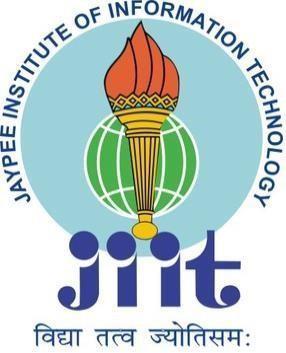
**Project**



Operating System and System Programming

**15B11CI412**

Submitted by:

**Jayant Goel (18103255) B8**

Under the supervision of:

**Dr. Alka Singhal & Dr. Taj Alam**

**Implementation and Performance comparison of four disk scheduling algorithms**

Hard disks are being employed to store vast info information in all modern computers. Disk drives should give quicker access time so as to optimize the speed of I/O operations. In a multitasking system with several processes, disk performance can be improved by incorporating a scheduling algorithm for maintaining several unfinished requests within the disk queue.

The Four disk scheduling algorithms (FCFS, SSTF, LOOK for each upward and downward direction, and C-LOOK) to measure their performance in terms of total head movement. The developed machine runs with success in a concurrent execution surroundings and therefore the results demonstrate that LOOK (downward direction) the algorithm provides the most effective results for given take a look at samples due to the reduction of an oversized range of unneeded head movements. As several wild swings are experienced by FCFS scheme so it gives the worst scheduling performance. SSTF is far higher compared to seem (upward direction) and C-LOOK. It’s has been noticed that LOOK is a lot of economical than C-LOOK at all hundreds whereas C-LOOK is best at high hundreds solely because it reduces the starvation problem.

**The performance of every algorithm, however, heavily depends on the amount and sort of requests.**

**DISK SCHEDULING**

Since most jobs depend heavily on the disk for loading and

I/O operations, it is important that disk service be as fast as possible. Disk speed is composed of three parts: **seek time, latency time and transfer time**. Every I/O device, including each disk drive, has queue of pending requests. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. This request specifies several pieces of information:

(1) Type of I/O operations,

(2) Address of disk (drive, cylinder, surface, and block),

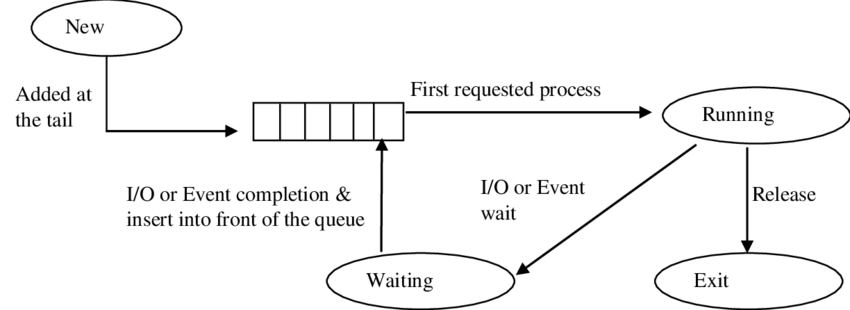
(3) Address of memory, and

(4) Amount of information is to be transferred (a byte or word count).

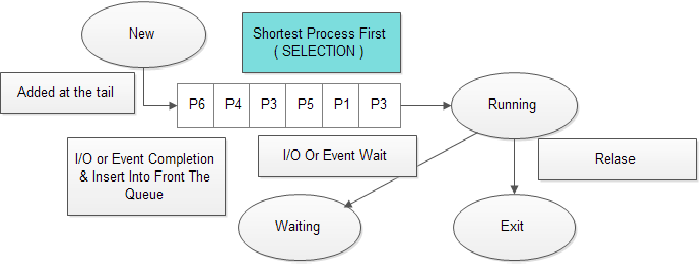
If the desired disk drive and controller are available, the request can be serviced immediately. However, while the drive or controller is serving one request, any additional requests will need to be queued. For multiprogramming system with many processes, the disk queue may often be nonempty. Thus, when a request is complete, a new request is picked from the queue and it is serviced. A disk service requires that the head be moved to the desired track; it must wait for latency and seek time, and finally transfer the data to memory.

Different algorithms such as **FCFS, SSTF, LOOK and CLOOK** are used for selecting request for servicing from the queue of requests.

