

Operating System

Experiment No. 10

Name: Ayush Jain

SAP ID: 60004200132

Batch: B2

Computer Engineering

Aim- Implement disk scheduling algorithm FCFS, SSTF, SCAN, CSCAN etc.

Problem Statement-

Perform following Disk Scheduling algorithms and also perform the Comparative Assessment of them 1) FCFS

2) SSTF

3) SCAN

4) CSCAN

5) LOOK

Description:

1) FCFS

All incoming requests are placed at the end of the queue. Whatever number that is next in the queue will be the next number served. Using this algorithm doesn't provide the best results. To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next.

2) SSTF

In this case request is serviced according to next shortest distance. Starting at 50, the next shortest distance would be 62 instead of 34 since it is only 12 tracks away from 62 and 16 tracks away from 34. The process would continue until all the process are taken care of.

3) SCAN

This approach works like an elevator does. It scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up

4) CSCAN

Circular scanning works just like the elevator to some extent. It begins its scan toward the nearest end and works it way all the way to the end of the system. Once it hits the bottom or top it jumps to the other end and moves in the same direction.

5) LOOK

This is just an enhanced version of C-SCAN. In this the scanning doesn't go past the last request in the direction that it is moving. It too jumps to the other end but not all the way to the end. Just to the furthest request.

SSFT

Code:

```
#include <bits/stdc++.h> using namespace std; 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]);
    }
} int findMIN(int diff[][2], int
n)
{   int index = -1;
int minimum = 1e9;

for(int i = 0; i < n; i++)
{
    if (!diff[i][1] && minimum > diff[i][0])
    {
        minimum = diff[i][0];
index = i;
    }
}   return
index;
```

```
}
```

```
void shortestSeekTimeFirst(int request[],
```

```
    int head, int n)
```

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

```
    }    int diff[n][2] = { { 0, 0 } };
```

```
int seekcount = 0;    int
```

```
seeksequence[n + 1] = {0};
```

```
for(int i = 0; i < n; i++)
```

```
{
```

```
    seeksequence[i] = head;
```

```
    calculatedifference(request, head, diff, n);
```

```
    int index = findMIN(diff, n);
```

```
    diff[index][1] = 1;    seekcount +=
```

```
    diff[index][0];    head = request[index];
```

```
}
```

```
seeksequence[n] = head;
```

```
cout << "Total number of seek operations = "
```

```
    << seekcount << endl;    cout <<
```

```
"Seek sequence is : " << "\n";    for(int i
```

```
= 0; i <= n; i++)
```

```
{
```

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

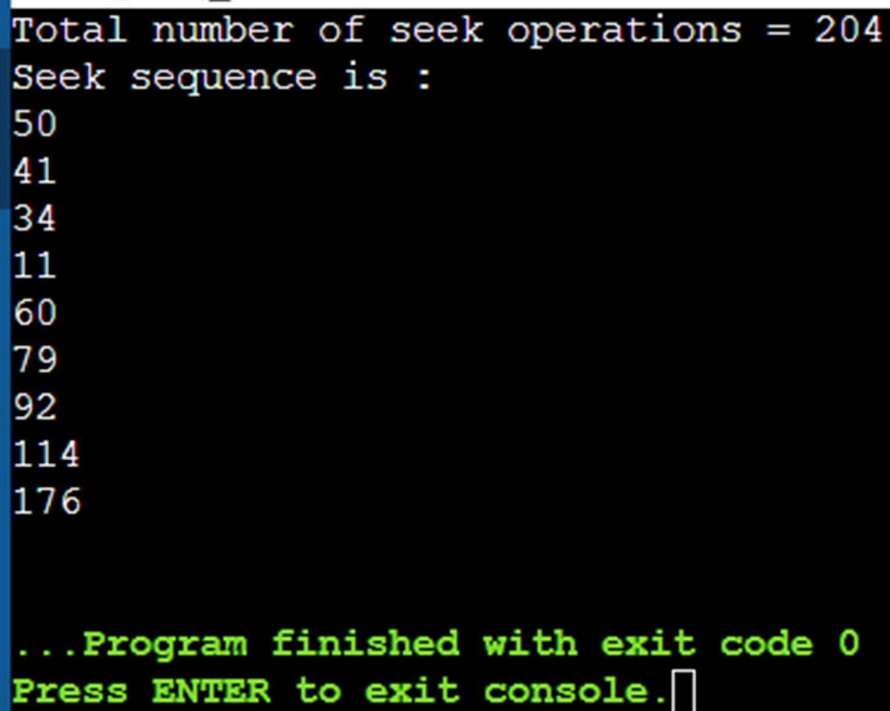
```
}
```

```
} 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  
  
...Program finished with exit code 0  
Press ENTER to exit console. □
```

SCAN

Code :

```
#include <bits/stdc++.h> using  
namespace std;  
int size = 8; int  
disk_size = 200;  
void SCAN(int arr[], int head, string direction)  
{    int seek_count = 0;    int  
distance, cur_track;    vector<int>  
left, right;    vector<int>
```

```

seek_sequence;    if (direction ==
"left")        left.push_back(0);
else if (direction == "right")
right.push_back(disk_size - 1);

    for (int i = 0; i < size; i++) {
if (arr[i] < head)
left.push_back(arr[i]);    if
(arr[i] > head)
right.push_back(arr[i]);

    }    std::sort(left.begin(),
left.end());    std::sort(right.begin(),
right.end());    int run = 2;    while
(run--) {        if (direction == "left")
{            for (int i = left.size() - 1; i
>= 0; i--) {                cur_track =
left[i];
seek_sequence.push_back(cur_track)
;                distance = abs(cur_track -
head);                seek_count +=
distance;                head = cur_track;

            }                direction
= "right";

        }

    else if (direction == "right") {        for (int
i = 0; i < right.size(); i++) {            cur_track
= right[i];
seek_sequence.push_back(cur_track);

```

```

distance = abs(cur_track - head);

seek_count += distance;          head =

cur_track;

    }

direction = "left";

    }

}

cout << "Total number of seek operations = "

    << seek_count << endl;

cout << "Seek Sequence is" << endl;

for (int i = 0; i < seek_sequence.size(); i++) {

cout << seek_sequence[i] << endl;

    }

} int main() {    int arr[size] = {

176, 79, 34, 60,          92,

11, 41, 114 };    int head = 50;

string direction = "left";

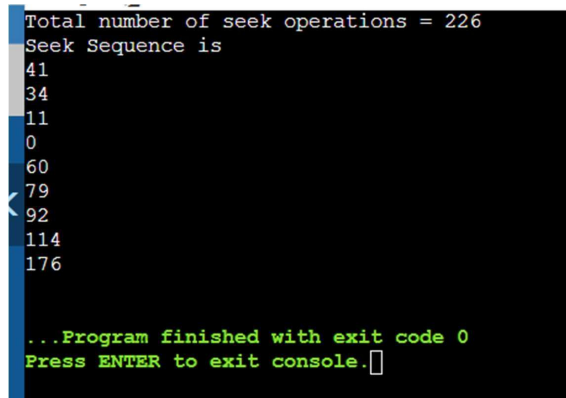
    SCAN(arr, head, direction);

    return 0;

}

```

Output:



```

Total number of seek operations = 226
Seek Sequence is
41
34
11
0
60
79
92
114
176

...Program finished with exit code 0
Press ENTER to exit console.

```