Monte Carlo Simulation of Disk Scheduling Algorithms.

1. Problem Statement

Write a Monte Carlo simulation of the FIFO, LIFO, SSTF and SCAN disk scheduling algorithms in a C/C++ program. The algorithms should be run on a set of 50 to 150 requests increasing in steps of 10. The sector and track locations for each request are to be a uniform random distribution. At the start of an experiment, the cylinder head is to be located at track 100 and sector 0. A total of 1000 simulations are to be run for each I/O request count. Following are the HDD's operating parameters

Rotational Speed: 12000 RPM

Average Seek Time: 2.5ms

Transfer Rate: 6GB/s

Number of tracks: 201

Number of sectors: 360

Block Size: 4KB

The importance of the project is to learn the concepts of I/O request scheduling for a disk by an OS. And to learn the use of a Monte Carlo simulation to test multiple disk scheduling algorithms over a certain dataset and compare their performance relative to each other.

2. Approach

Language used: C++11

Compiler used: g++

Editor used: micro

Debugging tools used: gdb.

First task was writing the data structures that would aid in the process of simulating a disk scheduler's function. The data structures called 'dataPoint', 'sectorTrackPair' and 'simulation' were created so as to hold information for each simulation's outcome, the pair of sector and track location of a block of data and the latter to hold information for each batch of requests simulated.

Next task was implementing the disk scheduling algorithms and testing for correctness. The algorithms were implemented as concurrent threads in the same loop. Thus, we could think of the system as having 4 identical HDDs with the same data being accessed simultaneously. Each of the HDDs has their own unique scheduling algorithm and have the same set of requests coming in during a given simulation instance. As the disk drive services these I/O requests, the request time for each request is stored for each of the algorithms. Data from 1000 such simulations is then aggregated for each request count, and the average is taken before reporting.

Last task was creating the uniformly distributed random dataset required for the Monte Carlo simulation. Using the C++ STL, 'random', and the class contained within it called 'uniform_int_distribution' a randomly distributed dataset of int ints within a range is generated.

3. Solution

No major bugs were detected while implementing the algorithms mentioned earlier.

The solution is merely an adaptation of the algorithms described in the course textbook.

To build the code g++ is invoked as follows.

$ g++ -o prog5 main_rds190000.cc

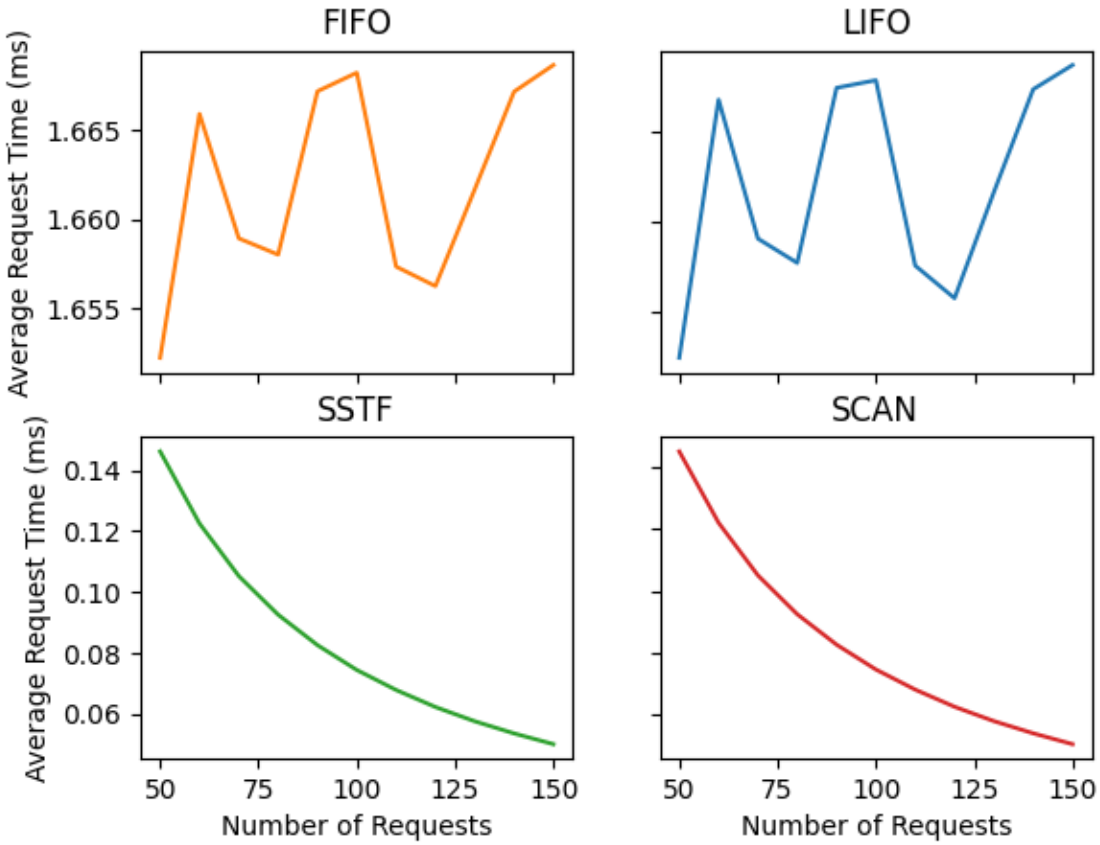
To run the code the following command is executed.

