Operating System (CS301)

Assignment - 9

U19CS012

1.) To Implement Shortest Seek Time First (SSTF) Disk Scheduling Algorithm.

Basic Idea: The tracks which are **closer to current disk head** position should be serviced first in order to *Minimise the seek* operations.

Code

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

// For Proper Presentation
void dash(char ch, int freq);

// To Print the Answer [Disk Request] & Total Seek Time
void print(vector<int> &answer, int totalSeekTime);

// Shortest Seek Time First (SSTF) Disk Scheduling Algorithm Implementation
void SSTF(vector<int> requests, int n, int head);

int main()
{
    int n;
    cout << "Enter the number of Disk Access request : ";
    cin >> n;

    vector<int> requests(n);
    cout << "Enter the index of Tracks : ";

    for (int i = 0; i < n; i++)
        cin >> requests[i];

    int head;
    cout << "Enter Disk head position : ";
    cin >> head;
    SSTF(requests, n, head);
    return 0;
}

// For Proper Presentation
void dash(char ch, int freq)
{
    for (int i = 0; i < freq; i++)
```

```
        cout << ch;
}

// To Print the Answer [Disk Request] & Total Seek Time
void print(vector<int> &answer, int totalSeekTime)
{
    cout << endl;
    dash('-', 100);
    cout << endl;

    cout << "Disk request will be served as : ";

    for (auto &x : answer)
        cout << x << " ";

    cout << "\nTotal Seek Time : " << totalSeekTime;

    cout << endl;
    dash('-', 100);
    cout << endl;
}

// Shortest Seek Time First (SSTF) Disk Scheduling Algorithm Implementation
void SSTF(vector<int> requests, int n, int head)
{
    int totalSeekTime = 0;

    auto ite = requests.begin();
    vector<int> answer;
    answer.push_back(head);

    while (n)
    {
        int min_seek_time = INT_MAX;

        auto itr = requests.begin();
        auto ind = requests.begin();

        while (itr != requests.end())
        {
            int dis = abs(head - (*itr));
            if (dis < min_seek_time)
            {
                min_seek_time = dis;
                ind = itr;
            }
            itr++;
        }

        head = (*ind);
```

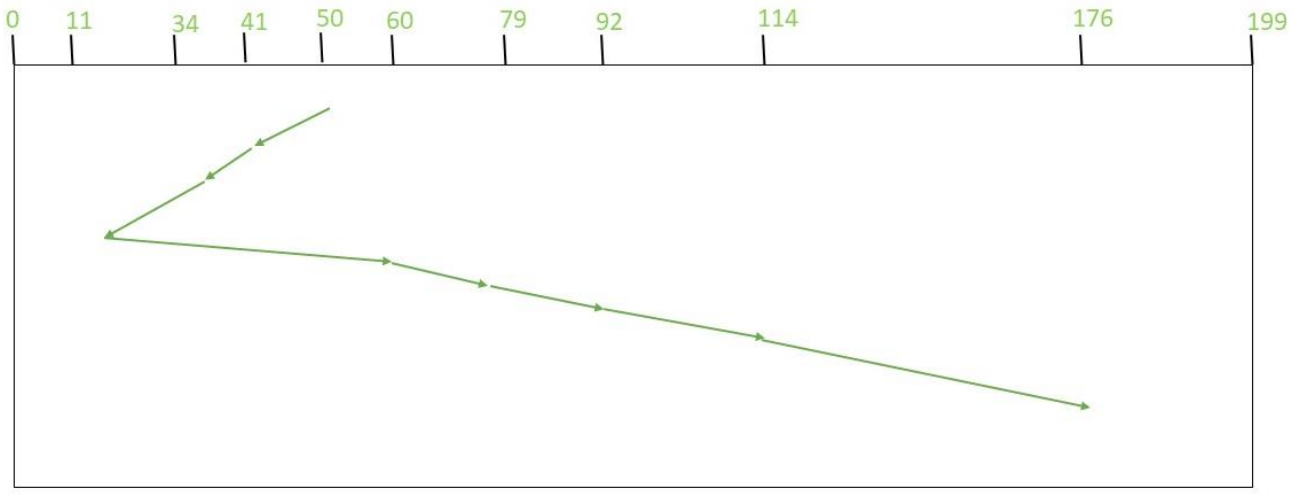
```
        answer.push_back(head);
        requests.erase(ind);

        totalSeekTime += min_seek_time;

        n--;
    }

    print(answer, totalSeekTime);
}
```

Test Case

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

Initial head position = 50

```
Enter the number of Disk Access request : 8
Enter the index of Tracks : 176 79 34 60 92 11 41 114
Enter Disk head position : 50

-----------------------------------------------------------------
Disk request will be served as : 50 41 34 11 60 79 92 114 176
Total Seek Time : 204
-----------------------------------------------------------------
```



Therefore, total seek count is calculated as:

