**Exp No.** 13 **Name:** PONNURI ANIRUDDHA

**Date**: 26/10/2022 **Reg No:** RA2112704010015

**Disk Scheduling algorithm**

**Aim: --**

To implement Disk Scheduling algorithm

**Procedure: --**

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.

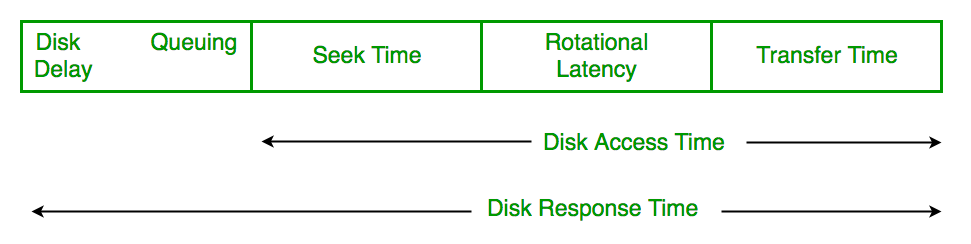
Disk scheduling is important because: 

* 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 request 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.

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.

**Disk Scheduling Algorithms**

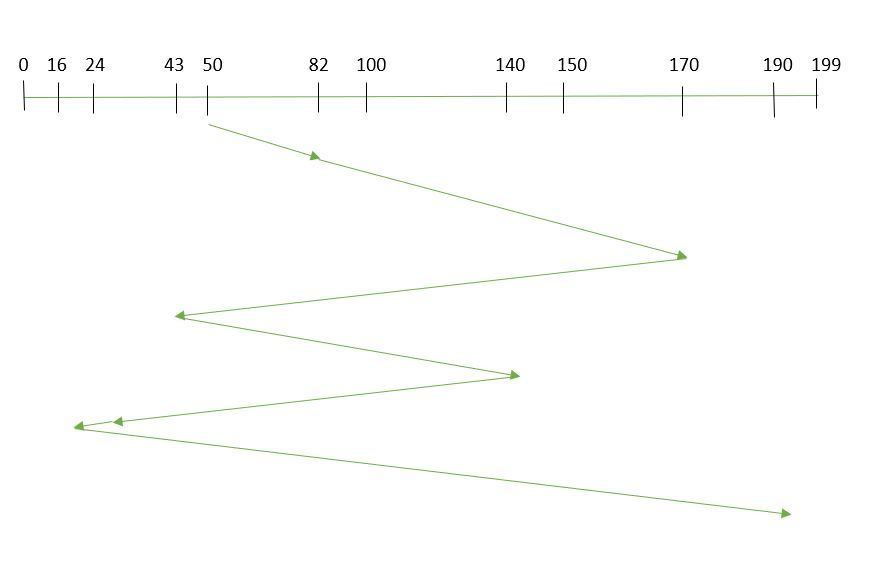
**FCFS:**

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. Let us understand this with the help of an example.

**Algorithm:**

* Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
* Let us one by one take the tracks in default order and calculate the absolute distance of the track from the head.
* Increment the total seek count with this distance.
* Currently serviced track position now becomes the new head position.
* Go to step 2 until all tracks in request array have not been serviced.

Example:

