

DISK SCHEDULING ALGORITHM

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.

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.

There are various Disk Scheduling algorithm such as:

1 FCFS

2 SSTF

3 SCAN

4 CSCAN

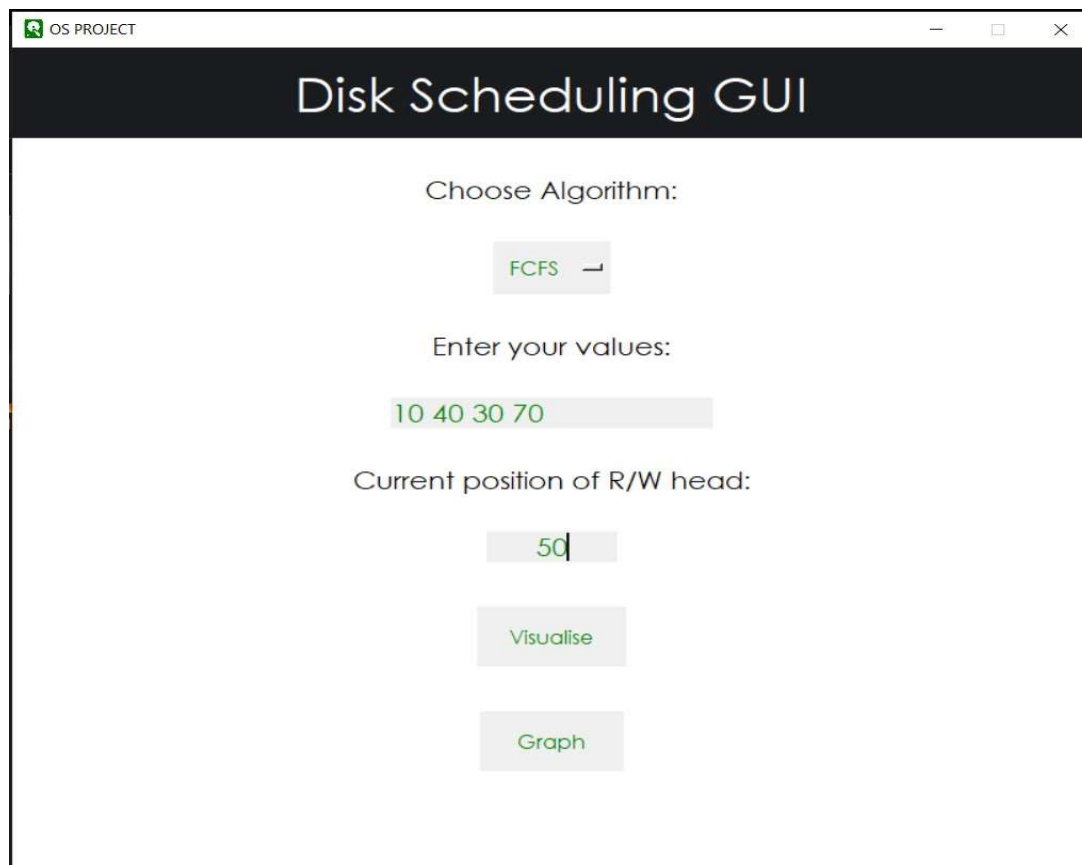
5 LOOK

6 CLOOK

THE MAIN SCREEN

The main screen of the project consists of:

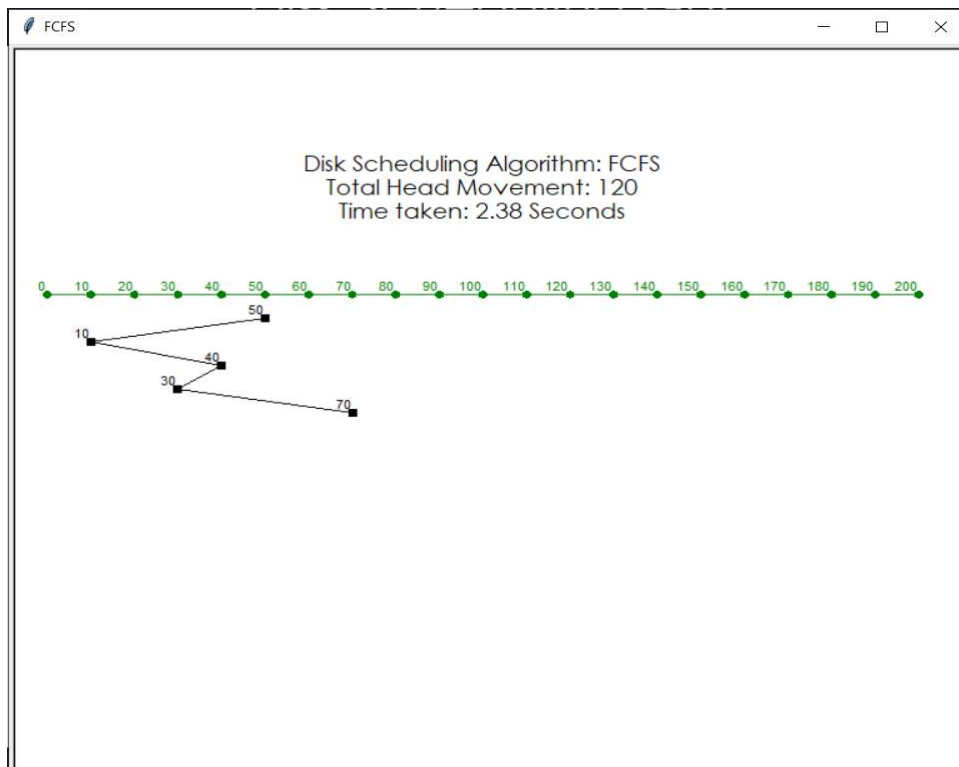
- 1 Choose Algorithm: User can choose from a set of algorithms
- 2 Enter Value: User provides the request array here
- 3 Current position of R/W head: User can assign the current head position
- 4 Visualize: It will help User to visualize the algorithm
- 5 Graph: It will give the Graph of the Algorithm



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.

In this example request array is-(10,40,30,70)



So, the Total seek time is $(50-10) + (40-10) + (40-30) + (70-30) = 120$

Advantages:

