DISK SCHEDULING SYSTEM

A Project Report Submitted in partial fulfilment of the

Requirements for the award of the Degree of

## BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

# V. SAI SUDEEP REDDY 1602-22-733-045

**S. HARSHA VARDHAN** **1602-22-733-011**

Under the guidance of

**Dr. D. BASWARAJ**

Professor. CSE



**Department of Computer Science & Engineering**] **Vasavi College of Engineering (Autonomous)**

(Affiliated to Osmania University)

**Ibrahimbagh, Hyderabad-31**

**2024**

**Vasavi College of Engineering (Autonomous)**

(Affiliated to Osmania University)

Hyderabad- 500 031

Department of Computer Science & Engineering



## DECLARATION BY THE CANDIDATE

We, **V. SAI SUDEEP REDDY** and **S. HARSHA**

**VARDHAN,** bearing hall ticket numbers, **1602-22-733-045** and **1602-22-733-011** hereby declare that the project report entitled “ **DISK SCHEDULING SYSTEM”** under the guidance of **DR.BASAWARAJ A DONAGAPURE,** PROFESSOR, Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering**.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**V. SAI SUDEEP REDDY (1602-22-733-045)**

**S. HARSHA VARDHAN (1602-22-733-011)**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Hyderabad-500 031**

**Department of Computer Science & Engineering**



**BONAFIDE CERTIFICATE**

This is to certify that the project entitled “**DISK SCHEDULING SYSTEM**” being submitted by **V. SAI SUDEEP REDDY** and **S. HARSHA VARDHAN**, bearing **1602-22-733-045** and **1602-22-733-011** in

partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

**Dr. D. BASAWARAJ DR.T .Adilakshmi**

**Professor. Professor & HOD**

Dept.of CSE.

## ACKNOWLEDGEMENT

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Dr. D. BASAWARAJ,** Department of Computer Science and Engineering, whose guidance was unforgettable and filled with innovative ideas as well as her constructive suggestions has made the presentation of my major project a grand success.

We are thankful to **Dr. T. Adilakshmi,** Head of Department (CSE), **Vasavi College of Engineering** for the help during our course work.

Finally we express our gratitude to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete our project work successfully.

## ABSTRACT

Disk scheduling algorithms play a crucial role in optimizing the performance of disk I/O operations in computer systems. These algorithms aim to minimize the seek time, which is the time required for the disk arm to move to the desired location on the disk. Various disk scheduling algorithms, such as FCFS, SSTF, SCAN, C-SCAN, and LOOK, employ different strategies to prioritize disk requests and reduce seek time. Each algorithm has its unique approach, advantages, and limitations, making them suitable for different types of workloads and system configurations.

Understanding and implementing disk scheduling algorithms are essential for enhancing system efficiency and improving overall performance.

## TABLE OF CONTENTS

1. INTRODUCTION………………………………………………………1
   1. OVERVIEW
   2. PROJECT SCOPE
2. SYSTEM REQUIREMENTS…………………………………………...2
   1. SOFTWARE REQUIREMENTS
   2. HARDWARE REQUIREMENTS
3. IMPLEMENTATION…………………………………………………...3
   1. PROJECT CODE
4. OUTPUTS OF PROJECT CODE……………………………………….9
5. CONCLUSION…………………………………………………………14
6. REFERENCES………………………………………………………….15

**.**

## INTRODUCTION

Disk scheduling algorithms are fundamental components of operating systems responsible for managing disk I/O operations efficiently. In computer systems, disk drives are essential for storing and retrieving data, but the mechanical nature of disk drives introduces latency due to the time required for the disk arm to move to the desired location on the disk. This latency, known as seek time, significantly impacts the overall performance of disk I/O operations.

Disk scheduling algorithms aim to minimize seek time and optimize the utilization of disk resources by determining the order in which disk requests are serviced. These algorithms prioritize disk requests based on various criteria, such as the current position of the disk arm, the distance to the requested track, and the direction of movement.

There are several disk scheduling algorithms, each with its own approach to minimizing seek time and improving disk performance. Some of the commonly used disk scheduling algorithms include First- Come, First-Served (FCFS), Shortest Seek Time First (SSTF), SCAN, Circular SCAN (C-SCAN), LOOK, and C-LOOK.

