

Disk Scheduling Algorithms

In C Language



Table of Contents

SL. NO		Page No.
I	Introduction 1.1 Problem Statement 1.2 Objectives of the project	
II	System Requirements 2.1 Functional Requirements 2.2 Software and Hardware Requirements	
III	System Design 3.1 Architecture/Data Flow Diagrams 3.2 Modules	
IV	System Implementation 4.1 Module Description 4.2 Pseudocode	
V	Output Screen Shots	
VI	Conclusion	

1. Introduction

1.1. Problem Statement

To analyse and understand the working of three disk scheduling algorithms namely FCFS (First Come First Serve), CLOOK (Circular LOOK) and CSCAN (Circular SCAN).

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more requests may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

The program offers a choice to the user to select between any three disk scheduling algorithms namely FCFS (First Come First Serve), CLOOK (Circular LOOK) and CSCAN (Circular SCAN). The user can input the number of disk locations, the position of the head and subsequent disk locations and the output will be the Disk sequence, the total seek time and the average seek time. The user can compare and assess the outputs from the different algorithms that have been implemented and develop a deeper and better understanding of the same.

1.2. Objectives

The objective of this project is to learn about disk scheduling and implement the algorithms using C - programming language. The aim is to comprehend concepts like seek sequence.

The project is also designed to help differentiate between the three algorithms namely FCFS, CSCAN and CLOOK.

The user will be allowed to enter the input and then make a choice to select which algorithm to implement.

The user can continue to check with another algorithm or to exit the application. The user can then view and compare different results to analyse which algorithm is more efficient and which one proves to be more effective

2. System Requirements

2.1. Functional Requirements

The functional requirements for this project are:

- **Input :**
 - number of disk locations
 - the position of the head
 - subsequent disk locations
- **Output:**
 - The Disk Sequence
 - Total Seek Time
 - Average Seek Time
- **Procedures involved :**
 - Implementing the CLOOK algorithm : In this process the head services requests only in one direction(either left or right) until all the requests in this direction are not serviced and then jumps back to the farthest request on the other direction and service the remaining requests which gives a better uniform servicing as well as avoids wasting seek time for going till the end of the disk.
 - Implementing the CSCAN algorithm: In this process the head from one end servicing all the requests to the other end. However, as soon as the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on the return trip and starts servicing again once reaches the beginning
 - Implementing the FCFS algorithm : In this process, requests are entertained in the order they arrive in the disk queue. The algorithm appears to be very fair and there is no starvation but generally, it does not provide the fastest service.
 - The three algorithms are displayed and a choice is given to the user to select the algorithms they prefer and continue with the disk scheduling process.

2.2. Software and Hardware requirements

Software used:

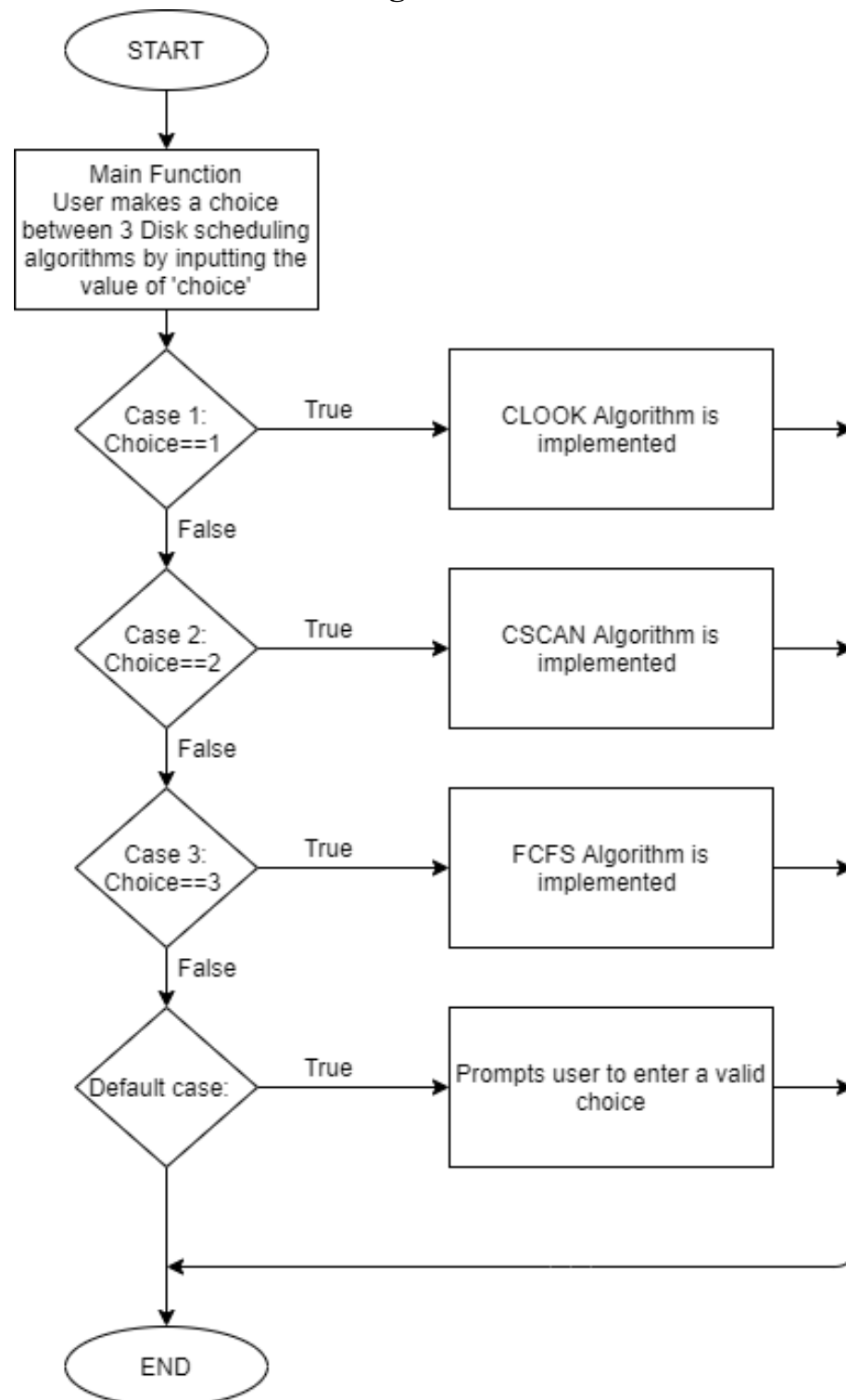
- Programming Language - C
- Compiler used - MinGW (GCC)
- Text Editor - Notepad ++ and Visual Studio Code

Hardware used:

- Intel i3 core or a higher version
- 4GB RAM or higher
- 32 Bit Operating System or higher
- Windows Specification - Windows 7 or higher

3. System Design

3.1. Architecture and Data Flow Diagram



3.2. Modules

Modules used in this project are:

- Main method
- FCFS
- CSCAN
- CLOOK

4. System Implementation

4.1. Algorithms

4.1.2. FCFC

- a)** Request array represents an array, storing indexes of tracks that have been requested in ascending order of arrival time and 'Head' denotes position of disk head.
- b)** Calculate the absolute distance of the track from the head.
- c)** Increment the total seek count with this distance.
- d)** Currently serviced track position becomes the new head position.
- e)** Go to step 'b' until all tracks in the request array have not been serviced.