- Every request gets a fair chance
- No indefinite postponement

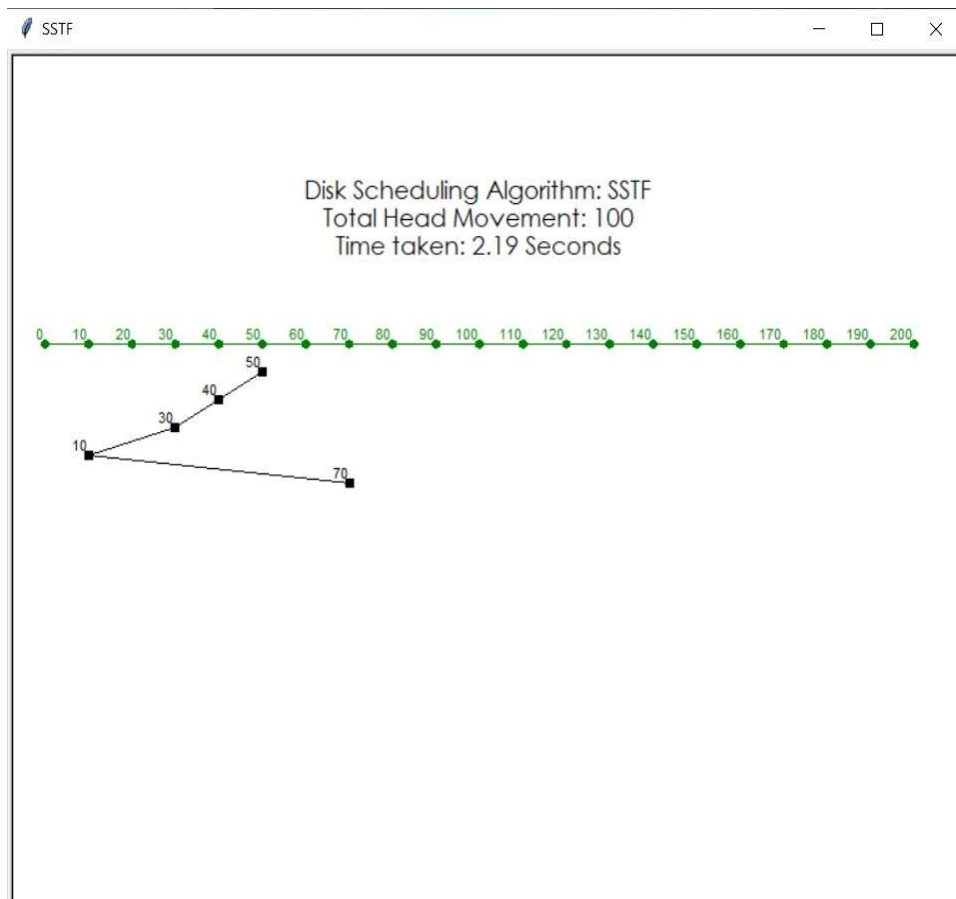
Disadvantages:

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

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 the 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.

In this example request array is-(10,40,30,70)



So, the Total seek time is $(50-40) + (40-30) + (30-10) + (70-10) = 100$

Advantages:

- Average Response Time decreases
- Throughput increases

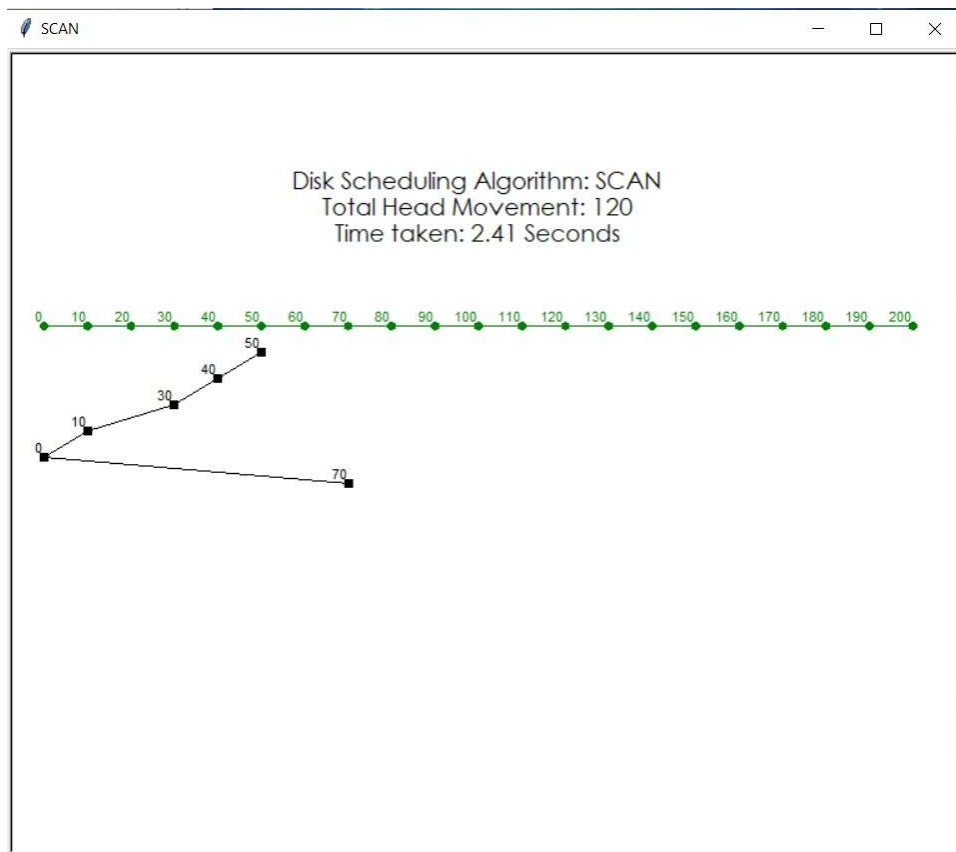
Disadvantages:

- 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 favors only some requests

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 as 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.

In this example request array is-(10,40,30,70)



So, the Total seek time is $(50-40) + (40-30) + (30-10) + (10-0) + (70-0) = 120$

Advantages:

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

Disadvantages:

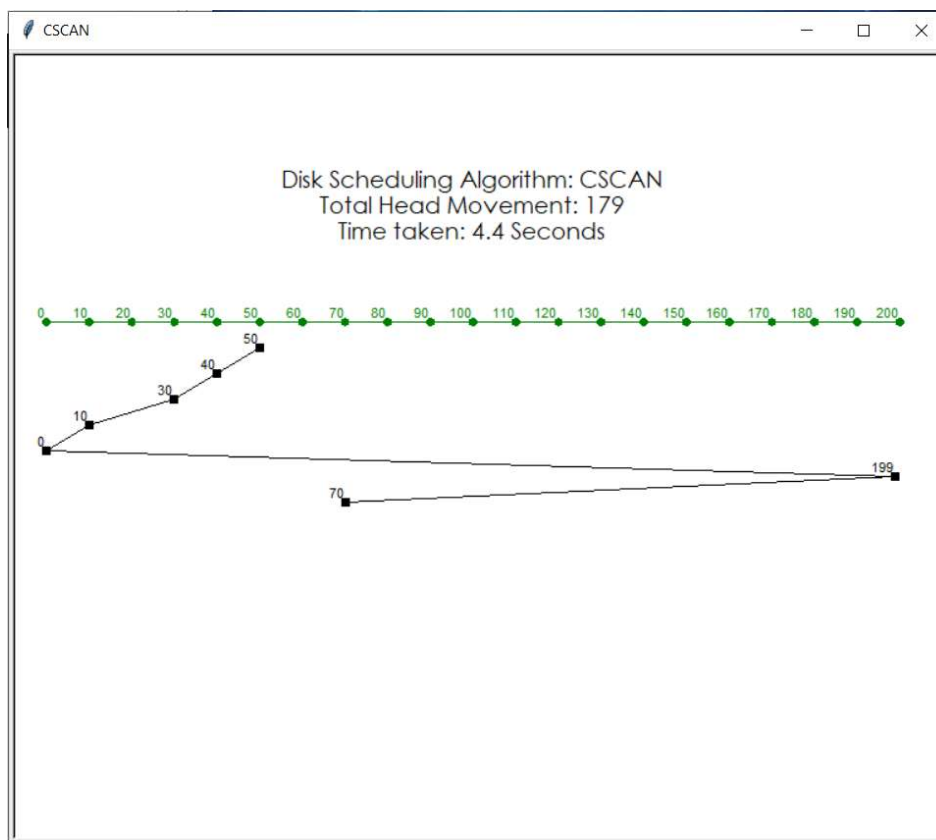
- Long waiting time for requests for locations just visited by disk arm

CSCAN

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 *CSCAN* 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).