## SYSTEM REQUIREMENTS

1. **Software Requirements:**

The major software requirements of our project include:

* 1. Operating system**:** Windows or any other
  2. Tools: J N U nano
  3. Language: c

1. **Hardware Requirements:**

The hardware requirements that preferably map towards the software are as follows:

* 1. RAM : 4GB
  2. Processor: Intel Core I5

1. **IMPLEMENTATION:**

**Project code:**

#include <stdio.h> #include <stdlib.h> #include <stdbool.h> #include <limits.h>

// Function prototypes

void FCFS(int requests[], int n, int head); void SSTF(int requests[], int n, int head);

void SCAN(int requests[], int n, int head, int disk\_size); void C\_SCAN(int requests[], int n, int head, int disk\_size); void LOOK(int requests[], int n, int head);

int findClosestRequest(int requests[], bool serviced[], int n, int head); int main() {

int n, head, disk\_size;

int algorithm\_choice;

printf("Enter the number of disk requests: "); scanf("%d", &n);

int requests[n];

printf("Enter the size of the disk: "); scanf("%d", &disk\_size);

printf("Enter the disk requests:\n"); for (int i = 0; i < n; i++) {

printf("Request %d: ", i + 1); scanf("%d", &requests[i]);

if (requests[i] >= disk\_size || requests[i] < 0) {

printf("Invalid request. Request must be less than disk size and non- negative.\n");

i--; // Reprompt for the same index

}

}

printf("Enter the initial head position: "); scanf("%d", &head);

printf("Select Disk Scheduling Algorithm:\n"); printf("1. FCFS\n");

printf("2. SSTF\n");

printf("3. SCAN\n");

printf("4. C-SCAN\n");

printf("5. LOOK\n"); printf("Enter your choice: "); scanf("%d", &algorithm\_choice);

switch (algorithm\_choice) { case 1:

FCFS(requests, n, head); break;

case 2:

SSTF(requests, n, head); break;

case 3:

SCAN(requests, n, head, disk\_size); break;

case 4:

C\_SCAN(requests, n, head, disk\_size); break;

case 5:

LOOK(requests, n, head); break;

default:

printf("Invalid choice!\n");

}

return 0;

}

// FCFS Disk Scheduling Algorithm

void FCFS(int requests[], int n, int head) { int total\_seek\_time = 0;

printf("FCFS Disk Scheduling:\n"); for (int i = 0; i < n; i++) {

printf("Servicing request at: %d\n", requests[i]); total\_seek\_time += abs(requests[i] - head); head = requests[i];

}

printf("Total seek time: %d\n\n", total\_seek\_time);

}

// SSTF Disk Scheduling Algorithm

int findClosestRequest(int requests[], bool serviced[], int n, int head) {

int min\_distance = INT\_MAX; int closest\_request\_index = -1; for (int i = 0; i < n; i++) {

if (!serviced[i] && abs(requests[i] - head) < min\_distance) { min\_distance = abs(requests[i] - head); closest\_request\_index = i;

}

}

return closest\_request\_index;

}

void SSTF(int requests[], int n, int head) { int total\_seek\_time = 0;

bool serviced[n];

for (int i = 0; i < n; i++) serviced[i] = false;

printf("SSTF Disk Scheduling:\n"); for (int i = 0; i < n; i++) {

int closest\_request\_index = findClosestRequest(requests, serviced, n, head);

serviced[closest\_request\_index] = true;

printf("Servicing request at: %d\n", requests[closest\_request\_index]); total\_seek\_time += abs(requests[closest\_request\_index] - head); head = requests[closest\_request\_index];

}

printf("Total seek time: %d\n\n", total\_seek\_time);

}

// SCAN Disk Scheduling Algorithm

void SCAN(int requests[], int n, int head, int disk\_size) { int total\_seek\_time = 0;

int i, j;

int sorted\_requests[n + 1];

for (i = 0; i < n; i++) sorted\_requests[i] = requests[i]; sorted\_requests[n] = head;

n++;

for (i = 0; i < n - 1; i++) {

for (j = 0; j < n - i - 1; j++) {

if (sorted\_requests[j] > sorted\_requests[j + 1]) { int temp = sorted\_requests[j]; sorted\_requests[j] = sorted\_requests[j + 1]; sorted\_requests[j + 1] = temp;

