|  |  |  |  |
| --- | --- | --- | --- |
| **Fr. Conceicao Rodrigues College of Engineering**  **Department of Computer Engineering** | | | |
| **Student’s Roll No** |  | **Students Name** |  |
| **Date of Performance** |  | **SE Computer – Div** | **A / B** |

**Aim:** Study Disk Management

**Lab Outcome:**

**CSL403.6:** Implement various Disk Management techniques and evaluate their performance.

**Problem Statements:**

Implement Disk Management Algorithms

(a)FCFS (b)SSTF (c )SCAN

Given the current head position and future disk block references wrt tracks or cylinders. Calculate the seek length based on above algorithms. Show the sequence in which the disk blocks will be accessed and no of tracks traversed in each algorithm.

**References:**

[**https://www.geeksforgeeks.org/disk-scheduling-algorithms/**](https://www.geeksforgeeks.org/disk-scheduling-algorithms/)

[**https://www.youtube.com/watch?v=9uoa\_p8q47Y**](https://www.youtube.com/watch?v=9uoa_p8q47Y)

|  |  |  |  |
| --- | --- | --- | --- |
| **On time Submission(2)** | **Knowledge of Topic(4)** | **Implementation and Demonstraion(4)** | **Total (10)** |
|  |  |  |  |
| **Signature of Faculty** |  | **Date of Submission** |  |

FCFS:

def FCFS(head, sequence):

    if not sequence:

        return 0

    seek\_operations = 0

    for i in sequence:

        if i!= head:

            print(head, " ==> ",i, end=" ")

            print("\n")

            difference = abs(head - i)

            seek\_operations += difference

            head = i

            print(i)

    return seek\_operations

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

    Number\_disk = int(input("Enter the number of disks: "))

    if Number\_disk > 0:

        head = int(input("Enter initial header position: "))

        while not head in range(Number\_disk + 1):

            head = int(input("Please enter valid initial head position: "))

        sequence = []

        sequence = list(map(int, input("Enter the sequence: ").split()))

        for i in sequence:

            if i < 0 or i >= Number\_disk:

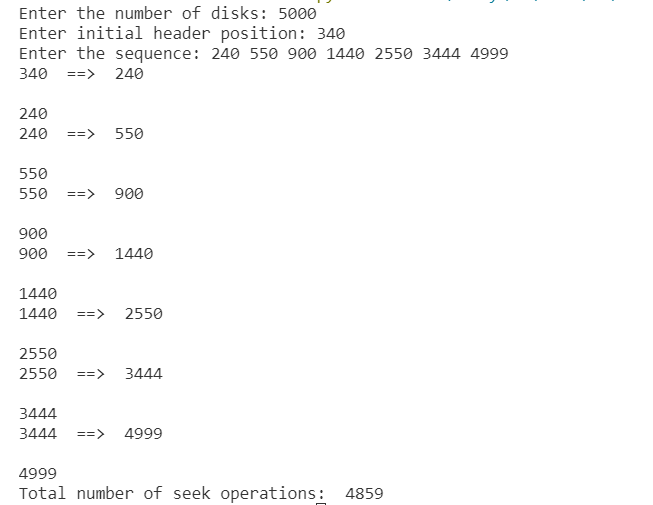
                print("Sequence out of range")

                exit(0)

        seek\_operations = FCFS(head, sequence)

        print("Total number of seek operations: ", seek\_operations)

Output:



SSTF:

def FCFS(head, sequence):

    if not sequence:

        return 0

    seek\_operations = 0

    for i in sequence:

        if i!= head:

            print(head, " ==> ",i, end=" ")

            print("\n")

            difference = abs(head - i)

            seek\_operations += difference

            head = i

    return seek\_operations

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

    Number\_disk = int(input("Enter the number of disks: "))

    if Number\_disk > 0:

        head = int(input("Enter initial header position: "))

        while not head in range(Number\_disk + 1):

            head = int(input("Please enter valid initial head position: "))

        sequence = []

        sequence = list(map(int, input("Enter the sequence: ").split()))

        for i in sequence:

            if i < 0 or i >= Number\_disk:

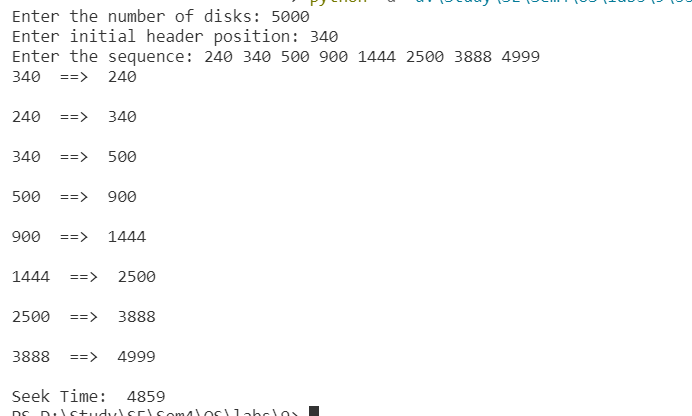
                print("Sequence out of range")

                exit(0)

        seek\_operations = FCFS(head, sequence)

        print("Seek Time: ", seek\_operations)

Output:



SCAN:

def SCAN(arr, head, direction):

    total=0

    seek\_count = 0

    distance, cur\_track = 0, 0

    left = []

    right = []

    seek\_sequence = []

    if (direction == "left"):

        left.append(0)

    elif (direction == "right"):

        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()

    # Run the while loop two times.

    # one by one scanning right

    # and left of the head

    run = 2

    while (run != 0):

        if (direction == "left"):

            for i in range(len(left) - 1, -1, -1):

                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

                total=total+head-min(arr)

                # Accessed track is now the new head

                head = cur\_track

            direction = "right"

        elif (direction == "right"):

            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

                total=total+max(arr)-head

                # Accessed track is now new head

                head = cur\_track

            direction = "left"

        run -= 1

    print("Total number of seek operations =",

        seek\_count)

    print("Seek Sequence is")

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

        print(seek\_sequence[i])

    print(f"Seek time: {total}")

# request array

disk\_size =int(input("Enter the number of disks:"))

head = int(input("Enter initial header position :   "))

direction = input("Enter the direction: ")

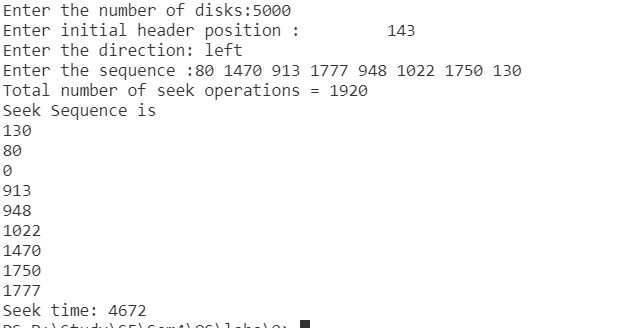
arr=[]

arr = list(map(int, input("Enter the sequence :").split()))

size = len(arr)

SCAN(arr, head, direction)

Output:



Text, letter

Description automatically generatedx