Assignment 4

21AIE111

Data Structure and Algorithms – SEM-II

Professor – Dr. Sachin Sir

Submitted By: Vikhyat Bansal [CB.EN.U4AIE21076]

AMRITA VISHWA VIDYAPEETHAM

DEEMED TO BE UNIVERSITY

1. Write a java code to generate 1 lakh random integer numbers between 1 and 1 lakh.

   a) Collect it to an array.

   b) Pick a number in the far end of the array

   c) search that number and observe the time taken (in seconds)

   (you can increase the total numbers to 10 lakh also).

CODE: [For 1 Lakh random numbers]

```
import java.util.Arrays;
import java.util.Scanner;
public class Search_Element {

    public static void main( String[] args ) {
        long start = System.currentTimeMillis();            //Start a
counter to keep track of time.
        int min = 1;
        int max = 100000;
        int n, x, flag = 0, i = 0;
        Scanner s = new Scanner(System.in);

        int[] arr = new int[1000001];
        for ( i = 0; i < 100000; i++) {

            int range = (max - min) + 1;
            int randomNum = (int)(Math.random() * range) + min;
            arr[i] = randomNum;
            System.out.println(arr[i]);

        }
        System.out.println("Last element of array is: " + arr[99999]);

        System.out.print("Enter the element you want to find:");
        x = s.nextInt();
        for( i = 0; i < arr.length; i++)
        {
            if(arr[i] == x)
            {
                flag = 1;
                break;
            }
            else
            {
                flag = 0;
            }
        }
        if(flag == 1)
        {
            System.out.println("Element found at position:"+(i + 1));

        }
        else
        {
            System.out.println("Element not found");
        }

          long end = System.currentTimeMillis();
```

```
            System.out.println("Total Time taken by the program to execute is
" +
                                        ((end - start)/1000) + " seconds");

    }
}
```

OUTPUT:

```
Terminal:  Local
55267
73229
8123
32881
71703
9170
64682
77405
94164
32065
10493
Last element of array is: 10493
Enter the element you want to find:10493
Element found at position:100000
Total Time taken by the program to execute is 15 seconds
```

2. Write a java code to generate 1 lakh random integer numbers between 1 and 1 lakh.

   a) Collect it to a single linked list.

   b) Pick a number in the far end of the list (traverse the list)

   c) search that number and observe the time taken (in seconds)

   (you can increase the total numbers to 10 lakh also).

CODE:

```
import java.util.Random;
public class LinkedRand {

   static class Node{
      int data;
      Node next;
      Node(int d) {
         this.data = d;
         this.next = null;
      }
   }
   public Node head = null;
   public Node tail = null;
   public void insertNode(int d) {
      Node object = new Node(d);
      if(head == null) {
         head = object;
      }
      else {
         Node node = head;
         while(node.next!=null) {
            node = node.next;
         }
         node.next = object;
      }
      tail = object;
   }
      public static void printList(LinkedRand list)
   {

      Node node = list.head;

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

      // Traverse through the LinkedList
      while (node != null) {
         // Print the data at current node
         System.out.println(node.data + " ");

         // Go to next node
         node = node.next;
      }
   }
   public int retrieveLastElement() {
      Node node = head;
      while(node.next!=null) {
         node = node.next;
      }
      System.out.println("Last element in linked list is:" +node.data);
```

```
      return node.data;
   }
   public void findMatch(int t) {
      Node node = head;
      int count = 0;
      while (node.next != null) {

            if (node.data == t) {
               System.out.println("Match of " + t + " found at location " +
count);
            }



            if (count == 99999) {
                  System.out.println("Match of " + t + " found at location"
+ count);

            }
         count += 1;
         node = node.next;
         }
      }

   public static void main(String[] args) {
      long start=System.currentTimeMillis();
      Random obj = new Random();
      LinkedRand list = new LinkedRand();
      for(int i = 0;i <= 100000;i++) {
         list.insertNode(obj.nextInt(100000));
      }
      printList(list);
      int target = list.retrieveLastElement();
      list.findMatch(target);

      long end=System.currentTimeMillis();
      float time=(end-start)/1000;
      System.out.println("the time taken is "+(time)+" seconds");
   }
}
```

OUTPUT:

```
Terminal:  Local
65293
66351
22234
73081
261
98783
68524
62773
54664
97560
9561
92596
19771
78079
41700
14057
Last element in linked list is:14057
Match of 14057 found at 99999
the time taken is 9.0 seconds
```

3. Write a java code to generate 1 lakh random integer numbers between 1 and 1 lakh.

a) Collect it to an array.

b) Pick 3 number in the far end of the array and create a sub-array

c) search the sub-array and observe the time taken (in seconds)

(you can increase the total numbers to 10 lakh also).