}

}

}

int head\_index = 0; for (i = 0; i < n; i++) {

if (sorted\_requests[i] == head) { head\_index = i;

break;

}

}

printf("SCAN Disk Scheduling:\n"); for (i = head\_index; i < n; i++) {

printf("Servicing request at: %d\n", sorted\_requests[i]); total\_seek\_time += abs(sorted\_requests[i] - head); head = sorted\_requests[i];

}

for (i = head\_index - 1; i >= 0; i--) {

printf("Servicing request at: %d\n", sorted\_requests[i]); total\_seek\_time += abs(sorted\_requests[i] - head); head = sorted\_requests[i];

}

printf("Total seek time: %d\n\n", total\_seek\_time);

}

// C-SCAN Disk Scheduling Algorithm

void C\_SCAN(int requests[], int n, int head, int disk\_size) { int total\_seek\_time = 0;

int i, j;

int sorted\_requests[n + 2];

for (i = 0; i < n; i++) sorted\_requests[i] = requests[i]; sorted\_requests[n] = head;

sorted\_requests[n + 1] = disk\_size - 1; n += 2;

for (i = 0; i < n - 1; i++) {

for (j = 0; j < n - i - 1; j++) {

if (sorted\_requests[j] > sorted\_requests[j + 1]) { int temp = sorted\_requests[j];

sorted\_requests[j] = sorted\_requests[j + 1]; sorted\_requests[j + 1] = temp;

}

}

}

int head\_index = 0; for (i = 0; i < n; i++) {

if (sorted\_requests[i] == head) { head\_index = i;

break;

}

}

printf("C-SCAN Disk Scheduling:\n"); for (i = head\_index; i < n; i++) {

printf("Servicing request at: %d\n", sorted\_requests[i]); total\_seek\_time += abs(sorted\_requests[i] - head); head = sorted\_requests[i];

}

for (i = 0; i < head\_index; i++) {

printf("Servicing request at: %d\n", sorted\_requests[i]); total\_seek\_time += abs(sorted\_requests[i] - head); head = sorted\_requests[i];

}

printf("Total seek time: %d\n\n", total\_seek\_time);

}

// LOOK Disk Scheduling Algorithm

void LOOK(int requests[], int n, int head) { int total\_seek\_time = 0;

int i, j;

int sorted\_requests[n + 1];

for (i = 0; i < n; i++) sorted\_requests[i] = requests[i]; sorted\_requests[n] = head;

n++;

for (i = 0; i < n - 1; i++) {

for (j = 0; j < n - i - 1; j++) {

if (sorted\_requests[j] > sorted\_requests[j + 1]) {

int temp = sorted\_requests[j]; sorted\_requests[j] = sorted\_requests[j + 1]; sorted\_requests[j + 1] = temp;

}

}

}

int head\_index = 0; for (i = 0; i < n; i++) {

if (sorted\_requests[i] == head) { head\_index = i;

break;

}

}

printf("LOOK Disk Scheduling:\n"); for (i = head\_index; i < n; i++) {

printf("Servicing request at: %d\n", sorted\_requests[i]); total\_seek\_time += abs(sorted\_requests[i] - head); head = sorted\_requests[i];