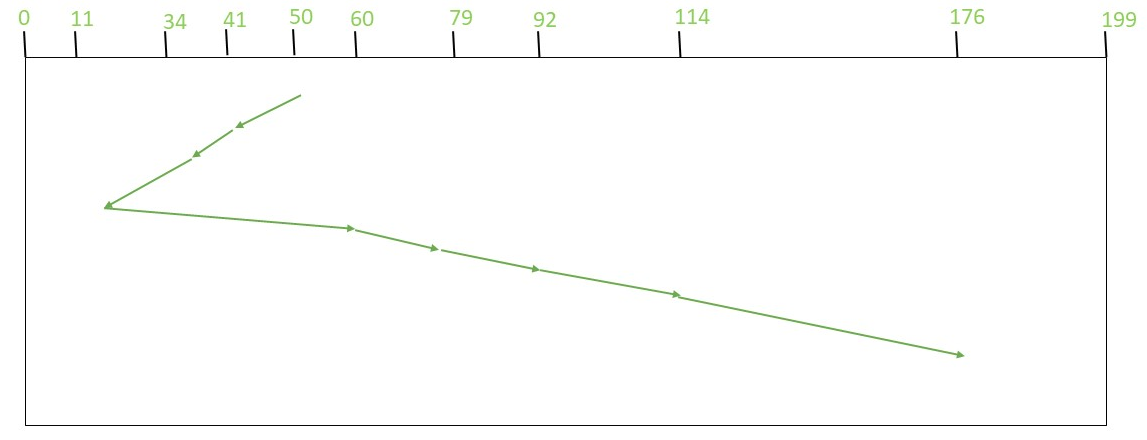
Suppose the order of request is- (82,170,43,140,24,16,190)  
And current position of Read/Write head is: 50

**Shortest Seek Time First:**

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 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. Let us understand this with the help of an example.

Example:

Suppose the order of request is- (176, 79, 34, 60, 92, 11, 41, 114)  
And current position of Read/Write head is : 50 



So, total seek time:

= (50-41)+(41-34)+(34-11)+(60-11)+(79-60)+(92-79)+(114-92)

+(176-114)

= 204

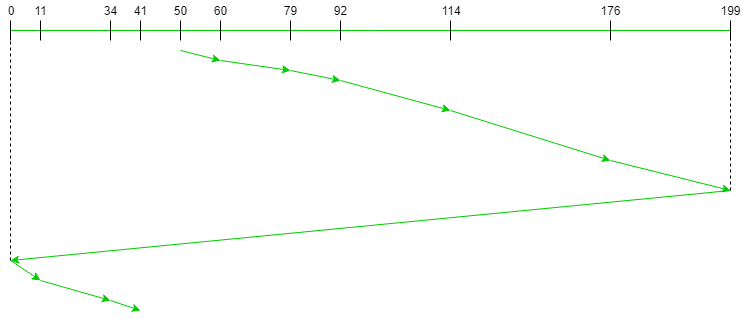
**CSCAN:**

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 CSCANalgorithm 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 (Circulars SCAN).

**Example:**

Suppose the requests to be addressed are- 176, 79, 34, 60, 92, 11, 41, 114. And the Read/Write arm is at 50, and it is also given that the disk arm should move “towards the larger value”.



Therefore, the total seek count is calculated as: 

= (60-50)+(79-60)+(92-79) + (114-92)+(176-114)+(199-176)+(199-0)

+(11-0)+(34-11)+(41-34)

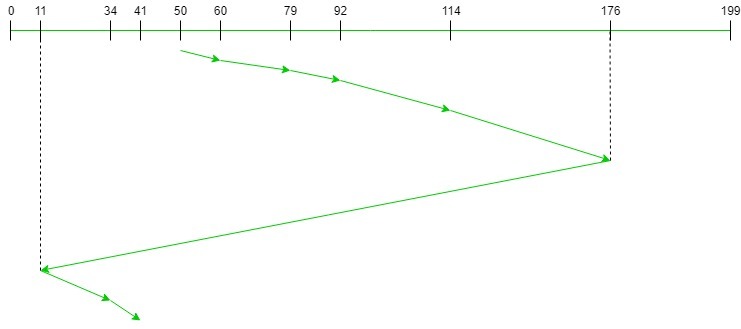
= 389

**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.

Example:

Suppose the requests to be addressed are- 176, 79, 34, 60, 92, 11, 41, 114. And the Read/Write arm is at 50 and it is also given that the disk arm should move “towards the larger value”



Therefore, the total seek count

= (60 – 50) + (79 – 60) + (92 – 79) + (114 – 92) + (176 – 114)

