

Disk Scheduling Algorithms

GROUP - 1

**Submitted to:
Dr L.T Jayprakash**

**Submitted by:
Anusha Modwal(MT2013024)
Ashutosh Trivedi(MT2013030)
Piyush Kaushik (MT2013108)
Sumit Singh Chauhan(MT2013155)**

1. Specification

The main features of the system are following-

- The system will show the comparison among different algorithms in the form of graphs.
- The system will also be able to keep track of all the previous results so that an average comparison can also be determined.
- The system will let the user to choose the algorithm that he or she wants to analyse.

2. System Architecture

2.1 Architectural Design

MVC architecture is used for the development of the system -

- A model is an object representing data or even activity, e.g. a database table.
- A view is some form of visualization of the state of the model.
- A controller offers facilities to change the state of the model.

2.2 Design Rationale

The reason behind choosing the MVC architecture is following -

- Project has a separate section for displaying the output and taking the input from the user which is not very coupled with other modules so we can put it in the view module.
- We need to have a separate section for the database entries and fetching and we are trying to keep it as independent as possible so we will create a separate data module and will put all these functions in that module only.
- The system will be doing a lot of calculation after fetching the data from database of getting the data from the user so this calculation will be done independently so all this will be kept in the control module.

3. Software Packages

3.1 Edu.iiitb.model – This package contains two classes Schedulemodel.java and legecyModel.java.

- Schedulemodel.java –It take input as algoType, trackinput and startPoint
- legecyModel.java - It take input type, avg, no, and head

3.2 Edu.iiitb.view- The sole purpose of this is to provide a depiction of the schedule graph. This is also used to show comparison among the different algorithms using bar graphs. It contains following classes.

- SelectAlgoView : It contains all the AWT components for the functionality of accepting the algorithm type user wants to select.
- InputTrackView – It contains all the AWT components for the functionality of accepting the track string from user.
- ScheduleAlgoView – It has Draw function which draw the movement of disk header and gives the user to select other algorithm at the same time with drop down java label.
- ScheduleGraphView – this component generate bar graph of the performance. It also takes the values from database.

3.3 Edu.iiitb.controller - This section holds the calculative representations of various algorithms used in our project. The output of each of these procedures will be with respect to seek time on a given input. It contains following classes.

- ScheduleGraphController – This module has all the algorithm implemented and some helper function.
- LegacyController – It contains all the functionality and functioning dealing with legacy comparision.
- DbController – This module has database connection, insert and fetch functions.
- AvgCompController- This module contains all the function which takes the raw data (track string) and calculate the seek time for legacy database to insert. Legacy database contains processed data but not the raw data.

4. Algorithm Functioning

The system essentially implements the following algorithms if we take a re- quest set as 95, 180, 34, 119, 11, 123, 62, 64

4.1 First Come -First Serve (FCFS)

All incoming requests are placed at the end of the queue. Whatever number that is next in the queue will be the next number served. Using this algorithm doesn't provide the best results. To determine the number of head movements you would simply and the number of tracks it took to move from one request to the next. For this case it went from 50 to 95 to 180 and so on. From 50 to 95 it moved 45 tracks. If you tally up the total number of tracks you will and how many tracks it had to go through before finishing the entire request. In this example, it had a total head movement of 640 tracks. The disadvantage of this algorithm is noted by the oscillation from track 50 to track 180 and 5 then back to track 11 to 123 then to 64. As you will soon see, this is the worse algorithm that one can use.

4.2 Shortest Seek Time First (SSTF)

In this case request is serviced according to next shortest distance. Starting at 50, the next shortest distance would be 62 instead of 34 since it is only 12 tracks away from 62 and 16 tracks away from 34. The process would continue until all the process are taken care of. For example the next case would be to move from 62 to 64 instead of 34 since there are only 2

tracks between them and not 18 if it were to go the other way. Although this seems to be a better service being that it moved a total of 236 tracks, this is not an optimal one. There is a great chance that starvation would take place. The reason for this is if there were a lot of requests close to each other the other requests will never be handled since the distance will always be greater.

4.3 SCAN

This approach works like an elevator does. It scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up. This process moved a total of 230 tracks. Once again this is more optimal than the previous algorithm, but it is not the best.

