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

*#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;

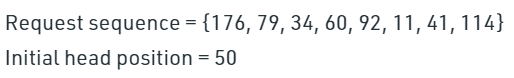
        n--;

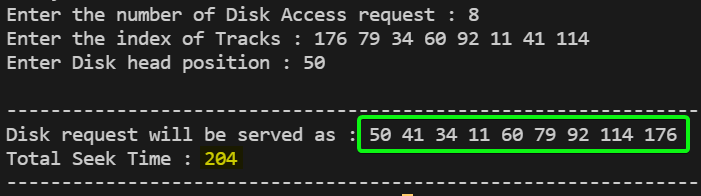
    }

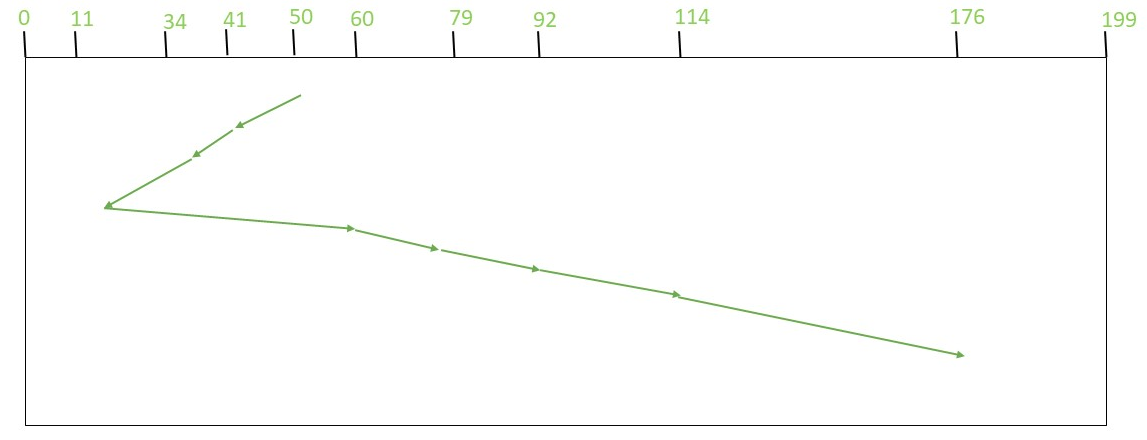
    print(answer, totalSeekTime);

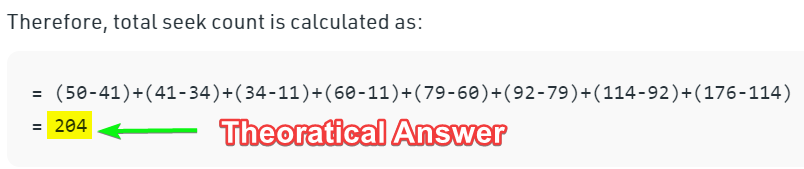
}

Test Case









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

*#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]);

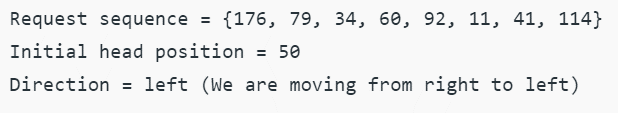
        head = requests[i];

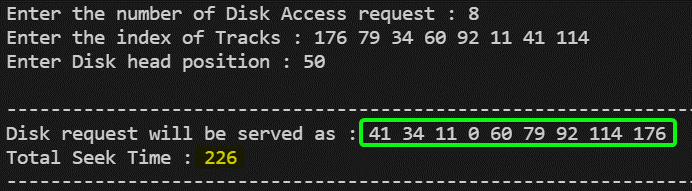
    }

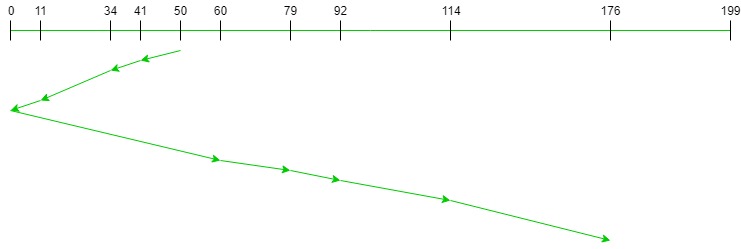
    print(answer, totalSeekTime);

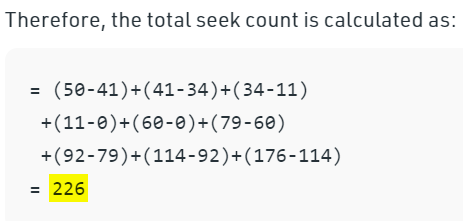
}

Test Case









SUBMITTED BY:

**U19CS012**

BHAGYA VINOD RANA