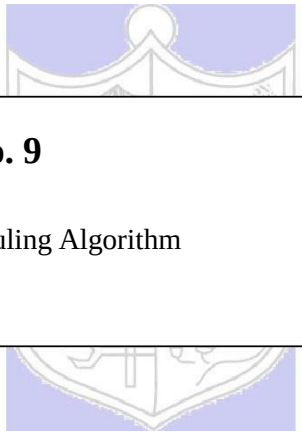


Experiment No. 9

Title: Disk Scheduling Algorithm

Batch: B2 **Roll No: 16010420117** **Experiment No: 9**

Aim: To implement any one disk scheduling algorithm.

Resources needed: Text editor and JAVA/C compiler.

Theory:

Pre lab/Prior concepts:

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.

Two or more request may be far from each other so can result in greater disk arm movement.

Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

Seek Time:Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.

Rotational Latency: Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.

Transfer Time: Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

Disk Access Time: Disk Access Time is:

```
Disk Access Time = Seek Time + Rotational Latency + Transfer Time
```

Disk Response Time: Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of the all requests. Variance Response Time is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

For more information refer to : https://www.geeksforgeeks.org/disk-scheduling-algorithms/

Results: Results should be saved in a separate text/ doc file saved as <Roll No_Batch No_Date> (No snapshot s to be attached or pasted in this file)

This file must contain on the top:

Name:
Roll No.
Exp No.
Batch:
Date:

And Students have to upload this document electronically.

C-SCAN Algorithm: In C-SCAN algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.

Code:

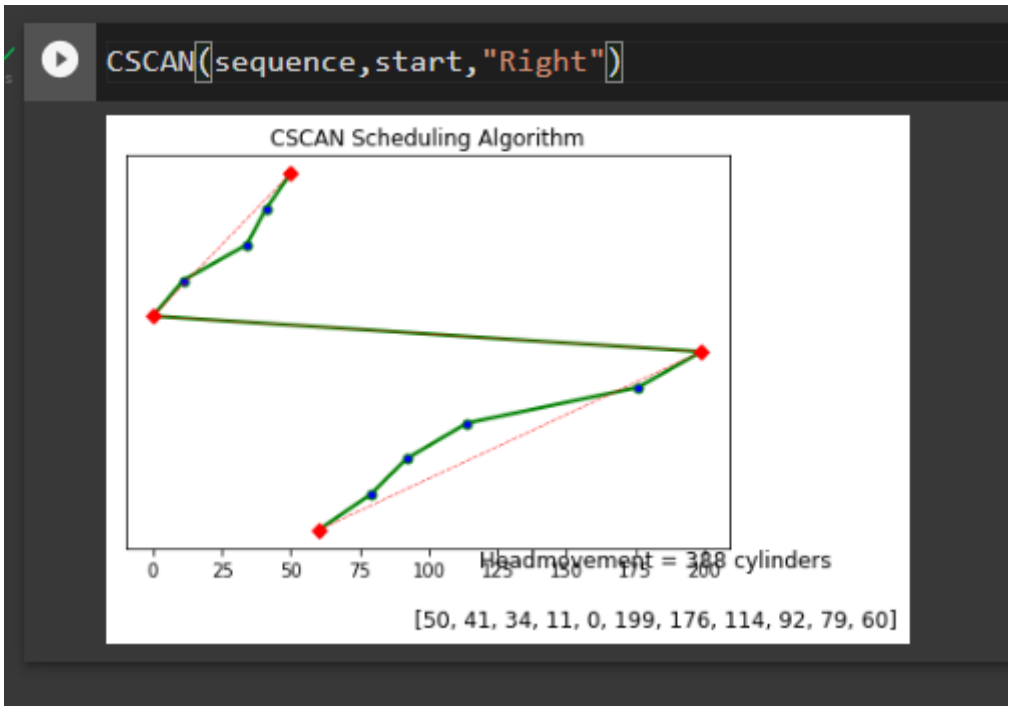
```
[1] from matplotlib import pyplot as plt
    import math

[26] sequence = [int(i) for i in input("Enter space separated order of request: ").split()]
    sequence.sort()
    # print(sequence)
    start = int(input("Enter the current position of READ/WRITE head: "))
    CYLINDER_MAX = 199

    Enter space separated order of request: 176 79 34 60 92 11 41 114
    Enter the current position of READ/WRITE head: 50
```

```
def CSCAN(sequence, start, direction):
    temp = sequence.copy()
    left = []
    right = []
    x = []
    y = []
    x_approx = []
    y_approx = []
    headmovement = 0
    headmovement_approx = 0
    x.append(start)
    if(direction=="Left"):
        for i in temp:
            if i<start:
                left.append(i)
            else:
                right.append(i)
        left.sort(reverse = True)
        for i in left:
```

```
            x.append(i)
        x.append(0)
        x.append(CYLINDER_MAX)
        right.sort(reverse = True)
        for i in right:
            x.append(i)
        x_approx.append(start)
        x_approx.append(min(x))
        x_approx.append(max(x))
        x_approx.append(x[-1])
        headmovement_approx = abs(start-0)
        headmovement_approx = headmovement_approx + abs(0-max(x))
        headmovement_approx = headmovement_approx + abs(0-x[-1])
    elif(direction=="Right"):
        for i in temp:
            if i>start:
                right.append(i)
            else:
                left.append(i)
        right.sort()
        for i in right:
            x.append(i)
        x.append(CYLINDER_MAX)
        x.append(0)
        left.sort()
        for i in left:
            x.append(i)
        x_approx.append(start)
        x_approx.append(CYLINDER_MAX)
        x_approx.append(0)
        x_approx.append(x[-1])
        headmovement_approx = abs(start-199)
        headmovement_approx = headmovement_approx + abs(199-0)
        headmovement_approx = headmovement_approx + abs(0-x[-1])



    y_approx.append(0)
    size = len(x)
    for i in range(0,size):
```

```
        y.append(-i)
        if(x[i]==0 or x[i]==199):
            y_approx.append(-i)
        if i!=(size-1):
            headmovement = headmovement + abs(x[i]-x[i+1])
        else:
            y_approx.append(-i)
    string = 'Headmovement = ' + str(headmovement) + ' cylinders'
    string2 = str(x)

    plt.plot(x,y, color="green", markerfacecolor = 'blue', marker='o', mar
kersize = 5, linewidth = 2, label="CSCAN")
    plt.plot(x_approx,y_approx, dashes = [6,2] , color="red", markerfaceco
lor = 'red', marker='D', markersize = 5, linewidth = 0.5, label="Approx CS
CAN")
    plt.ylim = (0,size)
    plt.xlim = (0,CYLINDER_MAX)
    plt.yticks([])
    plt.title("CSCAN Scheduling Algorithm")
    plt.text(182.5, -
10.85, string, horizontalalignment='center',verticalalignment='center',fon
tsize=12)
    plt.text(182.5, -
12.5, string2, horizontalalignment='center',verticalalignment='center',fon
tsize=12)
    plt.show()
```

