SYSTEM SOFTWARE LAB
EXPERIMENT 2 CYCLE I

September 8, 2020

AMEYA LIZBETH JAMES
S5 CSE,ROLL NO:8,REG:TVE18CS008
College of Engineering , Trivandrum
Department of Computer Science
Engineering

Contents

0.1 PROGRAM 1

Simulate the following disk scheduling algorithms.
a) FCFS

0.1.1 AIM

To find the seek time and sequence for FCFS disc scheduling algorithm

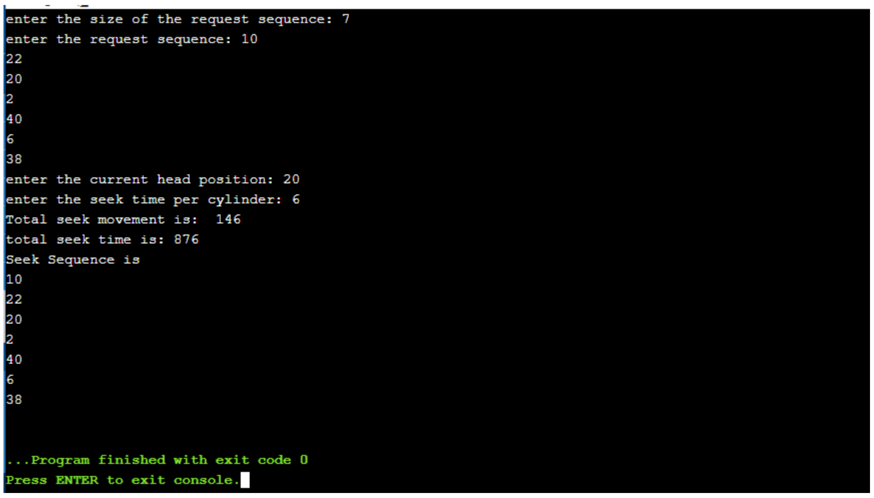
0.1.2 ALGORITHM

```
1.Let Request array represents an array storing indexes of tracks that have been
requested in ascending order of time of arrival.‘head’ is 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.
```

0.1.3 PROGRAM CODE

```
#include <bits/stdc++.h>
using namespace std;




void FCFS(int arr[], int head, int size, int t)
{
int seek_count = 0;
int distance, cur_track;

for (int i = 0; i < size; i++) {
cur_track = arr[i];
// calculate absolute distance
distance = abs(cur_track - head);

// increase the total count
seek_count += distance;

// accessed track is now new head
head = cur_track;
}

cout << "Total seek movement is:  "
<< seek_count << endl;
    cout<<"total seek time is: "<<t*seek_count<<endl;

// Seek sequence would be the same
// as request array sequence
cout << "Seek Sequence is" << endl;

for (int i = 0; i < size; i++) {
cout << arr[i] << endl;
}
}

// Driver code
int main()
{
    int size,head,t;
    std::cout<<"enter the size of the request sequence: ";
    std::cin>>size;
    int arr[size];
    std::cout<<"enter the request sequence: ";
    for(int i=0;i<size;i++)
    {
        std::cin>>arr[i];
    }
    std::cout<<"enter the current head position: ";
    std::cin>>head;
    std::cout<<"enter the seek time per cylinder: "
    std::cin>>t;

FCFS(arr, head, size,t);

return 0;
}
```

0.1.4 OUTPUT



```
enter the size of the request sequence: 7
enter the request sequence: 10
22
20
2
40
6
38
enter the current head position: 20
enter the seek time per cylinder: 6
Total seek movement is:  146
total seek time is: 876
Seek Sequence is
10
22
20
2
40
6
38


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

0.1.5 RESULT

The following code was executed and output was obtained.

0.2 PROGRAM 2

```
Simulate the following disk scheduling algorithms.
a) SCAN
```

0.2.1 AIM

To find the seek time and sequence for SCAN disc scheduling algorithm

0.2.2 ALGORITHM