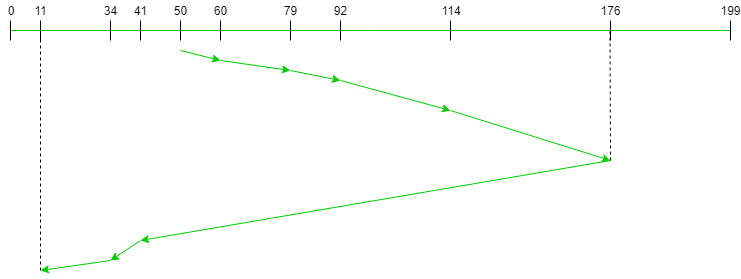
**First-Come, First-Served (FCFS)** is the simplest form of disk scheduling. This algorithm is easy to implement using FIFO queue. Wild swing of FCFS can increase total head movements.



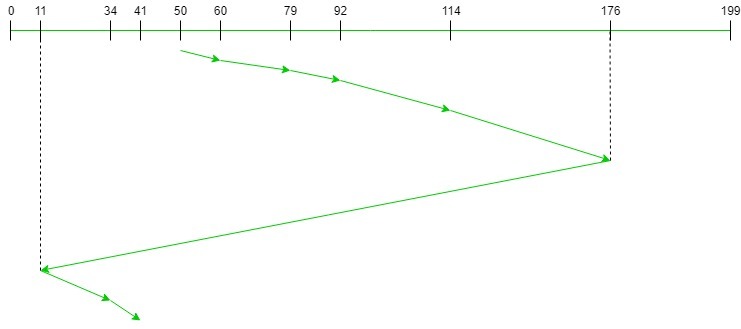
**Shortest-Seek-Time-First (SSTF)** seems reasonable to service together all requests close to the current head position, before moving the head far away to service another request. It selects the request with the minimum seek time from the current head position. Since the seek time is generally proportional to the track difference between the requests, it is implemented by moving the head to the closest track in the request queue. Some of the requests may wait indefinitely in this approach.



Recognition of the dynamic nature of the request queue leads to the **LOOK algorithm.** The read-write head starts from its present position, services requests while moving towards one end of the disk, after servicing the last request of this end, the direction of the head movement is reversed and servicing continues as far as the last request in this direction. LOOK scheme continuously scans the disk from one direction to the other keeping the swing to the last request on either side. In a real-time system, if a request arrives just in front of the head, it will be serviced almost immediately, whereas a request arriving just behind the head will have to wait until the head moves to the last request of one end of the disk, reverses direction and returns to service the leftover requests on its way to the other end. In practical systems the head is moved as far as the last request in each direction. As soon as there are no requests in the current direction, the head movement is reversed.

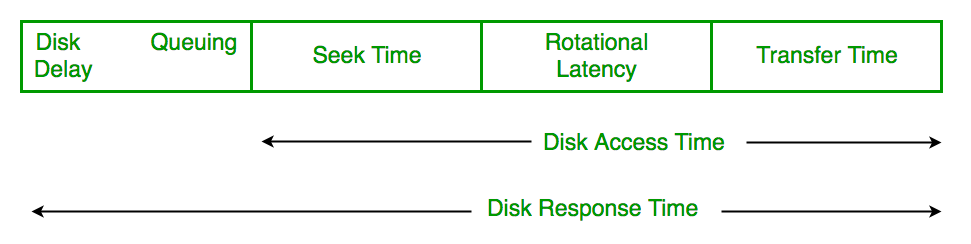


A variant of LOOK scheduling that is designed to provide a more uniform wait time is **C-LOOK (Circular LOOK)** scheduling. CLOOK moves the head towards one direction from its present position, after servicing the last request in the current direction it reverses its direction and immediately returns to the first request of the other end, without servicing any requests on the return trip. C-LOOK scheduling essentially treats the disk as though it were circular, with the last track adjacent to the first one.