+ (176 – 11) + (34 – 11) + (41 – 34)

= 321

**CODE (FCFS): --**

*def FCFS(arr, head):*

*seek\_count = 0;*

*distance, cur\_track = 0, 0;*

*for i in range(size):*

*cur\_track = arr[i]*

*# calculate absolute distance*

*distance = abs(cur\_track - head)*

*# increase the total count*

*seek\_count += distance*

*# accessed track is now new head*

*head = cur\_track*

*print("Total number of seek operations = ",seek\_count)*

*# Seek sequence would be the same*

*# as request array sequence*

*print("Seek Sequence is");*

*for i in range(size):*

*print(arr[i]);*

*# Driver code*

*if \_\_name\_\_ == '\_\_main\_\_':*

*# request array*

*arr = []*

*size = int(input("Enter size of request array:-- "))*

*for i in range(size):*

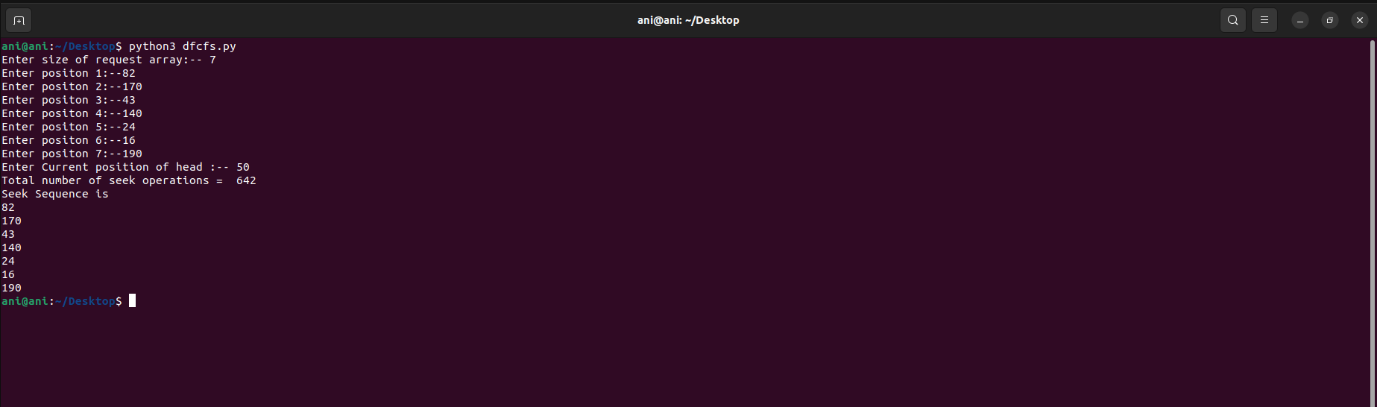
*x = int(input(f'Enter positon {i+1}:--'))*

