Experiment 07 – Disk Scheduling Algorithms-FCFS, SSTF

SE COMP C 34 YASH SINHA

Learning Objective: Implementation of FCFS and SSTF disk scheduling algorithms.

Tools: C/Java under Windows or Linux environment

Theory: Develop a program to implement FCFS and SSTF disk scheduling algorithms.

1. FCFS Disk Scheduling –

Given an array of disk track numbers and initial head position, our task is to find the total number of seek operations done to access all the requested tracks if **First Come First Serve (FCFS)** disk scheduling algorithm is used.

FCFS is the simplest disk scheduling algorithm. As the name suggests, this algorithm entertains requests in the order they arrive in the disk queue. The algorithm looks very fair and there is no starvation (all requests are serviced sequentially) but generally, it does not provide the fastest service.

Example:

Input:

Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}

Initial head position = 50

Output:

Total number of seek operations = 510

Seek Sequence is

176

79

34

60

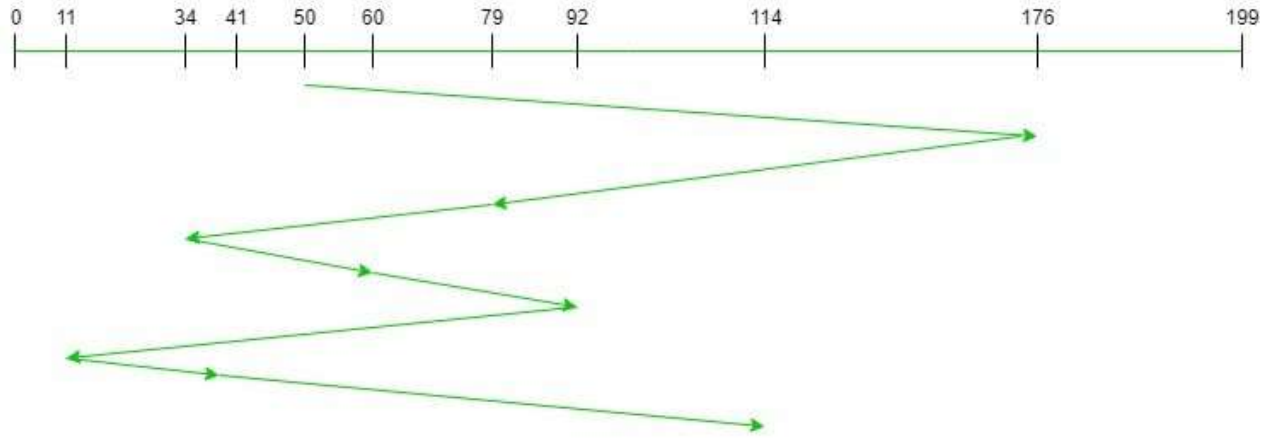
92

11

41

114

The following chart shows the sequence in which requested tracks are serviced using FCFS.



Therefore, the total seek count is calculated as:

= (176-50)+(176-79)+(79-34)+(60-34)+(92-60)+(92-11)+(41-11)+(114-41)

= 510

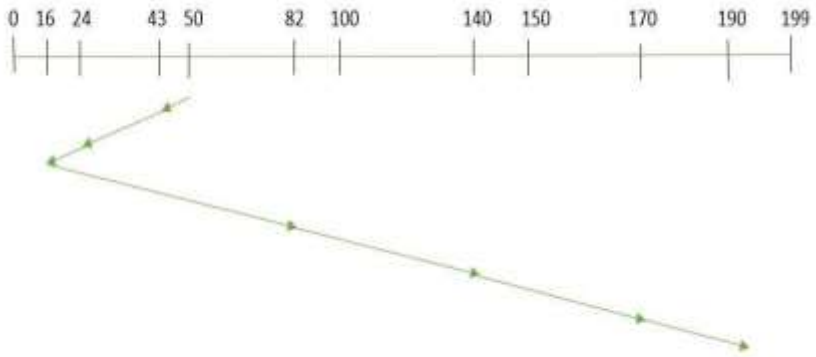
2. SSTF Disk Scheduling-

In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

Example:

Suppose the order of request is- (82,170,43,140,24,16,190)

And current position of Read/Write head is : 50



So, total seek time:

=(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-40)+(190-170)

=208

ALGORITHM:

1. **FCFS Disk Scheduling:**

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. 'head' is the position of disk head.
2. Let us one by one take the tracks in default order and calculate the absolute distance of the track from the head.
3. Increment the total seek count with this distance.
4. Currently serviced track position now becomes the new head position.
5. Go to step 2 until all tracks in request array have not been serviced.