In this example request array is-(10,40,30,70)



So, the Total seek time is $(50-40) + (40-30) + (30-10) + (10-0) + (199-70) = 179$

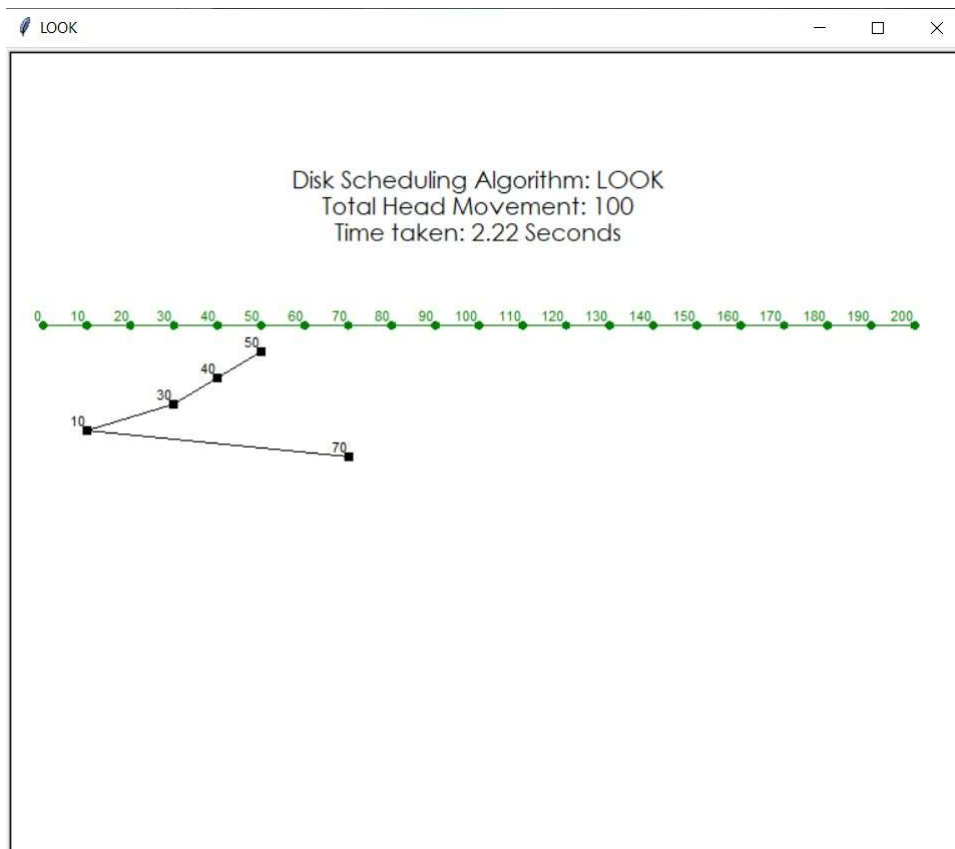
Advantages:

- Provides more uniform wait time compared to SCAN

LOOK

It is similar to the SCAN disk scheduling algorithm except for 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.

In this example request array is-(10,40,30,70)

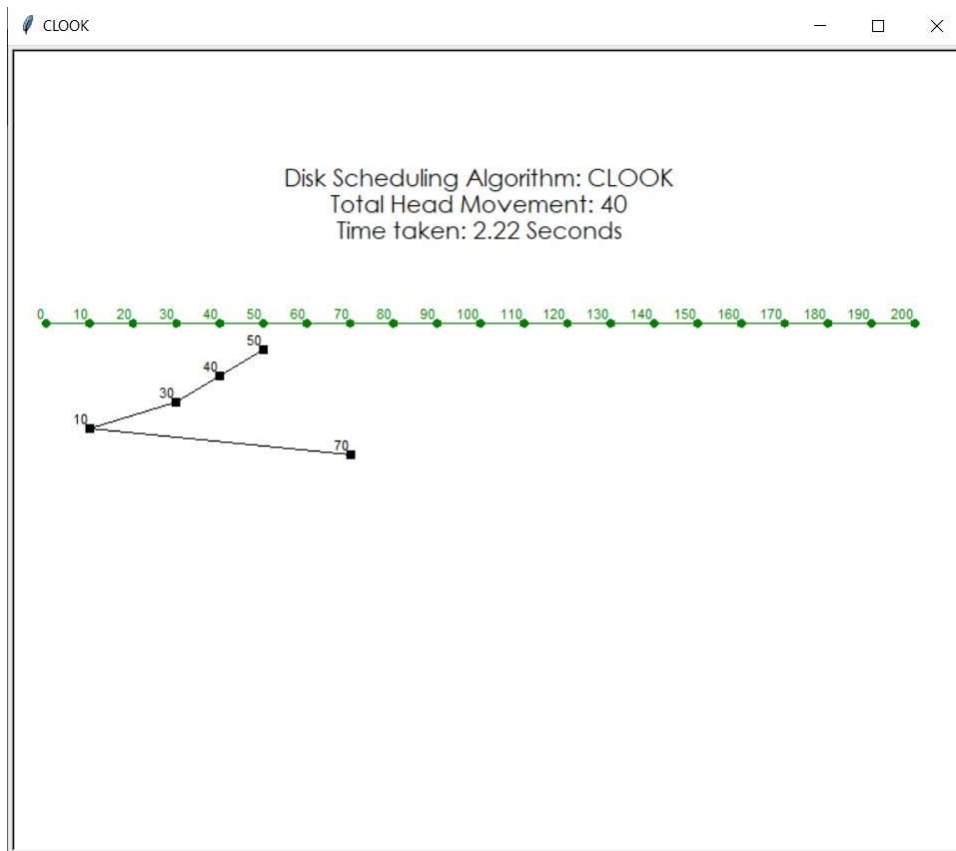


So, the Total seek time is $(50-40) + (40-30) + (30-10) + (70-10) = 100$

CLOOK

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.

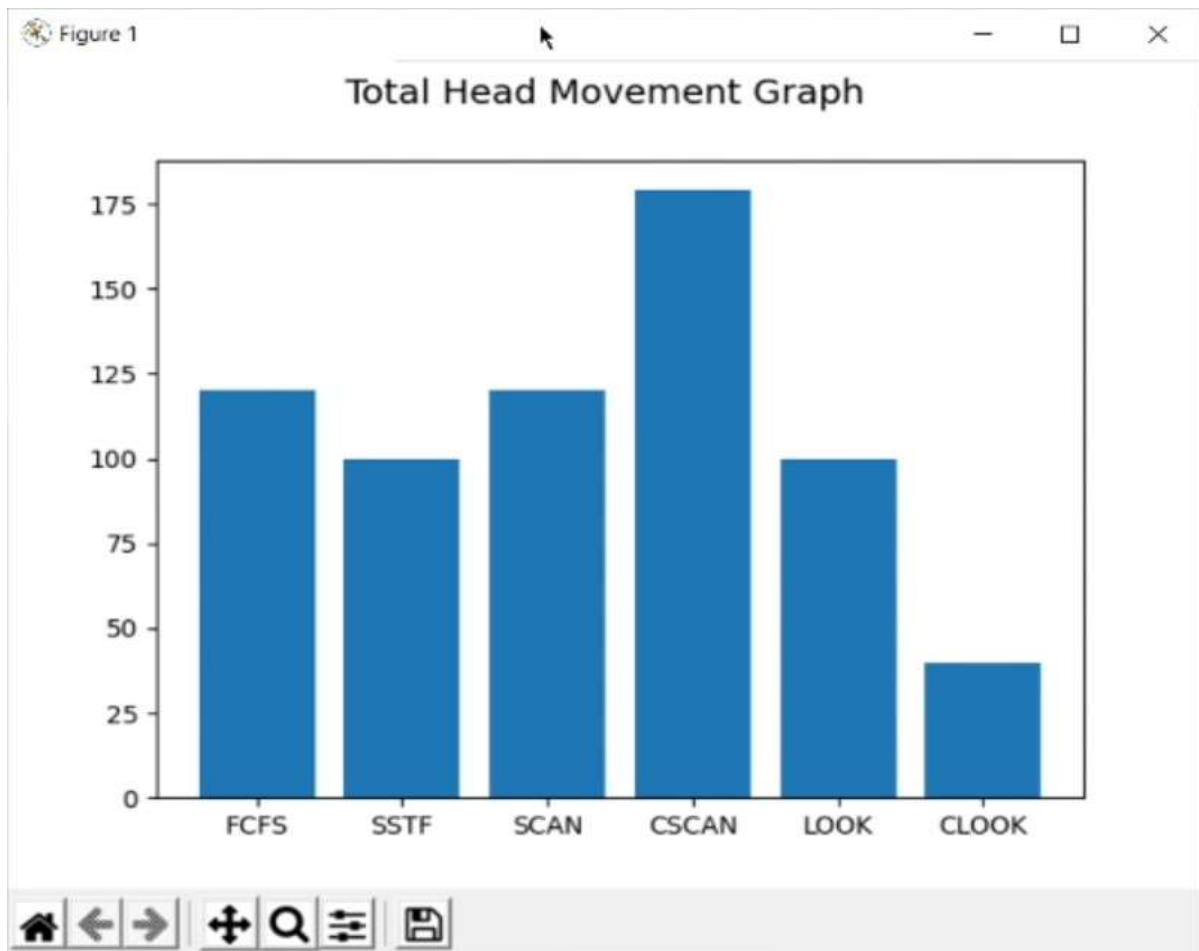
In this example request array is-(10,40,30,70)



