

OPERATING SYSTEMS PROJECT

on

VARIOUS DISK SCHEDULING **ALGORITHMS AND THEIR** **COMPARISON**

Group Members

NAME	REGISTRATION NUMBER
ARPIT RATHI	16BCI0065
PARMEET SINGH	16BCE0184

Under the guidance of
SHAIK NASEERA
School of Computer Science And Engineering
VIT University, Vellore



OCTOBER 2017

TABLE OF CONTENTS

INTRODUCTION	3-4
LITERATURE SURVEY	5-9
SYSTEM DESIGN AND EXPLANATION	9-13
OUTPUTS	14-15
IMPLEMENTATION TABLE	16
FUTURE REFERENCES	17
CONCLUSION	17
REFERENCE	17-18

INTRODUCTION

Disk Scheduling is the method in which the operating systems decide in which order I/O operations are executed. Thus it is sometimes also called Input/output (I/O) Scheduling.

In operating systems, seek time is very important. Since all device requests are linked in queues, the seek time is increased causing the system to slow down. **Disk Scheduling** Algorithms are used to reduce the total seek time of any request.

The main purpose of the I/O Schedulers is dependent on the purpose of the operating system:

1. Decrease the time wasted by hard-disk seeks
2. To prioritize certain processes' I/O request
3. To give a share of the bandwidth to each running process
4. To guarantee that each process request will be handled before a particular deadline.

In this project we will demonstrate various Disk Scheduling algorithms like

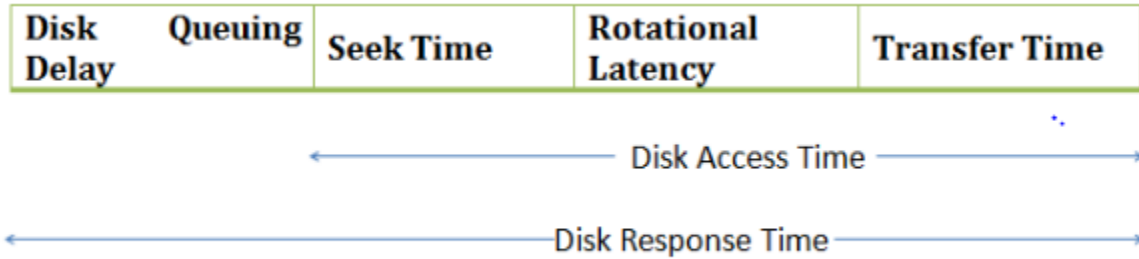
- i. FCFS
- ii. SSTF
- iii. SCAN
- iv. C-SCAN
- v. LOOK
- vi. C-LOOK

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:** Disk Access Time is:

Disk Access Time = Seek Time + Rotational Latency + Transfer Time

- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.



We will also try to implement some of the new disk scheduling algorithm proposed by scientist all around the world like:

- i.) SSEDV (Shortest Seek Time and Earliest Deadline by Value)
- ii.) SSEDV (Shortest Seek Time and Earliest Deadline by Value)

Finally we will compare all the different algorithms and come up with the best algorithm for different purposes.

We will also try to come up with one of our own algorithm and try to make its Disk Access Time better.

LITERATURE SURVEY

1.) Performance evaluation of two new disk scheduling algorithms for real-time systems

In this paper, we present two new disk scheduling algorithms for real-time systems. The two algorithms, called SSEDV (*Shortest Seek and Earliest Deadline by Ordering*) and SSEDV (*Shortest Seek and Earliest Deadline by Value*), combine *deadline* information and *disk service time* information in different ways. The basic idea behind these new algorithms is to give the disk I/O request with the earliest deadline a high priority, but if a request with a larger deadline is *very* close to the current disk arm position, then it may be assigned the highest priority. The performance of the SSEDV and SSEDV algorithms is compared with three other proposed real-time disk scheduling algorithms ED, P-SCAN, and FD-SCAN, as well as four conventional algorithms SSTF, SCAN, C-SCAN, and FCFS. An important aspect of the performance study is that the evaluation is not done in isolation with respect to the disk, but as part of an integrated collection of protocols necessary to support a real-time transaction system. The transaction system model is validated on an actual real-time transaction system test bed, called RT-CARAT. The performance results show that SSEDV outperforms SSEDV; that both of these new algorithms can improve performance of up to 38% over previously-known real-time disk scheduling algorithms; and that all of these real-time scheduling algorithms are significantly better than non-real-time algorithms in the sense of minimizing the transaction loss ratio.

