**EX.NO: 9 IMPLEMENTATION OF DISK SCHEDULING ALGORITHMS**

## Objectives:

To write a C program to implement Disk Scheduling Algorithm.

## FCFS SCHEDULING ALGORITHM 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.

**Sample Coding:** #include<stdio.h> #include<stdlib.h> int main()

{

int RQ[100],i,n,TotalHeadMoment=0,initial; printf("Enter the number of Requests\n"); scanf("%d",&n);

printf("Enter the Requests sequence\n"); for(i=0;i<n;i++)

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

printf("Enter initial head position\n"); scanf("%d",&initial);

// logic for FCFS disk scheduling for(i=0;i<n;i++)

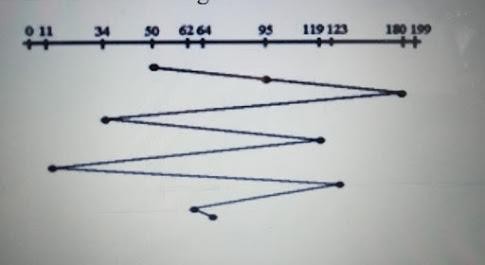
{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

printf("Total head moment is %d",TotalHeadMoment); return 0;

}



## Sample Output:

Enter the number of Request 8

Enter the Requests Sequence 95 180 34 119 11 123 62 64

Enter initial head position 50

Total head movement is 644

## SSTF (SHORTEST SEEK TIME FIRST) ALGORITHM Algorithm :

* + 1. Let Request array represents an array storing indexes of tracks that have been requested. ‘head’ is the position of disk head.
    2. Find the positive distance of all tracks in the request array from head.
    3. Find a track from requested array which has not been accessed/serviced yet and has minimum distance from head.
    4. Increment the total seek count with this distance.
    5. Currently serviced track position now becomes the new head position.
    6. Go to step 2 until all tracks in request array have not been serviced.

**Sample Coding:** #include<stdio.h> #include<stdlib.h> int main()

{

int RQ[100],i,n,TotalHeadMoment=0,initial,count=0; printf("Enter the number of Requests\n"); scanf("%d",&n);

printf("Enter the Requests sequence\n"); for(i=0;i<n;i++)

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

printf("Enter initial head position\n"); scanf("%d",&initial);

// logic for sstf disk scheduling

/\* loop will execute until all process is completed\*/ while(count!=n)

{

int min=1000,d,index; for(i=0;i<n;i++)

{

d=abs(RQ[i]-initial); if(min>d)

{

min=d; index=i;

}

}

TotalHeadMoment=TotalHeadMoment+min; initial=RQ[index];

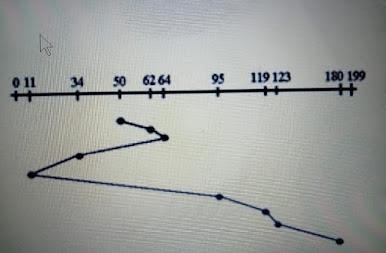
// 1000 is for max

// you can use any number RQ[index]=1000; count++;

}

printf("Total head movement is %d",TotalHeadMoment); return 0;

}



## Sample Output:

Enter the number of Request 8

Enter Request Sequence

95 180 34 119 11 123 62 64

Enter initial head Position 50

Total head movement is 236

## SCAN SCHEDULING ALGORITHM 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.

**Sample Coding:**

# include<stdio.h> #include<stdlib.h> int main()

{

int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move; printf("Enter the number of Requests\n"); scanf("%d",&n);

printf("Enter the Requests sequence\n"); for(i=0;i<n;i++)

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

printf("Enter initial head position\n"); scanf("%d",&initial);

printf("Enter total disk size\n"); scanf("%d",&size);

printf("Enter the head movement direction for high 1 and for low 0\n"); scanf("%d",&move);

// logic for Scan disk scheduling

/\*logic for sort the request array \*/ for(i=0;i<n;i++)

{

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

{ if(RQ[j]>RQ[j+1])

{

int temp; temp=RQ[j]; RQ[j]=RQ[j+1];

RQ[j+1]=temp;

}

}

}

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

{

if(initial<RQ[i])

{

index=i; break;

}

}

// if movement is towards high value if(move==1)

{

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

// last movement for max size TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1); initial = size-1;

for(i=index-1;i>=0;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

}

// if movement is towards low value else

{

for(i=index-1;i>=0;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

// last movement for min size TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0); initial =0;