**Software Design**

4 Different programs have been developed in C++ language for disk scheduling algorithms which includes four major scheduling algorithms (i.e. FCFS, SSTF, LOOK and C-LOOK). These algorithm runs and presents results based upon service requests and number of tracks involved in the given test sample. Tracks requests are read by the program from the relevant file. Algorithm automatically offers reordered list of the read requests and a queue is displayed to service the requests. When all requests are serviced, then the result in the form of total head movement is displayed. Each program can be run to get new data of requests from the keyboard. Total head movement is calculated based upon entered data. Various data structures such as linked lists, arrays, queues and records have been used for successful execution, recording of simulation results.



**Implementation**

**FCFS**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  void FCFS(int arr[], int head,int size){  int seek\_count = 0;  int distance, cur\_track;  for (int i = 0; i < size; i++) {  cur\_track = arr[i];  distance = abs(cur\_track - head);  seek\_count += distance;  head = cur\_track;  }  cout<< "Total number of seek operations = "<<seek\_count << endl;  cout << "Seek Sequence is" << endl;  for (int i = 0; i < size; i++) {  cout << arr[i] << endl;  }  }  int main(){  int n;  cin>>n;    int arr[n];  for(int i=0;i<n;i++){  cin>>arr[i];  }  int head;  cin>>head;    FCFS(arr, head, n);  return 0;  } |
|  |

**SSTF**

#include <bits/stdc++.h>

using namespace std;

void calculateDifference(int \*request, int head, int \*\*diff, int n){

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

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

}

}

int findMIN(int \*\*diff, int n){

int index = -1;

int minimum = 1e9;

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

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

minimum = diff[i][0];

index = i;

}

}

return index;

}

void shortestSeekTimeFirst(int \*request,int head, int n){

if (n == 0){

return;

}

int \*\*diff = new int\*[n];

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

diff[i] = new int [2];

}

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

for (int j = 0; j < 2; ++j) {

diff[i][j] = 0;

}

}

int seekCount = 0;

int \*seekSequence = new int[n+1];

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

seekSequence[i] = 0;

}

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

seekSequence[i] = head;

calculateDifference(request, head, diff, n);

int index = findMIN(diff, n);

diff[index][1] = 1;

seekCount += diff[index][0];

head = request[index];

}

seekSequence[n] = head;

cout << "Total number of seek operations = " << seekCount << endl;

cout << "Seek sequence is : " << "\n";

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

cout << seekSequence[i] << "\n";

}

}

int main(){

int n;

cin>>n;

int arr[n];

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

cin>>arr[i];

