A Course Based Project Report

On

Visualization of Disk Scheduling Algorithms

Submitted in partial fulfillment of requirement

for the completion of the

Operating Systems Laboratory

I B.Tech Computer Science and Engineering

of

VNR VJIET

By

A. NIRUP	**23071A05E4**
K. KOWSHIK REDDY	**23071A05G4**
G. AKSHAY KUMAR	**23071A05F9**
A. SANTHOSH	**23071A05D8**
A. NIKITHA	**23071A05E3**

2024-2025

VALLURIPALLI NAGESWARA RAO
VIGNANA JYOTHI INSTITUTE OF ENGINEERING &TECHNOLOGY
(AUTONOMOUS INSTITUTE)
NAAC ACCREDITED WITH ‘A++’ GRADE
NBA Accreditation for B. Tech Programs
Vignana Jyothi Nagar, Bachupally, Nizampet (S.O), Hyderabad 500090
Phone no: 040-23042758/59/60, Fax: 040-23042761
Email:postbox@vnrvjiet.ac.in Website: www.vnrvjiet.ac.in

A Project Report On

Visualization of Disk Scheduling Algorithms

Submitted in partial fulfillment of requirement
for the completion of the

Operating Systems Laboratory

B.Tech Computer Science and Engineering

of

VNRVJIET

2024-2025

Under the Guidance

of

DR. D N VASUNDHARA
Assistant Professor
Department of CSE

TAMASOMA JYOTHIRGAMAYA

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

(AUTONOMUS INSTITUTE)

NAAC ACCREDITED WITH 'A++' GRADE

CERTIFICATE

This is to certify that the project entitled "Visualization of disk scheduling algorithms" submitted in partial fulfillment for the course of Operating systems laboratory being offered for the award of Batch (CSE-C) by VNRVJIET is a result of the bonafide work carried out by **23071A05E4**, **23071A05G4, 23071A05F9** and **23071A05D8, 23071A05E3** during the year **2024-2025**.

This has not been submitted for any other certificate or course.

.

Signature of Faculty **Signature of Head of the Department**

ACKNOWLEDGEMENT

An endeavor over a long period can be successful only with the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We wish to express our profound gratitude to our honorable **Principal Dr. C.D.Naidu** and **HOD Dr.S.Nagini**, **CSE Department**, **VNR Vignana Jyothi Institute of Engineering and Technology** for their constant and dedicated support towards our career moulding and development.

With great pleasure we express our gratitude to the internal guide **DR. DN VASUNDHARA, Assistant Professor, CSE department** for his timely help, constant guidance, cooperation, support and encouragement throughout this project as it has urged us to explore many new things.

Finally, we wish to express my deep sense of gratitude and sincere thanks to our parents, friends and all our well-wishers who have technically and non-technically contributed to the successful completion of this course-based project.

DECLARATION

We hereby declare that this Project Report titled “**Visualization of Disk Scheduling Algorithms**” submitted by us of Computer Science & Engineering in **VNR Vignana Jyothi Institute of Engineering and Technology**, is a bonafide work undertaken by us and it is not submitted for any other certificate /Course or published any time before.

Name & Signature of the Students:

A. NIRUP	**23071A05E4**
K. KOWSHIK REDDY	**23071A05G4**
G. AKSHAY KUMAR	**23071A05F9**
A. SANTHOSH	**23071A05D8**
A. NIKITHA	**23071A05E3**

Date:

TABLE OF CONTENTS

ABSTRACT

In modern operating systems, disk scheduling is a vital component that determines the order in which I/O requests are served, aiming to reduce disk head movement and improve overall system efficiency. Understanding the logic and performance differences among various disk scheduling algorithms can be challenging through theory alone. This project addresses that gap by developing a web-based visualization tool for disk scheduling algorithms using JavaScript.

The project supports simulation of six widely used algorithms: First-Come-First-Serve (FCFS), Shortest Seek Time First (SSTF), SCAN, C-SCAN (Circular SCAN), LOOK, and C-LOOK (Circular LOOK). Users can input a custom disk request queue and initial head position, and the system visually animates the path taken by the disk head as it services requests based on the selected algorithm. Additionally, it calculates and displays the total head movement for each case, allowing users to compare the relative efficiency of different algorithms.

The tool is implemented using HTML, CSS, and JavaScript, ensuring that it runs efficiently in any modern web browser without the need for external installations or backend services. The interactive and responsive design makes it an effective educational aid, offering hands-on learning through real-time visual feedback.

This project is particularly useful for students and educators in operating systems courses, enabling a deeper understanding of algorithm behaviour through visual learning. It makes abstract scheduling strategies concrete and intuitive, while also offering a platform for experimentation. Future improvements could include graphical comparisons of performance metrics, additional algorithms, or enhancements for mobile compatibility.

INTRODUCTION

In modern computing systems, efficient management of disk input/output (I/O) operations is crucial for optimal performance. Disk scheduling algorithms play a pivotal role in determining the sequence in which disk I/O requests are serviced. By optimizing the order of these requests, these algorithms aim to reduce the total seek time—the time taken for the disk's read/write head to move between tracks—thereby enhancing overall system efficiency.

Traditional disk scheduling algorithms include:

- **First-Come-First-Serve (FCFS):** Processes requests in the order they arrive, without reordering.
- **Shortest Seek Time First (SSTF):** Selects the request closest to the current head position, minimizing immediate seek time.
- **SCAN (Elevator Algorithm):** Moves the disk head in one direction, servicing all requests until it reaches the end, then reverses direction.
- **C-SCAN (Circular SCAN):** Similar to SCAN but, upon reaching the end, the head returns to the beginning without servicing requests on the return trip.
- **LOOK:** Like SCAN but reverses direction when there are no further requests in the current direction.
- **C-LOOK:** A variant of LOOK where the head moves only as far as the last request in one direction, then jumps to the first request in the opposite direction.

Understanding the nuances of these algorithms can be challenging through theoretical study alone. Visual tools can bridge this gap by providing intuitive representations of how each algorithm operates, making it easier to grasp their behaviours and performance implications.
This project introduces an interactive, web-based visualization tool developed using JavaScript, HTML, and CSS. The tool simulates the aforementioned disk scheduling algorithms, allowing users to input custom request sequences and observe how each algorithm processes them. By animating the movement of the disk head and calculating metrics like total head movement, the tool offers a hands-on learning experience. It serves as an educational aid for students and enthusiasts aiming to deepen their understanding of operating system concepts related to disk scheduling.

METHODOLOGY

The development of the Disk Scheduling Visualization Tool involved a systematic approach encompassing algorithm selection, design, implementation, and testing phases. The primary objective was to create an interactive web-based application that effectively demonstrates the functioning of various disk scheduling algorithms, thereby enhancing the understanding of these concepts.

1. Algorithm Selection

The project focuses on six widely recognized disk scheduling algorithms:

- **First-Come-First-Serve (FCFS)**
- **Shortest Seek Time First (SSTF)**
- **SCAN**
- **C-SCAN (Circular SCAN)**
- **LOOK**
- **C-LOOK (Circular LOOK)**

These algorithms were selected based on their relevance in operating systems and their varying approaches to optimizing disk head movement.