So, the Total Seek Time =

THE GRAPH

The Graph will show the user the total number of Head Movements in all the Algorithms.



PAGE REPLACEMENT ALGORITHM

In a computer operating system that uses paging for virtual memory management, page replacement algorithms decide which memory pages to page out, sometimes called swap out, or write to disk, when a page of memory needs to be allocated. Page replacement happens when a requested page is not in memory (page fault) and a free page cannot be used to satisfy the allocation, either because there are none, or because the number of free pages is lower than some threshold.

When the page that was selected for replacement and paged out is referenced again it has to be paged in (read in from disk), and this involves waiting for I/O completion. This determines the *quality* of the page replacement algorithm: the less time waiting for page-ins, the better the algorithm. A page replacement algorithm looks at the limited information about accesses to the pages provided by hardware, and tries to guess which pages should be replaced to minimize the total number of page misses, while balancing this with the costs (primary storage and processor time) of the algorithm itself.

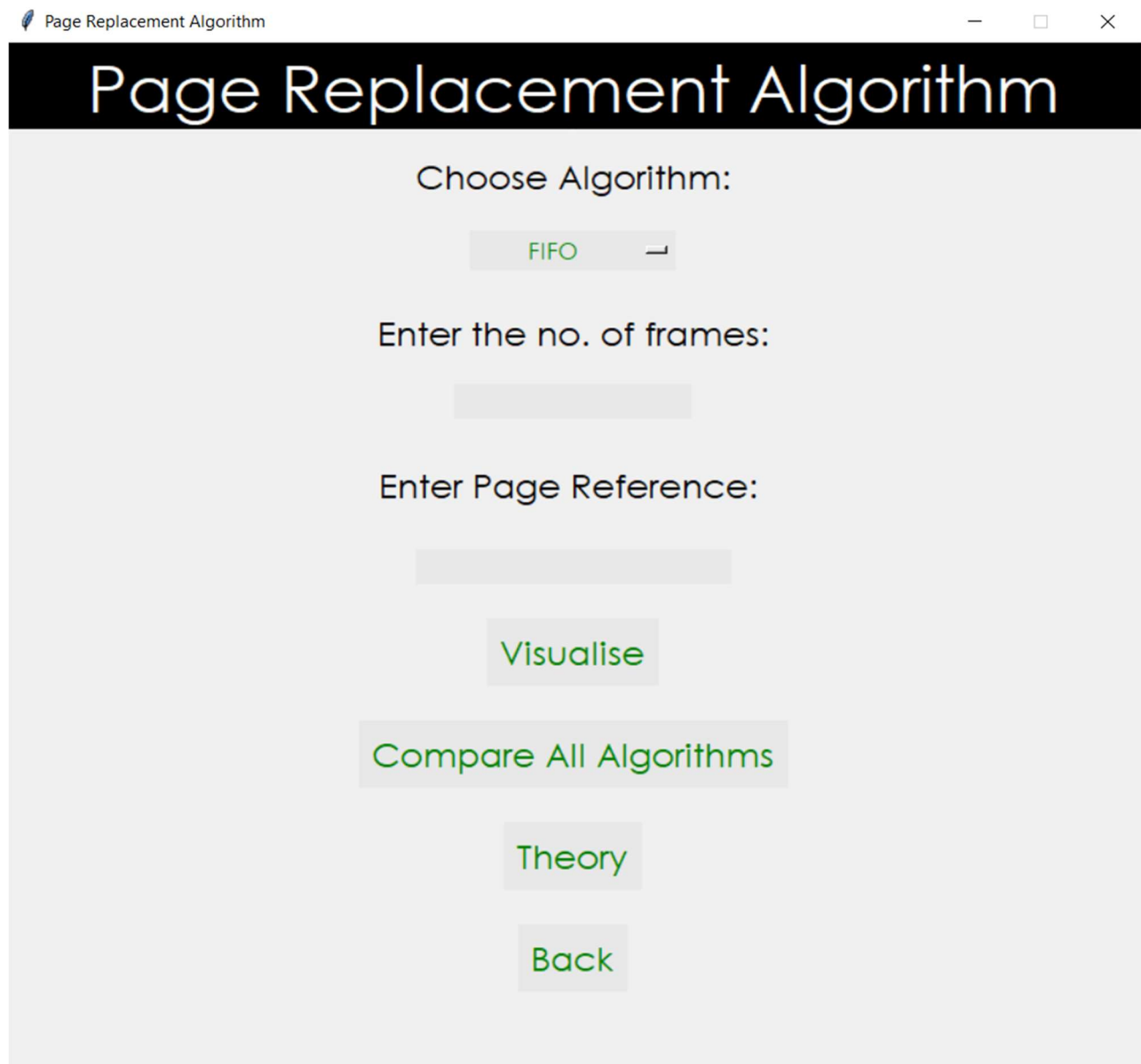
There are various Disk Scheduling algorithm such as:

1. FIFO
2. LIFO
3. Optimal PRA
4. Random PRA
5. MRU
6. LRU

THE MAIN SCREEN

The main screen of the project consists of:

- 1 Choose Algorithm: User can choose from a set of algorithms
- 2 Enter the no. of frames: User provides the no. of frames
- 3 Enter Page Reference: User provides the request array here
- 4 Visualize: It will help User to visualize the algorithm
- 5 Compare All Algorithms: It will give the Graph of the Algorithm
- 6 Theory: It will help User to understand the algorithm



The screenshot shows a web application titled "Page Replacement Algorithm". The interface is centered and features the following elements:

- Choose Algorithm:** A dropdown menu with "FIFO" selected.
- Enter the no. of frames:** An empty input field.
- Enter Page Reference:** An empty input field.
- Visualise** button.
- Compare All Algorithms** button.
- Theory** button.
- Back** button.

First In First Out (FIFO):

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced, page in the front of the queue is selected for removal.

Example:

Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames.

