**Experiment 9**

**Keyul Jain**

**1902055**

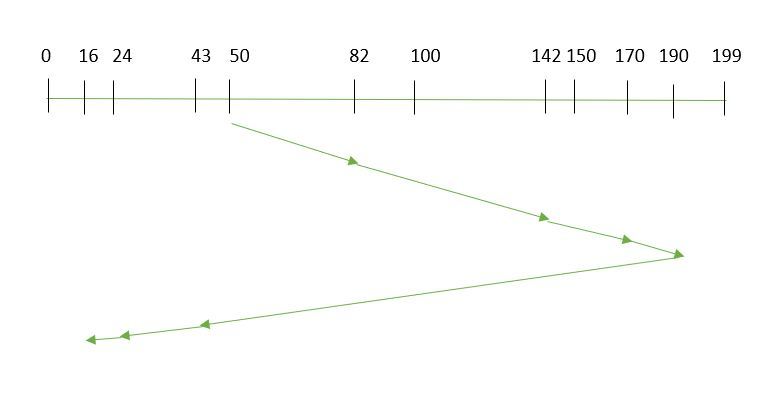
**Aim: To demonstrate the use of Disk scheduling algorithm using LOOK and C-LOOK.**

**Theory:**

**Disk Scheduling Algorithms**

LOOK: It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

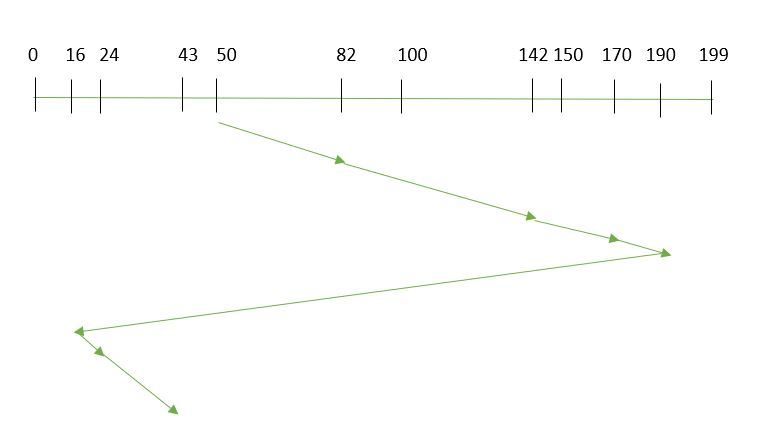
Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move “towards the larger value”.  


So, the seek time is calculated as:

=(190-50)+(190-16)  
=314

**CLOOK:** As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end’s last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move “towards the larger value”  


So, the seek time is calculated as:

=(190-50)+(190-16)+(43-16)  
=341

**Code:**

import java.util.\*;

class exp9 {

    public static void main(String[] args) {

        int ini\_head, n, prev\_head, max\_loc = 0, min\_loc = 10000;

        System.out.println("Enter initial header location: ");

        Scanner sc = new Scanner(System.in);

        ini\_head = sc.nextInt();

        int cini\_head = ini\_head;

        System.out.println("Enter previous header location: ");

        prev\_head = sc.nextInt();

        int cprev\_head = prev\_head;

        System.out.println("Enter number of disk locations: ");

        n = sc.nextInt();

        int[] disk\_location = new int[n];

        int seek\_time = 0, i, j, f = 0, d = 0;

        System.out.println("Enter disk locations: ");

        for (i = 0; i < n; i++) {

            disk\_location[i] = sc.nextInt();

            if (disk\_location[i] < min\_loc)

                min\_loc = disk\_location[i];

            if (disk\_location[i] > max\_loc)

                max\_loc = disk\_location[i];

        }

        int[] visited = new int[n + 1];

        int[] cvisited = new int[n + 1];

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

        System.out.println("Disk Location \t Seek Time");

        if (prev\_head <= ini\_head)

            f = 0;

        else

            f = 1;

        for (i = 0; i < n; i++) {

            int pos = -1;

            int min = 10000;

            for (j = 0; j < n; j++) {

                if (f == 0) {

                    if (disk\_location[j] > ini\_head && min > Math.abs(disk\_location[j] - ini\_head) && visited[j] == 0) {

                        min = Math.abs(disk\_location[j] - ini\_head);

                        pos = j;

                    }

                } else if (f == 1) {

                    if (disk\_location[j] <= ini\_head && min > Math.abs(disk\_location[j] - ini\_head)

                            && visited[j] == 0) {

                        pos = j;

                        min = Math.abs(disk\_location[j] - ini\_head);

                    }

                }

            }

            if (pos == -1) {

                if (f == 0) {

                    f = 1;

                } else {

                    f = 0;

                }

                System.out.println("Changing Directions");

                System.out.println("Disk Location \t Seek Time");

                i--;

                continue;

            }

            visited[pos] = 1;

            seek\_time += Math.abs(disk\_location[pos] - ini\_head);

            System.out.println(disk\_location[pos] + "\t\t " + seek\_time);

            ini\_head = disk\_location[pos];

        }

        System.out.println();

        System.out.println("Total Seek Time :" + seek\_time);

        System.out.println("\n\*\*\*C-LOOK\*\*\*");

        seek\_time = 0;

        System.out.println("Disk Location \t Seek Time");

        if (prev\_head <= cini\_head)

            f = 0;

        else

            f = 1;

        for (i = 0; i < n; i++) {

            int pos = -1;

            int min = 10000;

            for (j = 0; j < n; j++) {

                if (d == 1) {

                    if (disk\_location[j] == cini\_head) {

                        pos = j;

                        break;

                    }

                } else {

                    if (f == 0) {

                        if (disk\_location[j] > cini\_head && min > disk\_location[j] - cini\_head && cvisited[j] == 0) {

                            min = disk\_location[j] - cini\_head;

                            pos = j;

                        }

                    } else if (f == 1) {

                        if (disk\_location[j] <= cini\_head && min > cini\_head - disk\_location[j] && cvisited[j] == 0) {

                            pos = j;

                            min = cini\_head - disk\_location[j];

                        }

                    }

                }

            }

            if (pos == -1) {

                if (f == 0) {

                    cini\_head = min\_loc;

                    d = 1;

                } else {

                    cini\_head = max\_loc;

                    d = 1;

                }

                System.out.println("Circling around");

                System.out.println("Disk Location \t Seek Time");

                i--;

                continue;

            }

            cvisited[pos] = 1;

            if (d == 0)

                seek\_time += Math.abs(disk\_location[pos] - cini\_head);

            else if (d == 1)

                d = 0;

            System.out.println(disk\_location[pos] + "\t\t " + seek\_time);

            cini\_head = disk\_location[pos];

        }

        System.out.println();

        System.out.println("Total Seek Time :" + seek\_time);

        sc.close();

    }

}

**Output :**