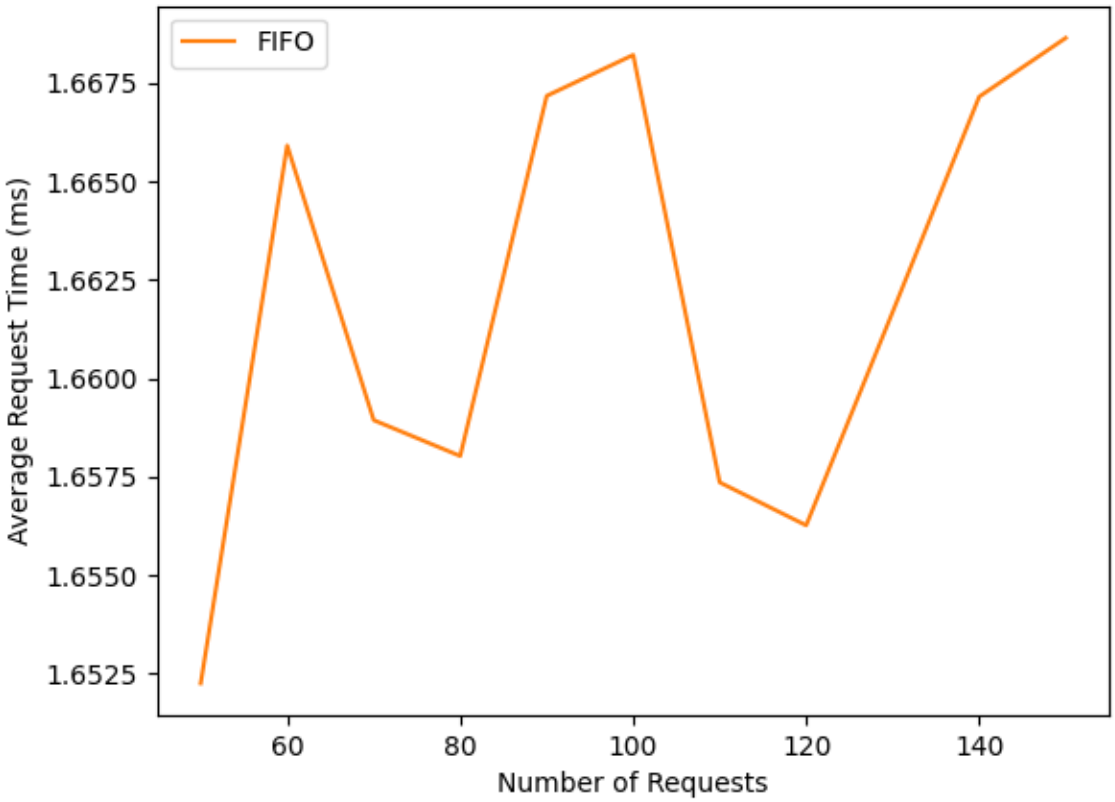
$ ./prog5

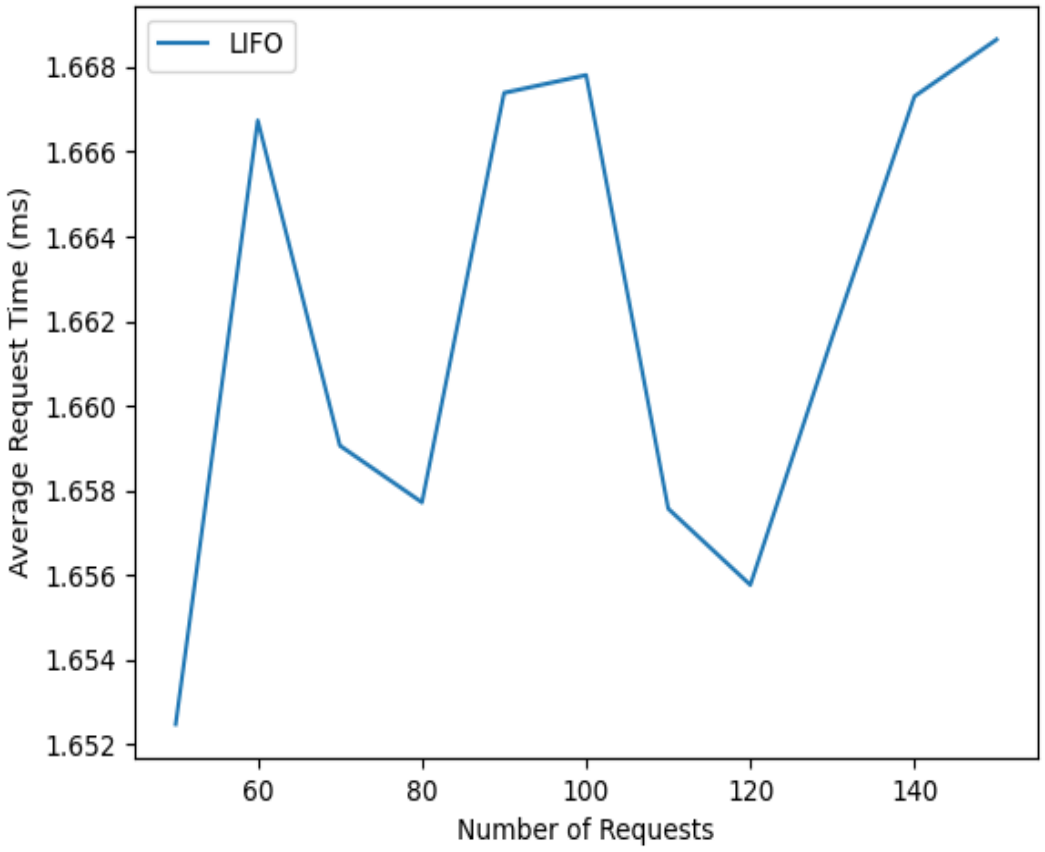
The code generates a data.csv file that contains the plot points. To parse this file and plot the graphs a python script is used.