Visualisation Of Algorithm: First In First Out

Reference String	1	3	0	3	5	6
Frame 3			0	0	0	0
Frame 2		3	3	3	3	6
Frame 1	1	1	1	1	5	5
Page Faults	Fault	Fault	Fault	Hit	Fault	Fault

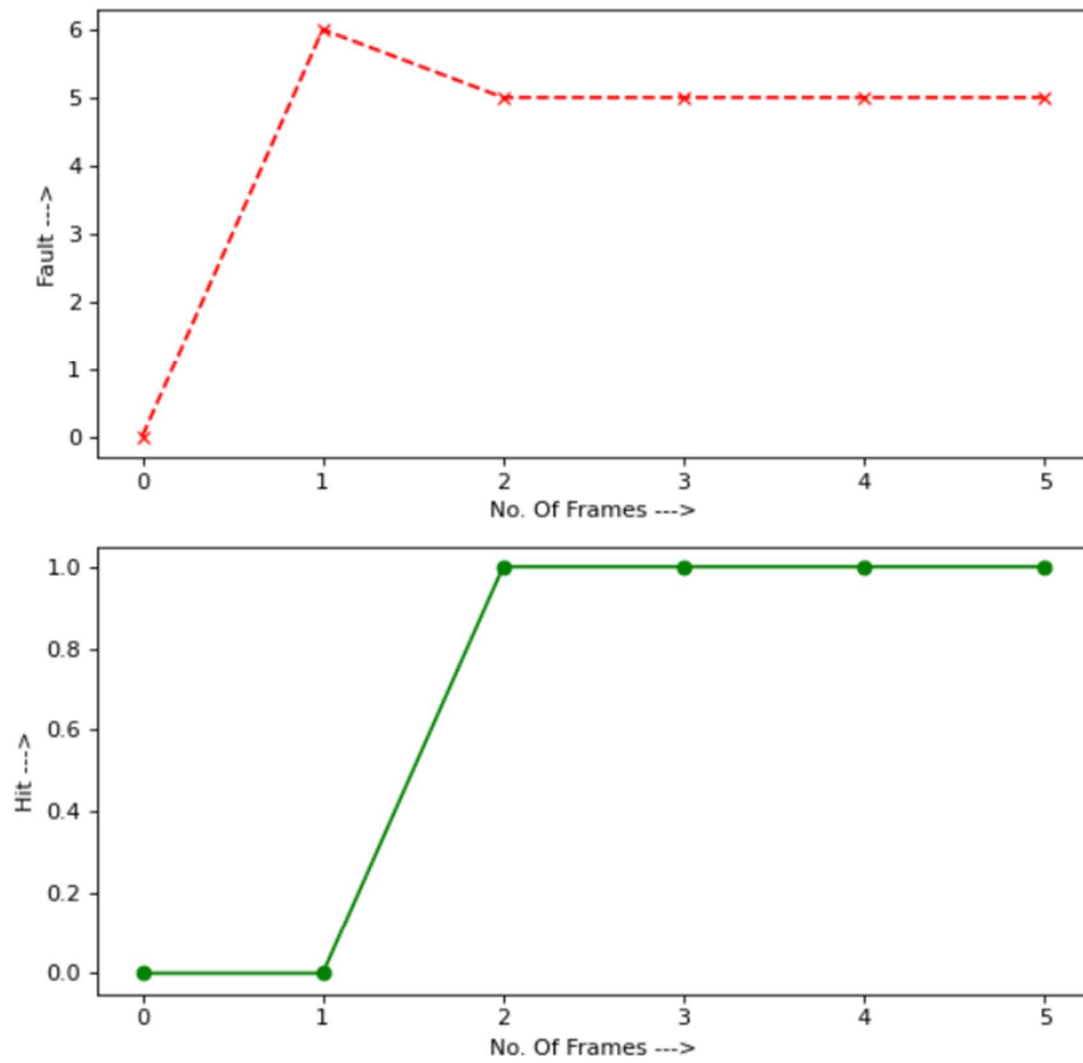
First In First Out Page Fault Ratio

Hit Ratio: = 0.1666666667

Fault Ratio: = 0.8333333333

- Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults**.
- When 3 comes, it is already in memory so —> **0 Page Faults**.
- Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —> **1 Page Fault**.
- 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —> **1 Page Fault**.

Figure 1

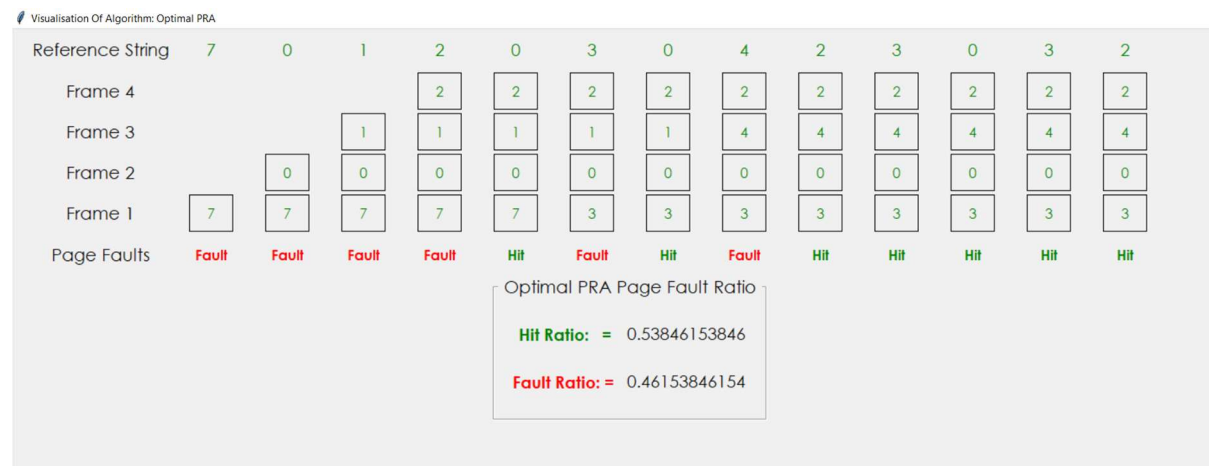


Optimal Page replacement:

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example:

Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame.