2.) Scheduling algorithms for modern disk drives

Disk subsystem performance can be dramatically improved by dynamically ordering, or scheduling, pending requests. Via strongly validated simulation, we examine the impact of complex logical-to-physical mappings and large prefetching caches on scheduling effectiveness. Using both synthetic workloads and traces captured from six different user environments, we arrive at three main conclusions: (1) Incorporating complex mapping information into the scheduler provides only a marginal (less than 2%) decrease in response times for seek-reducing algorithms. (2) Algorithms which effectively utilize prefetching disk caches provide significant performance improvements for workloads with read sequentiality. The cyclical scan algorithm (C-LOOK), which always schedules requests in ascending logical order, achieves the highest performance among seek-reducing algorithms for such workloads. (3) Algorithms that reduce overall positioning delays produce the highest performance provided that they recognize and exploit a prefetching cache.

3.) Disk Scheduling: FCFS vs. SSTF Revisited

We report on a rather extensive simulation effort directed at evaluating the merits of two scheduling strategies, FCFS and SSTF, for moving-arm disks under stationary request arrival process. For First-Come-First-Served (FCFS) scheduling, analytic results for the mean waiting time are also given (in a closed form). If the objective of a schedule is to minimize the mean waiting time (or queue size) and its variance, the results seem to confirm the overall superiority of Shortest-Seek-Time-First (SSTF), particularly for medium and heavy traffic. This holds also when the input is highly correlated or addresses the cylinders nonuniformly. These results contradict some statements published in recent years. The domain of policies where SSTF is optimal is considered. The simulation methodology is described in some detail.

Disk Scheduling Algorithms

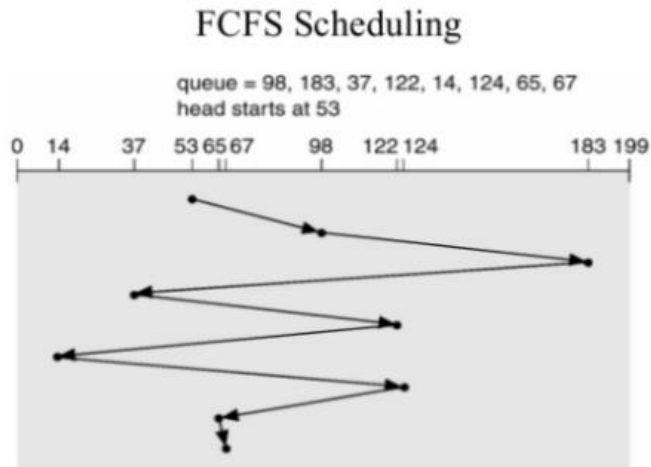
1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time
- May not provide the best possible service



2. **SSTF:** In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

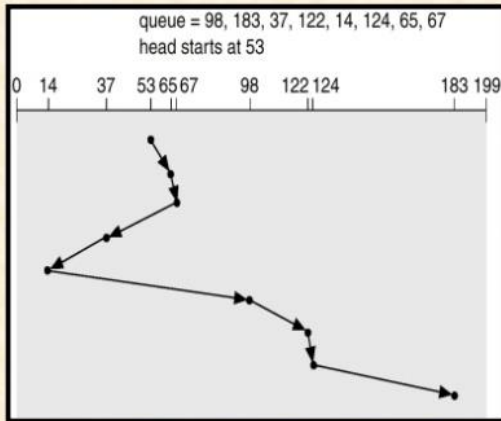
Advantages:

- Average Response Time decreases
- Throughput increases

Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests

SSTF



3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence also known as **elevator algorithm**. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

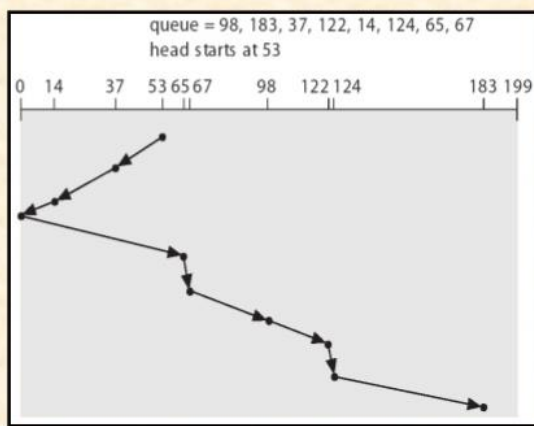
Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