5.4 Circular Scan (C-SCAN)

Circular scanning works just like the elevator to some extent. It begins its scan toward the nearest end and works its way all the way to the end of the system. Once it hits the bottom or top it jumps to the other end and moves in the same direction. Keep in mind that the huge jump doesn't count as a head movement. The total head movement for this algorithm is only 187 tracks, but still it is not the most efficient.

5.5 C-LOOK

This is just an enhanced version of C-SCAN. In this the scanning doesn't go past the last request in the direction that it is moving. It too jumps to the other end but not all the way to the end. Just to the furthest request. C-SCAN had a total movement of 187 but this scan (C-LOOK) reduced it down to 157 tracks. From this you were able to see a scan change from 644 total head movements to just 157. You should now have an understanding as to why your operating system truly relies on the type of algorithm it needs when it is dealing with multiple processes.

5.6 Look

As we described them, both SCAN and C-SCAN move the disk arm across the full width of the disk. In practice, neither algorithm is often implemented this way. More commonly, the arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to the end of the disk. Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they look for a request before continuing to move in a given direction.

5.7 N-Step Scan

N-step scan takes the input tracks and break it into N steps. On every step SCAN algorithm which is explained above will run. Here we have implemented 3-step scan.

5.8 Pickup

Pickup algorithm is combination of FCFS and SCAN. It will take the input as FCFS bases but process them based SCAN.

5. User Interface Design

The user essentially provides the input track numbers along with the starting header position. On continuing to the next page the graphical representation so as of how the seek time for the various track requests are given, is depicted. On the main page there is a button for depicting the legacy data in the form a bar chart. Also, if required the user can compare the results of all the 8 algorithms shown in the bar chart format.



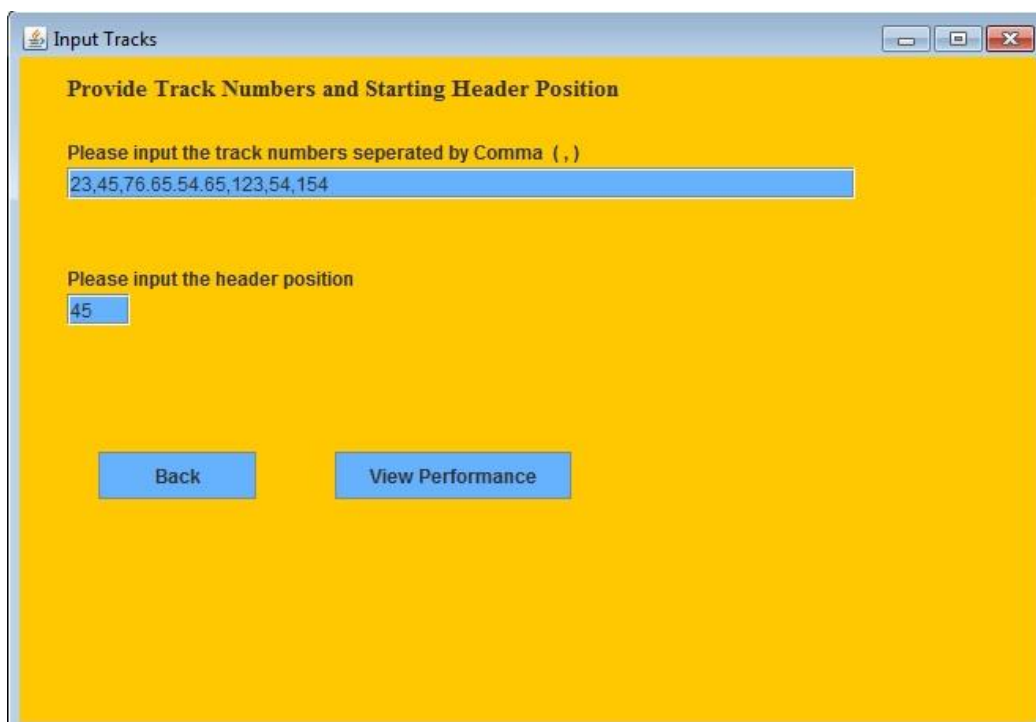
Select Algorithm

Select Any One of the Algorithm Below

click "Next" to input the sequence or click "Average Comparison" to see comparison on already existing data.

