**PROGRAM CODE:**

import java.util.Scanner;

import java.util.ArrayList;

public class FCFSdisk {

public static void main(String[] args) {

int i, n, m, x, h;

float sum = 0;

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of disk:");

m = sc.nextInt();

System.out.println("Enter number of requests:");

n = sc.nextInt();

// Creating arrays to store request and arrival time

ArrayList<Integer> requests = new ArrayList<>();

ArrayList<Integer> arrivalTimes = new ArrayList<>();

System.out.println("Enter the requests and their arrival times (separated by a space):");

for (i = 0; i < n; i++) {

x = sc.nextInt();

int arrivalTime = sc.nextInt();

requests.add(x);

arrivalTimes.add(arrivalTime);

if (x > m) {

System.out.println("Error, Unknown position " + x);

return;

}

}

System.out.println("Enter the head position:");

h = sc.nextInt();

// Head will be at h at the starting

int temp = h;

System.out.print(temp);

for (i = 0; i < n; i++) {

System.out.print(" -> " + requests.get(i) + " ");

// Calculating the difference for the head movement

sum += Math.abs(requests.get(i) - temp);

// Head movement due to waiting time (arrival time difference)

if (i > 0)

sum += (arrivalTimes.get(i) - arrivalTimes.get(i - 1));

// Head is now at the next I/O request

temp = requests.get(i);

}

System.out.println();

System.out.println("Total head movements = " + sum);

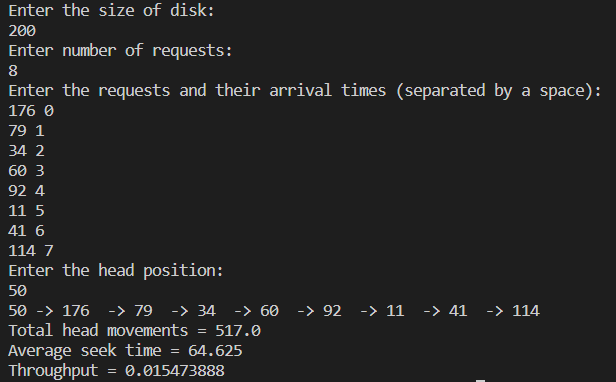
System.out.println("Average seek time = " + sum / n);

System.out.println("Throughput = " + n / sum);

}

}

**OUTPUT:**



**RESULT:**

The program has been executed successfully and the output is noted.

**PROGRAM CODE:**

import java.util.Scanner;

public class SSTFdisk {

// Calculates difference of each

// track number with the head position

static void calculatedifference(int request[], int head,

int diff[][], int n) {

for (int i = 0; i < n; i++) {

diff[i][0] = Math.abs(head - request[i]);

}

}

// Find unaccessed track which is

// at minimum distance from head

static int findMIN(int diff[][], int n) {

int index = -1;

int minimum = Integer.MAX\_VALUE;

for (int i = 0; i < n; i++) {

if (diff[i][1] == 0 && minimum > diff[i][0]) {

minimum = diff[i][0];

index = i;

}

}

return index;

}

static void shortestSeekTimeFirst(int request[], int head, int n) {

if (n == 0) {

return;

}

// Create array of objects of class node

int[][] diff = new int[n][2];

// Count total number of seek operation

float seekcount = 0;

// Stores sequence in which disk access is done

int[] seeksequence = new int[n + 1];

for (int i = 0; i < n; i++) {

seeksequence[i] = head;

calculatedifference(request, head, diff, n);

int index = findMIN(diff, n);

diff[index][1] = 1;

// Increase the total count

seekcount += diff[index][0];

// Accessed track is now new head

head = request[index];

}

seeksequence[n] = head;

System.out.println("Total number of seek operations = " + seekcount);

System.out.println("Average seek time = " + seekcount / n);

System.out.println("Throughput = " + n / seekcount);

System.out.print("Seek sequence is: ");

// Print the sequence

for (int i = 0; i <= n; i++) {

if (i < n) {

System.out.print(seeksequence[i] + "->");

} else {

System.out.print(seeksequence[i]);

}

}

}

// Driver code

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n, h;

System.out.print("Enter number of requests: ");

n = sc.nextInt();

int[] proc = new int[n];

System.out.print("Enter request sequence: \n");

for (int i = 0; i < n; i++) {

proc[i] = sc.nextInt();

}

System.out.print("Enter initial head position: ");

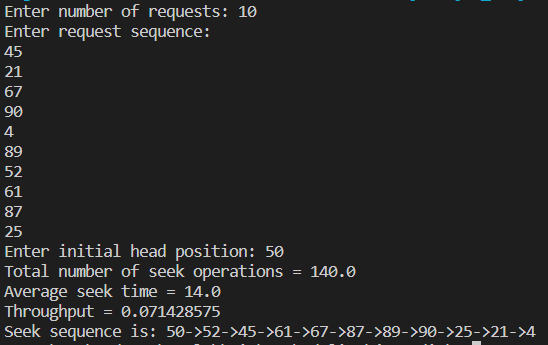
h = sc.nextInt();

shortestSeekTimeFirst(proc, h, n);

}

}

**OUTPUT:**



**RESULT:**

The program has been executed successfully and the output is noted.

**PROGRAM CODE:**

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class SCANdisk {

static int diskSize = 100;

static int n;

static void SCAN(int[] arr, int head, String direction) {

float seekCount = 0;

int distance, curTrack;

ArrayList<Integer> left = new ArrayList<>();

ArrayList<Integer> right = new ArrayList<>();

ArrayList<Integer> seekSequence = new ArrayList<>();

// appending end values

// which has to be visited

// before reversing the direction

if (direction.equals("left")) {

left.add(0);

} else if (direction.equals("right")) {

right.add(diskSize - 1);

}

for (int i = 0; i < n; i++) {

if (arr[i] < head) {

left.add(arr[i]);

}

if (arr[i] > head) {

right.add(arr[i]);

}

}

// sorting left and right ArrayLists

Collections.sort(left);

Collections.sort(right);

// run the while loop two times.

