#include<math.h>

#include<stdio.h>

#include<stdlib.h>

int main()

{

int i,n,k,req[50],mov=0,cp,index[50],min,a[50],j=0,mini,cp1; printf("enter the current position\n");

scanf("%d",&cp);

printf("enter the number of requests\n");

scanf("%d",&n);

cp1=cp;

printf("enter the request order\n");

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

{

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

}

for(k=0;k<n;k++)

{

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

{

index[i]=abs(cp-req[i]); // calculate distance of each request from current position

} // to find the nearest request

min=index[0];

mini=0;

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

{

if(min>index[i])

{

min=index[i];

mini=i;

}

}

a[j]=req[mini];

j++;

cp=req[mini]; // change the current position value to next request req[mini]=999;

} // the request that is processed its value is changed so that it is not processed again

printf("Sequence is : ");

printf("%d",cp1);

mov=mov+abs(cp1-a[0]); // head movement

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

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

{

mov=mov+abs(a[i]-a[i-1]); ///head movement

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

} printf("\n");

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

}

## Algorithm ****Shortest Seek Time First (SSTF)****

**Step 1:**Let the Request array represents an array storing indexes of tracks that have been requested. ‘head’ is the position of the disk head.

**Step 2:**Find the positive distance of all tracks in the request array from the head.

**Step 3:**Find a track from the requested array which has not been accessed/serviced yet and has a minimum distance from the head.

**Step 4:**Increment the total seek count with this distance.

**Step 5:**Currently serviced track position now becomes the new head position.

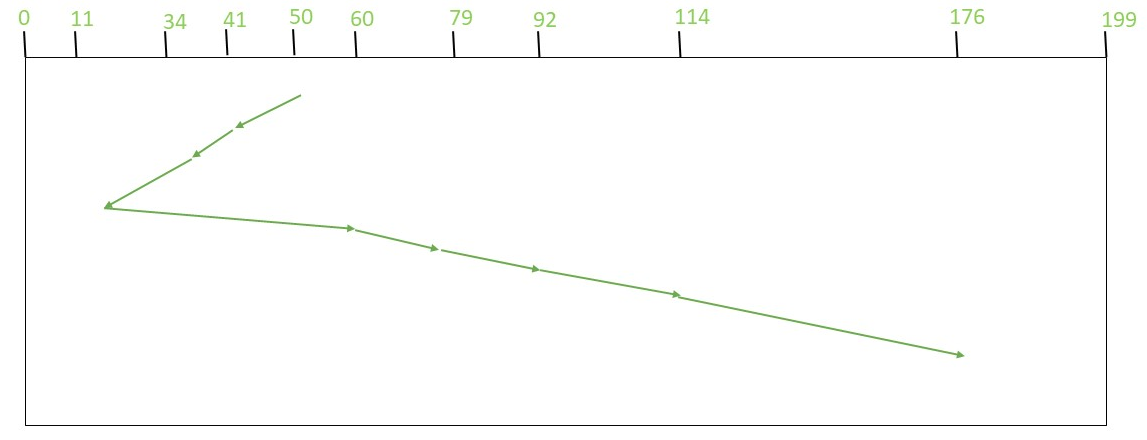
**Step 6:**Go to step 2 until all tracks in the request array have not been serviced.

**Example:**

Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}   
Initial head position = 50

The following chart shows the sequence in which requested tracks are serviced using SSTF.

Therefore, the total seek count is calculated as:



*SSTF (Shortest Seek Time First)*

= (50-41)+(41-34)+(34-11)+(60-11)+(79-60)+(92-79)+(114-92)+(176-114)  
= **204**  
which can also be directly calculated as: (50-11) + (176-11)