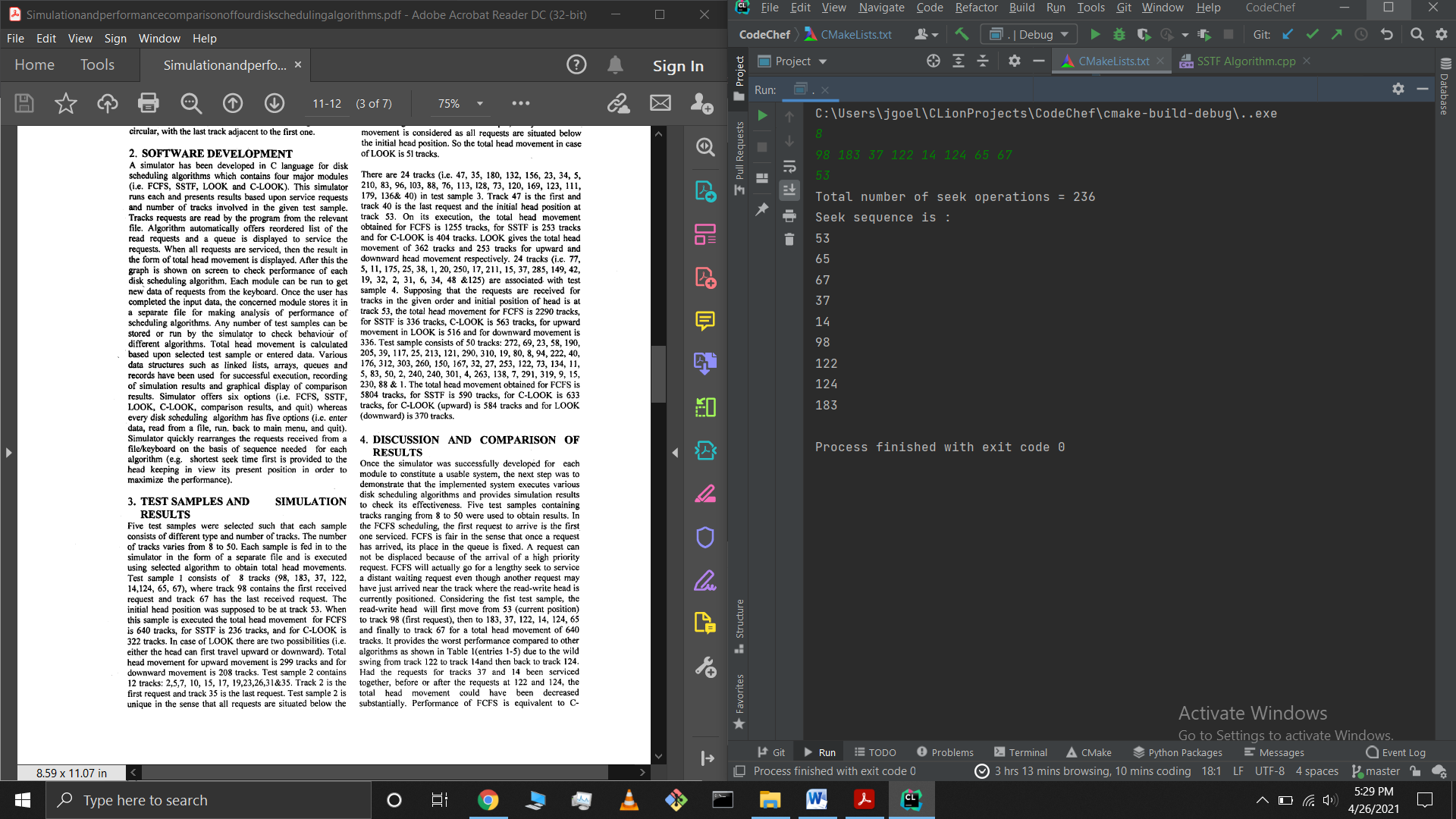
}

int head;

cin>>head;

shortestSeekTimeFirst(arr, head, n);

}



**FCFS**

#include <bits/stdc++.h>

using namespace std;

void LOOK(int arr[], int head, string direction,int size){

int seek\_count = 0;

int distance, cur\_track;

vector<int> downward, upward;

vector<int> seek\_sequence;

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

if (arr[i] < head)

downward.push\_back(arr[i]);

if (arr[i] > head)

upward.push\_back(arr[i]);

}

sort(downward.begin(), downward.end());

sort(upward.begin(), upward.end());

int run = 2;

while (run--) {

if (direction == "downward") {

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

cur\_track = downward[i];

seek\_sequence.push\_back(cur\_track);

distance = abs(cur\_track - head);

seek\_count += distance;

head = cur\_track;

}

direction = "upward";

}

else if (direction == "upward") {

for (int i : upward) {

cur\_track = i;

seek\_sequence.push\_back(cur\_track);

distance = abs(cur\_track - head);

seek\_count += distance;

head = cur\_track;

}

direction = "downward";

}

}

cout << "Total number of seek operations = "<< seek\_count << endl;

cout << "Seek Sequence is" << endl;

for (int i : seek\_sequence) {

cout << i << endl;

}

}

int main(){

int n;

cin>>n;

int arr[n];

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

cin>>arr[i];

}

int head;

cin>>head;