2. System Design

The system was designed with a user-centric approach to facilitate ease of use and interactivity:

- **User Interface (UI):** Developed using HTML and CSS to provide a clean and intuitive layout. The UI includes input fields for the request queue and initial head position, buttons to select the algorithm, and a canvas area to display the animation.
- **Visualization:** Utilized the HTML5 Canvas API to render the disk tracks and the movement of the disk head. Each algorithm's operation was animated to visually represent the sequence of head movements.
- **Interactivity:** JavaScript was employed to handle user inputs, algorithm logic, and dynamic updates to the visualization. Event listeners were implemented to respond to user actions such as input submission and algorithm selection.

3. Algorithm Implementation

Each disk scheduling algorithm was implemented as a separate function in JavaScript:

- **FCFS:** Processes requests in the order they arrive, moving the disk head from one request to the next sequentially.
- **SSTF:** Selects the request closest to the current head position, minimizing the seek time for each operation.
- **SCAN:** Moves the disk head in one direction, servicing requests until the end is reached, then reverses direction to service remaining requests.

- **C-SCAN:** Similar to SCAN but, upon reaching the end, the head returns to the beginning without servicing requests, then continues in the same direction.
- **LOOK:** Operates like SCAN but reverses direction when the last request in that direction is serviced, rather than reaching the end.
- **C-LOOK:** Functions like C-SCAN but reverses direction when the last request in that direction is serviced.

Each algorithm was tested with various input scenarios to ensure correct functionality and to compare performance metrics such as total head movement.

4. Testing and Validation

The tool underwent rigorous testing to validate its accuracy and performance:

- **Unit Testing:** Each algorithm function was tested independently with predefined inputs to verify correctness.
- **Integration Testing:** The entire system was tested to ensure that the UI, JavaScript logic, and visualization components worked seamlessly together.
- **User Testing:** Feedback was gathered from potential users to identify areas for improvement in usability and functionality.

5. Deployment

The final application was deployed as a static website, accessible through modern web browsers. No server-side components were required, making the tool lightweight and easy to access.

OBJECTIVES

The primary objective of this project is to design and develop an interactive, web-based tool that simulates and visualizes various disk scheduling algorithms to enhance the understanding of their impact on disk I/O operations in operating systems. This system is designed with the following specific objectives:

1. **Implementation of Multiple Disk Scheduling Algorithms**
   - To accurately implement six fundamental disk scheduling algorithms: First-Come-First-Serve (FCFS), Shortest Seek Time First (SSTF), SCAN, Circular SCAN (C-SCAN), LOOK, and Circular LOOK (C-LOOK).
   - To allow users to select and compare these algorithms to evaluate their performance under different scenarios.

2. **Interactive Disk Request Input Handling**
   - To enable users to dynamically input disk request sequences and initial head positions.
   - To provide flexibility in simulating various disk access patterns and scenarios, facilitating a deeper understanding of each algorithm's behaviour.

3. **Disk Head Movement Simulation and Visualization**
   - To simulate the movement of the disk head based on the selected algorithm and animate the process in real-time.
   - To display Gantt Charts and Timeline Charts using visualization libraries for a clear and intuitive understanding of the scheduling process.

4. **Performance Metrics Evaluation**
   - To calculate and display key performance metrics such as Total Head Movement, Seek Time, and Throughput for each algorithm.
   - To provide comparative analysis tools to assess the efficiency and effectiveness of different scheduling strategies.

5. **User-Friendly Interface**
   - To create an intuitive and responsive user interface that encourages interaction and facilitates learning through real-time feedback and visualization.
   - To ensure accessibility across various devices and platforms, providing a consistent user experience.

By achieving these objectives, the project aims to provide a comprehensive and engaging platform for understanding and analysing disk scheduling algorithms, thereby enhancing the learning experience for students and professionals in the field of operating sys

IMPLEMENTATION OF PROGRAM

Code:

Disk.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8">

    <title>Disk Scheduling Algorithms</title>

    <!--CSS and Script linking-->
    <link rel="stylesheet" href="mystyle.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
  </head>

  <body>
    <header>
      <nav id="header-nav" class="navbar navbar-default">
        <div class="container">
          <div class="navbar-header">

            <div class="navbar-brand">
              <a href="#"><h1>Disk Scheduling Algorithms</h1></a>
            </div>

            <button id="navbarToggle" type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#collapsable-nav" aria-expanded="false">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
          </div>

          <div id="collapsable-nav" class="collapse navbar-collapse">
            <ul id="nav-list" class="nav navbar-nav navbar-right">
              <li>
                <a href="Disk.html">
                  <i class="fa fa-file-text" aria-hidden="true"></i><br
class="hidden-xs"> Wiki</a>
              </li>
              <li>
                <a href="Simulate.html">
                  <i class="fa fa-area-chart" aria-hidden="true"></i><br
```

```
class="hidden-xs"> Simulate</a>
          </li>
          <li>
            <a href="#">
              <i class="fa fa-user-circle" aria-hidden="true"></i><br
class="hidden-xs"> About</a>
          </li>
        </ul><!-- #nav-list -->
      </div><!-- .collapse .navbar-collapse -->
    </div><!-- .container -->
   </nav><!-- #header-nav -->
  </header>

  <div class="diskinfo">
   <p class="hd">What are Disk Scheduling Algorithms?</p>

   <div id="intro">
    <img id="diskimg" src="img1.png" alt="disk image">
     A process needs two type of time, CPU time and IO time. For I/O, it
requests the Operating system to access the disk.However, the operating system
must be fare enough to satisfy each request and at the same time, operating
system must maintain the efficiency and speed of process execution.The technique
that operating system uses to determine the request which is to be satisfied
next is called disk scheduling.Disk scheduling is done by operating systems to
schedule I/O requests arriving for the disk. Disk scheduling is also known as
I/O Scheduling.
   </div>

   <div id="terms">
    <img id="diskimg2" src="img2.png" alt="disk image">
    <span id="hd2">Some important terminologies :</span><br><br>
    <b id="bold">Seek Time</b>
    - <br>Seek time is the time taken in locating the disk arm to a specified
track where the read/write request will be satisfied.<br>
    <b id="bold">Rotational Latency</b>
    - <br>It is the time taken by the desired sector to rotate itself to the
position from where it can access the R/W heads.<br>
    <b id="bold">Transfer Time</b>
    - <br> It is the time taken to transfer the data.<br>
    <b id="bold"> Disk Access Time</b>
    - <br>Disk access time is given as, <br>Disk Access Time = Rotational
Latency + Seek Time + Transfer Time<br>
    <b id="bold">Disk Response Time:</b>
    - <br>It is the average of time spent by each request waiting for the IO
operation.
   </div>