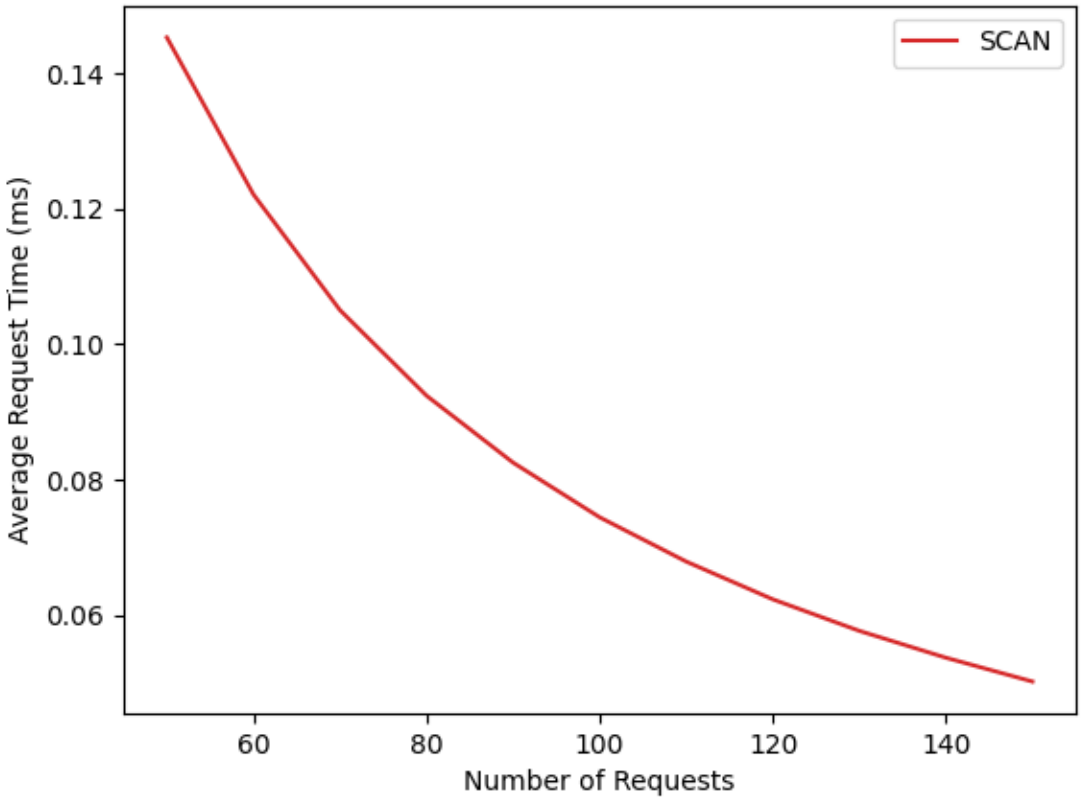
$ python3 plots_rds190000.py

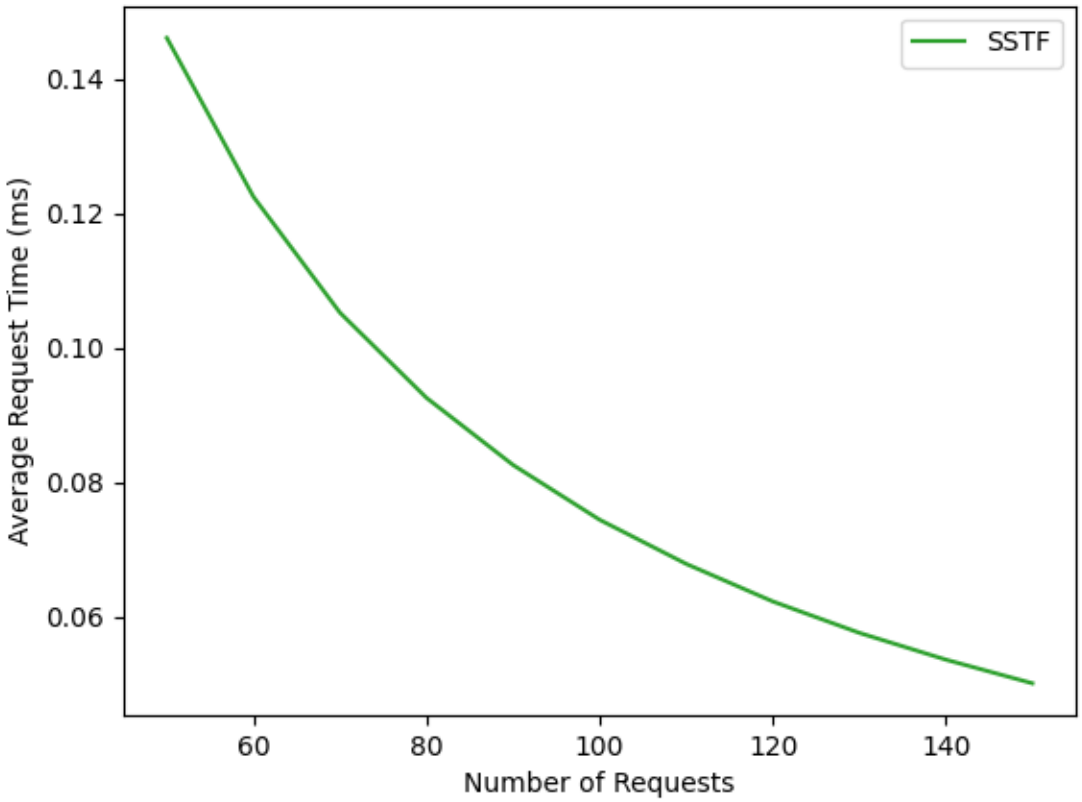
Comparison of the disc scheduling algorithms