- Long waiting time for requests for locations just visited by disk arm

SCAN

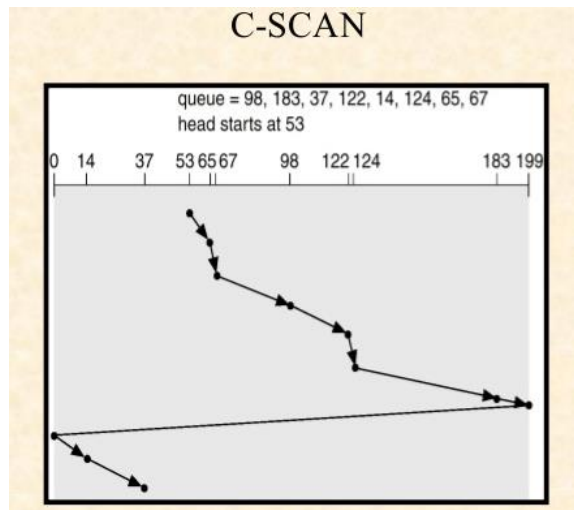


4. **C-SCAN:** In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

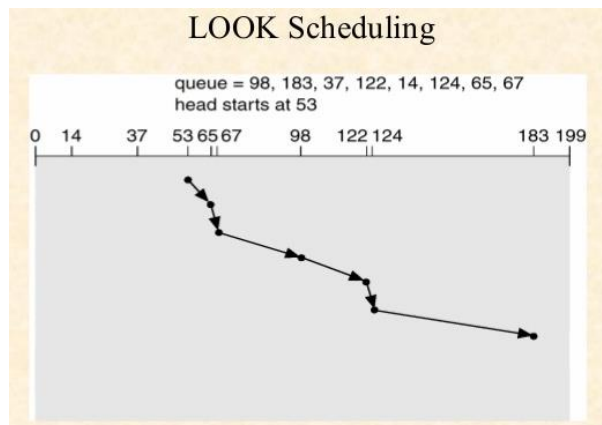
These situations are avoided in CSAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Advantages:

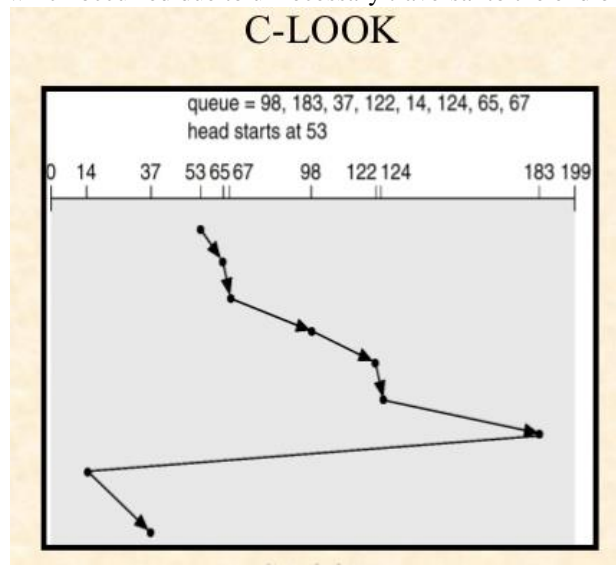
- Provides more uniform wait time compared to SCAN



5. **LOOK:** It is similar to the SCAN disk scheduling algorithm except the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.