CODE: [For 1 lakh random numbers]

```
import java.util.Arrays;
import java.util.Scanner;
public class random {

    public static void main( String[] args ) {
        long start = System.currentTimeMillis();            //Start a
counter to keep track of time.
        int min = 1;
        int max = 100000;
        int x, flag = 0, i =0;
        int k = 0;
        int temp[] = new int[3];


        int[] arr = new int[1000000];
        for ( i = 0; i < 100000; i++) {

            int range = (max - min) + 1;
            int randomNum = (int)(Math.random() * range) + min;
            arr[i] = randomNum;
            System.out.println(arr[i]);

        }
        System.out.println();


          int[] subarr = Arrays.copyOfRange(arr, 99997, 100000);
          for (int p : subarr) {
            System.out.print("Subarray of size 3 from initial array: " +p +
"  ");

            System.out.println();

        }
          System.out.println();


        for (int j = 0; j <arr.length-1; j++) {
            for (int l = 0; l < 3; l++) {
                if (arr[j] == subarr[0] && arr[j + 1] == subarr[1] && arr[j
+ 2] == subarr[2]) {
                    System.out.println("the element present in the array
are: " +subarr[l]);
                    System.out.println("the location(index) at which all
elements of subarray are: " +(((arr.length-1))-l));
//
```

```
                    k = 1;
                }

            }
        }
        if (k==0) {
            System.out.println("the subarray is not present in the array");
        }


        long end = System.currentTimeMillis();
          System.out.println("Total time taken by the program to execute
is: " +
                    ((end - start)/1000) + " seconds");


    }
}
```

OUTPUT:

```
Terminal:  Local
5526
17291
69654
51715
13721
42077

Subarray of size 3 from initial array: 51715
Subarray of size 3 from initial array: 13721
Subarray of size 3 from initial array: 42077

the element present in the array are: 51715
the location(index) at which all elements of subarray are: 999999
the element present in the array are: 13721
the location(index) at which all elements of subarray are: 999998
the element present in the array are: 42077
the location(index) at which all elements of subarray are: 999997
Total time taken by the program to execute is: 3 seconds
```

4. Write java code to find the possible substrings (size of substring is atleast 2) from the given sequences

a) ATGCT b) AGCT

a) CCGTCG b) CCGCG

CODE: a)

```
public class Searchsub {
   public static void main(String[]args) {
      String arr[]= {"A","T","G","C","T"};
      String subarr[]= {"A","G","C","T"};
      String temp[]= new String[2];
      String temp2[]= new String[3];
      for (int i = 0; i <arr.length-1; i++) {
         for (int j = 0; j <3 ; j++) {
            if (arr[i]==subarr[j] && arr[i+1]==subarr[j+1]) {
               System.out.println("common seqence with two elements");
               for (int k = 0; k <temp.length; k++) {
                  temp[k]=subarr[j+k];
                  System.out.println(temp[k]);
               }
               break;
            }
         }
      }
      for (int i = 0; i <arr.length-2; i++) {
         for (int j = 0; j <2; j++) {
            if (arr[i]==subarr[j] && arr[i+1]==subarr[j+1] &&
arr[i+2]==subarr[j+2]) {
               System.out.println("the common seqence with three elemets");
               for (int k = 0; k <temp2.length; k++) {
                  temp2[k]=subarr[j+k];
                  System.out.println(temp2[k]);
               }
               break;
            }
         }
      }
   }
}
```

OUTPUT: a)

```
Terminal:  Local
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\CODING\JAVA\VS Code> java .\Searchsub.java
common seqence with two elements
G
C
common seqence with two elements
C
T
the common seqence with three elemets
G
C
T
```

b)

```java
public class Searchsub {
   public static void main(String[]args) {
      String arr[]= {"C","C","G","T","C","G"};
      String subarr[]= {"C","C","G","C","G"};
      String temp[]= new String[2];
      String temp2[]= new String[3];
      for (int i = 0; i <arr.length-1; i++) {
         for (int j = 0; j <4 ; j++) {
            if (arr[i]==subarr[j] && arr[i+1]==subarr[j+1]) {
               System.out.println("common seqence with two elements");
               for (int k = 0; k <temp.length; k++) {
                  temp[k]=subarr[j+k];
                  System.out.println(temp[k]);
               }
               break;
            }
         }
      }
      for (int i = 0; i <arr.length-2; i++) {
         for (int j = 0; j <3; j++) {
            if (arr[i]==subarr[j] && arr[i+1]==subarr[j+1] &&
arr[i+2]==subarr[j+2]) {
               System.out.println("the common seqence with three elemets");
               for (int k = 0; k <temp2.length; k++) {
                  temp2[k]=subarr[j+k];
                  System.out.println(temp2[k]);
               }
               break;
            }
         }
      }
   }
}
```

OUTPUT: b)

```
Terminal:  Local

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\CODING\JAVA\VS Code> java .\Searchsub.java
common seqence with two elements
C
C
common seqence with two elements
C
G
common seqence with two elements
C
G
the common seqence with three elemets
C
C
G
```

Mention machine specifications like OS, RAM, GPU

General specifications

Operating system	Windows 11 Home Single Language 64-bit Version: 22000.675
Microprocessor	AMD Ryzen 7 5800H with Radeon Graphics
System memory	16 GB
Memory slot 1	8GB Hynix 3200MHz
Memory slot 2	8GB Hynix 3200MHz
System board	88DE 96.31
System BIOS	F.15

Video

Graphic device 1	NVIDIA GeForce RTX 3050 Laptop GPU