4.1.3. CSCAN

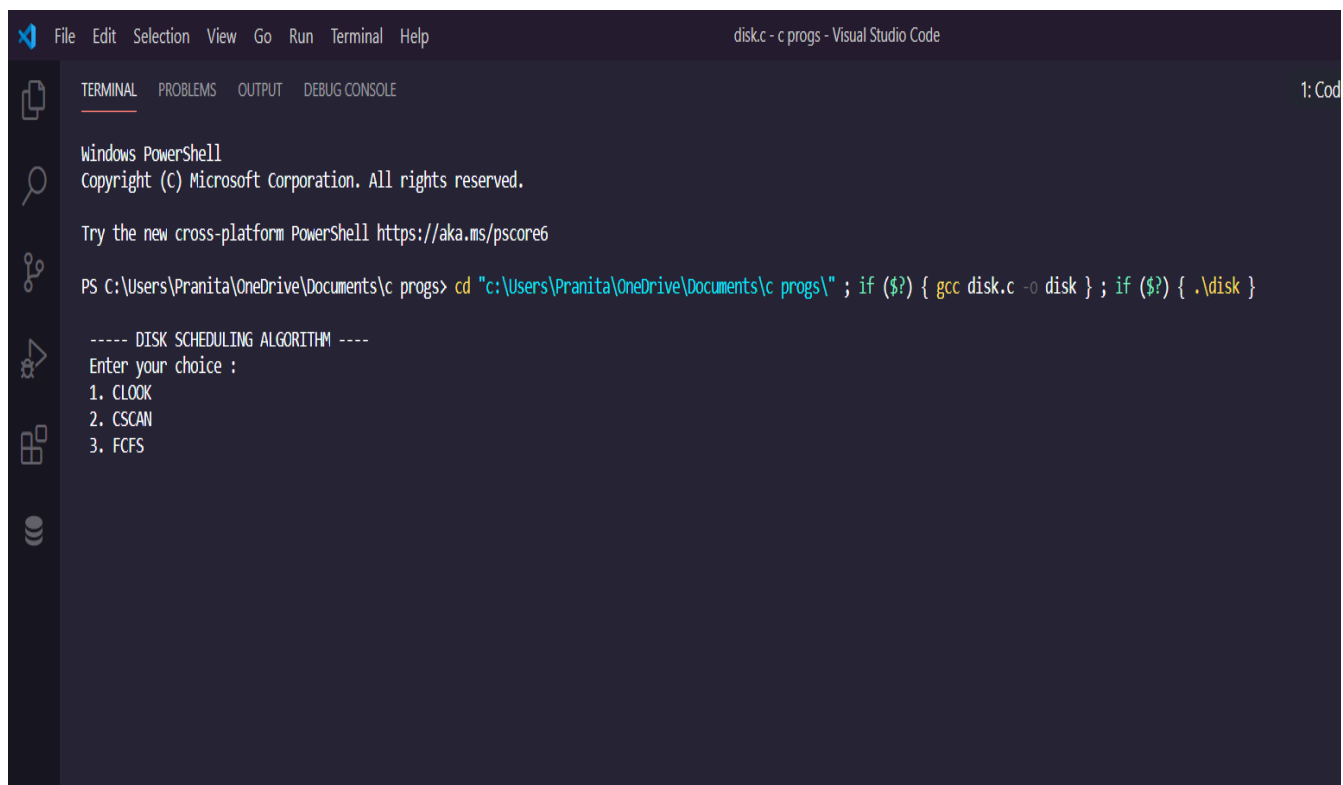
- a)** Request array represents an array storing indexes of tracks that have been requested in ascending order of arrival time and 'Head' is the position of disk head.
- b)** The head services only in the right direction from 0 to size of the disk.
- c)** While moving in the left direction do not service any of the tracks.
- d)** When we reach at the beginning(left end) reverse the direction.
- e)** While moving in the right direction, it services all tracks sequentially.
- f)** While moving in the right direction, calculate the absolute distance of the track from the head.
- g)** Increment the total seek count with this distance.
- h)** Currently serviced track position becomes the head position.
- i)** Go to step 'f' until the right end of the disk is reached.
- j)** If we reach at the right end of the disk reverse the direction and go to step 'c' until all tracks in the request array have not been serviced.

4.1.4. CLOOK

- a)** Request array represents array storing indexes of the tracks that have been requested in ascending order of arrival time and 'head' is the position of the disk head.
- b)** The initial direction in which the head is moving is given and it services in the same direction.
- c)** The head services all the requests sequentially in the direction it is moving.
- d)** The head continues to move in the same direction until all the requests in this direction have been serviced.
- e)** While moving in this direction, calculate the absolute distance of the tracks from the head.
- f)** Increment the total seek count with this distance.

- g) Currently serviced track position becomes the head position.
- h) Go to step 'e' until the last request is reached in this direction.
- i) If the last request is reached in the current direction, then reverse the direction and move the head in this direction until the last request is reached - that is needed to be serviced in this direction without servicing the intermediate requests.
- j) Reverse the direction and go to step 'c' until all the requests have not been serviced.

