7. Disk Scheduling

Aim:

To implement a disk scheduling algorithm.

Program:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.Collections;

/**
 * JAVA program to demonstrate the LOOK
 * disk scheduling algorithm.
 *
 * @author jimil
 */
public class LookAlgorithmImpl {

    static int workFlowSize = 0;
    static int totalCylinders = 0;
    static int initDir = 0;
    static Integer workflow[];
    static int movement = 0;
    static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

    /**
     * Driver function for the program
     *
     * @param args
     */
    public static void main(String[] args) throws Exception {
        int headStart;
        System.out.println("Enter the total number of cylinders: ");
        totalCylinders = Integer.parseInt(br.readLine());
        System.out.println("Enter the number of cylinder requests: ");
        workFlowSize = Integer.parseInt(br.readLine());
        workflow = new Integer[workFlowSize];
```

```
        System.out.println("Enter the cylinder requests: ");
        for(int i = 0; i < workFlowSize; i++) {
            workflow[i] = Integer.parseInt(br.readLine());
        }
        System.out.println("Enter the starting location of the Head: ");
        headStart = Integer.parseInt(br.readLine());
        System.out.println("Head Start Loc = " + headStart);
        System.out.println("Enter the initial direction for head movement:
");
        System.out.println("0.  Left");
        System.out.println("1.  Right");
        initDir = Integer.parseInt(br.readLine());
        int prev;
        int current = 0;
        if(initDir == 0) {
            Arrays.sort(workflow, Collections.reverseOrder());
            prev = headStart;
            int unprocessed[] = new int[20];
            int idx = 0;
            int min = getMinValue(workflow, workFlowSize);
            int count = 0;
            for(int i = 0; i < workFlowSize; i++) {
                if(workflow[i] < headStart && workflow[i] >= min) {
                    if(count == 0) {
                        count++;
                        current = workflow[i];
                        System.out.println(headStart + "-" + current + ",
movement = " + Math.abs(headStart-current));
                        movement += Math.abs(headStart-current);
                    } else {
                        count ++;
                        prev = current;
                        current = workflow[i];
                        System.out.println(prev + "-" + current + ",
movement = " + Math.abs(prev-current));
                        movement += Math.abs(prev-current);
                    }
                } else {
                    unprocessed[idx++] = workflow[i];
                }
            }
            /**
```

```
             * TODO Comment below sort call to implement C-LOOK instead of
LOOK
             */
            Arrays.sort(unprocessed);
            for(int i = 0; i < unprocessed.length; i++) {
                if(unprocessed[i] > 0) {
                    prev = current;
                    current = unprocessed[i];
                    System.out.println(prev + "-" + current + ", movement =
" + Math.abs(prev-current));
                    movement += Math.abs(prev-current);
                }
            }
            System.out.println("Total Movement = " + movement);
        } else if(initDir == 1) {
            Arrays.sort(workflow);
            prev = headStart;
            Integer unprocessed[] = new Integer[20];
            int idx = 0;
            int max = getMaxValue(workflow, workFlowSize);
            int count = 0;
            for(int i = 0; i < workFlowSize; i++) {
                if(workflow[i] > headStart && workflow[i] <= max) {
                    if(count == 0) {
                        count++;
                        current = workflow[i];
                        System.out.println(headStart + "-" + current + ",
movement = " + Math.abs(headStart-current));
                        movement += Math.abs(headStart-current);
                    } else {
                        count ++;
                        prev = current;
                        current = workflow[i];
                        System.out.println(prev + "-" + current + ",
movement = " + Math.abs(prev-current));
                        movement += Math.abs(prev-current);
                    }
                } else {
                    unprocessed[idx++] = workflow[i];
                }
            }
            /**
```

```
             * TODO Comment below sort call to implement C-LOOK instead of
LOOK
             */
            Arrays.sort(unprocessed, 0, idx, Collections.reverseOrder());
            for(int i = 0; i < unprocessed.length; i++) {
                if(unprocessed[i] != null) {
                    if(unprocessed[i] > 0) {
                        prev = current;
                        current = unprocessed[i];
                        System.out.println(prev + "-" + current + ",
movement = " + Math.abs(prev-current));
                        movement += Math.abs(prev-current);
                    }
                }
            }
            System.out.println("Total Movement = " + movement);
        }
    }

    /**
     * Utility function to get min from an array.
     *
     * @param arr
     * @param len
     * @return
     */
    public static int getMinValue(Integer arr[], int len) {
        int min = arr[0];
        for(int i = 1; i < len; i++) {
            if(arr[i] < min) {
                min = arr[i];
            }
        }
        return min;
    }

    /**
     * Utility function to get max from an array.
     *
     * @param arr
     * @param len
     * @return
```

```
     */
    public static int getMaxValue(Integer arr[], int len) {
        int max = arr[0];
        for(int i = 1; i < len; i++) {
            if(arr[i] > max) {
                max = arr[i];
            }
        }
        return max;
    }
}
```

Output:

Case 1 - Head initially moving in the right direction

```
Enter the total number of cylinders: 200
Enter the number of cylinder requests: 5
Enter the cylinder requests:
23 89 132 42 187
Enter the starting location of the Head:
100
Enter the initial direction for head movement:
0.  Left
1.  Right
1
--------------------------------------------------------------------------------
100-132, movement = 32
132-187, movement = 55
187-89, movement = 98
89-42, movement = 47
42-23, movement = 19
Total Movement = 251
```

Case 2 - Head initially moving in the left direction

```
Enter the total number of cylinders: 200
Enter the number of cylinder requests: 5
Enter the cylinder requests:
23 89 132 42 187
Enter the starting location of the Head: 100
Enter the initial direction for head movement:
0.  Left
1.  Right
0
--------------------------------------------------------------------------------
100-89, movement = 11
89-42, movement = 47
42-23, movement = 19
23-132, movement = 109
132-187, movement = 55
Total Movement = 241
```

Conclusion:

Thus, through this experiment, I understood the need for a disk scheduling algorithm and implemented the LOOK disk scheduling algorithm.