

# OS Project Disk Scheduling(Final Submission)

BY Amartya Vibhu 201413

Code:

```
#include<bits/stdc++.h>
using namespace std;
class FCFS
{
public:
void fcfs(int arr[], int head, int size)
{
int seek_count = 0;
int distance, cur_track;
for (int i = 0; i < size; i++)
{
cur_track = arr[i];
distance = abs(cur_track - head);
seek_count += distance;
head = cur_track;
}
cout << "Total number of seek operations = " <<
seek_count << endl;
cout << "Seek Sequence is" << endl;
for (int i = 0; i < size; i++)
{
cout << arr[i] << endl;
}
}
};
class SSTF
{
public:
void calculatedifference(int request[], int head,
int diff[][2], int size)
{
for(int i = 0; i < size; i++)
{
diff[i][0] = abs(head - request[i]);
}
}
int findMIN(int diff[][2], int size)
{
int index = -1;
int minimum = 1e9;
for(int i = 0; i < size; i++)
{
if (!diff[i][1] && minimum > diff[i][0])
{
minimum = diff[i][0];
}
```

```

index = i;
}
}
return index;
}
void sstf(int request[],int head, int size)
{
if (size == 0)
{
return;
}
int diff[size][2] = { { 0, 0 } };
int seekcount = 0;
int seeksequence[size + 1] = {0};
for(int i = 0; i < size; i++)
{
seeksequence[i] = head;
calculatedifference(request, head, diff,
size); int index = findMIN(diff, size);
diff[index][1] = 1;
seekcount += diff[index][0];
head = request[index];
}
seeksequence[size] = head;
cout << "Total number of seek operations =
" << seekcount << endl;
cout << "Seek sequence is : " << "\n";
for(int i = 0; i <= size; i++)
{
cout << seeksequence[i] << "\n";
}
}
};
class SCAN
{
public:
void scan(int arr[], int head, string direction, int size, int
disk_size)
{
int seek_count = 0;
int distance, cur_track;
vector<int> left, right;
vector<int> seek_sequence;
// appending end values
// which has to be visited
// before reversing the direction
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]);
}
// sorting left and right vectors
sort(left.begin(), left.end());
sort(right.begin(), right.end());
// run the while loop two times.
// one by one scanning right
// and left of the head
int run = 2;
while (run-- > 0) {
if (direction == "left") {
for (int i = left.size() - 1; i >= 0; i--) {
cur_track = left[i];
// appending current track to seek sequence
seek_sequence.push_back(cur_track);
distance = abs(cur_track - head);
// increase the total count
seek_count += distance;
// accessed track is now the new
head head = cur_track;
}
direction = "right";
}
else if (direction == "right") {
for (int i = 0; i < right.size(); i++) {
cur_track = right[i];
// appending current track to seek sequence
seek_sequence.push_back(cur_track); //
calculate absolute distance
distance = abs(cur_track - head);
seek_count += distance;
// accessed track is now new head
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;
}
}
};
class CSCAN{
public:
void cscan(int arr[], int head, int size, int disk_size)
{
int seek_count = 0;
int distance, cur_track;
vector<int> left, right;

```

```

vector<int> seek_sequence;
// appending end values
// which has to be visited
// before reversing the direction
left.push_back(0);
right.push_back(disk_size - 1);
// tracks on the left of the
// head will be serviced when
// once the head comes back
// to the beginning (left end).
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]);
}
// sorting left and right vectors
std::sort(left.begin(), left.end());
std::sort(right.begin(), right.end());
// first service the requests
// on the right side of the
// head.
for (int i = 0; i < right.size(); i++) {
    cur_track = right[i];
    // appending current track to seek sequence
    seek_sequence.push_back(cur_track);
    // calculate absolute distance
    distance = abs(cur_track - head);
    // increase the total count
    seek_count += distance;
    head = cur_track;
}
// once reached the right end
// jump to the beginning.
head = 0;
// adding seek count for head returning from 199 to 0
seek_count += (disk_size - 1);
// Now service the requests again
// which are left.
for (int i = 0; i < left.size(); i++) {
    cur_track = left[i];
    // appending current track to seek sequence
    seek_sequence.push_back(cur_track);
    // calculate absolute distance
    distance = abs(cur_track - head);
    // increase the total count
    seek_count += distance;
    // accessed track is now the new head
    head = cur_track;
}
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;
}
};
class CLOOK{
public:
void clook(int arr[], int head, int size)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;
    // Tracks on the left of the
    // head will be serviced when
    // once the head comes back
    // to the beginning (left end)
    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]);
    }
    // Sorting left and right vectors
    sort(left.begin(), left.end());
    sort(right.begin(), right.end());
    for (int i = 0; i < right.size(); i++) {
        cur_track = right[i];
        // Appending current track to seek sequence
        seek_sequence.push_back(cur_track);
        // Calculate absolute distance
        distance = abs(cur_track - head);
        // Increase the total count
        seek_count += distance;
        // Accessed track is now new head
        head = cur_track;
    }
    seek_count += abs(head - left[0]);
    head = left[0];
    // Now service the requests again
    // which are left
    for (int i = 0; i < left.size(); i++) {
        cur_track = left[i];
        // Appending current track to seek sequence
        seek_sequence.push_back(cur_track);
        // Calculate absolute distance
        distance = abs(cur_track - head);
        // Increase the total count
        seek_count += distance;
        // Accessed track is now the new head
        head = cur_track;
    }
    cout << "Total number of seek operations = "