   <p class="hd">Why do we need Disk Scheduling Algorithms?</p>

   <div id="why">
    Disk scheduling is important because:
    <ul>
     <li>Multiple I/O requests may arrive by different processes and only one
I/O request can be served at a time by the disk controller.Thus other I/O
requests need to wait in the waiting queue and need to be scheduled.</li>
     <li>Two or more request may be far from each other so can result in
```

```
greater disk arm movement.</li>
       <li>Hard drives are one of the slowest parts of the computer system and
thus need to be accessed in an efficient manner.</li>
      </ul>

    </div>

    <p class="hd">Various Disk Scheduling Algorithms</p>

    <div class="cont">
      <div>
       <a id="algo" href="Simulate.html"><h2 class=" heady">First Come First
Serve (FCFS)<br></h2><hr style="width: 30%; border-width: 2px;"></a><br>
       <p class="description">FCFS is the simplest disk scheduling algorithm. As
the name suggests, this algorithm entertains requests in the order they arrive
in the disk queue. The algorithm looks very fair and there is no starvation (all
requests are serviced sequentially) but generally, it does not provide the
fastest service.</p>
      </div>
      <div>
       <a id="algo" href="Simulate.html"><h2 class=" heady">Shortest Seek Time
First (SSTF)<br></h2><hr style="width: 30%; border-width: 2px;"></a><br>
       <p class="description">Shortest seek time first (SSTF) algorithm selects
the disk I/O request which requires the least disk arm movement from its current
position regardless of the direction. It reduces the total seek time as compared
to FCFS. It allows the head to move to the closest track in the service
queue.</p>
      </div>
      <div>
       <a id="algo" href="Simulate.html"><h2 class=" heady">SCAN<br><br></h2><hr
style="width: 30%; border-width: 2px;"></a><br>
       <p class="description">In SCAN disk scheduling algorithm, head starts
from one end of the disk and moves towards the other end, servicing requests in
between one by one and reach the other end. Then the direction of the head is
reversed and the process continues as head continuously scan back and forth to
access the disk.</p>
      </div>
      <div>
       <a id="algo" href="Simulate.html"><h2 class=" heady">Circular SCAN (C-
SCAN)<br></h2><hr style="width: 30%; border-width: 2px;"></a><br>
       <p class="description">Circular SCAN (C-SCAN) scheduling algorithm is a
modified version of SCAN disk scheduling algorithm that deals with the
inefficiency of SCAN algorithm by servicing the requests more uniformly. Like
SCAN (Elevator Algorithm) C-SCAN moves the head from one end servicing all the
requests to the other end.</p>
      </div>
      <div>
       <a id="algo" href="Simulate.html"><h2 class=" heady">LOOK<br></h2><hr
style="width: 30%; border-width: 2px;"></a><br>
       <p class="description">The LOOK algorithm services request similarly as
SCAN algorithm meanwhile it also “looks” ahead as if there are more tracks that
are needed to be serviced in the same direction. If there are no pending
requests in the moving direction the head reverses the direction and start
servicing requests in the opposite direction. </p>
      </div>
      <div class=" heady">
       <a id="algo" href="Simulate.html"><h2 class="heady">Circular
```

```
LOOK<br></h2><hr style="width: 30%; border-width: 2px;"></a><br>
       <p class="description">In this algorithm, the head services requests only
in one direction until all the requests in this direction are not serviced and
then jumps back to the farthest request on the other direction and service the
remaining requests which gives a better uniform servicing as well as avoids
wasting seek time for going till the end of the disk.</p>
      </div>
     </div>
    </div>


    <script src="js/jquery-2.1.4.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="myscript.js"></script>

  </body>

</html>
```

Scrpit.js

```
$(function () {
      $("#navbarToggle").blur(function (event) {
        var screenWidth = window.innerWidth;
        if (screenWidth < 768) {
          $("#collapsable-nav").collapse('hide');
        }
      });

      $("#navbarToggle").click(function (event) {
        $(event.target).focus();
      });
});


$("#animate-button").click(function() {
      var btn1 = document.getElementById("animate-button");
      btn1.disabled= true;
      var b = document.forms["myForm"]["bitstream-input"].value;
      var i = document.forms["myForm"]["initial-input"].value;
      if(b == ""){
            alert("Enter the Sequence of Request queue!");
            return false;
      }
      if (b!= "" && i == "") {
            alert("Enter the value of Initial Cylinder!");
            return false;
      }

      var ini = parseInt(document.getElementById('initial-input').value);
      var final =  parseInt(document.getElementById('final-input').value);
```

```
	var str = document.getElementById('bitstream-input').value;
	var dir = document.getElementById('direction').value;

	var inp=[],r2=str.split(" "),r3;
	for(a1=0;a1<r2.length;++a1){
		if(r2[a1]==""){continue;}
		r3=parseInt(r2[a1]);
		inp.push(r3);

		if((r3>parseInt(final)) || (parseInt(ini)>parseInt(final))){
				alert("Invalid Input: Final cylinder has to be 
Greater!");
				return;
		}
	}

	final=parseInt(final);
	ini=parseInt(ini);

	if($('div.left').hasClass('transform') && window.matchMedia("(min-width: 
1249px)").matches) {
		$('.left').css("width", "30%");
		$('.left').css("margin", "30px");
		$('#plot-button').css("margin-left", "30px");
		$('#plot-button').css("margin-bottom", "5%");
		$('#animate-button').css("margin-bottom", "5%");
		$('#cmpr-button').css("margin-left", "25%");
		$('.container2').css("top", "800px");
		$('.container3').css("top", "1500px");



		setTimeout(function(){
			document.getElementById("canvas").style.visibility = "visible";
			myalgorithm(document.getElementById('algorithm').value, inp, 
ini, final, dir);
		}, 500);

	}

	else if(window.matchMedia("(min-width: 992px)").matches) {
		document.getElementById("canvas").style.visibility = "visible";
		myalgorithm(document.getElementById('algorithm').value, inp, ini, 
final, dir);
		$('.container2').css("top", "1250px");
	}

	else if(window.matchMedia("(min-width: 768px").matches) {
		document.getElementById("canvas").style.visibility = "visible";
```

```
            myalgorithm(document.getElementById('algorithm').value, inp, ini,
final, dir);
            $('.container2').css("top", "1500px");
      }

      else if(window.matchMedia("(min-width: 600px").matches){
            document.getElementById("canvas").style.visibility = "visible";
            myalgorithm(document.getElementById('algorithm').value, inp, ini,
final, dir);
            $('.container2').css("top", "1450px");

      }
      else {
            document.getElementById("canvas").style.visibility = "visible";
            myalgorithm(document.getElementById('algorithm').value, inp, ini,
final, dir);
            $('.container2').css("top", "1350px");
      }
});