```
1. Let Request array represents an array storing indexes of tracks that have been
  requested in ascending order of time of arrival.‘head’ is position of disk head.
2.Let direction represents whether the head is moving towards left or right.
3.In the direction in which head is moving service all tracks one by one.
4.Calculate the absolute distance of the track from the head.
5.Increment the total seek count with this distance.
6.Currently serviced track position now becomes the new head position.
7.Go to step 3 until we reach at one of the ends of the disk.
8.If we reach at the end of the disk reverse the direction and go to step 2 until
  all tracks in request array have not been serviced.
```

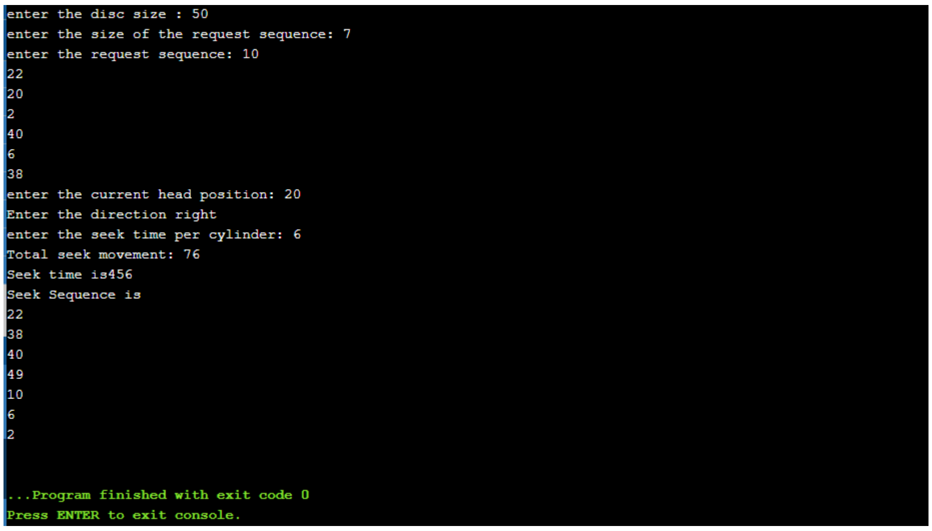
0.2.3 PROGRAM CODE

```
#include <bits/stdc++.h>
using namespace std;



void SCAN(int arr[], int head,int size,int disk_size, string direction,int t)
{
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
std::sort(left.begin(), left.end());
std::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--) {
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);

// 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;
}
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);

// increase the total count
seek_count += distance;

// accessed track is now new head
head = cur_track;
}
direction = "left";
}
}

cout << "Total seek movement: "
<< seek_count << endl;
    cout << "Seek time is" << seek_count*t<<endl;
cout << "Seek Sequence is" << endl;


for (int i = 0; i < seek_sequence.size(); i++) {
cout << seek_sequence[i] << endl;
}
}
```

```
// Driver code
int main()
{
    int size,head,disc_size,t;
    std::cout<<"enter the disc size : ";
    std::cin>>disc_size;
    std::cout<<"enter the size of the request sequence: ";
    std::cin>>size;
    int arr[size];
    std::cout<<"enter the request sequence: ";
    for(int i=0;i<size;i++)
    {
        std::cin>>arr[i];
    }
    std::cout<<"enter the current head position: ";
    std::cin>>head;
    string str;
    std::cout << "Enter the direction ";
    std::cin>>str;

    std::cout<<"enter the seek time per cylinder: ";
    std::cin>>t;


    SCAN(arr, head,size,disc_size, str,t);


return 0;
}
```

0.2.4 OUTPUT



```
enter the disc size : 50
enter the size of the request sequence: 7
enter the request sequence: 10
22
20
2
40
6
38
enter the current head position: 20
Enter the direction right
enter the seek time per cylinder: 6
Total seek movement: 76
Seek time is456
Seek Sequence is
22
38
40
49
10
6
2


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

0.2.5 RESULT

The following code was executed and output was obtained.