*arr.append(x)*

*head = int(input("Enter Current position of head :-- "))*

*FCFS(arr, head);*

**OUTPUT (FCFS): --**



**CODE (SSTF): --**

*def calculateDifference(queue, head, diff):*

*for i in range(len(diff)):*

*diff[i][0] = abs(queue[i] - head)*

*# find unaccessed track which is*

*# at minimum distance from head*

*def findMin(diff):*

*index = -1*

*minimum = 999999999*

*for i in range(len(diff)):*

*if (not diff[i][1] and*

*minimum > diff[i][0]):*

*minimum = diff[i][0]*

*index = i*

*return index*

*def shortestSeekTimeFirst(request, head):*

*if (len(request) == 0):*

*return*

*l = len(request)*

*diff = [0] \* l*

*# initialize array*

*for i in range(l):*

*diff[i] = [0, 0]*

*# count total number of seek operation*

*seek\_count = 0*

*# stores sequence in which disk*

*# access is done*

*seek\_sequence = [0] \* (l + 1)*

*for i in range(l):*

*seek\_sequence[i] = head*

*calculateDifference(request, head, diff)*

*index = findMin(diff)*

*diff[index][1] = True*

*# increase the total count*

*seek\_count += diff[index][0]*

*# accessed track is now new head*

*head = request[index]*

*# for last accessed track*

*seek\_sequence[len(seek\_sequence) - 1] = head*

*print("Total number of seek operations =",*

*seek\_count)*

*print("Seek Sequence is")*

*# print the sequence*

*for i in range(l + 1):*

*print(seek\_sequence[i])*

*# Driver code*

*if \_\_name\_\_ =="\_\_main\_\_":*

*# request array*

*arr = []*

*size = int(input("Enter size of request array:-- "))*

*for i in range(size):*

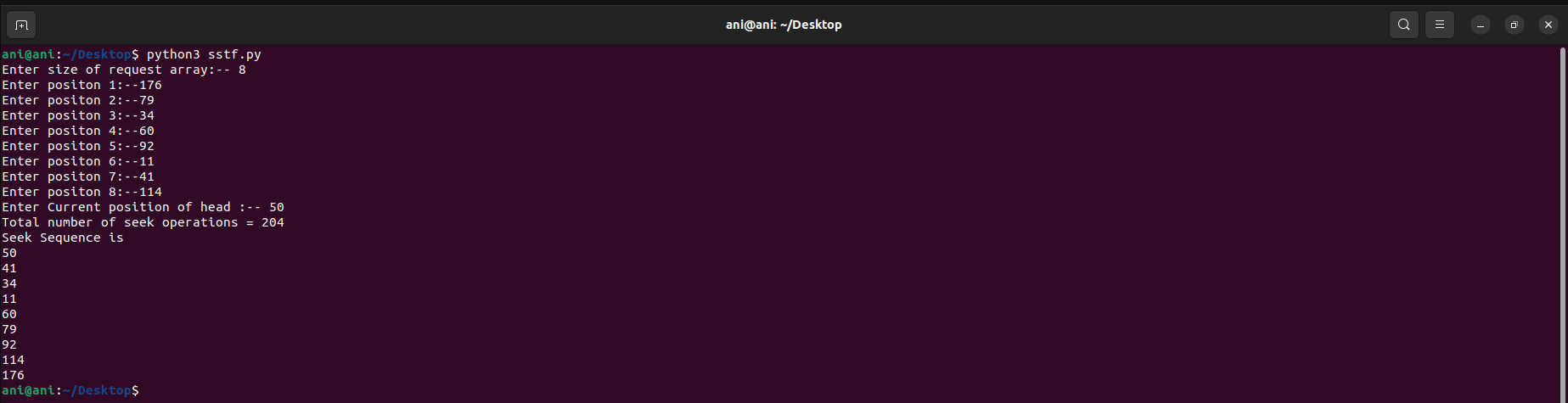
*x = int(input(f'Enter positon {i+1}:--'))*