/**** ANIMATION ****/


function myalgorithm(alg, inp, ini, final, dir){

      if(alg=="fcfs"){
            var op = fcfs(inp, ini, final);
            var target = op[0];
            var seek = op[1];
            animation(target[0].x);
            document.getElementById("am_alg_name").innerHTML = "FCFS";
            document.getElementById("am_alg_seek").innerHTML = "Total Seek Time:
" + seek;
      }

      if(alg=="sstf"){
            var op = sstf(inp, ini, final);
            var target = op[0];
            var seek = op[1];
            animation(target[0].x);
            document.getElementById("am_alg_name").innerHTML = "SSTF";
            document.getElementById("am_alg_seek").innerHTML = "Total Seek Time:
" + seek;
      }

      if(alg=="scan"){
            var f = document.forms["myForm"]["final-input"].value;
            if(f == ""){
                  alert("Enter the value of Final Cylinder");
                  return false;
            }

            var op = scan(inp, ini, final, dir);
            var target = op[0];
            var seek = op[1];
            console.log(seek);
            animation(target[0].x);
            document.getElementById("am_alg_name").innerHTML = "SCAN";
            document.getElementById("am_alg_seek").innerHTML = "Total Seek Time: 
" + seek;
      }

      if(alg=="c-scan"){
            var f = document.forms["myForm"]["final-input"].value;

            if(f == ""){
                  alert("Enter the value of Final Cylinder");
                  return false;
            }

            var op = cscan(inp, ini, final, dir);
            var target = op[0];
            var seek = op[1];
            var seq = [...target[0].x, ...target[1].x, ...target[2].x];
            animation(seq);
            document.getElementById("am_alg_name").innerHTML = "C-SCAN";
            document.getElementById("am_alg_seek").innerHTML = "Total Seek Time: 
" + seek;
      }

      if(alg=="look"){
            var op = look(inp, ini, final, dir);
            var target = op[0];
            var seek = op[1];
            animation(target[0].x);
            document.getElementById("am_alg_name").innerHTML = "LOOK";
            document.getElementById("am_alg_seek").innerHTML = "Total Seek Time: 
" + seek;
      }

      if(alg=="c-look"){
            var op = clook(inp, ini, final, dir);
            var target = op[0];
            var seek = op[1];
            var seq = [...target[0].x, ...target[1].x, ...target[2].x];
            animation(seq);
            document.getElementById("am_alg_name").innerHTML = "C-LOOK";
            document.getElementById("am_alg_seek").innerHTML = "Total Seek Time:
" + seek;
      }

}

function animation(values) {

      var canvas = document.getElementById("canvas");
      var ctx = canvas.getContext("2d");

      var cx = 350;
      var cy = 320;
      var PI2 = Math.PI * 2;
      var radius = 0;
      var totRadius = 0;

      var circles = [];

      const target = values;
      var max = Math.max(...target);

      if(max > 30){
            alert("Please Enter values beween 1 to 30 to visualize Animation !");
            return;
      }

      addCircle(20, "black");

      for(i=0; i<30; i++) {
         addCircle(10, "#A79C9D");
      }

      var targetIndex = 1;


      function addCircle(lineWidth, color) {
        if (radius == 0) {
          radius = lineWidth / 2;
        } else {
          radius += lineWidth;
        }
        totRadius = radius + lineWidth / 2;
        circles.push({
          radius: radius,
          color: color,
          width: lineWidth
        });
      }


      function drawCircle(circle, color) {
        ctx.beginPath();
        ctx.arc(cx, cy, circle.radius, 0, PI2);
        ctx.closePath();
        ctx.lineWidth = circle.width;
        ctx.strokeStyle = color;
        ctx.stroke();
      }

      function canvas_arrow(context, fromx, fromy, tox, toy) {

        var headlen = 5; // length of head in pixels
        var dx = tox - fromx;
        var dy = toy - fromy;
        var angle = Math.atan2(dy, dx);
        context.moveTo(fromx, fromy);
        context.lineTo(tox, toy);
        context.lineTo(tox - headlen * Math.cos(angle - Math.PI / 6), toy
-   headlen * Math.sin(angle - Math.PI / 6));
        context.moveTo(tox, toy);
        context.lineTo(tox - headlen * Math.cos(angle + Math.PI / 6), toy
-   headlen * Math.sin(angle + Math.PI / 6));
      }


      var fps = 1;
      let request;
      var sik ="Seek-Time: ";
      function animate() {


          setTimeout(function() {
          request = requestAnimationFrame(animate);

          // Drawing code goes here
          sik = sik + "|" + target[targetIndex].toString() + "-" +
target[targetIndex-1].toString() + "|";
          document.getElementById("cl-seek").innerHTML = sik;
          sik = sik + "+";


          ctx.clearRect(0, 0, canvas.width, canvas.height);
          for(var i=0; i< circles.length; i++)
             {
                var circle = circles[i];
                var color = circle.color;
                drawCircle(circles[i], color);
             }

          for (var i = 0; i < circles.length; i++) {
            var circle = circles[i];
            var color = circle.color;
            var p_circle = circles[target[targetIndex-1]];

            if (i == target[targetIndex]) {
             color = "white";
             ctx.font = "10px Arial";
             ctx.fillStyle = "black";
             drawCircle(circles[i], color);

             ctx.font = "15px Arial";
             ctx.fillStyle = "darkblue";
             ctx.fillText(i, 350 + circle.radius, 310);
             ctx.fillStyle = "black";
             ctx.fillText(target[targetIndex-1], 350 + p_circle.radius, 310);
             ctx.textAlign = "center";
             ctx.beginPath();
             canvas_arrow(ctx, 350+p_circle.radius, 320, 350+circle. radius, 320);
             ctx.strokeStyle = "black";
             ctx.lineWidth = 3;
             ctx.stroke();
            }
          }


          ctx.beginPath();
          ctx.arc(cx, cy, totRadius, 0, PI2);
          ctx.closePath();
          ctx.strokeStyle = "black";
          ctx.lineWidth = 5;
          ctx.stroke();

          targetIndex++;
          if(targetIndex==target.length)
             {
                   cancelAnimationFrame(request);
                   var btn = document.getElementById("animate-button");
                   btn.disabled= false;

             }
         }, 3000);

       }

       animate();
```

```
} // END OF ANIMATION()
/***** GRAPH *****/

var pre,v1,v2,v3,v4,v5,v6;

function fcfs(inp, ini, final){
      var x1=[];
      var y1=[];
      var seek=0;
      x1.push(ini);
      y1.push(0);
      var a1;
      for(a1=1;a1<=inp.length;++a1){
            x1.push(inp[a1-1]);
            y1.push(-1*a1);
            if(a1==1){
                  seek=seek+Math.abs(ini-inp[a1-1]);
            }
            else{
                  seek=seek+Math.abs(inp[a1-2]-inp[a1-1]);
            }
      }

      var trace1 = {
            x: x1,
            y: y1,
            type: 'scatter'
      };

      var data = [trace1];
      v1=seek;

      return [data, seek];