0.3 PROGRAM 3

```
Simulate the following disk scheduling algorithms.
a)C-SCAN
```

0.3.1 AIM

To find the seek time and sequence for C-SCAN disc scheduling algorithm

0.3.2 ALGORITHM

```
1.Let Request array represents an array storing indexes of tracks that have been
  requested in ascending order of  time of arrival.‘head’ is position of disk head.
2.The head services only in the right direction from 0 to size of the disk.
3.While moving in the left direction do not service any of the tracks.
4.When we reach at the beginning(left end) reverse the direction.
5.While moving in right direction it services all tracks one by one.
```

```
6.While moving in right direction calculate the absolute
  distance of the track from the head.
7.Increment the total seek count with this distance.
8.Currently serviced track position now becomes the new head position.
9.Go to step 6 until we reach at right end of the disk.
10.If we reach at the right end of the disk reverse the direction and go to step 3
  until all tracks in request array have not been serviced
```

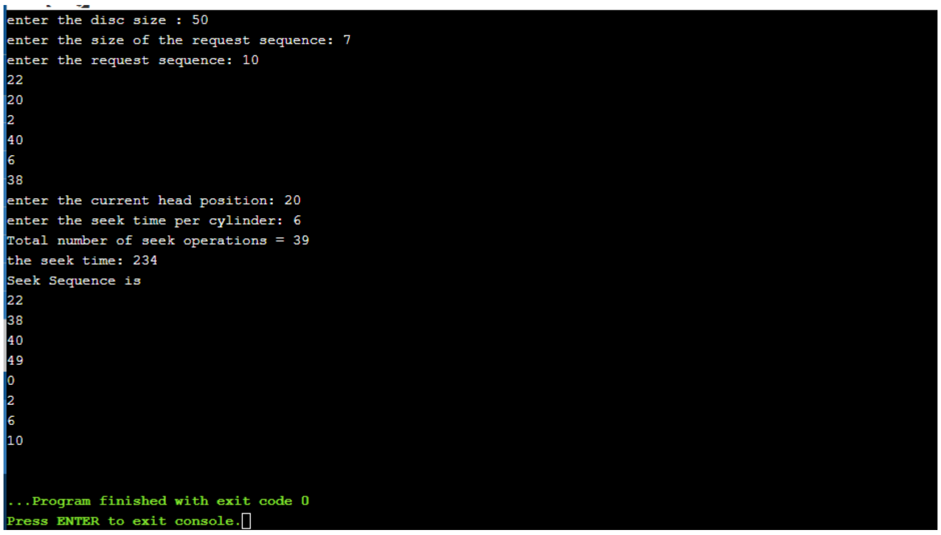
0.3.3 PROGRAM CODE

```
#include <bits/stdc++.h>
using namespace std;





void CSCAN(int arr[], int head,int size,int disk_size,int t)
{
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 beggining (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;

// accessed track is now new head
head = cur_track;
}

// once reached the right end
// jump to the beggining.
head = 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;
    std::cout<<"the seek time: "<<t*seek_count<<endl;
cout << "Seek Sequence is" << endl;

for (int i = 0; i < seek_sequence.size(); i++) {
cout << seek_sequence[i] << endl;
}
}

// Driver code


// Driver code
int main()
{
    int size,head,disc_size,t;
    std::cout<<"enter the disc size : ";
    std::cin>>disc_size;
    std::cout<<"enter the size of the request sequence: ";
    std::cin>>size;
    int arr[size];
    std::cout<<"enter the request sequence: ";
    for(int i=0;i<size;i++)
    {
        std::cin>>arr[i];
    }
    std::cout<<"enter the current head position: ";
    std::cin>>head;
    string str;


    std::cout<<"enter the seek time per cylinder: ";
    std::cin>>t;


    CSCAN(arr, head,size,disc_size,t);


return 0;
}
```

0.3.4 OUTPUT



```
enter the disc size : 50
enter the size of the request sequence: 7
enter the request sequence: 10
22
20
2
40
6
38
enter the current head position: 20
enter the seek time per cylinder: 6
Total number of seek operations = 39
the seek time: 234
Seek Sequence is
22
38
40
49
0
2
6
10


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

0.3.5 RESULT

The following code was executed and output was obtained.