6. **C-LOOK:** As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

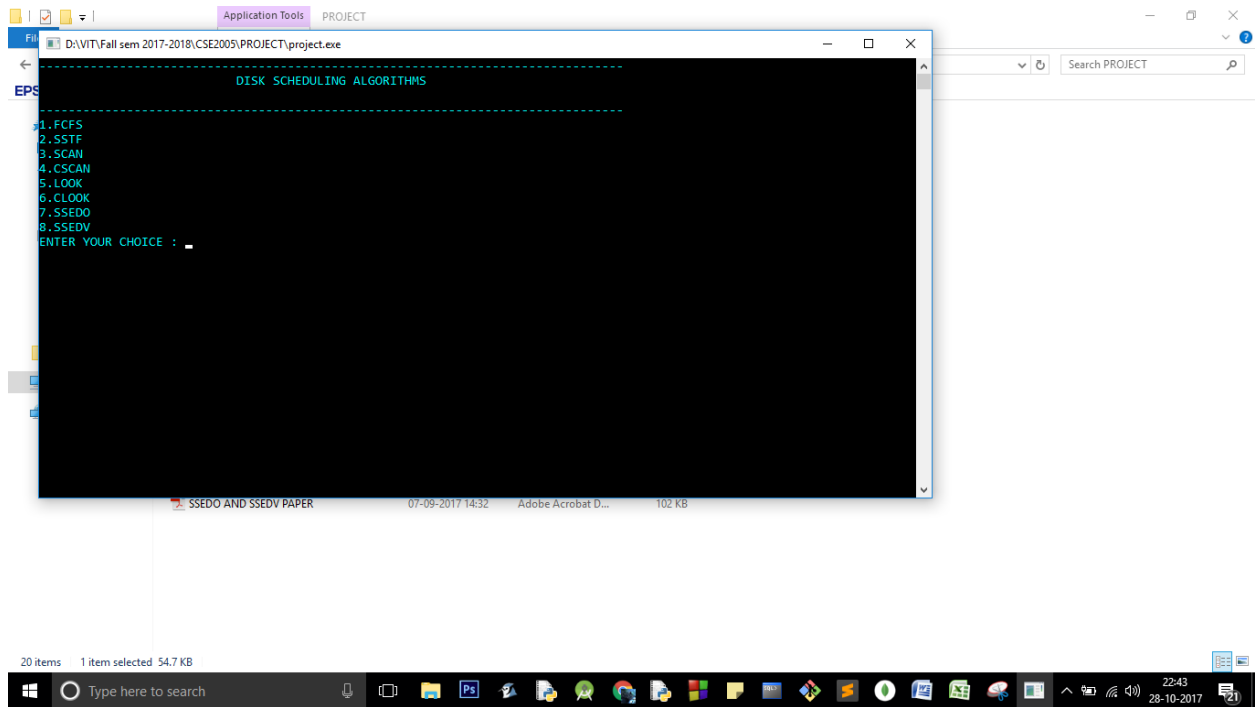


7. **SSEDO** (Shortest Seek Time and Earliest Deadline by Ordering) and **SSEDV** (Shortest Seek Time and Earliest Deadline by Value) :
The two algorithms, called SSEDO and SSEDV, combine deadline information and disk service time information in different ways. The basic idea behind these new algorithms is to give the disk I/O request with the earliest deadline a high priority, but if a request with a larger deadline is very close to the current disk arm position, then it may be assigned the highest priority. SSEDO and SSEDV both consider deadline and seek time, but put more weight on deadline. It also requires parameter tuning.
The performance results show that SSEDV outperforms SSEDO; that both of these new algorithms can improve performance of up to 38% over previously-known real-time disk scheduling algorithms; and that all of these real-time scheduling algorithms are significantly better than non-real-time algorithms in the sense of minimizing the transaction loss ratio.
8. **Own Algorithm** : Studying the above two algorithms i.e. SSEDO and SSEDV, we have tried to make our own algorithm. First we calculate the product of the element in the disk and their corresponding deadline and calculate its average. Then we sort the elements in the disk according to the product of element and its deadline divided by the average in ascending order and then we calculate the total seek time.