- ☐ First Come -First Serve (FCFS)
- ☐ Shortest Seek Time First (SSTF)
- ☐ SCAN
- ☐ Circular Scan (C-SCAN)
- ☐ C-LOOK
- ☐ Look
- ☐ 3-Step Scan
- ☐ Pickup

Next Average Comparison



Input Tracks

Provide Track Numbers and Starting Header Position

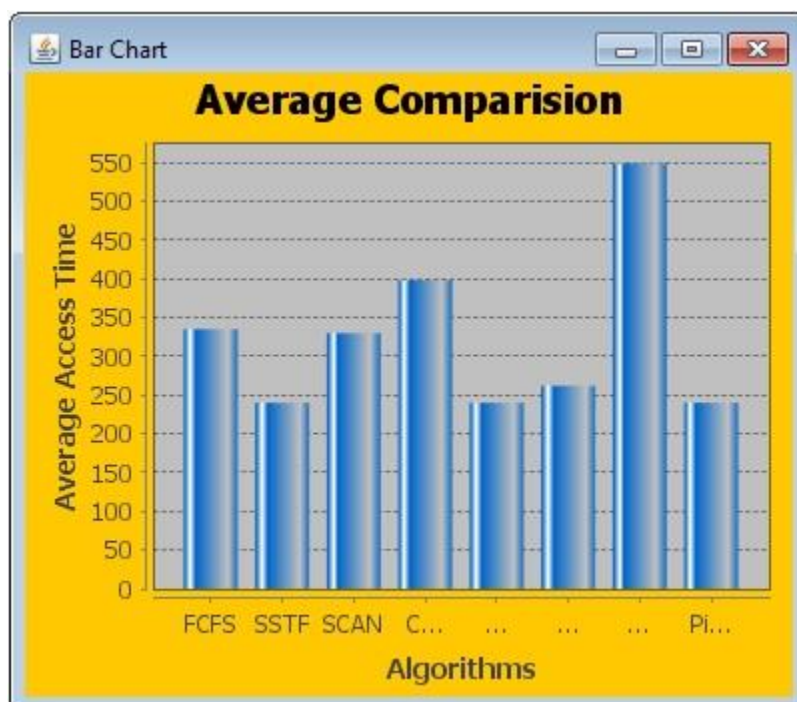
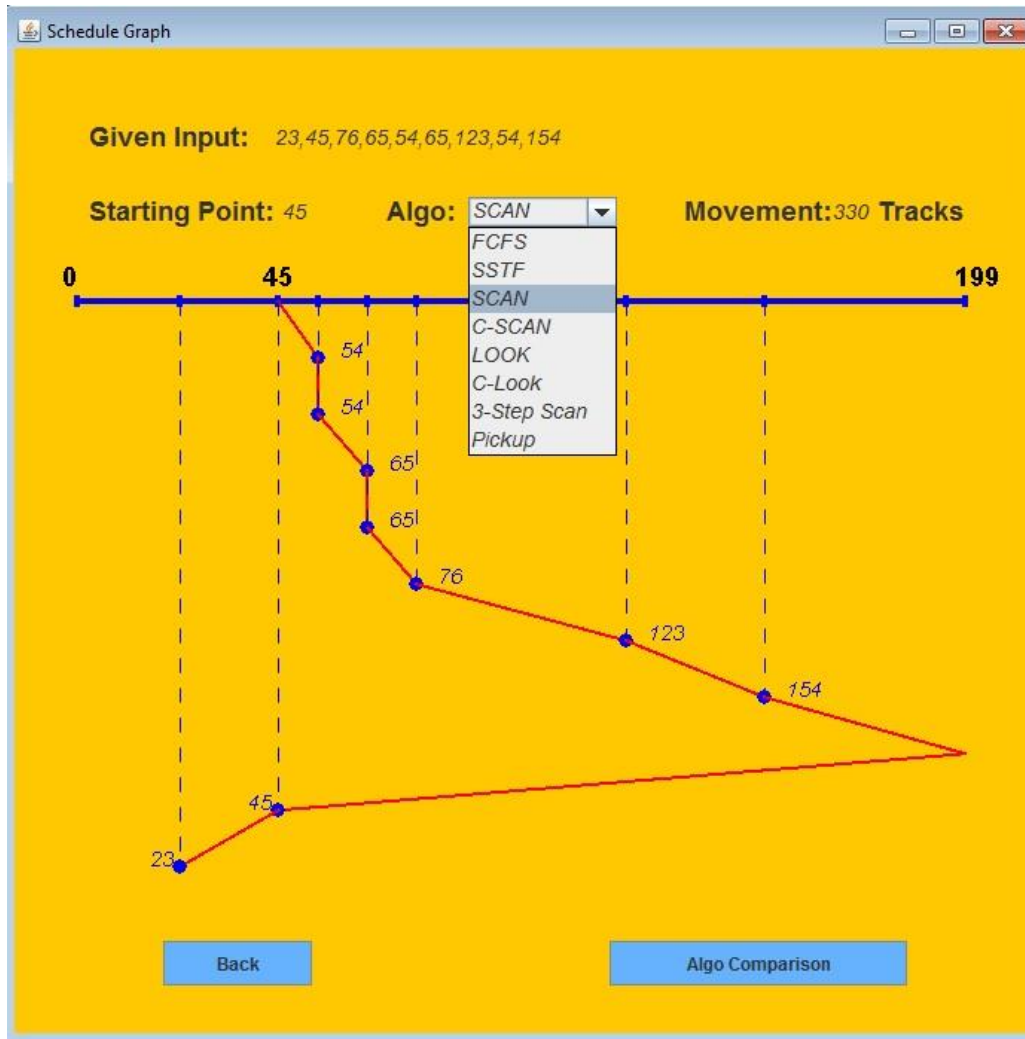
Please input the track numbers seperated by Comma (,)