Output:


CSCAN(sequence,start,"Right")
CSCAN Scheduling Algorithm
0
25
50
75
100
125
150
175
200
Headmovement = 388 cylinders
[50, 41, 34, 11, 0, 199, 176, 114, 92, 79, 60]

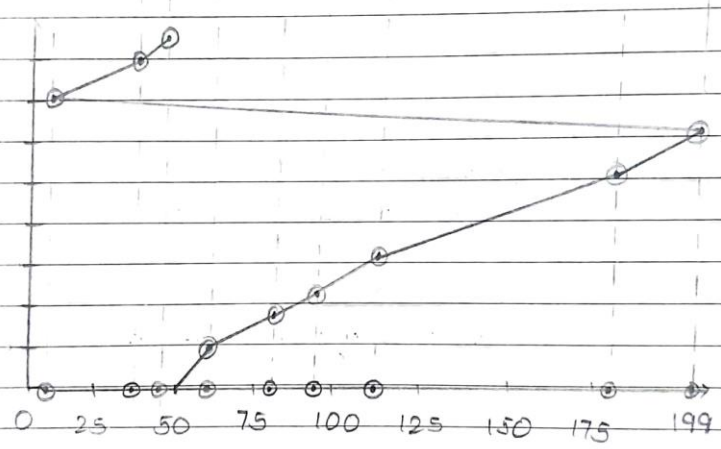
Questions:

Solve the same example of Disk Scheduling Algorithm given to the system, on paper and compare the results.

Ans:

Ans CSCAN - disk scheduling algorithm

Given
request sequence = {176, 79, 34, 60, 92, 11, 41, 114}
initial head position = 50
direction = right (we are moving from left to right)



seek sequence is = {60, 79, 92, 114, 176, 199, 0, 11, 34, 41}

Total seek count is calculated as = (60-50) + (79-60) + (92-79) + (114-92) + (176-114) + (199-176) + (199-0) + (11-0) + (34-11) + (41-34)

= 388

Outcomes: CO2: Demonstrate use of inter process communication

Conclusion: (Conclusion to be based on outcomes achieved)
Understood and Successfully Implemented CSCAN Disc Scheduling algorithm

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

Referred Books:

1. Applied Operating System Concepts, 1st ed. Silberschatz, Galvin and Gagne, John Wiley Publishers.
2. Modern Operating Systems, Tanenbaum, PHI.
3. Operating System, 4th Edition, William Stallings, Pearson