*arr.append(x)*

*head = int(input("Enter Current position of head :-- "))*

*shortestSeekTimeFirst(arr,head)*

**OUTPUT (SSTF): --**



**CODE(CSCAN): --**

*disk\_size = 200*

*def CSCAN(arr, head):*

*seek\_count = 0*

*distance = 0*

*cur\_track = 0*

*left = []*

*right = []*

*seek\_sequence = []*

*left.append(0)*

*right.append(disk\_size - 1)*

*for i in range(size):*

*if (arr[i] < head):*

*left.append(arr[i])*

*if (arr[i] > head):*

*right.append(arr[i])*

*# Sorting left and right vectors*

*left.sort()*

*right.sort()*

*# First service the requests*

*# on the right side of the*

*# head.*

*for i in range(len(right)):*

*cur\_track = right[i]*

*# Appending current track*

*# to seek sequence*

*seek\_sequence.append(cur\_track)*

*# Calculate absolute distance*

*distance = abs(cur\_track - head)*

*# Increase the total count*

*seek\_count += distance*

*# Accessed track is now new head*

*head = cur\_track*

*# Once reached the right end*

*# jump to the beggining.*

*head = 0*

*# adding seek count for head returning from 199 to 0*

*seek\_count += (disk\_size - 1)*

*# Now service the requests again*

*# which are left.*

*for i in range(len(left)):*

*cur\_track = left[i]*

*# Appending current track*

*# to seek sequence*

*seek\_sequence.append(cur\_track)*

*# Calculate absolute distance*

*distance = abs(cur\_track - head)*

*# Increase the total count*

*seek\_count += distance*

*# Accessed track is now the new head*

*head = cur\_track*

*print("Total number of seek operations =",seek\_count)*

*print("Seek Sequence is")*

*print(\*seek\_sequence, sep="\n")*

*# Driver code*

*if \_\_name\_\_ =="\_\_main\_\_":*

*# request array*

*arr = []*

*size = int(input("Enter size of request array:-- "))*

*for i in range(size):*

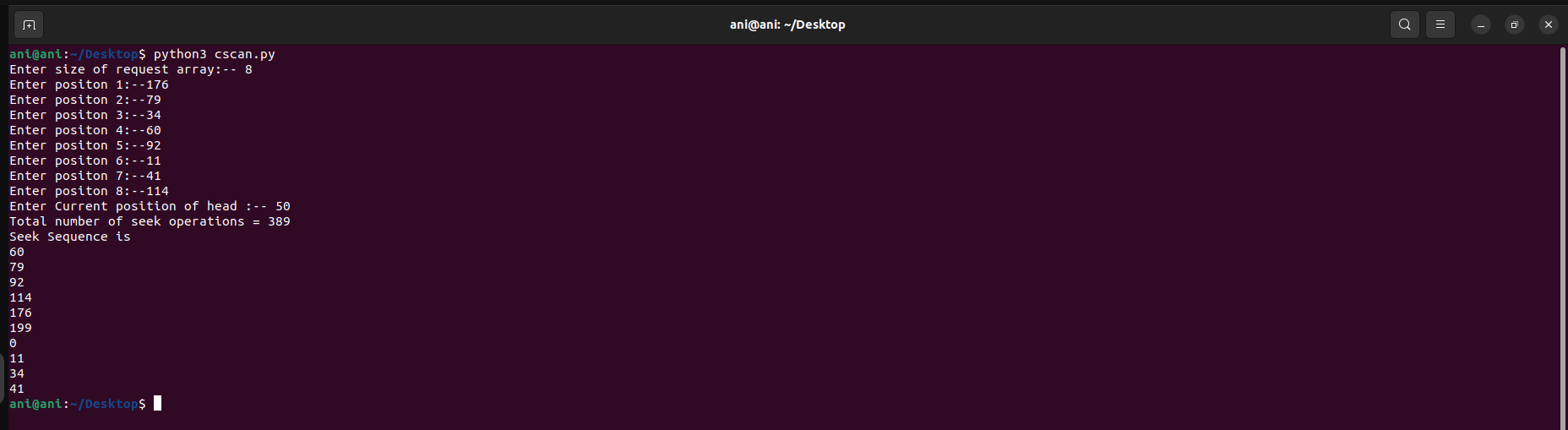
*x = int(input(f'Enter positon {i+1}:--'))*