// one by one scanning right

// and left of the head

int run = 2;

while (run-- > 0) {

if (direction.equals("left")) {

for (int i = left.size() - 1; i >= 0; i--) {

curTrack = left.get(i);

// appending current track to seek sequence

seekSequence.add(curTrack);

// calculate absolute distance

distance = Math.abs(curTrack - head);

// increase the total count

seekCount += distance;

// accessed track is now the new head

head = curTrack;

}

direction = "right";

} else if (direction.equals("right")) {

for (int i = 0; i < right.size(); i++) {

curTrack = right.get(i);

// appending current track to seek sequence

seekSequence.add(curTrack);

// calculate absolute distance

distance = Math.abs(curTrack - head);

// increase the total count

seekCount += distance;

// accessed track is now new head

head = curTrack;

}

direction = "left";

}

}

System.out.println("Total number of seek operations = " + seekCount);

System.out.println("Average seek time = " + seekCount / n);

System.out.println("Throughput = " + n / seekCount);

System.out.print("Seek Sequence is: ");

// Print the sequence

for (int i = 0; i < seekSequence.size(); i++) {

if (i < n) {

System.out.print(seekSequence.get(i) + "->");

} else {

System.out.print(seekSequence.get(i));

}

}

}

// Driver code

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int h;

System.out.print("Enter number of requests: ");

n = sc.nextInt();

int[] proc = new int[n];

System.out.print("Enter request sequence: \n");

for (int i = 0; i < n; i++) {

proc[i] = sc.nextInt();

}

System.out.print("Enter initial head position: ");

h = sc.nextInt();

System.out.print("Enter direction: (left or right)\n");

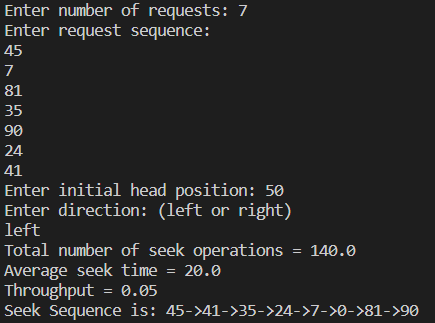
String direction = sc.next();

SCAN(proc, h, direction);

}

}

**OUTPUT:**



**RESULT:**

The program has been executed successfully and the output is noted.

**PROGRAM CODE:**

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class CSCANdisk {

static int n;

static int diskSize = 100;

static void CSCAN(int[] arr, int head) {

float seekCount = 0;

int distance, curTrack;

ArrayList<Integer> left = new ArrayList<>();

ArrayList<Integer> right = new ArrayList<>();

ArrayList<Integer> seekSequence = new ArrayList<>();

// appending end values

// which has to be visited

// before reversing the direction

left.add(0);

right.add(diskSize - 1);

// tracks on the left of the

// head will be serviced when

// once the head comes back

// to the beginning (left end).

for (int i = 0; i < n; i++) {

if (arr[i] < head) {

left.add(arr[i]);

}

if (arr[i] > head) {

right.add(arr[i]);

}

}

// sorting left and right ArrayLists

Collections.sort(left);

Collections.sort(right);

// first service the requests

// on the right side of the

// head.

for (int i = 0; i < right.size(); i++) {

curTrack = right.get(i);

// appending current track to seek sequence

seekSequence.add(curTrack);

// calculate absolute distance

distance = Math.abs(curTrack - head);

// increase the total count

seekCount += distance;

// accessed track is now new head

head = curTrack;

}

// once reached the right end

// jump to the beginning.

head = 0;

// adding seek count for head returning from 99 to 0

seekCount += (diskSize - 1);

// Now service the requests again

// which are left.

for (int i = 0; i < left.size(); i++) {

curTrack = left.get(i);

// appending current track to seek sequence

seekSequence.add(curTrack);

// calculate absolute distance

distance = Math.abs(curTrack - head);

// increase the total count

seekCount += distance;

// accessed track is now the new head

head = curTrack;

}

System.out.println("Total number of seek operations = " + seekCount);

System.out.println("Average seek time = " + seekCount / n);

System.out.println("Throughput = " + n / seekCount);

System.out.print("Seek Sequence is: ");

// Print the sequence

for (int i = 0; i < seekSequence.size(); i++) {

if (i < seekSequence.size() - 1) { // Check if it's not the last element

System.out.print(seekSequence.get(i) + "->");

} else {

System.out.print(seekSequence.get(i));

}

}

}

// Driver code

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int h;

System.out.print("Enter number of requests: ");

n = sc.nextInt();

int[] proc = new int[n];

System.out.print("Enter request sequence: ");

for (int i = 0; i < n; i++) {

proc[i] = sc.nextInt();

}

System.out.print("Enter initial head position: ");

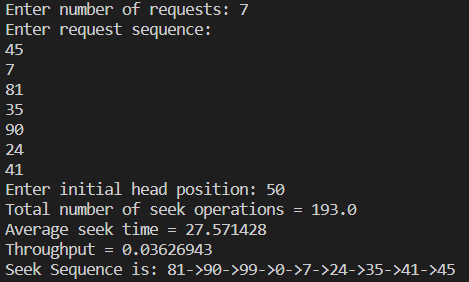
h = sc.nextInt();

CSCAN(proc, h);

}

}

**OUTPUT:**



**RESULT:**

The program has been executed successfully and the output is noted.