FIFO
Average Request Time (ms)
1.6675
1.6650
1.6625
1.6600
1.6575
1.6550
1.6525
60
80
100
120
140
Number of Requests


LIFO
Average Request Time (ms)
1.668
1.666
1.664
1.662
1.660
1.658
1.656
1.654
1.652
60
80
100
120
140
Number of Requests

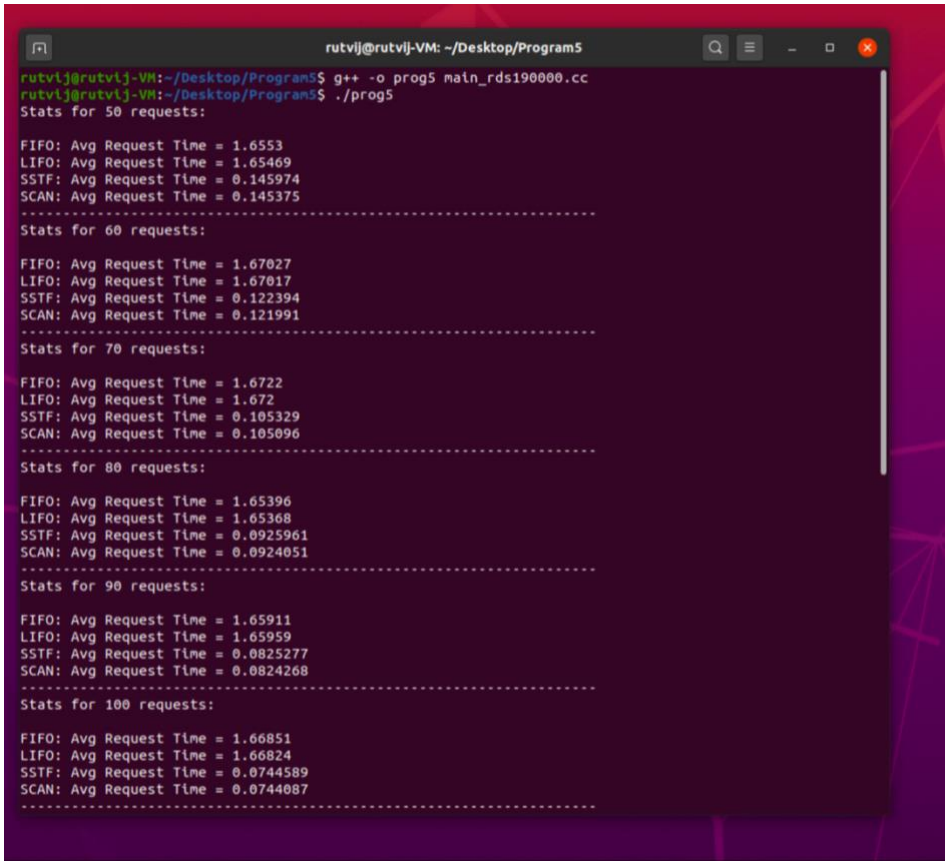

SCAN
0.14
0.12
0.10
0.08
0.06
Average Request Time (ms)
60
80
100
120
140
Number of Requests


SSTF
0.14
0.12
0.10
0.08
0.06
Average Request Time (ms)
60
80
100
120
140
Number of Requests