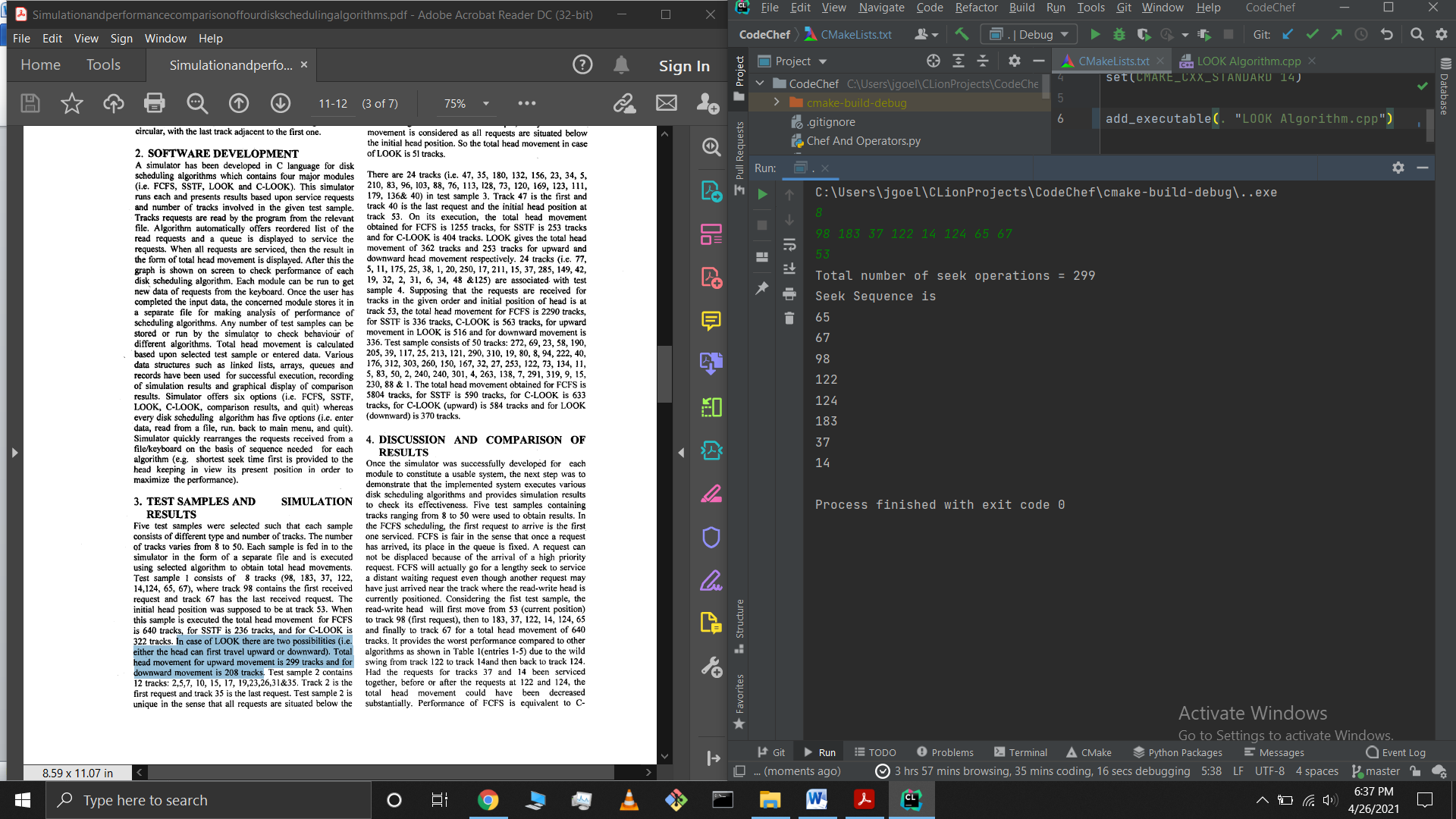
//string direction = "upward";

string direction = "downward";

