## AIM: Implementation of various Disk Scheduling Algorithms (FCFS, SCAN).

**Objective:** To study and implement various disk scheduling algorithms.

#### Theory:

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.

Algorithm:

1. 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.
2. Let us one by one take the tracks in default order and calculate the absolute distance of the track from the head.
3. Increment the total seek count with this distance.
4. Currently serviced track position now becomes the new head position.
5. Go to step 2 until all tracks in request array have not been serviced.

**Source code:**

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

int main() {

int i, n, req[50], mov = 0, cp;

printf("Enter the current position of the disk head: ");

scanf("%d", &cp);

printf("Enter the number of requests: ");

scanf("%d", &n);

printf("Enter the request order: ");

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

scanf("%d", &req[i]);

}

mov = abs(cp - req[0]);

printf("%d -> %d", cp, req[0]);

for(i = 1; i < n; i++) {

mov += abs(req[i] - req[i - 1]);

printf(" -> %d", req[i]);

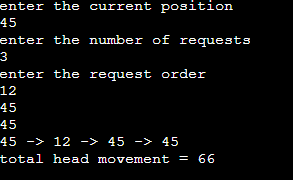
}

printf("\nTotal head movement = %d\n", mov);

return 0;

}

**Output:**



#### SCAN:

It is also called as Elevator Algorithm. In this algorithm, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path, and then it turns backend moves in the reverse direction satisfying requests coming in its path. It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.

Algorithm:

1. 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.
2. Let direction represents whether the head is moving towards left or right.
3. In the direction in which head is moving service all tracks one by one.
4. Calculate the absolute distance of the track from the head.
5. Increment the total seek count with this distance.
6. Currently serviced track position now becomes the new head position.
7. Go to step 3 until we reach at one of the ends of the disk.
8. If we reach at the end of the disk reverse the direction and go to step 2 until all tracks in request array have not been serviced.

**Source code:**

#include <stdio.h>

#include <math.h>

int main()

{

int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],

queue2[20], temp1 = 0, temp2 = 0;

float avg;

printf("Enter the max range of disk\n");

scanf("%d", &max);

printf("Enter the initial head position\n");

scanf("%d", &head);

printf("Enter the size of queue request\n");

scanf("%d", &n);

printf("Enter the queue of disk positions to be read\n");

for (i = 1; i <= n; i++)

{

scanf("%d", &temp);

if (temp >= head)

{

queue1[temp1] = temp;

temp1++;

}

else

{

queue2[temp2] = temp;

temp2++;

}

}

for (i = 0; i < temp1 - 1; i++)

{

for (j = i + 1; j < temp1; j++)

{

if (queue1[i] > queue1[j])

{

temp = queue1[i];

queue1[i] = queue1[j];

queue1[j] = temp;

}

}

}

for (i = 0; i < temp2 - 1; i++)

{

for (j = i + 1; j < temp2; j++)

{

if (queue2[i] < queue2[j])

{

temp = queue2[i];

queue2[i] = queue2[j];

queue2[j] = temp;

}

}

}

for (i = 1, j = 0; j < temp1; i++, j++)

queue[i] = queue1[j];

queue[i] = max;

for (i = temp1 + 2, j = 0; j < temp2; i++, j++)

queue[i] = queue2[j];

queue[i] = 0;

queue[0] = head;

for (j = 0; j <= n + 1; j++)

{

diff = abs(queue[j + 1] - queue[j]);

seek += diff;

printf("Disk head moves from %d to %d with seek %d\n", queue[j],

queue[j + 1], diff);

}

printf("Total seek time is %d\n", seek);

avg = seek / (float)n;

printf("Average seek time is %f\n", avg);

return 0;

}

**Output:**