}

for (i = head\_index - 1; i >= 0; i--) {

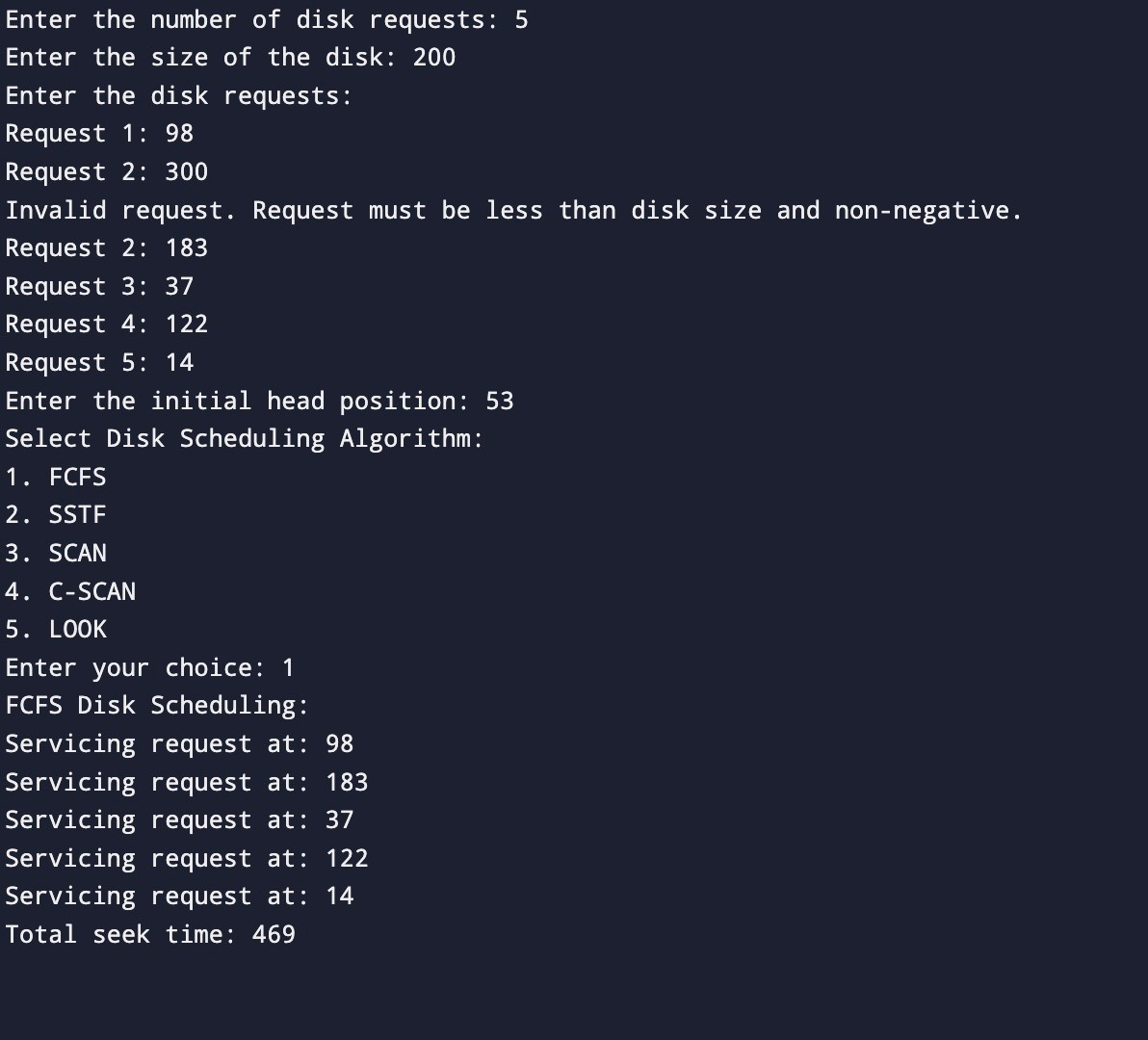
printf("Servicing request at: %d\n", sorted\_requests[i]); total\_seek\_time += abs(sorted\_requests[i] - head); head = sorted\_requests[i];

}

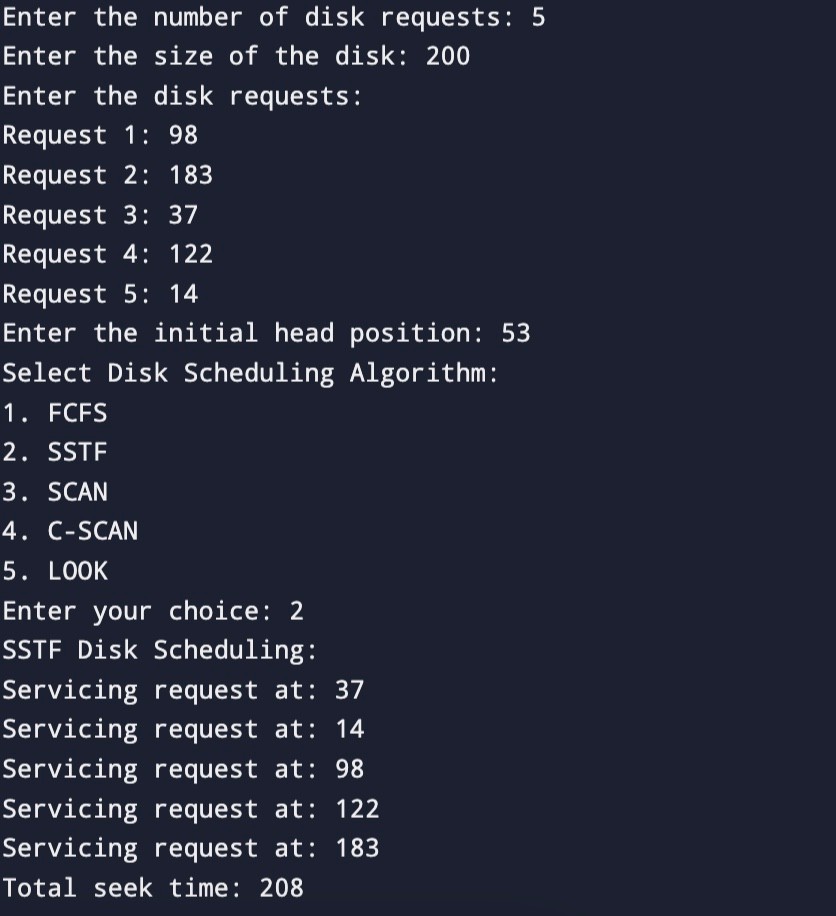
printf("Total seek time: %d\n\n", total\_seek\_time