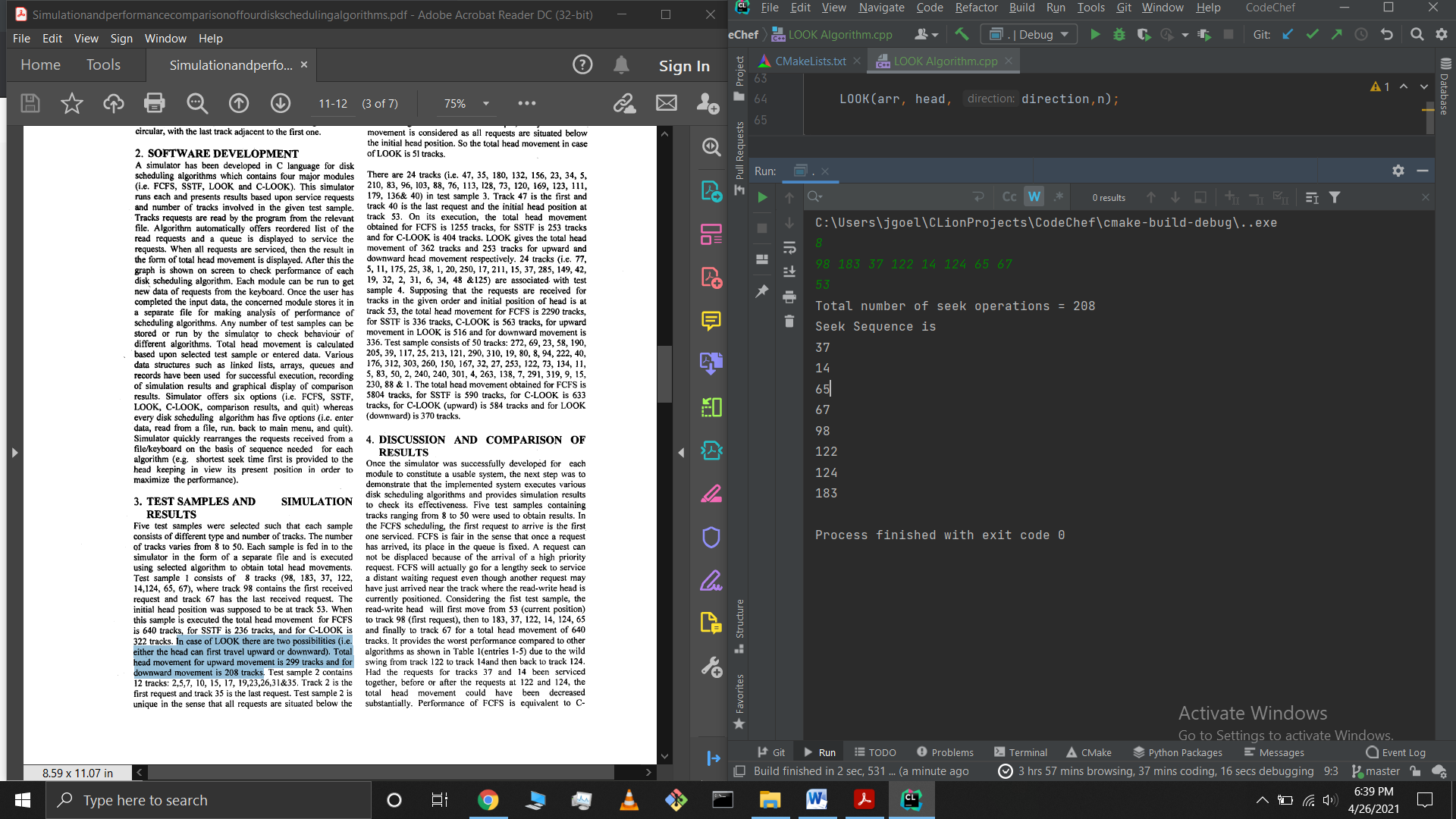
LOOK(arr, head, direction,n);

}

**Upward**



**Downward**



**CLOOK**

#include <bits/stdc++.h>

using namespace std;

void CLOOK(int arr[], int head,int size){

int seek\_count = 0;

int distance, cur\_track;

vector<int> left, right;

vector<int> seek\_sequence;

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

if (arr[i] < head)

left.push\_back(arr[i]);

if (arr[i] > head)

right.push\_back(arr[i]);

}

sort(left.begin(), left.end());

sort(right.begin(), right.end());

for (int i : right) {

cur\_track = i;

seek\_sequence.push\_back(cur\_track);

distance = abs(cur\_track - head);

seek\_count += distance;

head = cur\_track;

}

seek\_count += abs(head - left[0]);

head = left[0];

for (int i : left) {

cur\_track = i;

seek\_sequence.push\_back(cur\_track);

distance = abs(cur\_track - head);

seek\_count += distance;

head = cur\_track;

}

cout << "Total number of seek operations = "<< seek\_count << endl;

cout << "Seek Sequence is" << endl;

for (int i : seek\_sequence) {

cout << i << endl;

}

}

int main(){

int n;

cin>>n;

int arr[n];

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

cin>>arr[i];

}

int head;

cin>>head;

CLOOK(arr, head,n);

}