5. Output Screenshot



```
File Edit Selection View Go Run Terminal Help
disk.c - c progs - Visual Studio Code

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

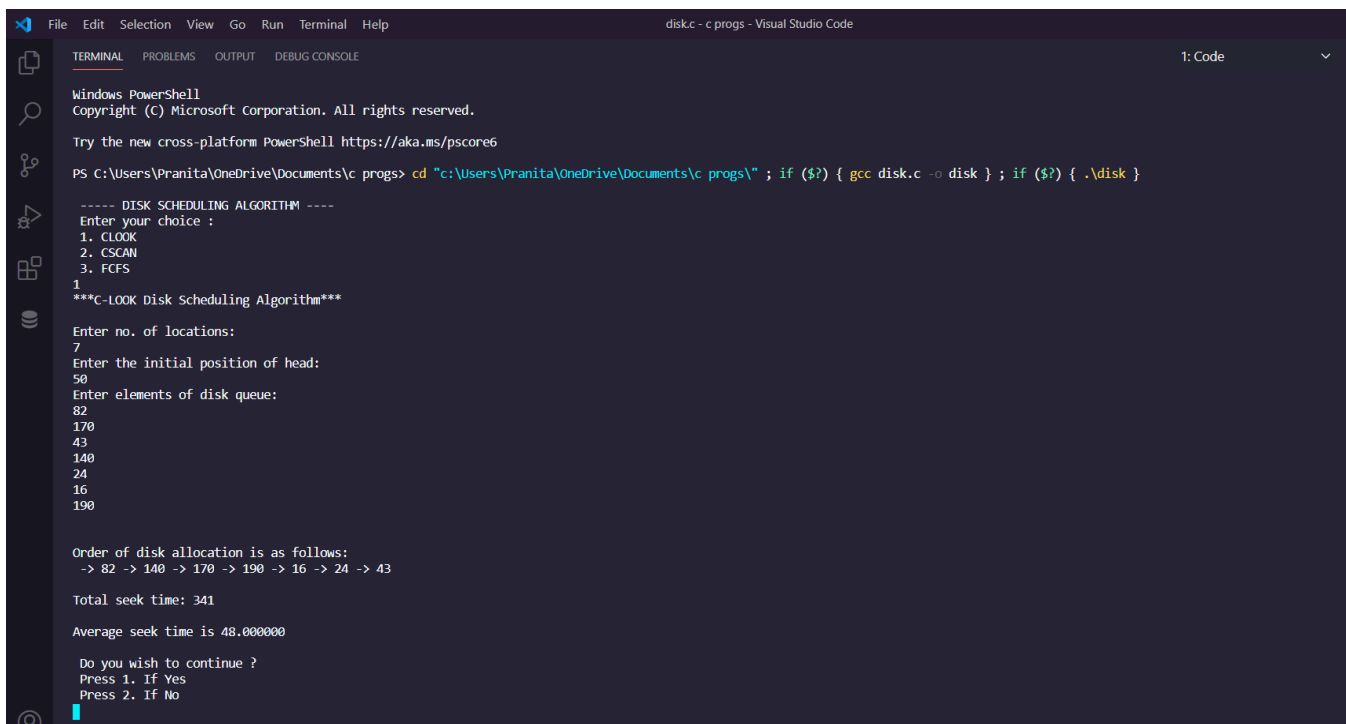
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Pranita\OneDrive\Documents\c progs> cd "c:\Users\Pranita\OneDrive\Documents\c progs\" ; if ($?) { gcc disk.c -o disk } ; if ($?) { .\disk }

----- DISK SCHEDULING ALGORITHM -----
Enter your choice :
1. CLOOK
2. CSCAN
3. FCFS
```