# OUTPUTS / SCREENSHOTS:

SAMPLE OUTPUT 1:



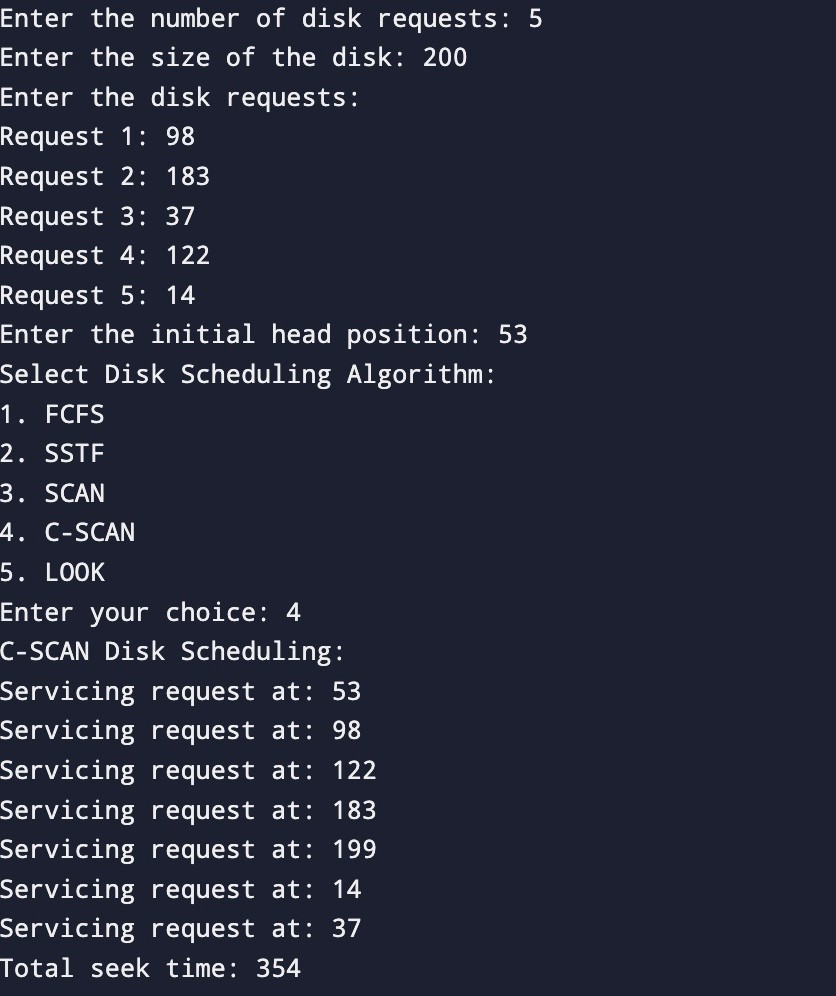
SAMPLE OUTPUT 2:



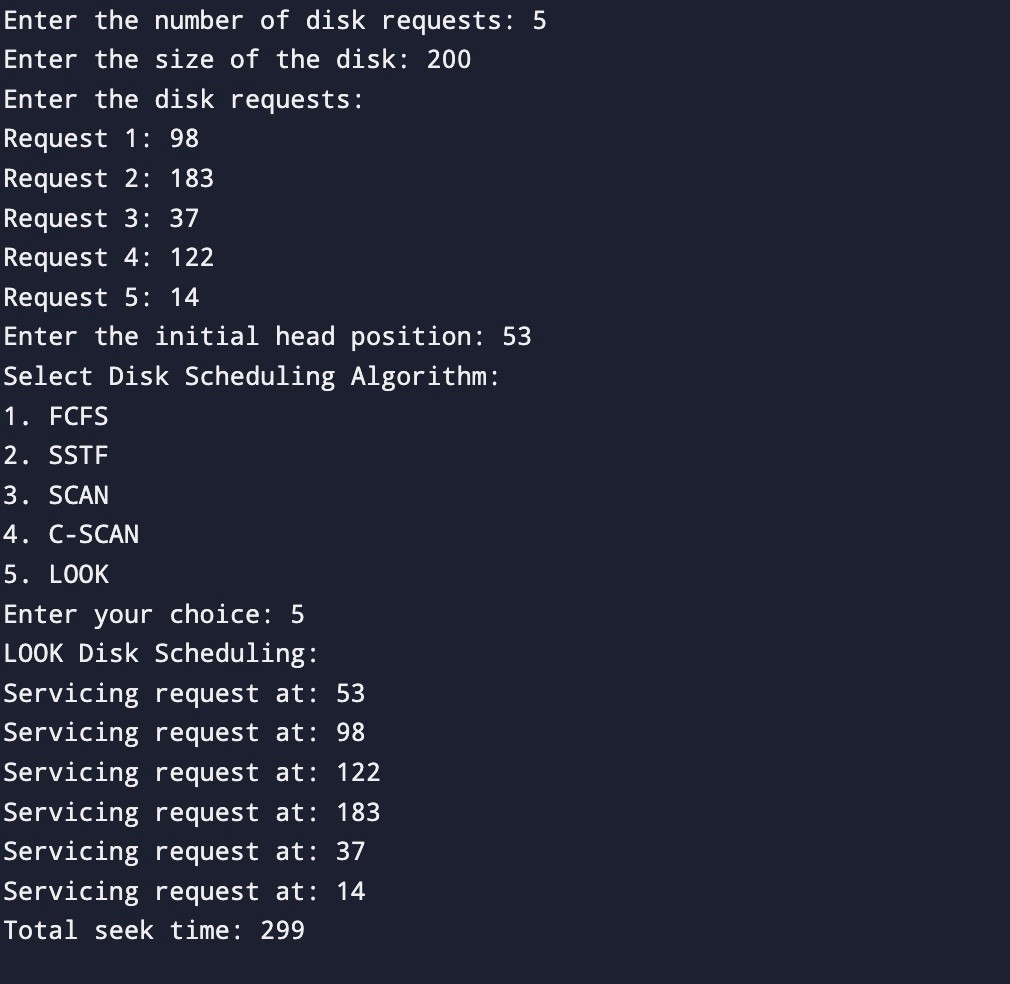
SAMPLE OUTPUT 3:



SAMPLE OUTPUT 4:



SAMPLE OUTPUT 5:



## CONCLUSION

Disk scheduling algorithms are used to determine the order in which disk I/O requests are serviced. They aim to minimize seek time and improve disk performance. Some common disk scheduling algorithms include FCFS (First-Come, First-Served), SSTF (Shortest Seek Time First), SCAN, C-SCAN, LOOK, and C-LOOK. Each algorithm employs different strategies to optimize disk access and minimize head movement, considering factors like seek time, rotational latency, and throughput.

## REFERENCES

1. Main Idea from includehelp.com Submitted by Amit Shukla, on August 14, 2017 < [https://www.ine.ludehelp.com/operating-systems/file-management-in-operating-](http://www.ine.ludehelp.com/operating-systems/file-management-in-operating-) system.aspx>
2. How to rename a file answer by Mazhar MIK on askubuntu.com
3. <https://askubuntu.com/questions/280768/how-to-rename-a-file-in- terminal#:textA%20simple%20way%20to%20rename.from%20one%20name%20to

%20another &text where%20%E2%80%90filel>

1. Use of stat command answer by <https: //lmuxhint.com/linux star command>