23,45,76,65,54,65,123,54,154

Please input the header position

45

Back View Performance



6. Test-Cases:

Test Case-1:

Initial Header Position: 50

Input String	Algorithm	Expected output	Output obtained
34,54,65,27,124,164	FCFS	34,54,65,27,124,164	34,54,65,27,124,164
34,54,65,27,124,164	SSTF	54,65,34,27,124,164	54,65,34,27,124,164
34,54,65,27,124,164	SCAN	54,65,124,164,199,34,27	54,65,124,164,199,34,27
34,54,65,27,124,164	C-SCAN	54,65,124,164,199,27,34	54,65,124,164,199,27,34
34,54,65,27,124,164	LOOK	54,65,124,164,34,27	54,65,124,164,34,27
34,54,65,27,124,164	C-LOOK	54,65,124,164,27,34	54,65,124,164,27,34
34,54,65,27,124,164	3 step	54,65,199,34,27,124,164	54,65,199,34,27,124,164
34,54,65,27,124,164	Pickup	34,54,65,27,124,164	34,54,65,27,124,164

Test Case-2:

Initial Header Position: 43

Input String	Algorithm	Expected output	Output obtained
65,98,11,23,156,87,9	FCFS	65,98,11,23,156,87,9	65,98,11,23,156,87,9
65,98,11,23,156,87,9	SSTF	23,11,9,65,87,98,156	23,11,9,65,87,98,156
65,98,11,23,156,87,9	SCAN	65,87,98,156,199,23,11,9	65,87,98,156,199,23,11,9
65,98,11,23,156,87,9	C-SCAN	65,87,98,156,199,9,11,23	65,87,98,156,199,9,11,23
65,98,11,23,156,87,9	LOOK	65,87,98,156,23,11,9	65,87,98,156,23,11,9
65,98,11,23,156,87,9	C-LOOK	65,87,98,156,9,11,23	65,87,98,156,9,11,23
65,98,11,23,156,87,9	3 step	65,98,199,11,23,87,156,9	65,98,199,11,23,87,156,9
65,98,11,23,156,87,9	Pickup	65,87,98,23,11,156,9	65,87,98,23,11,156,9

Test Case-3:

Initial Header Position: 87

Input String	Algorithm	Expected output	Output obtained
12,134,78,145	FCFS	12,134,78,145	12,134,78,145
12,134,78,145	SSTF	78,134,145,12	78,134,145,12
12,134,78,145	SCAN	134,145,199,78,12	134,145,199,78,12
12,134,78,145	C-SCAN	134,145,199,12,78	134,145,199,12,78
12,134,78,145	LOOK	134,145,78,12	134,145,78,12
12,134,78,145	C-LOOK	134,145,12,78	134,145,12,78
12,134,78,145	3 step	134,199,78,12,145	134,199,78,12,145
12,134,78,145	Pickup	78,12,134,145	78,12,134,145

7. Limitations:

1. The track numbers are assumed to be between 0 to 199.
2. Maximum 20 track requests can be taken at a time.
3. In the legacy system, the real input strings cannot be traced.