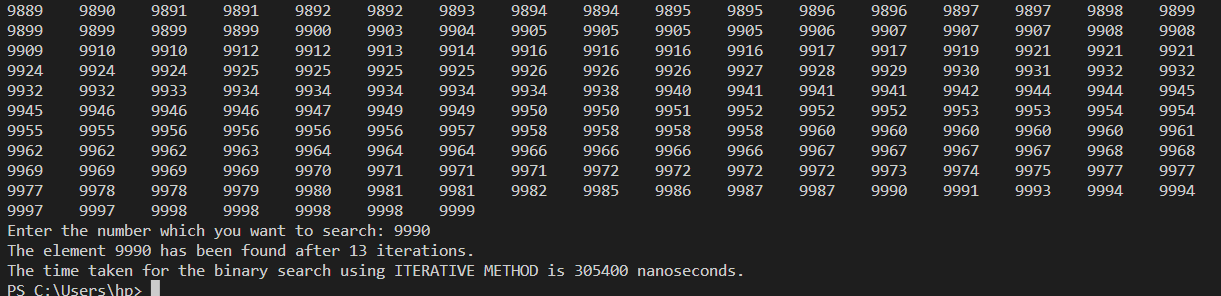


Case 3: When the number of elements is 15,000

Recursive approach: 653,800 nanoseconds

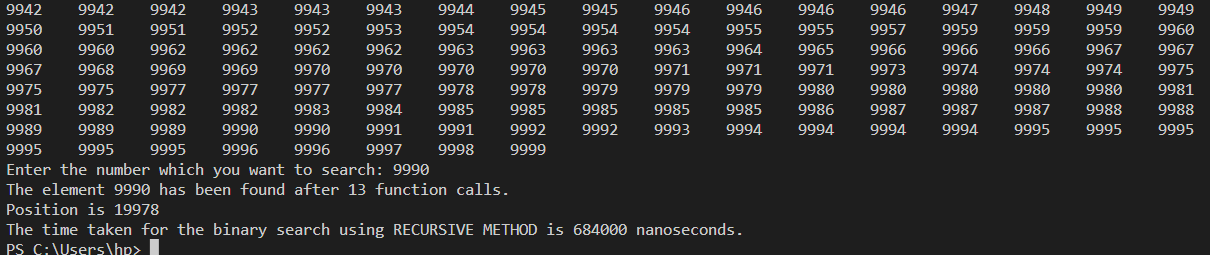


Iterative approach: 305,400 nanoseconds

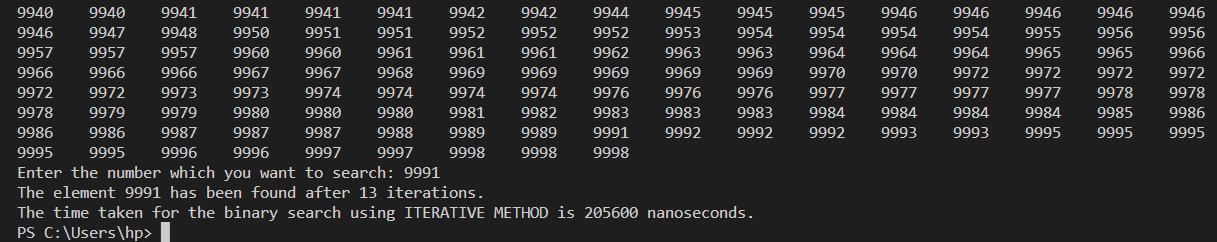


Case 4: When the number of elements is 20,000

Recursive approach: 684,000 nanoseconds

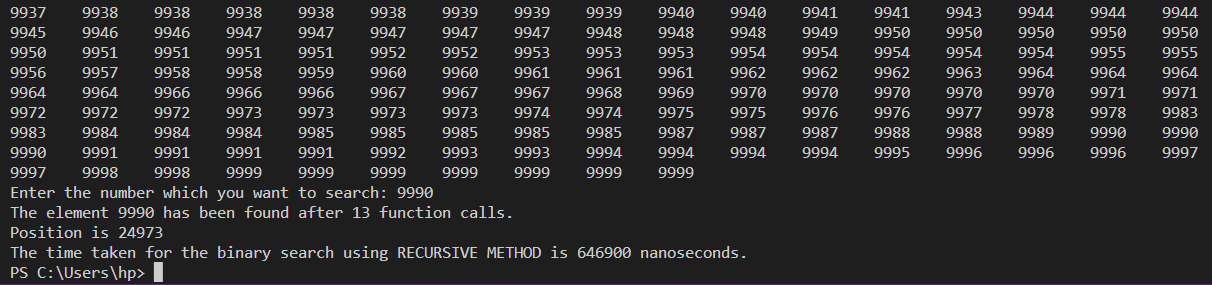


Iterative approach: 205,600 nanoseconds



Case 5: When the number of elements is 25,000

Recursive approach: 646,900 nanoseconds



Iterative approach: 300,100 nanoseconds