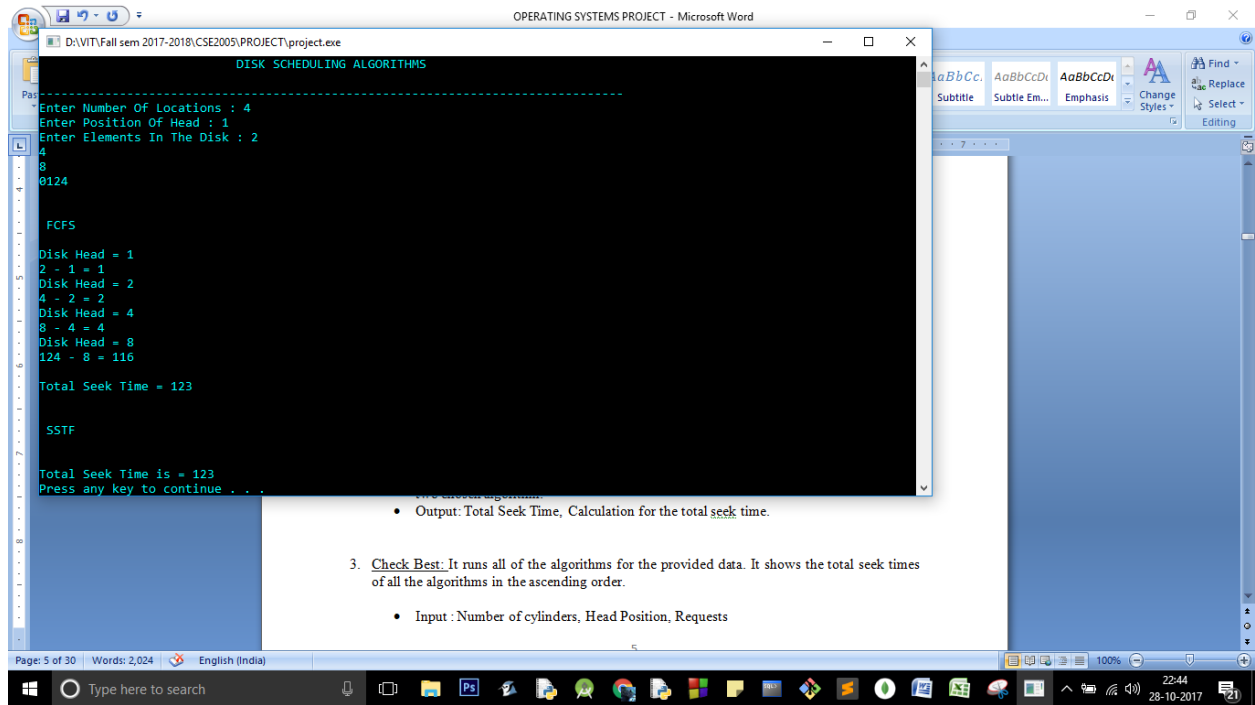
SYSTEM DESIGN AND EXPLANATION

The Various Modules Used In Our Application :

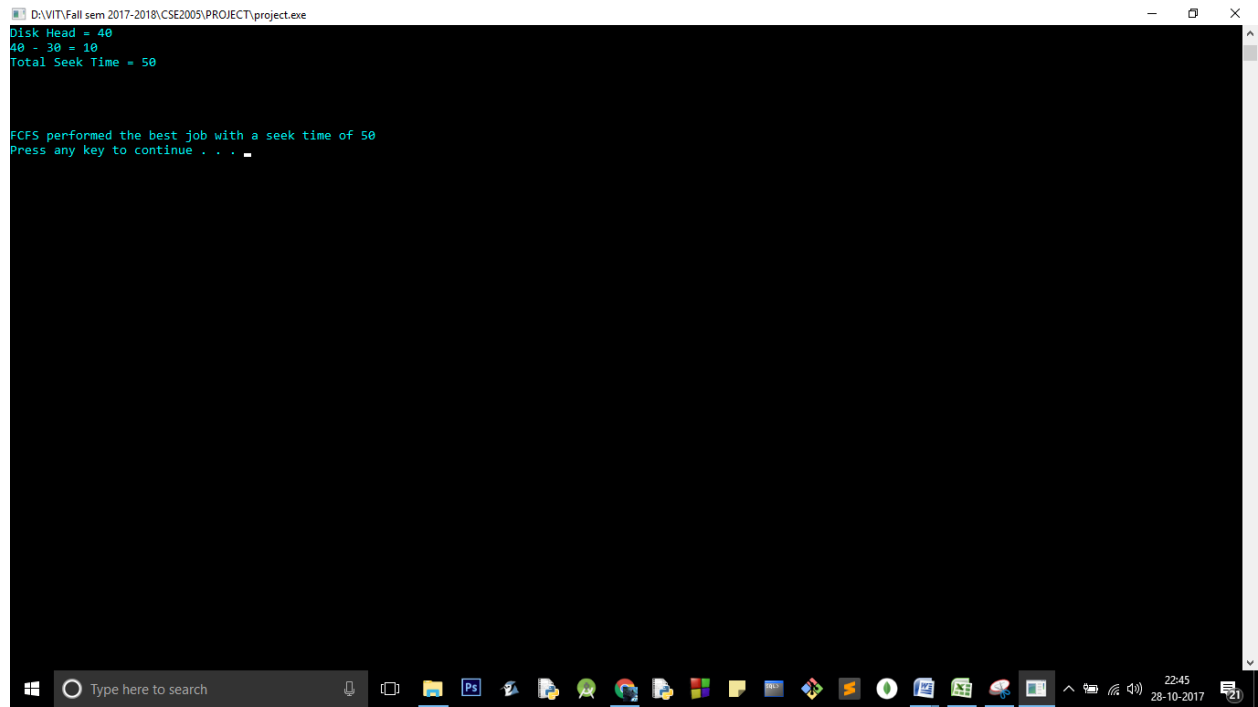
1. **Individual Analysis**: It chooses which algorithm to use and inputs the data for the algorithm. It shows all the calculations and the total seek time.
 - Input : Number of cylinders, Head Position, Requests
 - Working : It takes in the specified input and then calculates the total seek time of the chosen algorithm.
 - Output : Total Seek Time, Calculation for the total seek time.