```

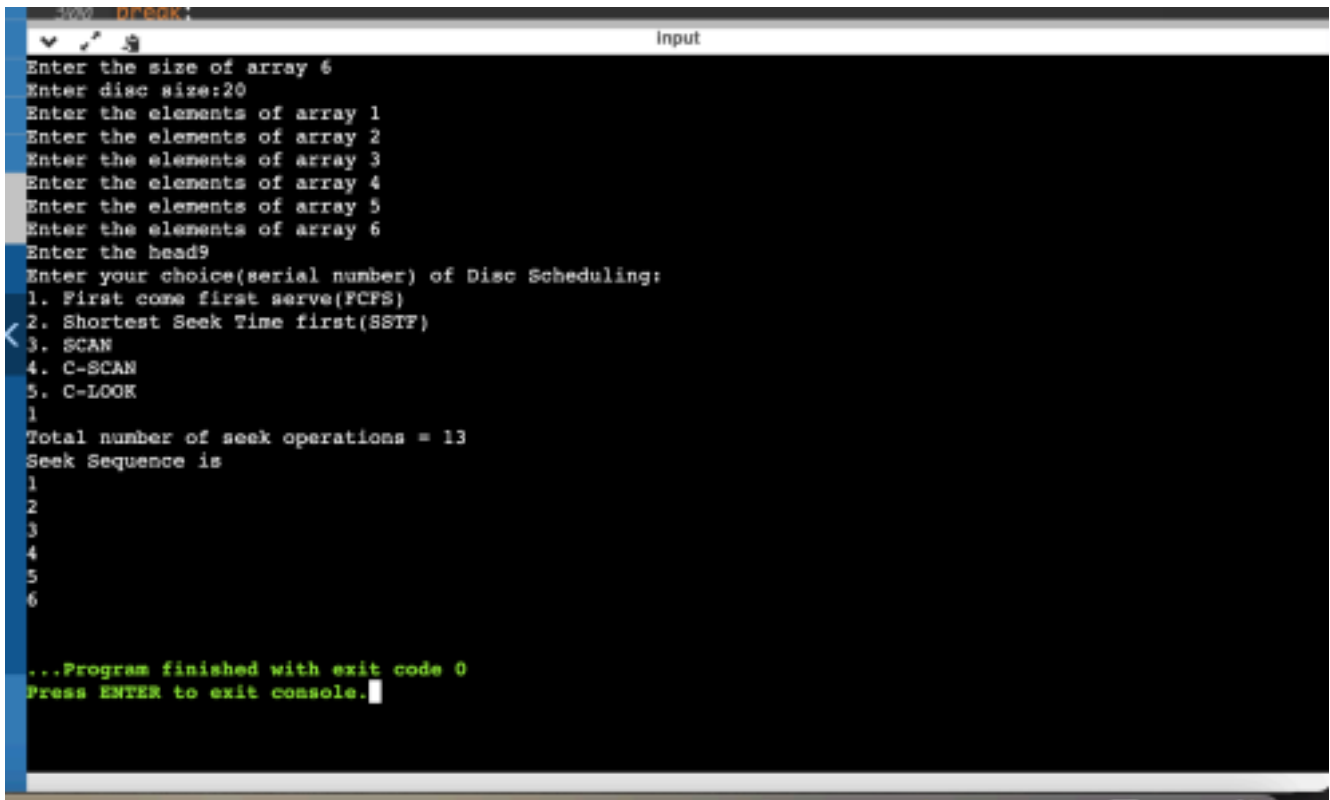
```

<< seek_count << endl;
cout << "Seek Sequence is" << endl;
for (int i = 0; i < seek_sequence.size(); i++) {
    cout << seek_sequence[i] << endl;
}
};
void menu()
{
}
int main()
{
    int size ,head, disk_size ;
    cout<<"Enter the size of array ";
    cin>>size;
    cout<<"Enter disc size:";
    cin>>disk_size;
    int arr[size];
    for(int i=0;i<size;i++)
    {
        cout<<"Enter the elements of array ";
        cin>>arr[i];
    }
    cout<<"Enter the head";
    cin>>head;
    string direction = "left";
    int inpt;
    cout<<"Enter your choice(serial number) of Disc
    Scheduling:"<<endl; cout<<"1. First come first serve(FCFS)"<<endl;
    cout<<"2. Shortest Seek Time first(SSTF)"<<endl;
    cout<<"3. SCAN "<<endl;
    cout<<"4. C-SCAN"<<endl;
    cout<<"5. C-LOOK"<<endl;
    cin>>inpt;
    switch (inpt) {
    case 1:
        FCFS obj01;
        obj01.fcfs(arr , head , size);
        break;
    case 2:
        SSTF obj02;
        obj02.sstf(arr, head, size);
        break;
    case 3:
        SCAN obj03;
        obj03.scan(arr, head , direction, size , disk_size);
        break;
    case 4:
        CSCAN obj04;
        obj04.cscan(arr, head, size, disk_size);
        break;
    case 5:
        CLOOK obj05;

```

```
obj05.clook(arr, head, size);
break;
default:
cout<<"Please fill the correct choices(1/2/3/4/5)";
menu();
break;
}
return 0;
}
```

## OUTPUT



```
input
Enter the size of array 6
Enter disc size:20
Enter the elements of array 1
Enter the elements of array 2
Enter the elements of array 3
Enter the elements of array 4
Enter the elements of array 5
Enter the elements of array 6
Enter the head9
Enter your choice(serial number) of Disc Scheduling:
1. First come first serve(FCFS)
2. Shortest Seek Time first(SSTF)
3. SCAN
4. C-SCAN
5. C-LOOK
1
Total number of seek operations = 13
Seek Sequence is
1
2
3
4
5
6

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