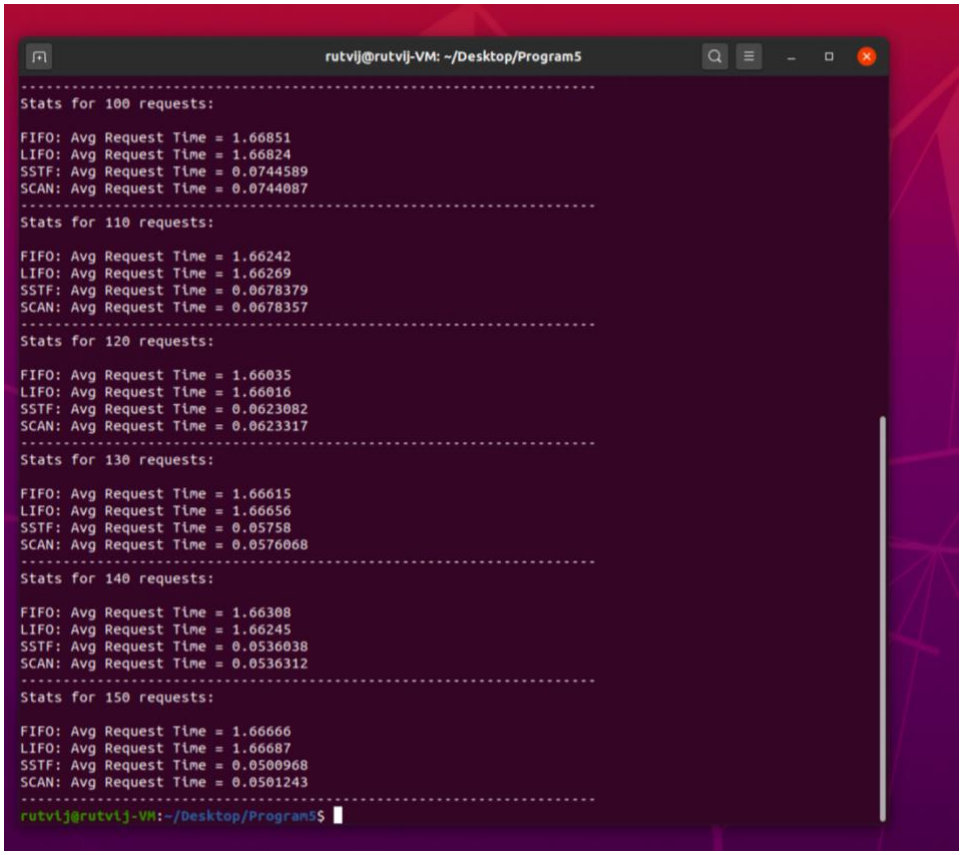
Screenshot of Program Correctness, Compilation and Execution