}

function sstf(inp, ini, final){
      var x1=[];
      var y1=[];
      var seek=0;
      var visited=[];
      var a1,a2;
      for(a1=0;a1<inp.length;++a1){
            visited[a1]=0;
      }

      x1.push(ini);
      y1.push(0);
      var hold=ini;
```

```
        for(a1=1;a1<=inp.length;++a1){
              var mn=10000;
              var idx;
              for(a2=0;a2<inp.length;++a2){
                    if(visited[a2]==0){
                          if(Math.abs(hold-inp[a2])<mn){
                                idx=a2;
                                mn=Math.abs(hold-inp[a2]);
                          }
                    }
              }
              seek=seek+Math.abs(hold-inp[idx]);
              visited[idx]=1;
              hold=inp[idx];
              x1.push(inp[idx]);
              y1.push(-1*a1);
        }

        var trace1 = {
              x: x1,
              y: y1,
              type: 'scatter'
        };

        var data = [trace1];
        v2=seek;

        return [data,seek];
  }

  function scan(inp, ini, final, dir){
        var x1=[];
        var y1=[];
        var seek=0;
        var visited=[];
        var a1,a2;
        console.log(inp);
        for(a1=0;a1<inp.length;++a1){
              visited[a1]=0;
        }

        x1.push(ini);
        y1.push(0);
        inp.sort(function(a, b){return a-b});
        if((ini<inp[0])||(ini>inp[inp.length-1])){
              var scan_use = sstf(inp,ini,final);
              var data = scan_use[0];
              var seek = scan_use[1];
              seek=v2;
```

```
                v3=seek;
                return [data,seek];
        }
        if(dir=="left"){
                var store,hold=ini;
                for(a1=0;a1<inp.length;++a1){if(inp[a1]<=ini){store=a1;}}
                var count=1;
                for(a1=store;a1>=0;--a1){
                        x1.push(inp[a1]);
                        y1.push(-1*count);
                        ++count;
                        seek=seek+Math.abs(hold-inp[a1]);
                        hold=inp[a1];
                }
                x1.push(0);
                y1.push(-1*count);
                seek=seek+hold;
                hold=0;
                ++count;
                for(a1=store+1;a1<inp.length;++a1){
                        x1.push(inp[a1]);
                        y1.push(-1*count);
                        ++count;
                        seek=seek+Math.abs(hold-inp[a1]);
                        hold=inp[a1];
                }
        }
        else{
                var store,hold=ini;
                for(a1=0;a1<inp.length;++a1){if(inp[a1]>=ini){store=a1;break}}
                var count=1;
                for(a1=store;a1<inp.length;++a1){
                        x1.push(inp[a1]);
                        y1.push(-1*count);
                        ++count;
                        seek=seek+Math.abs(hold-inp[a1]);
                        hold=inp[a1];
                }
                x1.push(final);
                y1.push(-1*count);
                seek=seek+parseInt(final)-hold;
                hold=final;
                ++count;
                for(a1=store-1;a1>=0;--a1){
                        x1.push(inp[a1]);
                        y1.push(-1*count);
                        ++count;
                        seek=seek+Math.abs(hold-inp[a1]);
                        hold=inp[a1];
```

```
            }

        }

        var trace1 = {
            x: x1,
            y: y1,
            type: 'scatter'
        };

        var data = [trace1];
        v3=seek;
        console.log(seek);
        console.log(x1);
        return [data, seek];
}

function cscan(inp, ini, final, dir){
        var x1=[];
        var y1=[];
        var x2=[];
        var y2=[];
        var x3=[];
        var y3=[];
        var seek=0;
        var visited=[];
        var a1,a2;
        for(a1=0;a1<inp.length;++a1){
            visited[a1]=0;
        }

        x1.push(ini);
        y1.push(0);
        inp.sort(function(a, b){return a-b});
        if((ini<inp[0])||(ini>inp[inp.length-1])){
            var cscan_use = sstf(inp,ini,final);
            var data = cscan_use[0];
            var seek = cscan_use[1];
            seek=v2;
            v4=seek;
            return [data,seek];
        }
        if(dir=="left"){
            var store,hold=ini;
            for(a1=0;a1<inp.length;++a1){if(inp[a1]<=ini){store=a1;}}
            var count=1;
            for(a1=store;a1>=0;--a1){
                x1.push(inp[a1]);
                y1.push(-1*count);
                ++count;
                seek=seek+Math.abs(hold-inp[a1]);
                hold=inp[a1];
            }
            x1.push(0);
            y1.push(-1*count);
            seek=seek+hold;
            hold=final;
            x2.push(0);
            y2.push(-1*count);
            x2.push(final);
            y2.push(-1*count);

            x3.push(final);
            y3.push(-1*count);
            ++count;
            for(a1=inp.length-1;a1>store;--a1){
                x3.push(inp[a1]);
                y3.push(-1*count);
                ++count;
                seek=seek+Math.abs(hold-inp[a1]);
                hold=inp[a1];
            }
        }
        else{
            var store,hold=ini;
            for(a1=0;a1<inp.length;++a1){if(inp[a1]>=ini){store=a1;break;}}
            var count=1;
            for(a1=store;a1<inp.length;++a1){
                x1.push(inp[a1]);
                y1.push(-1*count);
                ++count;
                seek=seek+Math.abs(hold-inp[a1]);
                hold=inp[a1];
            }
            x1.push(final);
            y1.push(-1*count);
            seek=seek+final-hold;
            hold=0;
            x2.push(final);
            y2.push(-1*count);
            x2.push(0);
            y2.push(-1*count);

            x3.push(0);
            y3.push(-1*count);
            ++count;
            for(a1=0;a1<store;++a1){
                x3.push(inp[a1]);
                y3.push(-1*count);
                ++count;
                seek=seek+Math.abs(hold-inp[a1]);
                hold=inp[a1];
            }

        }
        var trace1 = {
            x: x1,
            y: y1,
            type: 'scatter',
            name: ''
        };
        var trace2 = {
            x: x2,
            y: y2,
            mode: 'lines',
            name: '',
            line: {
                dash: 'dashdot',
                width: 4
            }
        };
        var trace3 = {
            x: x3,
            y: y3,
            type: 'scatter',
            name: ''
        };

        v4=seek;

        var data = [trace1,trace2,trace3];
        var max1 = Math.max(...trace1.x);
        var max2 = Math.max(...trace2.x);
        var max3 = Math.max(...trace3.x);

        var min1 = Math.min(...trace1.x);
        var min2 = Math.min(...trace2.x);
        var min3 = Math.min(...trace3.x);

        seek = max1 + max2 + max3 - min1 - min2 - min3;




        return [data, seek];
    }
```

```
function look(inp, ini, final, dir){
      var x1=[];
      var y1=[];
      var seek=0;
      var visited=[];
      var a1,a2;
      for(a1=0;a1<inp.length;++a1){
            visited[a1]=0;
      }

      x1.push(ini);
      y1.push(0);
      inp.sort(function(a, b){return a-b});
      if((ini<inp[0])||(ini>inp[inp.length-1])){
            var look_use = sstf(inp,ini,final);
            var data = look_use[0];
            var seek = look_use[1];
            seek=v2;
            v5=seek;
            return [data,seek];
      }
      if(dir=="left"){
            var store,hold=ini;
            for(a1=0;a1<inp.length;++a1){if(inp[a1]<=ini){store=a1;}}
            var count=1;
            for(a1=store;a1>=0;--a1){
                  x1.push(inp[a1]);
                  y1.push(-1*count);
                  ++count;
                  seek=seek+Math.abs(hold-inp[a1]);
                  hold=inp[a1];
            }

            for(a1=store+1;a1<inp.length;++a1){
                  x1.push(inp[a1]);
                  y1.push(-1*count);
                  ++count;
                  seek=seek+Math.abs(hold-inp[a1]);
                  hold=inp[a1];
            }
      }
      else{
            var store,hold=ini;
            for(a1=0;a1<inp.length;++a1){if(inp[a1]>=ini){store=a1;break}}
            var count=1;
            for(a1=store;a1<inp.length;++a1){
                  x1.push(inp[a1]);
                  y1.push(-1*count);
```