*arr.append(x)*

*head = int(input("Enter Current position of head :-- "))*

*CSCAN(arr,head)*

**OUTPUT (CSCAN): --**



**CODE (CLOOK): --**

*disk\_size = 200*

*def CLOOK(arr, head):*

*seek\_count = 0*

*distance = 0*

*cur\_track = 0*

*left = []*

*right = []*

*seek\_sequence = []*

*for i in range(size):*

*if (arr[i] < head):*

*left.append(arr[i])*

*if (arr[i] > head):*

*right.append(arr[i])*

*# Sorting left and right vectors*

*left.sort()*

*right.sort()*

*# First service the requests*

*# on the right side of the*

*# head*

*for i in range(len(right)):*

*cur\_track = right[i]*

*# Appending current track*

*# seek sequence*

*seek\_sequence.append(cur\_track)*

*# Calculate absolute distance*

*distance = abs(cur\_track - head)*

*# Increase the total count*

*seek\_count += distance*

*# Accessed track is now new head*

*head = cur\_track*

*# Once reached the right end*

*# jump to the last track that*

*# is needed to be serviced in*

*# left direction*

*seek\_count += abs(head - left[0])*

*head = left[0]*

*# Now service the requests again*

*# which are left*

*for i in range(len(left)):*

*cur\_track = left[i]*

*# Appending current track to*

*# seek sequence*

*seek\_sequence.append(cur\_track)*

*# Calculate absolute distance*

*distance = abs(cur\_track - head)*

*# Increase the total count*

*seek\_count += distance*

*# Accessed track is now the new head*

*head = cur\_track*

*print("Total number of seek operations =",*

*seek\_count)*

*print("Seek Sequence is")*

*for i in range(len(seek\_sequence)):*

*print(seek\_sequence[i])*

*# Driver code*

*if \_\_name\_\_ =="\_\_main\_\_":*

*# request array*

*arr = []*

*size = int(input("Enter size of request array:-- "))*

*for i in range(size):*

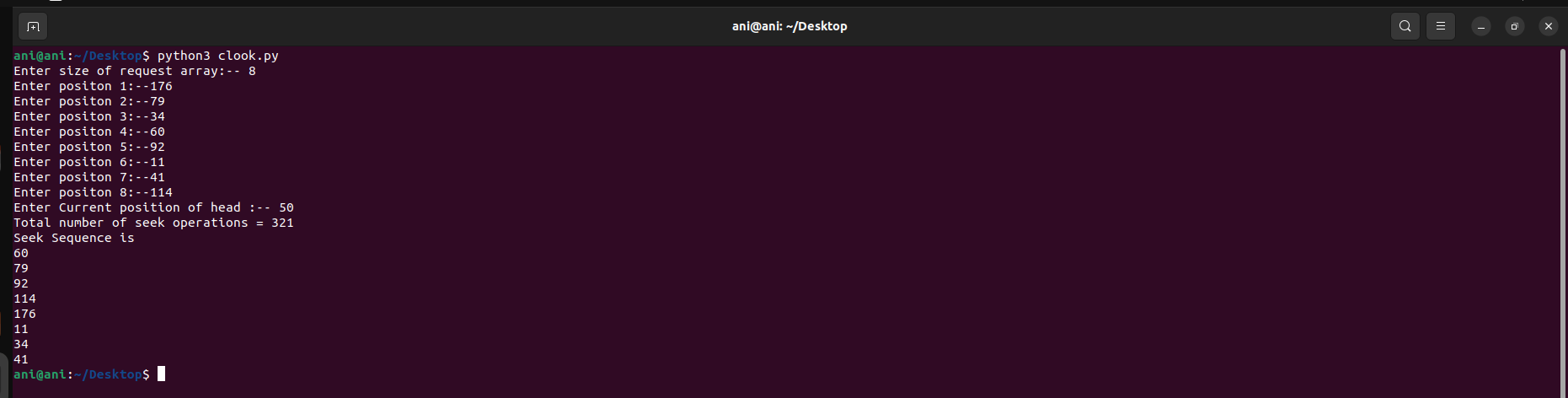
*x = int(input(f'Enter positon {i+1}:--'))*

*arr.append(x)*

*head = int(input("Enter Current position of head :-- "))*

*CLOOK(arr,head)*

**OUTPUT (CLOOK): --**



**RESULT: --**

Successfully implemented disk scheduling algorithms in python.