**Question:**

Write a C program to solve the following problem: Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order is:

86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance (in cylinders)that the disk arm moves to satisfy all the pending requests for each of the FCFS disk-scheduling algorithms?

**Description:**

FCFS (First Come First Serve) Disk-Scheduling algorithm is an algorithm used for scheduling. Out of all the scheduling algorithms, it is the simplest and easiest scheduling algorithm.

As the names suggests, in places the processes in the queue in the same order as they arrive for execution. Or we can simply say that it places the processes in the queue based on the arrival time of the process.

The process that enters in the CPU first will be executed first and after its execution next process will enter the CPU, i.e., next process will get the CPU only after the process which is present in the CPU completes its execution.

Distance moved by the disk from a process ‘A’ to process ‘B’ can be calculated by subtracting the value of process ‘A’ from the value of process ‘B’.

**Algorithm:**

In this problem, I have used a simple algorithm for the calculation of the total distance moved by the disk arm.

total = array[1] - array[0];

if(total<0)

{

total = total\*-1;

}

for(i=2;i<n;i++)

{

if((array[i]-array[i-1])>0)

total=total+(array[i]-array[i-1]);

else

total=total+(array[i-1]-array[i]);

}

**Complexity:**

Here in the algorithm we see that only one loop, i.e., for loop is contributing to the complexity.

So, we can say that the complexity of the mentioned code is **‘n’** which is the number of processes.

Or we can say that the complexity of the given problem depends on the number of processes that are present for execution.

**Code:**

The code is implemented using the concepts of system call.

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

#include<string.h>

#include<fcntl.h>

#include<sys/stat.h>

#include<sys/types.h>

int main()

{

int array[] = {143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130};

int size = sizeof(array)/sizeof(array[0]);

int total = 0;

int i;

total = array[1] - array[0];

if(total<0)

{

total = total\*-1;

}

for(i=2;i<10;i++)

{

total=total+abs(array[i]-array[i-1]);

}

char str[50];

snprintf(str, 50, "%d\n", total);

int fd;

fd = open("File.txt",O\_CREAT|O\_RDWR,0777);

write(fd, str, strlen(str));

lseek(fd,0,SEEK\_SET);

char buffer[50];

int a;

a = read(fd,buffer,50);

write(1,"Total Distance moved by Disk Arm: ",40);

write(1,buffer,a);

}

**Constraints:**

In this problem, the constraint is that the distance moved by the disk arm can be positive as well as negative. But the distance moved by the disk arm is always positive. So, while we are calculating the distance moved by disk arm we have used **abs()** function which returns the absolute value of the integer, i.e., a positive integer.

**Test Cases:**

**Input:**

Head is at 143.

Queue is: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

**Processing:**

Distance = abs(86 ­­­­- 143) = 57

Distance = 57 + abs(1470 - 86) = 1441

Distance = 1441 + abs(913 - 1470) = 1998

Distance = 1998 + abs(1774 - 913) = 2859

Distance = 2859 + abs(948 - 1774) = 3685

Distance = 3685 + abs(1509 - 948) = 4246

Distance = 4246 + abs(1022 - 1509) = 4733

Distance = 4733 + abs(1750 - 1022) = 5461

Distance = 5461 + abs(130 - 1750) = 7081

**Output:**

A screenshot of a cell phone

Description automatically generated