```
                    ++count;
                    seek=seek+Math.abs(hold-inp[a1]);
                    hold=inp[a1];
              }

              for(a1=store-1;a1>=0;--a1){
                    x1.push(inp[a1]);
                    y1.push(-1*count);
                    ++count;
                    seek=seek+Math.abs(hold-inp[a1]);
                    hold=inp[a1];
              }

        }

        var trace1 = {
              x: x1,
              y: y1,
              type: 'scatter'
        };

        var data = [trace1];
        v5=seek;

        return [data, seek];
}

function clook(inp, ini, final, dir){
        var x1=[];
        var y1=[];
        var x2=[];
        var y2=[];
        var x3=[];
        var y3=[];
        var seek=0;
        var visited=[];
        var a1,a2;
        for(a1=0;a1<inp.length;++a1){
              visited[a1]=0;
        }

        x1.push(ini);
        y1.push(0);
        inp.sort(function(a, b){return a-b});
        if((ini<inp[0])||(ini>inp[inp.length-1])){
              var clook_use = sstf(inp,ini,final);
              var data = clook_use[0];
              var seek = clook_use[1];
              seek=v2;
```

```
		v6=seek;
		return [data, seek];
	}
	if(dir=="left"){
		var store,hold=ini;
		for(a1=0;a1<inp.length;++a1){if(inp[a1]<=ini){store=a1;}}
		var count=1;
		for(a1=store;a1>=0;--a1){
			x1.push(inp[a1]);
			y1.push(-1*count);
			++count;
			seek=seek+Math.abs(hold-inp[a1]);
			hold=inp[a1];
		}

		x2.push(hold);
		y2.push(-1*(count-1));
		x2.push(inp[inp.length-1]);
		y2.push(-1*(count-1));
		x3.push(inp[inp.length-1]);
		y3.push(-1*(count-1));

		hold=inp[inp.length-1];
		for(a1=inp.length-2;a1>store;--a1){
			x3.push(inp[a1]);
			y3.push(-1*count);
			++count;
			seek=seek+Math.abs(hold-inp[a1]);
			hold=inp[a1];
		}
	}
	else{
		var store,hold=ini;
		for(a1=0;a1<inp.length;++a1){if(inp[a1]>=ini){store=a1;break;}}
		var count=1;
		for(a1=store;a1<inp.length;++a1){
			x1.push(inp[a1]);
			y1.push(-1*count);
			++count;
			seek=seek+Math.abs(hold-inp[a1]);
			hold=inp[a1];
		}

		x2.push(hold);
		y2.push(-1*(count-1));
		x2.push(inp[0]);
		y2.push(-1*(count-1));

		x3.push(inp[0]);
```

```
			y3.push(-1*(count-1));

			hold=inp[0];
			for(a1=1;a1<store;++a1){
				x3.push(inp[a1]);
				y3.push(-1*count);
				++count;
				seek=seek+Math.abs(hold-inp[a1]);
				hold=inp[a1];
			}

		}

		var trace1 = {
			x: x1,
			y: y1,
			type: 'scatter',
			name: ''
		};
		var trace2 = {
			x: x2,
			y: y2,
			mode: 'lines',
			name: '',
			line: {
				dash: 'dashdot',
				width: 4
			}
		};
		var trace3 = {
			x: x3,
			y: y3,
			type: 'scatter',
			name: ''
		};

		var data = [trace1,trace2,trace3];
		v6=seek;

		var data = [trace1,trace2,trace3];
		var max1 = Math.max(...trace1.x);
		var max2 = Math.max(...trace2.x);
		var max3 = Math.max(...trace3.x);

		var min1 = Math.min(...trace1.x);
		var min2 = Math.min(...trace2.x);
		var min3 = Math.min(...trace3.x);

		seek = max1 + max2 + max3 - min1 - min2 - min3;
```

```
      return [data, seek];
}

function getBitStreamAndPlot(event, r1, ini, final, alg, side){
      var b = document.forms["myForm"]["bitstream-input"].value;
      var i = document.forms["myForm"]["initial-input"].value;
      if(b == ""){
            alert("Enter the Sequence of Request queue!");
            return false;
      }
      if (b!= "" && i == "") {
            alert("Enter the value of Initial Cylinder!");
            return false;
      }

      var inp=[],r2=r1.split(" "),r3;
      for(a1=0;a1<r2.length;++a1){
            if(r2[a1]==""){continue;}
            r3=parseInt(r2[a1]);
            inp.push(r3);

            if((r3>parseInt(final)) || (parseInt(ini)>parseInt(final))){
                        alert("Invalid Input: Final cylinder has to be
Greater!");
                        return;
            }
      }

      final=parseInt(final);
      ini=parseInt(ini);
      dir=side;
      pre=1;

      if(alg=="fcfs"){
            var alg_use = fcfs(inp, ini, final);
            var plt_alg = "FCFS";
      }
      if(alg=="sstf"){
            var alg_use = sstf(inp, ini, final);
            var plt_alg = "SSTF";

      }
      if(alg=="scan"){
            var f = document.forms["myForm"]["final-input"].value;

            if(f == ""){
                  alert("Enter the value of Final Cylinder");
                  return false;
```

```
        }

        var alg_use = scan(inp, ini, final, dir);
        var plt_alg = "SCAN";

    }
    if(alg=="c-scan"){
        var alg_use = cscan(inp, ini, final, dir);
        var plt_alg = "C-SCAN";

        var f = document.forms["myForm"]["final-input"].value;

        if(f == ""){
            alert("Enter the value of Final Cylinder");
            return false;
        }

    }
    if(alg=="look"){
        var alg_use = look(inp, ini, final, dir);
        var plt_alg = "LOOK";
    }
    if(alg=="c-look"){
        var alg_use = clook(inp, ini, final, dir);
        var plt_alg = "C-LOOK";
    }

    var data = alg_use[0];
    var seek = alg_use[1];

    var layout = {
        xaxis: {
            autorange: true,
            showgrid: true,
            zeroline: false,
            showline: true,
            autotick: true,
            ticks: '',
            showticklabels: true,
            title: 'Cylinder Number'
        },
        yaxis: {
            autorange: true,
            showgrid: false,
            zeroline: false,
            showline: false,
            autotick: true,
            ticks: '',
            showticklabels: false,
            }
      };

      if(pre){
            Plotly.newPlot('graph_area', data, layout);
            var val = data[0].x;
            var tot_seek = "Seek-Time: ";
            for(var i=1; i<val.length; i++){
                  tot_seek = tot_seek + "|" + val[i].toString() + "-" + val[i-
1].toString()  + "|" ;
                  if(i<val.length-1)
                        tot_seek = tot_seek + "+";