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

{

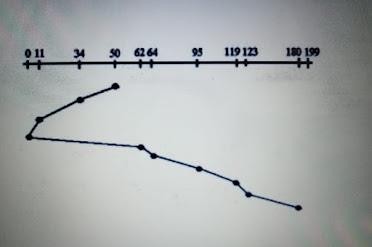
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

}

printf("Total head movement is %d",TotalHeadMoment); return 0;

}



## Sample Output:

Enter the number of Request 8

Enter the Requests Sequence 95 180 34 119 11 123 62 64

Enter initial head position 50

Enter total disk size 200

Enter the head movement direction for high 1 and for low 0 1

Total head movement is 337

## C-SCAN SCHEDULING ALGORITHM 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. The head services only in the right direction from 0 to the size of the disk.
    3. While moving in the left direction do not service any of the tracks.
    4. When we reach the beginning(left end) reverse the direction.
    5. While moving in the right direction it services all tracks one by one.
    6. While moving in the right direction calculate the absolute distance of the track from the head.
    7. Increment the total seek count with this distance.
    8. Currently serviced track position now becomes the new head position.
    9. Go to step 6 until we reach the right end of the disk.
    10. If we reach the right end of the disk reverse the direction and go to step 3 until all tracks in the request array have not been serviced.

**Sample Coding:**

# include<stdio.h> #include<stdlib.h> int main()

{

int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move; printf("Enter the number of Requests\n"); scanf("%d",&n);

printf("Enter the Requests sequence\n"); for(i=0;i<n;i++)

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

printf("Enter initial head position\n"); scanf("%d",&initial);

printf("Enter total disk size\n"); scanf("%d",&size);

printf("Enter the head movement direction for high 1 and for low 0\n"); scanf("%d",&move);

// logic for C-Scan disk scheduling

/\*logic for sort the request array \*/ for(i=0;i<n;i++)

{

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

{ if(RQ[j]>RQ[j+1])

{

int temp; temp=RQ[j]; RQ[j]=RQ[j+1];

RQ[j+1]=temp;

}

}

}

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

{

if(initial<RQ[i])

{

index=i; break;

}

}

// if movement is towards high value if(move==1)

{

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

// last movement for max size TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);

/\*movement max to min disk \*/ TotalHeadMoment=TotalHeadMoment+abs(size-1-0); initial=0;

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

}

// if movement is towards low value else

{

for(i=index-1;i>=0;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

// last movement for min size TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);

/\*movement min to max disk \*/ TotalHeadMoment=TotalHeadMoment+abs(size-1-0);

initial =size-1;

for(i=n-1;i>=index;i--)

{

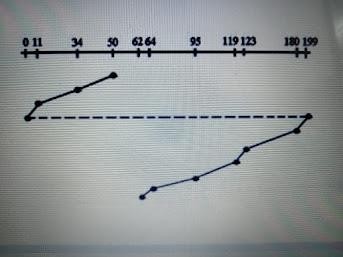
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

}

printf("Total head movement is %d",TotalHeadMoment); return 0;

}



## Sample Output:

Enter the number of Request 8

Enter the Requests Sequence 95 180 34 119 11 123 62 64

Enter initial head position 50

Enter total disk size 200

Enter the head movement direction for high 1 and for low 0 1

Total head movement is 382

## LOOK DISK SCHEDULING ALGORITHM 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. The initial direction in which head is moving is given and it services in the same direction.
    3. The head services all the requests one by one in the direction head is moving.
    4. The head continues to move in the same direction until all the request in this direction are finished.
    5. While moving in this direction calculate the absolute distance of the track from the head.
    6. Increment the total seek count with this distance.
    7. Currently serviced track position now becomes the new head position.
    8. Go to step 5 until we reach at last request in this direction.
    9. If we reach where no requests are needed to be serviced in this direction reverse the direction and go to step 3 until all tracks in request array have not been serviced.

