Saurav M. Hiremath
18BCB0082

Exercise X

17/10/19

Disk Scheduling Algorithms

Q1 C++code for FCFS disk scheduling algorithm.

A.

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

int size = 8;

void FCFS (int arr[], int head)
{
    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;
```

PTO →

```
for (int i=0; i<size; i++) {
    cout << arr[i] << endl;
}
}
int main()
{
    int arr[size] = {176, 79, 34, 60, 92, 11,
    41, 114};
    int head = 50;
    FCFS(arr, head);
    return 0;
}
```

Q2. Python program for SSTF disk scheduling algorithm.

A.

```
def calculateDiff(queue, head, diff):
    for i in range(len(diff)):
        diff[i][0] = abs(queue[i] - head)
```

PTO →

```
def findMin(diff):
    index = -1
    min = 99999999
    for i in range(len(diff)):
        if (not diff[i][1] and min>
        diff[i][0]):
            min = diff[i][0]
            index = i
    return index

def shortestSeekTimeFirst(request, head):
    if (len(request) == 0):
        return
    l = len(request)
    diff = [0] * 1
    for i in range(l):
        diff[i] = [0, 0]

    seek_count = 0
    seek_sequence = [0] * (l+1)
```

PTO →

```
    for i in range (l):
        seek_sequence[i] = head
        calculateDiff (request, head, diff)
        index = findMin (diff)
        diff[index][1] = True
        seek_count += diff[index][0]
        head = request[index]
    seek_sequence[len(seek_sequence)-1]=
    head
    print ("Total number of seek operations=",
    seek_count)
    print ("Seek Sequence is")
    for i in range (l+1):
        print (seek_sequence[i])

if __name__ == "__main__":
    proc = [176, 79, 34, 60, 92, 11, 41, 114]
    shortestSeekTimeFirst (proc, 50)
```

PTO
→

Q3 C++ code for SCAN disk scheduling algorithm

A.

```
#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]);
    }
```

PTO →

```
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--){
            curtrack = 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";
    }
}
```

PTO →

```
    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;
}
```

PTO →

Q4 C++ code for C-SCAN disk scheduling algorithm

A.

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

int size = 8;
int disk_size = 200;

void CSCAN(int arr[], int head)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;
    left.push_back(0);
    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());
```

PTO →

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

PTO →

```
int main()
{
    int arr[size] = { 176, 79, 34, 60, 92, 11, 41,
                      114 };
    int head = 50;
    cout << "Initial position of head: " << head
    << endl;
    CSCAN(arr, head);
    return 0;
}
```

Q_5 C++ code for LOOK disk scheduling problem.

A.

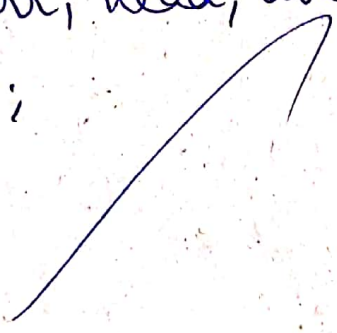
```
#include <bits/stdc++.h>
using namespace std;

int size = 8;
int disk_size = 200;

void LOOK( int arr[], int head, string direction)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;
```

PTO →

```
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";
    }
}
```

PTO →

```
    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 = "right";
    cout << "Initial position of head" << head
        << endl;
    LOOK (arr, head, direction);
    return 0;
}
```



PTO

→

Q6 C++ program for C-LOOK disk scheduling algorithm

A.

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

int size = 8;
int disk_size = 200;

void CLOOK (int arr[], int head)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;
    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()); ;
    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;
    }
```

PTO →

```
seek_count += abs(head - left[0]);
head = left[0];
for(int i=0; i < left.size(); i++){
    cur_track = left[i];
    seek_sequence.push_back(cur_track);
    distance = abs(cur_track - head);
    seek_count += distance;
    head = cur_track;
}
cout << "Total number of seek operations = "
     << seek_count << endl;
cout << "Sequence is " << endl;
for(int i=0; i < seek_sequence.size(); i++)
{
    cout << seek_sequence[i] << endl;
}
}
```

8.5

17/10/19

```
int main()
{
    int arr[size] = {176, 79, 34, 60, 92, 11,
    41, 114};
    int head = 50
    cout << "Initial position of head: " << head << endl;
    CLOOK(arr, head)
    return 0;
}
```