            }
            document.getElementById("plt_alg_name").innerHTML = plt_alg;
            document.getElementById("cal-seek").innerHTML = tot_seek + " = " +
seek;

            document.getElementById("graph_area").style.visibility = "visible";
      }

}
```

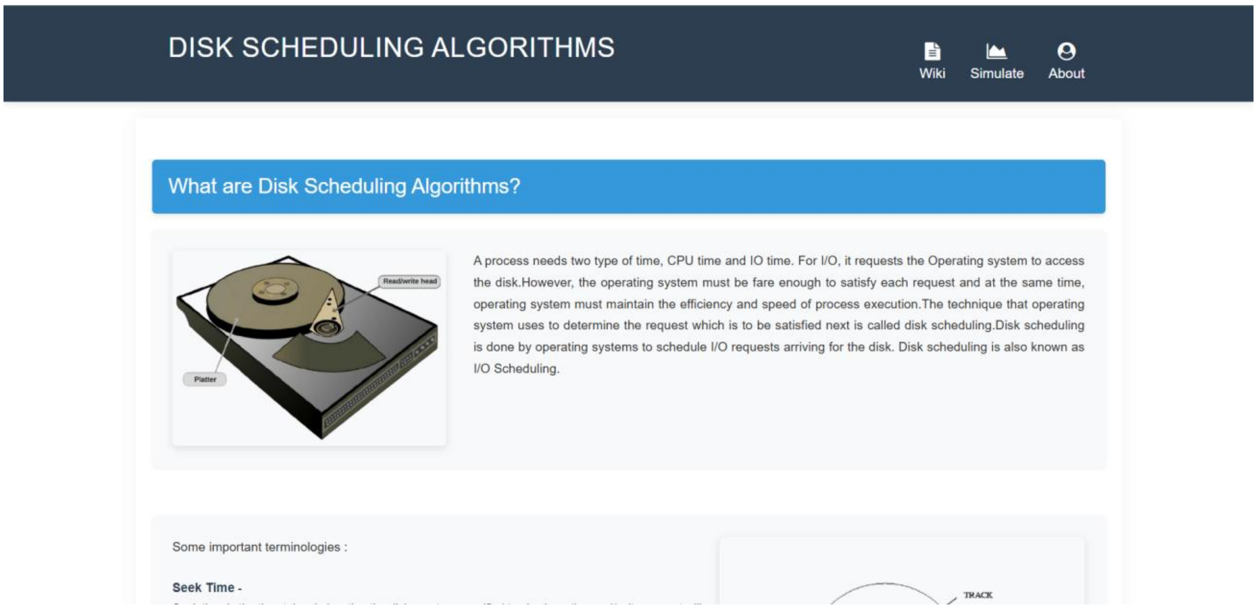
Styles.css

```
 * {
   box-sizing: border-box;
 }

 body {
   background-color: rgb(114, 110, 110);
   background-size: 400% 400%;
   animation: gradient 15s ease infinite;
   font-family:Verdana ;
 }

 @keyframes gradient {
   0% {
     background-position: 0% 50%;
   }
   50% {
     background-position: 100% 50%;
```

```
  }
  100% {
    background-position: 0% 50%;
  }
}

/** HEADER **/
#header-nav {
  background-color: #2c3e50;
  border-radius: 0;
  border: 0;
  box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.1);
  position: relative;
  z-index: 999;
  padding: 10px 0;
}

.navbar-brand {
  padding-top: 15px;
}

.navbar-brand h1 {
  padding: 10px;
  color: #ffffff;
  font-size: 1.8em;
  text-transform: uppercase;
  font-weight: 500;
  letter-spacing: 1px;
  margin-top: 0;
  margin-bottom: 0;
  line-height: 1;
  transition: all 0.3s ease;
}

.navbar-brand a:hover, .navbar-brand a:focus {
  text-decoration: none;
}

.navbar-brand a:hover h1 {
  color: #3498db;
}

#nav-list {
  margin-top: 10px;
}
```

```
#nav-list a {
  color: #ffffff;
  text-align: center;
  font-size: 1.2em;
  margin-top: 10px;
  transition: all 0.3s ease;
  border-radius: 4px;
}

#nav-list a:hover {
  background: #3498db;
  color: #ffffff;
  transform: translateY(-2px);
}

#nav-list a i {
  font-size: 1.3em;
  padding-bottom: 5px;
}

.navbar-header button.navbar-toggle, .navbar-header
.icon-bar {
  border: 1px solid #fff;
}