**Sample Coding:**

# include<stdio.h> #include<stdlib.h> int main()

{

int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move; printf("Enter the number of Requests\n"); scanf("%d",&n);

printf("Enter the Requests sequence\n"); for(i=0;i<n;i++)

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

printf("Enter initial head position\n"); scanf("%d",&initial);

printf("Enter total disk size\n"); scanf("%d",&size);

printf("Enter the head movement direction for high 1 and for low 0\n"); scanf("%d",&move);

// logic for look disk scheduling

/\*logic for sort the request array \*/ for(i=0;i<n;i++)

{

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

{ if(RQ[j]>RQ[j+1])

{

int temp; temp=RQ[j]; RQ[j]=RQ[j+1];

RQ[j+1]=temp;

}

}}

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

{

if(initial<RQ[i])

{

index=i; break;

}

}

// if movement is towards high value if(move==1)

{

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

for(i=index-1;i>=0;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

}

// if movement is towards low value else

{

for(i=index-1;i>=0;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

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

{

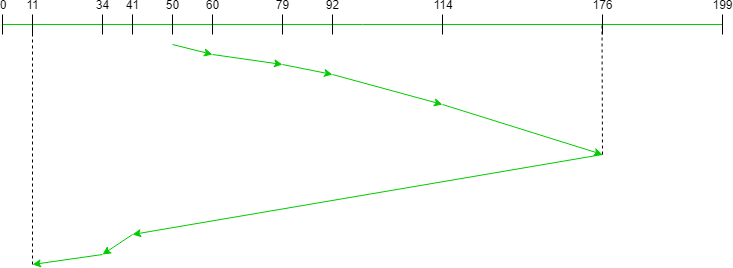
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

}

printf("Total head movement is %d",TotalHeadMoment); return 0;

}



## Sample Output:

Enter the number of Request 8

Enter the Requests Sequence 95 180 34 119 11 123 62 64

Enter initial head position 50

Enter the head movement direction for high 1 and for low 0 1

Total head movement is 299

## C-LOOK DISK SCHEDULING ALGORITHM Algorithm :

* + 1. Let Request array represents an array storing indexes of the tracks that have been requested in ascending order of their time of arrival and **head** is the position of the disk head.
    2. The initial direction in which the head is moving is given and it services in the same direction.
    3. The head services all the requests one by one in the direction it is moving.
    4. The head continues to move in the same direction until all the requests in this direction have been serviced.
    5. While moving in this direction, calculate the absolute distance of the tracks from the head.
    6. Increment the total seek count with this distance.
    7. Currently serviced track position now becomes the new head position.
    8. Go to step 5 until we reach the last request in this direction.
    9. If we reach the last request in the current direction then reverse the direction and move the head in this direction until we reach the last request that is needed to be serviced in this direction without servicing the intermediate requests.
    10. Reverse the direction and go to step 3 until all the requests have not been serviced.

**Sample Coding:**

# include<stdio.h> #include<stdlib.h> int main()

{

int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move; printf("Enter the number of Requests\n"); scanf("%d",&n);

printf("Enter the Requests sequence\n"); for(i=0;i<n;i++)

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

printf("Enter initial head position\n"); scanf("%d",&initial);

printf("Enter total disk size\n"); scanf("%d",&size);

printf("Enter the head movement direction for high 1 and for low 0\n"); scanf("%d",&move);

// logic for C-look disk scheduling

/\*logic for sort the request array \*/ for(i=0;i<n;i++)

{

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

{ if(RQ[j]>RQ[j+1])

{

int temp; temp=RQ[j]; RQ[j]=RQ[j+1];

RQ[j+1]=temp;

}}

}

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

{

if(initial<RQ[i])

{

index=i; break;

}}

// if movement is towards high value if(move==1)

{

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}}

// if movement is towards low value else

{

for(i=index-1;i>=0;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}

for(i=n-1;i>=index;i--)

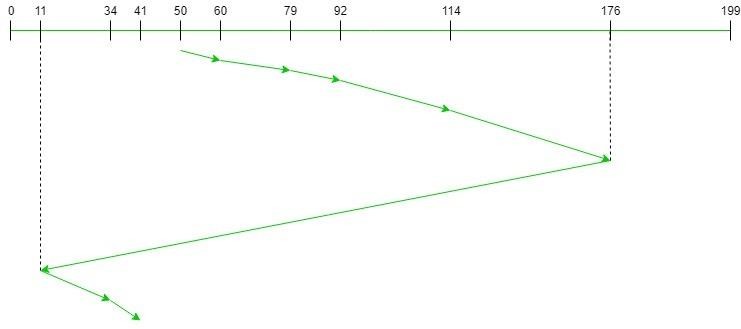
{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial); initial=RQ[i];

}}

printf("Total head movement is %d",TotalHeadMoment); return 0;

}



## Sample Output:

Enter the number of Request 8

Enter the Requests Sequence 95 180 34 119 11 123 62 64

Enter initial head position 50

Enter the head movement direction for high 1 and for low 0 1

Total head movement is 322

## Conclusion:

Thus the C program to implement the different disk scheduling algorithms are executed and outputs are verified.