Fig 5.1 : Initial view of the output screen



```
File Edit Selection View Go Run Terminal Help
disk.c - c progs - Visual Studio Code

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Pranita\OneDrive\Documents\c progs> cd "c:\Users\Pranita\OneDrive\Documents\c progs\" ; if ($?) { gcc disk.c -o disk } ; if ($?) { .\disk }

----- DISK SCHEDULING ALGORITHM -----
Enter your choice :
1. CLOOK
2. CSCAN
3. FCFS
1
***C-LOOK Disk Scheduling Algorithm***

Enter no. of locations:
7
Enter the initial position of head:
50
Enter elements of disk queue:
82
170
43
140
24
16
190

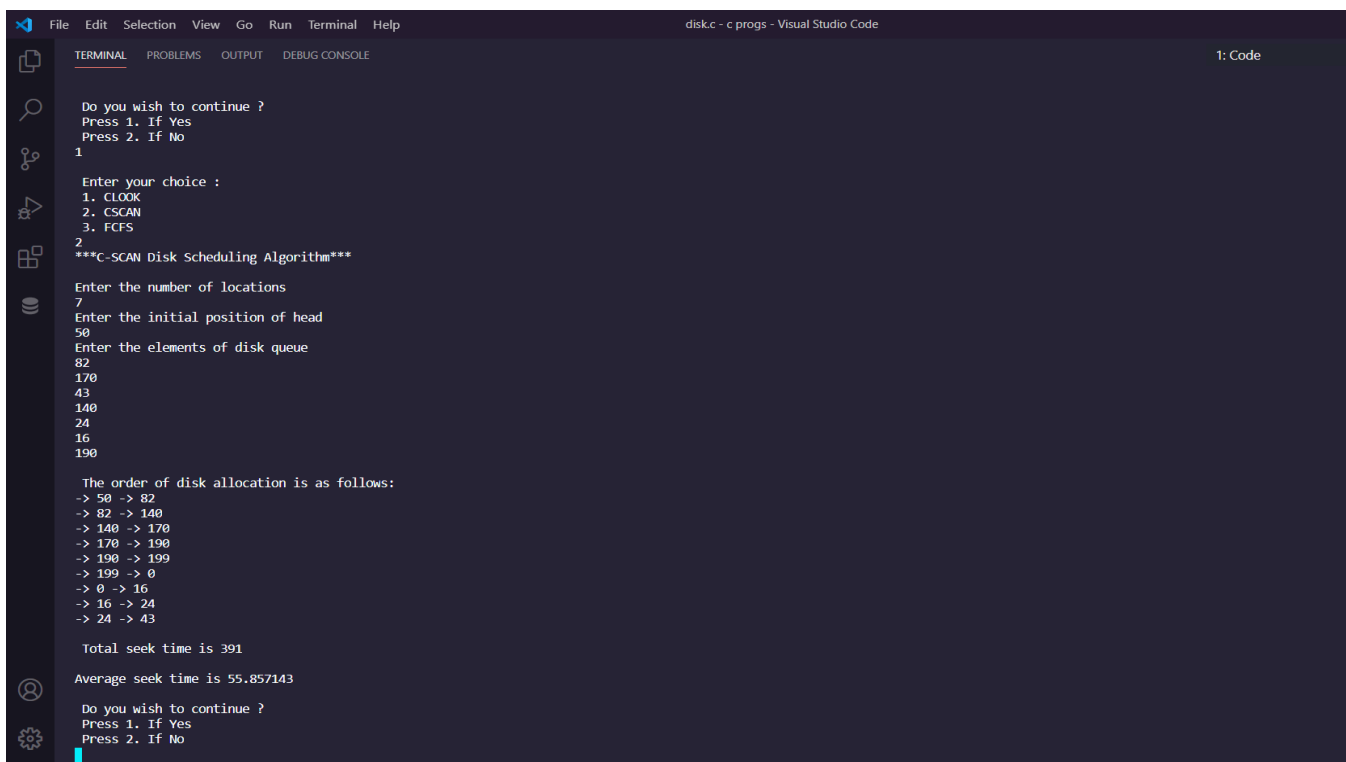
Order of disk allocation is as follows:
-> 82 -> 140 -> 170 -> 190 -> 16 -> 24 -> 43

Total seek time: 341

Average seek time is 48.000000

Do you wish to continue ?
Press 1. If Yes
Press 2. If No
1
```