2. SSTF Disk Scheduling:

1. Let Request array represents an array storing indexes of tracks that have been requested. 'head' is the position of disk head.

2. Find the positive distance of all tracks in the request array from head.

3.Find a track from requested array which has not been accessed/serviced yet and has minimum distance from head.

4.Increment the total seek count with this distance.

5.Currently serviced track position now becomes the new head position.

6.Go to step 2 until all tracks in request array have not been serviced.

Implementation:

main.cpp +

```cpp
// C++ program for implementation of
// SSTF disk scheduling
#include <bits/stdc++.h>
using namespace std;

// Calculates difference of each
// track number with the head position
void calculatedifference(int request[], int head,
                         int diff[][2], int n)
{
    for(int i = 0; i < n; i++)
    {
        diff[i][0] = abs(head - request[i]);
    }
}

// Find unaccessed track which is
// at minimum distance from head
```

```
19 int findMIN(int diff[][2], int n)
20 {
21     int index = -1;
22     int minimum = 1e9;
23
24     for(int i = 0; i < n; i++)
25     {
26         if (!diff[i][1] && minimum > diff[i][0])
27         {
28             minimum = diff[i][0];
29             index = i;
30         }
31     }
32     return index;
33 }
34
35 void shortestSeekTimeFirst(int request[],
36                            int head, int n)
```

```
{
    if (n == 0)
    {
        return;
    }

    // Create array of objects of class node
    int diff[n][2] = { { 0, 0 } };

    // Count total number of seek operation
    int seekcount = 0;

    // Stores sequence in which disk access is done
    int seeksequence[n + 1] = {0};

    for(int i = 0; i < n; i++)
    {
        seeksequence[i] = head;
```

```
55          calculatedifference(request, head, diff, n);
56          int index = findMIN(diff, n);
57          diff[index][1] = 1;
58
59          // Increase the total count
60          seekcount += diff[index][0];
61
62          // Accessed track is now new head
63          head = request[index];
64      }
65      seeksequence[n] = head;
66
67      cout << "Total number of seek operations = "
68           << seekcount << endl;
69      cout << "Seek sequence is : " << "\n";
70
71      // Print the sequence
72      for(int i = 0; i <= n; i++)
```

```
    {
        cout << seeksequence[i] << "\n";
    }
}

// Driver code
int main()
{
    int n = 8;
    int proc[n] = { 176, 79, 34, 60, 92, 11, 41, 114 };

    shortestSeekTimeFirst(proc, 50, n);

    return 0;
}
```

OUTPUT:-

```
Total number of seek operations = 204
Seek sequence is :
50
41
34
11
60
79
92
114
176


** Process exited - Return Code: 0 **
```

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){

    int i,j,k,n,m,sum=0,x,y,h;
    cout<<"Enter the size of disk\n";
    cin>>m;
    cout<<"Enter number of requests\n";
    cin>>n;
    cout<<"Enter the requests\n";

    // creating an array of size n
    vector <int> a(n);
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    for(i=0;i<n;i++){
        if(a[i]>m){
            cout<<"Error, Unknown position "<<a[i]<<"\n";
            return 0;
        }
    }
    cout<<"Enter the head position\n";
    cin>>h;

    // head will be at h at the starting
    int temp=h;
    cout<<temp;
    for(i=0;i<n;i++){
        cout<<" -> "<<a[i]<<' ';
        // calculating the difference for the head movement
        sum+=abs(a[i]-temp);
        // head is now at the next I/O request
        temp=a[i];
    }
    cout<<'\n';
    cout<<"Total head movements = "<< sum<<'\n';
```

```
        return 0;
    }
```

OUTPUT:

```
Enter the size of disk
199
Enter number of requests
8
Enter the requests
98 183 37 122 14 124 65 67
Enter the head position
53
53 -> 98  -> 183  -> 37  -> 122  -> 14  -> 124  -> 65  -> 67
Total head movements = 640
```

Result and Discussion: ….

In this experiment we learned about Implementation of FCFS and SSTF disk scheduling algorithms. We performed the program to implement FCFS and SSTF disk scheduling algorithms.

Learning Outcomes: The student should have the ability to

LO1: **explain** FCFS and SSTF disk scheduling algorithms

LO2: **apply** FCFS and SSTF disk scheduling algorithms for the given problem

LO3: **write** a code to implement FCFS and SSTF disk scheduling algorithms

Course Outcomes: Upon completion of the course students will be able to make use of FCFS and SSTF disk scheduling algorithms.

Conclusion:

In this experiment we learned about Implementation of FCFS and SSTF disk scheduling algorithms. We performed the program to implement FCFS and SSTF disk scheduling algorithms.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				