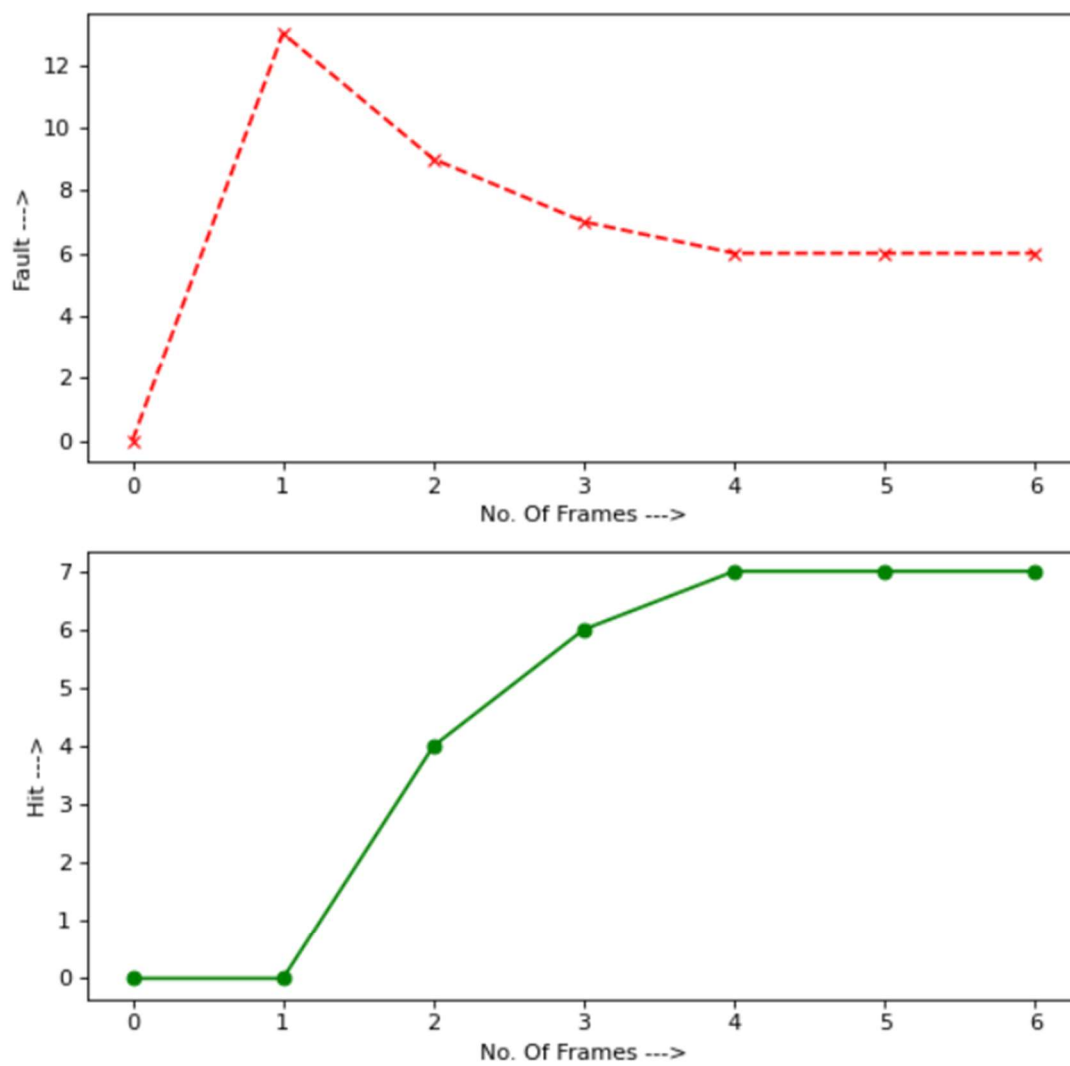


- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
- 0 is already there so —> **0 Page fault.**
- When 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**
- 0 is already there so —> **0 Page fault..**
- 4 will takes place of 1 —> **1 Page Fault.**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

Figure 1



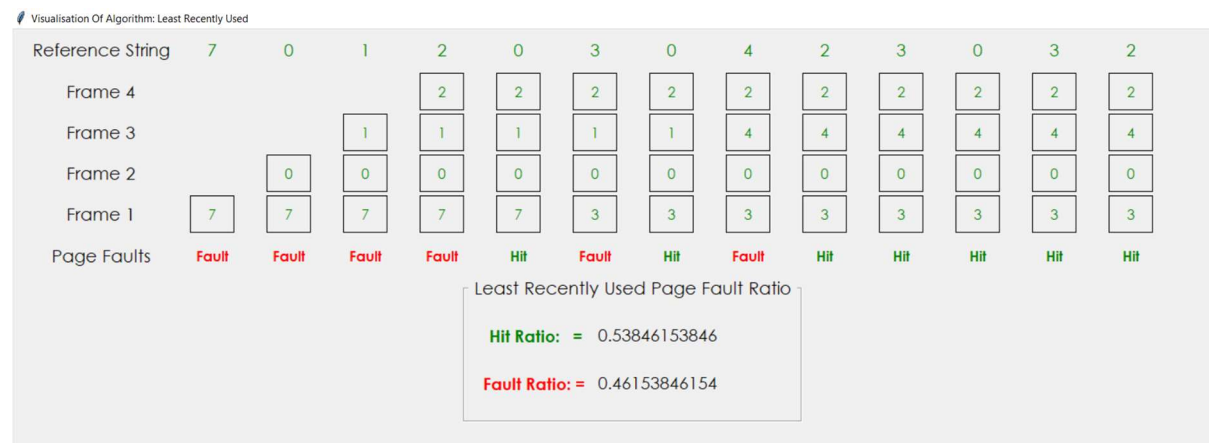
x=5.45 y=5.24

Least Recently Used:

In this algorithm page will be replaced which is least recently used. Page which has not been used for the longest time in main memory is the one which will be selected for replacement

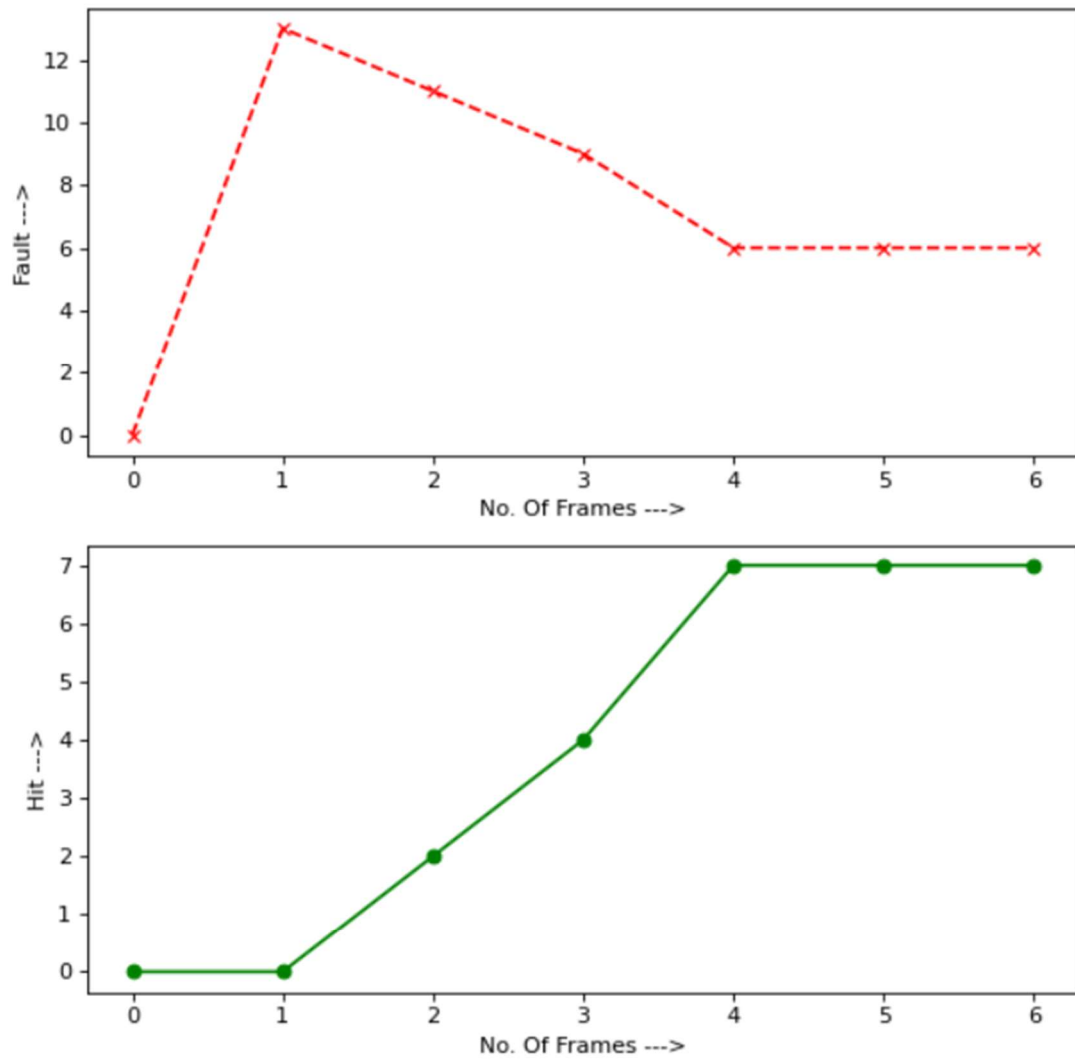
Example:

Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames.



- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
- 0 is already there so —> **0 Page fault.**
- When 3 came it will take the place of 7 because it is least recently used —> **1 Page fault**
- 0 is already in memory so —> **0 Page fault.**
- 4 will take place of 1 —> **1 Page Fault**
- Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Figure 1



Last In First Out (LIFO):

In this algorithm, the operating system keeps track of all pages in the memory in a de-queue, the oldest page is in the front of the queue. When a page needs to be replaced, page at the end of the queue is selected for removal.

Example:

Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames.

Visualisation Of Algorithm: Last In First Out

Reference String	1	3	0	3	5	6
Frame 3			0	0	5	6
Frame 2		3	3	3	3	3
Frame 1	1	1	1	1	1	1
Page Faults	Fault	Fault	Fault	Hit	Fault	Fault

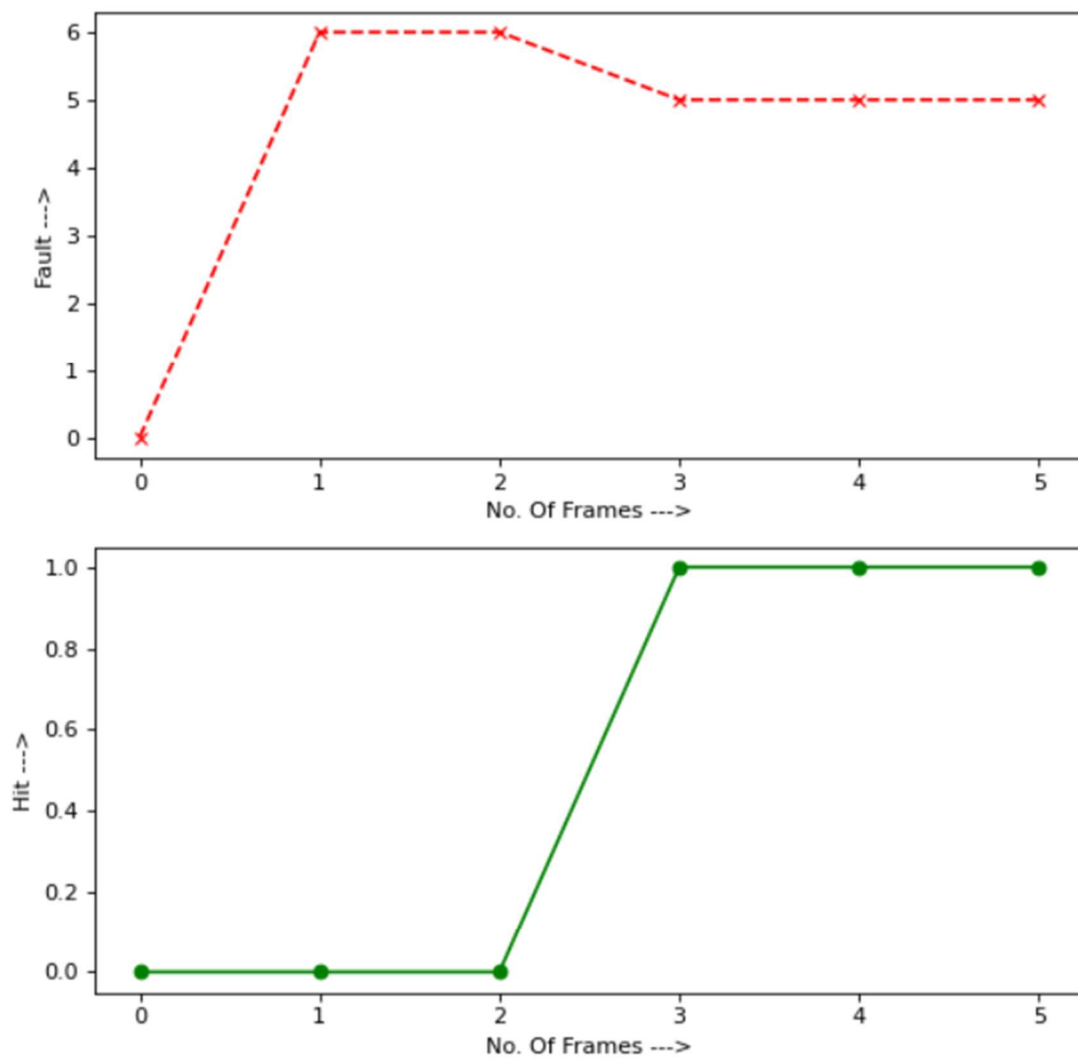
Last In First Out Page Fault Ratio

Hit Ratio: = 0.1666666667

Fault Ratio: = 0.8333333333

- Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
- When 3 comes, it is already in memory so —> **0 Page Faults.**
- Then 5 comes, it is not available in memory so it replaces the newest page slot i.e 0. —> **1 Page Fault.**
- 6 comes, it is also not available in memory so it replaces the newest page slot i.e 5 —> **1 Page Fault.**

Figure 1

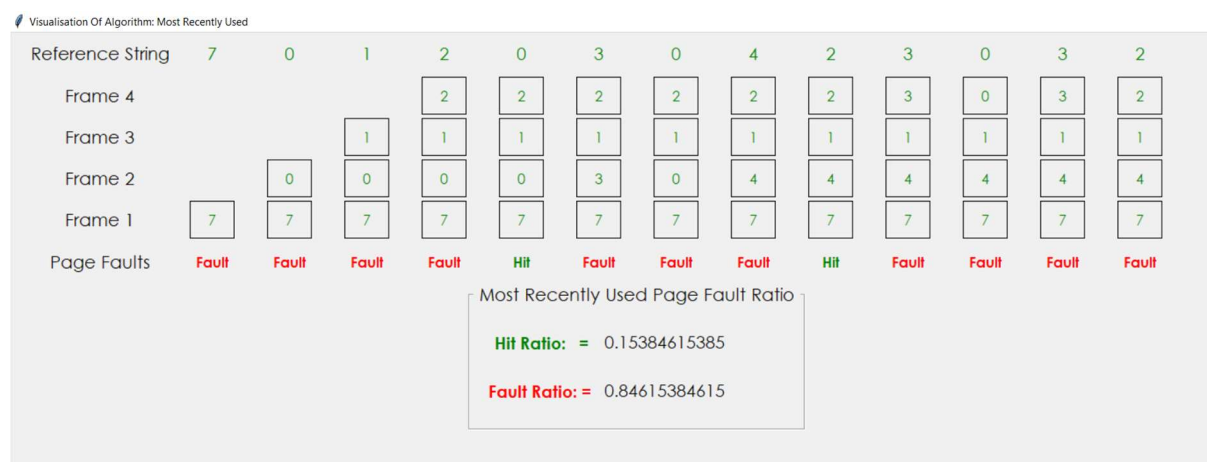


Most Recently Used:

In this algorithm page will be replaced which is most recently used. Page which has not been used for the lowest time in main memory is the one which will be selected for replacement