2. Comparison: It chooses two out of many algorithms for the given data. It shows the difference in the seek time of the two.
- Input : Number of cylinders, Head Position, Requests
 - Working: It takes in the specified input and then calculates the total seek time of the two chosen algorithm.
 - Output: Total Seek Time, Calculation for the total seek time.



3. Check Best: It runs all of the algorithms for the provided data. It shows the total seek times of all the algorithms in the ascending order.
 - Input : Number of cylinders, Head Position, Requests
 - Working: It takes in the specified input and then calculates the total seek time of all the algorithms.
 - Output: Total Seek Time, Calculation for the total seek time, best performance algorithm.



The screenshot shows a Windows command prompt window titled "D:\VIT\Fall sem 2017-2018\CSE2005\PROJECT\project.exe". The output text is as follows:

```
Disk Head = 40
40 - 30 = 10
Total Seek Time = 50

FCFS performed the best job with a seek time of 50
Press any key to continue . . .
```

The window's taskbar at the bottom shows the Windows Start button, a search bar, and several application icons. The system clock in the bottom right corner displays "22:45 28-10-2017".

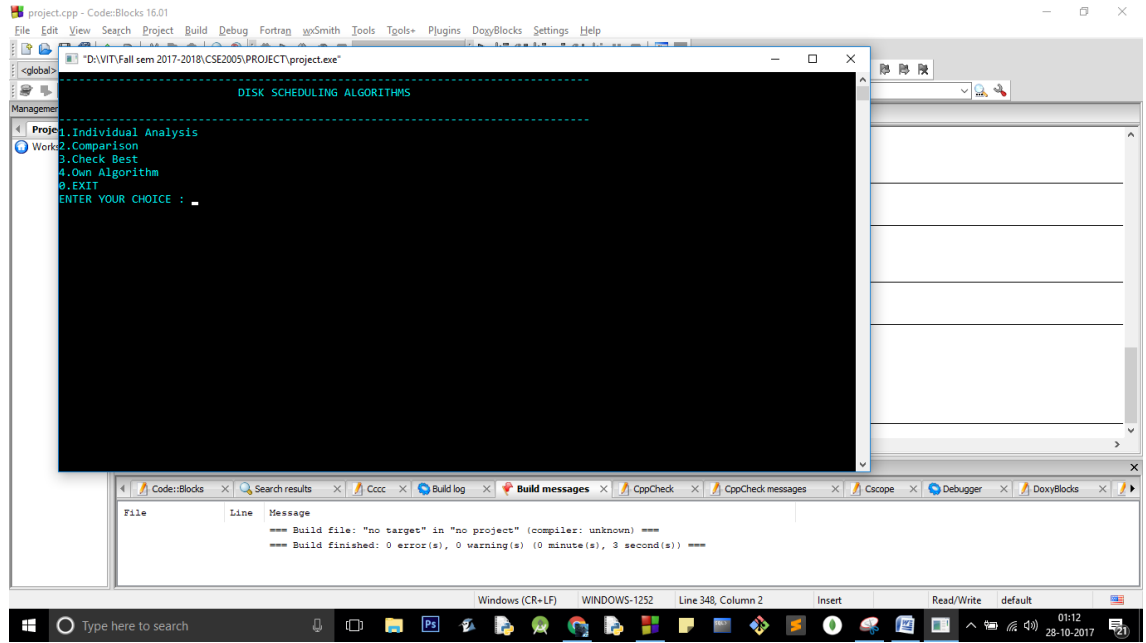
4. Own Algorithm: The given data runs on our algorithm and shows all the calculations and the total seek time and comparison between the other six algorithms.
- Input : Number of cylinders, Head Position, Requests, Deadlines of each request
 - Working: It takes in the specified input and then calculates the total seek time of our algorithm.
 - Output: Total Seek Time, Calculation for the total seek time.

```
D:\VIT\Fall sem 2017-2018\CSE2005\PROJECT\project.exe
-----
DISK SCHEDULING ALGORITHMS
-----
Enter Number Of Locations : 4
Enter Position Of Head : 10
Enter Elements In The Disk : 20
30
40
50
Enter the deadline of each of the following
1
4
5
9
20 -> 0
30 -> 1
40 -> 2
50 -> 3
TOTAL SEEK TIME : 40
Press any key to continue . . .
```

We have implemented our program on C++.