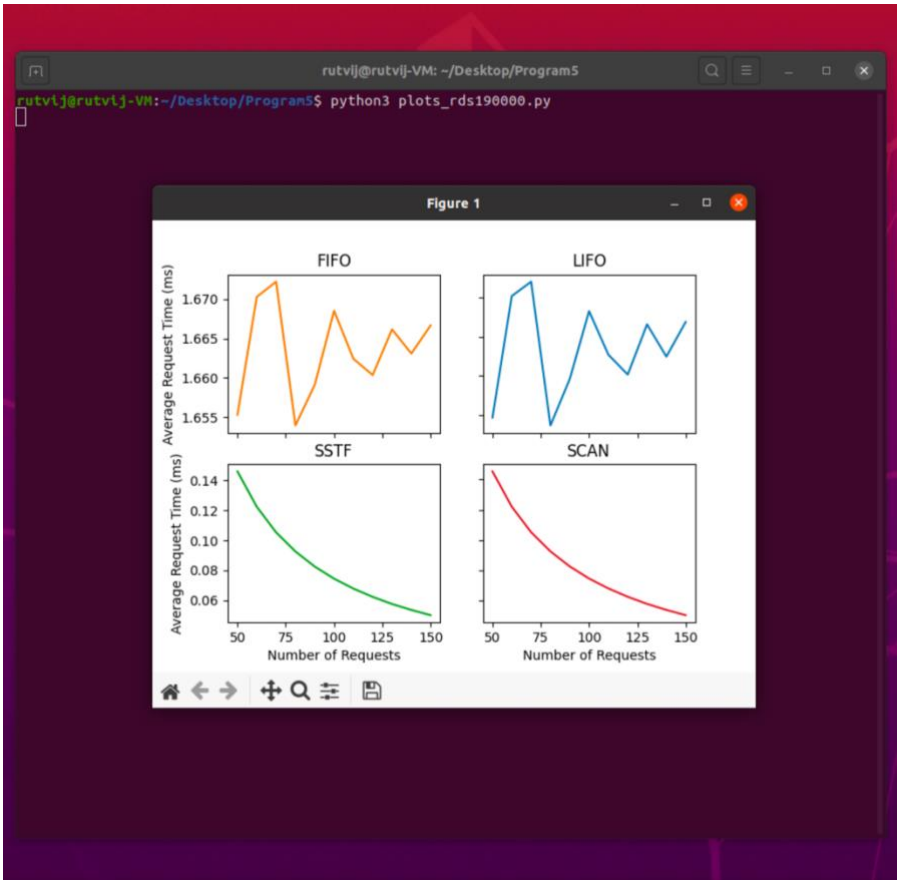
```
→  Program5 g++ main_rds190000.cc
→  Program5 ./a.out
0| FIFO = 45 | LIFO = 84 | SSTF = 10 | SCAN = 50 |
1| FIFO = 3 | LIFO = 146 | SSTF = 32 | SCAN = 10 |
2| FIFO = 19 | LIFO = 112 | SSTF = 3 | SCAN = 24 |
3| FIFO = 21 | LIFO = 10 | SSTF = 16 | SCAN = 94 |
4| FIFO = 72 | LIFO = 70 | SSTF = 1 | SCAN = 32 |
5| FIFO = 70 | LIFO = 72 | SSTF = 20 | SCAN = 3 |
6| FIFO = 10 | LIFO = 21 | SSTF = 132 | SCAN = 16 |
7| FIFO = 112 | LIFO = 19 | SSTF = 10 | SCAN = 1 |
8| FIFO = 146 | LIFO = 3 | SSTF = 24 | SCAN = 20 |


Avg seek length for FIFO = 55.3333
Avg seek length for LIFO = 59.6667
Avg seek length for SSTF = 27.5556
Avg seek length for SCAN = 27.7778
```

This picture has a modified version of the program being run to demonstrate the correctness of the algorithm by simulating the textbook data. The data that has been simulated and the output corresponds to the Table 11.2 of the textbook.



The program is compiled without any special compiler flags and the output is reported as shown. The csv file generated is then processed by a python script to generate the graphs.


rutvij@rutvij-VM: ~/Desktop/Program5
Stats for 100 requests:
FIFO: Avg Request Time = 1.66851
LIFO: Avg Request Time = 1.66824
SSTF: Avg Request Time = 0.0744589
SCAN: Avg Request Time = 0.0744087
Stats for 110 requests:
FIFO: Avg Request Time = 1.66242
LIFO: Avg Request Time = 1.66269
SSTF: Avg Request Time = 0.0678379
SCAN: Avg Request Time = 0.0678357
Stats for 120 requests:
FIFO: Avg Request Time = 1.66035
LIFO: Avg Request Time = 1.66016
SSTF: Avg Request Time = 0.0623082
SCAN: Avg Request Time = 0.0623317
Stats for 130 requests:
FIFO: Avg Request Time = 1.66615
LIFO: Avg Request Time = 1.66656
SSTF: Avg Request Time = 0.05758
SCAN: Avg Request Time = 0.0576068
Stats for 140 requests:
FIFO: Avg Request Time = 1.66308
LIFO: Avg Request Time = 1.66245
SSTF: Avg Request Time = 0.0536038
SCAN: Avg Request Time = 0.0536312
Stats for 150 requests:
FIFO: Avg Request Time = 1.66666
LIFO: Avg Request Time = 1.66687
SSTF: Avg Request Time = 0.0500968
SCAN: Avg Request Time = 0.0501243
rutvij@rutvij-VM:~/Desktop/Program5$


rutvij@rutvij-VM: ~/Desktop/Program5
rutvij@rutvij-VM:~/Desktop/Program5$ python3 plots_rds190000.py
Figure 1
FIFO
LIFO
SSTF
SCAN
Average Request Time (ms)
1.670
1.665
1.660
1.655
0.14
0.12
0.10
0.08
0.06
50
75
100
125
150
Number of Requests