Example:

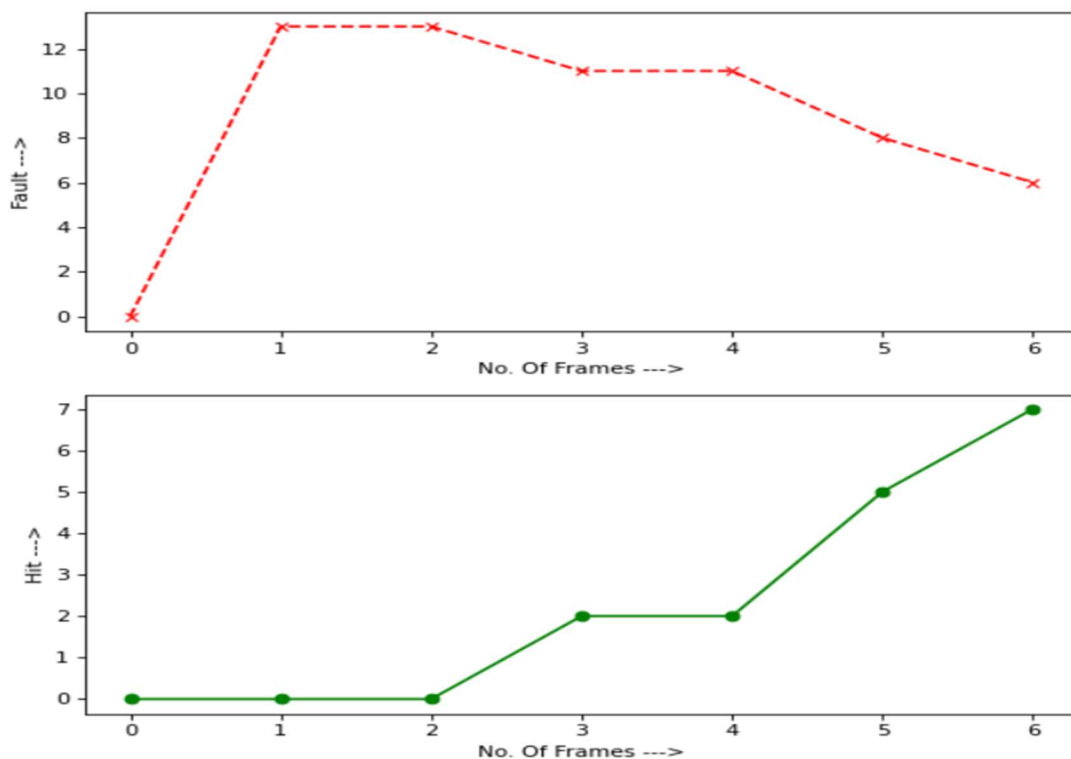
Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames.



- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots
—> **4 Page faults**
- 0 is already there so —> **0 Page fault.**
- When 3 came it will take the place of 0 because it was most recently used
—> **1 Page fault**
- When 0 came it will take the place of 3 because it was most recently used
—> **1 Page fault**
- When 4 came it will take the place of 0 because it was most recently used
—> **1 Page fault**
- 2 is already there so —> **0 Page fault.**
- When 3 came it will take the place of 2 because it was most recently used
—> **1 Page fault**

- When 0 came it will take the place of 3 because it was most recently used —> **1 Page fault**
- When 0 came it will take the place of 3 because it was most recently used —> **1 Page fault**
- When 3 came it will take the place of 0 because it was most recently used —> **1 Page fault**
- When 2 came it will take the place of 3 because it was most recently used —> **1 Page fault**

Figure 1

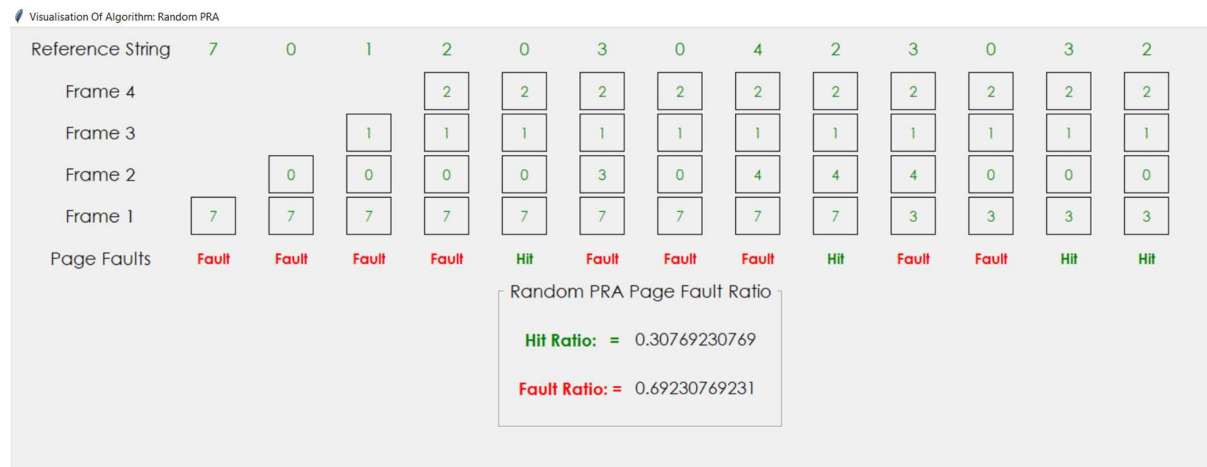
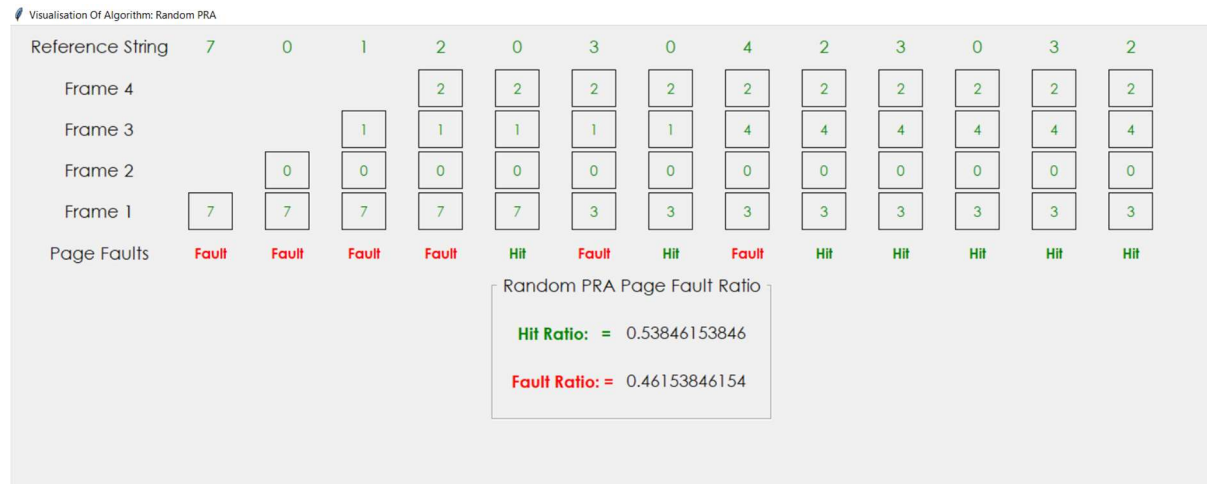


Random Page replacement:

In this algorithm, pages are replaced randomly in any order.

Example:

Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame.



THE GRAPH

The Graph will show the user the total Fault Ratio in all the Algorithms.