- First Individual Algorithms
- Then connectivity
- Last our own algorithm

OUTPUTS

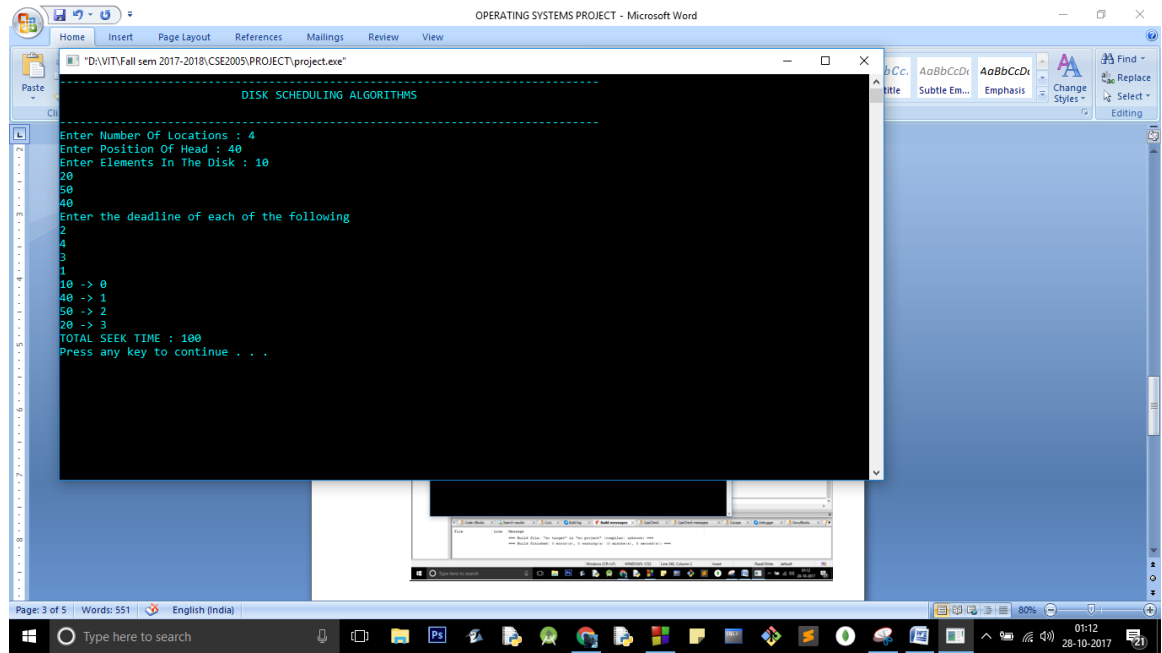


```
project.cpp - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\VIT\Fall sem 2017-2018\CSE2005\PROJECT\project.exe"
DISK SCHEDULING ALGORITHMS
1. Individual Analysis
2. Comparison
3. Check Best
4. Own Algorithm
5. EXIT
ENTER YOUR CHOICE : _
```

Build messages

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknown) ===
		=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 3 second(s)) ===

Windows (CR+LF) WINDOWS-1252 Line 348, Column 2 Insert Read/Write default 01:12 28-10-2017



```
OPERATING SYSTEMS PROJECT - Microsoft Word
Home Insert Page Layout References Mailings Review View
"D:\VIT\Fall sem 2017-2018\CSE2005\PROJECT\project.exe"
DISK SCHEDULING ALGORITHMS
Enter Number Of Locations : 4
Enter Position Of Head : 40
Enter Elements In The Disk : 10
20
50
40
Enter the deadline of each of the following
2
4
3
1
10 -> 0
40 -> 1
50 -> 2
20 -> 3
TOTAL SEEK TIME : 100
Press any key to continue . . .
```