```
= (50-41)+(41-34)+(34-11)+(60-11)+(79-60)+(92-79)+(114-92)+(176-114)
= 204
```

Theoratical Answer

2.) To Implement SCAN (Elevator) Algorithm for Disk Scheduling Algorithm.

- ✓ Head starts from one end of the disk and moves towards the other end, servicing requests in between one by one and reach the other end.
- ✓ Then the *direction of the head is reversed* and the process continues as head continuously scan back and forth to access the disk. So, this algorithm works as an elevator and hence also known as the **elevator algorithm**.
- ✓ As a result, the requests at the midrange are **serviced more** and those arriving behind the disk arm will have to wait.

Code

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

// For Proper Presentation
void dash(char ch, int freq);

// To Print the Answer [Disk Request] & Total Seek Time
void print(vector<int> &answer, int totalSeekTime);

// SCAN Algorithm for Disk Scheduling Algorithm Implementation
void SCAN(vector<int> requests, int n, int head);

int main()
{
    int n;
    cout << "Enter the number of Disk Access request : ";
    cin >> n;

    vector<int> requests(n);
    cout << "Enter the index of Tracks : ";
    for (int i = 0; i < n; i++)
        cin >> requests[i];

    int head;
    cout << "Enter Disk head position : ";
    cin >> head;

    SCAN(requests, n, head);

    return 0;
}

// For Proper Presentation
void dash(char ch, int freq)
{
```

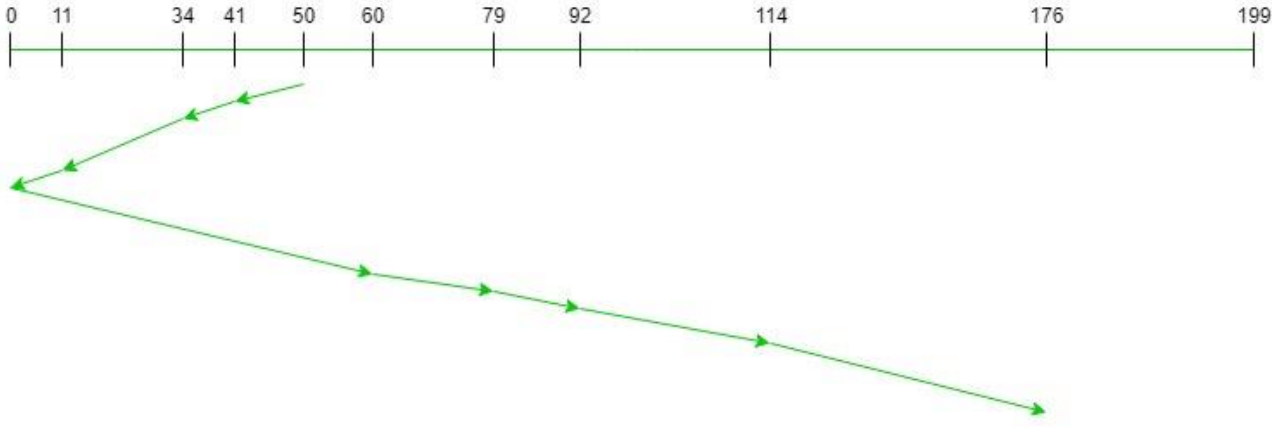
```
    for (int i = 0; i < freq; i++)
        cout << ch;
}

// To Print the Answer [Disk Request] & Total Seek Time
void print(vector<int> &answer, int totalSeekTime)
{
    cout << endl;
    dash('-', 100);
    cout << endl;

    cout << "Disk request will be served as : ";

    for (auto &x : answer)
        cout << x << " ";

    cout << "\nTotal Seek Time : " << totalSeekTime;

    cout << endl;
    dash('-', 100);
    cout << endl;
}

// SCAN Algorithm for Disk Scheduling Algorithm Implementation
void SCAN(vector<int> requests, int n, int head)
{
    int totalSeekTime = 0;

    sort(requests.begin(), requests.end());

    vector<int> answer;
    int ind = -1;
    if (head <= requests[0])
    {
        ind = 0;
    }

    for (int i = 0; i < n - 1 and ind == -1; i++)
    {
        if (requests[i] <= head and head < requests[i + 1])
        {
            ind = i;
            break;
        }
    }

    if (ind == -1)
    {
        ind = n - 1;
    }
```

```
    for (int i = ind; i >= 0; i--)
    {
        answer.push_back(requests[i]);
        totalSeekTime += abs(head - requests[i]);
        head = requests[i];
    }

    totalSeekTime += head;
    head = 0;
    answer.push_back(head);
    for (int i = ind + 1; i < n; i++)
    {
        answer.push_back(requests[i]);
        totalSeekTime += abs(head - requests[i]);
        head = requests[i];
    }

    print(answer, totalSeekTime);
}
```

Test Case

```
Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}
Initial head position = 50
Direction = left (We are moving from right to left)
```

```
Enter the number of Disk Access request : 8
Enter the index of Tracks : 176 79 34 60 92 11 41 114
Enter Disk head position : 50

-----------------------------------------------------------------
Disk request will be served as : 41 34 11 0 60 79 92 114 176
Total Seek Time : 226
-----------------------------------------------------------------
```



Therefore, the total seek count is calculated as:

```
= (50-41)+(41-34)+(34-11)
 +(11-0)+(60-0)+(79-60)
 +(92-79)+(114-92)+(176-114)
= 226
```

SUBMITTED BY:

U19CS012

BHAGYA VINOD RANA