Fig 5.2 : Working of C-LOOK



```
File Edit Selection View Go Run Terminal Help
disk.c - c progs - Visual Studio Code

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: Code

Do you wish to continue ?
Press 1. If Yes
Press 2. If No
1

Enter your choice :
1. CLOOK
2. CSCAN
3. FCFS
2
***C-SCAN Disk Scheduling Algorithm***

Enter the number of locations
7
Enter the initial position of head
50
Enter the elements of disk queue
82
170
43
140
24
16
190

The order of disk allocation is as follows:
-> 50 -> 82
-> 82 -> 140
-> 140 -> 170
-> 170 -> 190
-> 190 -> 199
-> 199 -> 0
-> 0 -> 16
-> 16 -> 24
-> 24 -> 43

Total seek time is 391

Average seek time is 55.857143

Do you wish to continue ?
Press 1. If Yes
Press 2. If No
1
```

Fig 5.3 : Working of C-SCAN


```

File Edit Selection View Go Run Terminal Help
disk.c - c prog - Visual Studio Code
1: Code
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

Do you wish to continue ?
Press 1. If Yes
Press 2. If No
1

Enter your choice :
1. CLOOK
2. CSCAN
3. FCFS
3
***FCFS Disk Scheduling Algorithm***

Enter the number of locations
7
Enter initial position of head
50
Enter the elements of disk queue

82
170
43
140
24
16
190

The order of disk allocation is as follows:
-> 50 -> 82
-> 82 -> 170
-> 170 -> 43
-> 43 -> 140
-> 140 -> 24
-> 24 -> 16
-> 16 -> 190

Total seek time is 642
Average seek time is 91.714287
Do you wish to continue ?
Press 1. If Yes
Press 2. If No
2
PS C:\Users\Pranita\OneDrive\Documents\c prog>

```

Fig 5.4 : Working of FCFS

6. Conclusion

Over the duration of this project, our group has managed to understand, comprehend and correctly identify and label the various algorithms used in the disk scheduling process. Concepts like seek time, rotational latency, transfer time, disk access time and disk response time have been understood and learnt thoroughly.

We have studied and applied the CLOOK, CSAN and FCFS algorithms in C language, creating an interface on the python console for the same.

The user can continue to check with another algorithm or to exit the application. The user can then view and compare different results to analyze which algorithm is more efficient.

Comparison between CLOOK, CSCAN and FCFS:

CLOOK	CSCAN	FCFS
C-LOOK algorithm has the best performance in all disk scheduling algorithms.	Whereas C-SCAN lags in performance, when compared to C-LOOK	FCFS algorithm is easy to understand and implement but it lags in performance
C-LOOK provides low variance in response time and waiting time.	C-SCAN provides uniform waiting time and response time.	In FCFS algorithm there is high variance in response time and waiting time.

In C-LOOK algorithm Throughput increases.	Here there is increment in Throughput.	In FCFS algorithm Throughput decreases.
In C-LOOK algorithm neither the requests suffers starvation nor Convoy effect.	In C-SCAN algorithm neither the requests suffers starvation nor Convoy effect.	FCFS doesn't cause starvation to any request, but request can experience Convoy effect.

After analysing the working of the three algorithms, it can be concluded that CLOOK algorithm offers the best performance in comparison to CSCAN and FCFS as it does not lead to starvation and it provides low variance and waiting time.

References:

1. <https://www.geeksforgeeks.org/disk-scheduling-algorithms/>
2. <http://www.cs.iit.edu/~cs561/cs450/disksched/disksched.html>
3. <http://www2.cs.uregina.ca/~hamilton/courses/330/notes/io/node8.html>
4. http://faculty.salina.k-state.edu/tim/oss/Files/disk_scheduling.html
5. <https://github.com/>
6. <https://www.geeksforgeeks.org/c-look-disk-scheduling-algorithm/>
7. <https://www.wikipedia.org/>
8. <https://www.thecompletecodes.com/2019/08/c-programming-codes-for-c-scan-disk-scheduling-algorithm.html>
9. <https://www.guru99.com/fcfs-scheduling.html#:~:text=First%20Come%20First%20Serve%20>
10. https://en.wikipedia.org/wiki/Elevator_algorithm
11. <https://www.geeksforgeeks.org/c-scan-disk-scheduling-algorithm/>
12. <https://www.geeksforgeeks.org/fcfs-disk-scheduling-algorithms>