Page: 3 of 5 Words: 551 English (India) 01:12 28-10-2017

```
"D:\WIT\Fall sem 2017-2018\CSE2005\PROJECT\project.exe"
-----
Enter Number Of Locations : 4
Enter Position Of Head : 20
Enter Elements In The Disk : 10
50
100
30

FCFS

Disk Head = 20
20 - 10 = 10
Disk Head = 10
50 - 10 = 40
Disk Head = 50
100 - 50 = 50
Disk Head = 100
100 - 30 = 70
Total Seek Time = 170

SCAN

Disk Head = 20
20 - 0 = 20
Disk Head = 0
30 - 0 = 30
Disk Head = 30
50 - 30 = 20
Disk Head = 50
100 - 50 = 50
Total Seek Time = 120
Press any key to continue . . .
```

IMPLEMENTATION TABLE

ALGORITHM	No. Of Locations	Head Position	Elements in Disk	Total Seek Time
FCFS	4	15	2,9,19,24	35
	5	95	15,65,89,124,155	220
	6	560	101,169,374,555,846,999	1357
SSTF	4	15	2,9,19,24	51
	5	95	15,65,89,124,155	330
	6	560	101,169,374,555,846,999	3063
SCAN	4	15	2,9,19,24	39
	5	95	15,65,89,124,155	250
	6	560	101,169,374,555,846,999	1559
C-SCAN	4	15	2,9,19,24	44
	5	95	15,65,89,124,155	304
	6	560	101,169,374,555,846,999	1993
LOOK	4	15	2,9,19,24	36
	5	95	15,65,89,124,155	134
	6	560	101,169,374,555,846,999	893
C-LOOK	4	15	2,9,19,24	24
	5	95	15,65,89,124,155	150
	6	560	101,169,374,555,846,999	1269

From the above tabulations, we get to know that LOOK scheduling program performs the best

ALGORITHM	No. Of Locations	Head Position	Elements in Disk	Deadline of elements	Total Seek Time
SSEDO	4	15	2,9,19,24	6,1,3,10	55
	5	65	21,36,59,75,88	11,7,23,37,2	144
SSEDV	4	15	2,9,19,24	6,1,3,10	68
	5	65	21,36,59,75,88	11,7,23,37,2	167
OWN	4	15	2,9,19,24	6,1,3,10	35
	5	65	21,36,59,75,88	11,7,23,37,2	136

From the above tabulations, we get to know that our own scheduling program performs the best

FUTURE ENHANCEMENTS

We will try to modify our algorithm to make it more better by decreasing its total seek time and the transaction loss ratio. The above results show that the current algorithm – is the best.

We can also use machine learning to make the system learn which algorithm to use when and for what types of inputs to get the best performance and this result can further be used to modify our algorithm further.

CONCLUSION

We have successfully shown the working of the common six disk scheduling algorithm as well as the other two algorithms i.e. SSEDV and SSEDV.

We also have created and successfully ran our own algorithm.

From the results obtained, we have come to the conclusion that _____ algorithm is the best for most of the cases.

Our own algorithm also showed a promisable performance against the previously proposed algorithms – SSEDV and SSEDV performing better than them in some of the cases.

REFERENCES

1. Performance Evaluation of two new disk scheduling algorithms for real time systems –
Shenze Chen, John A. Stankovic, James F. Kurose and Don Towsley
Department of Computer & Information Science – University of Massachusetts
2. Disk Scheduling with Quality of Service Guarantees
John Bruno, Jos6 Brustoloni, Eran Gabber, Banu Ozden and Abraham Silberschatz
Bell Laboratories, Lucent Technologies
3. Cello: A Disk Scheduling Framework for Next Generation Operating Systems
Prashant Shenoy(Department of Computer Science, University of Massachusetts at Amherst
University) Harrick M. Vin(Department of Computer Sciences, University of Texas at
Austin)
4. Krithi Ramamritham, “Real-Time Databases,” Journal of Distributed and Parallel Databases,
Vol. 1, No. 2, 1993,
pp.199-226., Kluwer Academic Publishers Hingham, MA,USA, April 1993.
5. Lund K, Goebel V., “Adaptive Disk Scheduling In A Multimedia DBMS”, Proceedings of
the eleventh ACM
international conference on Multimedia, Berkeley, CA, USA, 65 – 74, 2003.

6. Disk Scheduling Algorithms and presentation of mathematical model with database - Uma Shanker Jayswal, Gaurav Kumar Jain, Ravi Ranjan