.navbar-header button.navbar-toggle {
  clear: both;
  margin-top: -30px;
}
/* END HEADER */
```

Output:

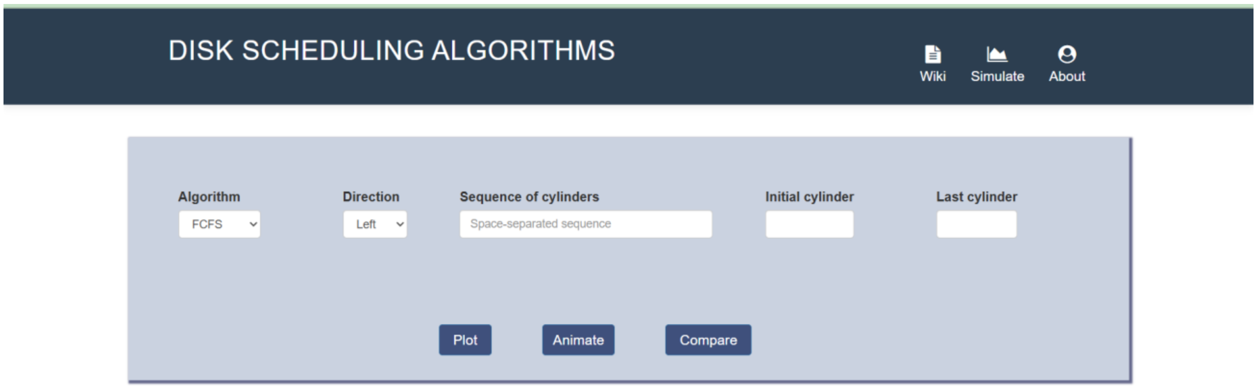

DISK SCHEDULING ALGORITHMS

What are Disk Scheduling Algorithms?

A process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk.However, the operating system must be fare enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O Scheduling.

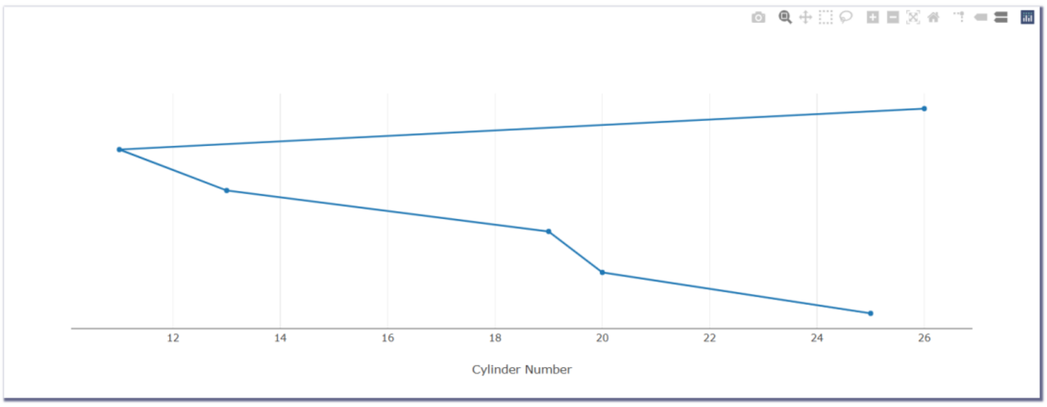
Some important terminologies :

Seek Time -


DISK SCHEDULING ALGORITHMS

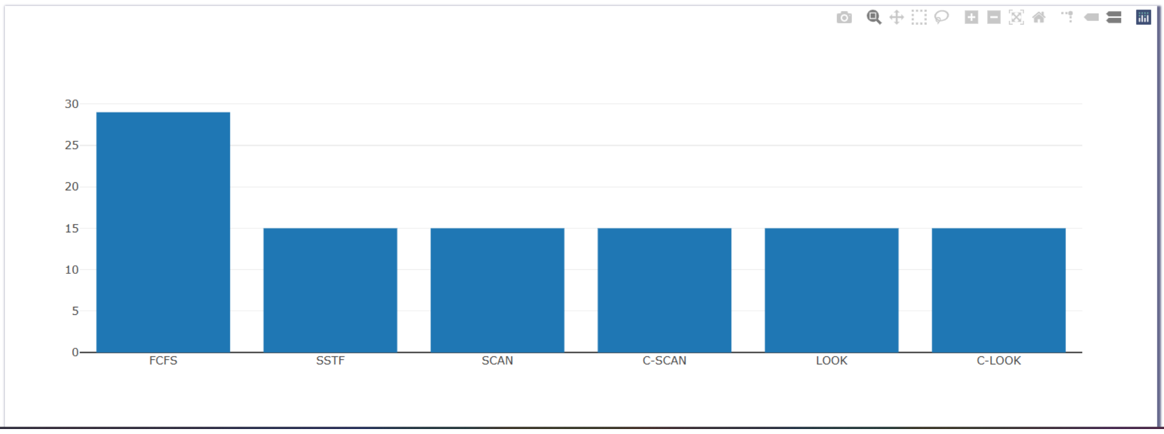
Algorithm	Direction	Sequence of cylinders	Initial cylinder	Last cylinder
FCFS	Left	Space-separated sequence		

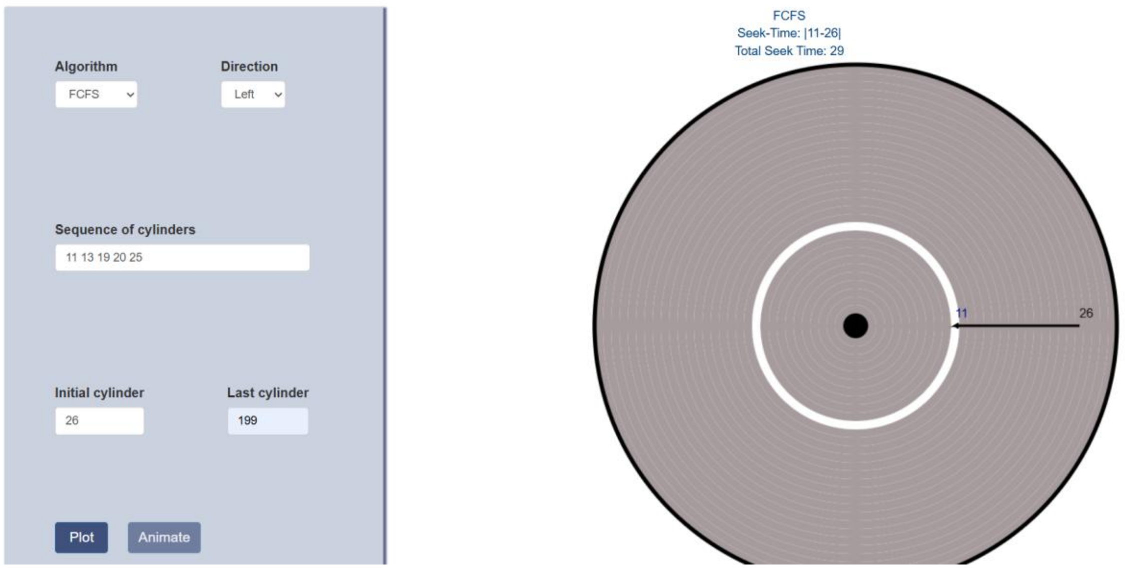
FCFS



Seek-Time: |11-26|+|13-11|+|19-13|+|20-19|+|25-20| = 29

Comparison-Chart





Algorithm
FCFS
Direction
Left

Sequence of cylinders
11 13 19 20 25

Initial cylinder
26
Last cylinder
199

FCFS
Seek-Time: |11-26|
Total Seek Time: 29

CONCLUSION

The **Disk Scheduling Visualization Tool** serves as an invaluable educational resource, bridging the gap between theoretical concepts and practical understanding in the realm of operating systems. By simulating and visually representing the operations of six fundamental disk scheduling algorithms—First-Come-First-Serve (FCFS), Shortest Seek Time First (SSTF), SCAN, Circular SCAN (C-SCAN), LOOK, and Circular LOOK (C-LOOK)—the tool provides users with a dynamic platform to observe and compare the efficiency of each algorithm in real-time.

Through interactive simulations, users can input custom disk request sequences and initial head positions, enabling them to visualize the movement of the disk head and the sequence in which requests are serviced. This hands-on approach facilitates a deeper comprehension of how each algorithm manages disk I/O operations, highlighting their strengths and limitations in various scenarios.

The inclusion of performance metrics such as total head movement, seek time, and throughput further enriches the learning experience, allowing users to quantitatively assess the impact of different scheduling strategies on system performance. By comparing these metrics across algorithms, users can gain insights into the trade-offs involved in optimizing disk access.
In essence, this project not only enhances the understanding of disk scheduling algorithms but also fosters critical thinking and analytical skills among students and professionals in the field of computer science. It underscores the importance of efficient disk scheduling in optimizing system performance and provides a foundation for exploring more advanced topics in operating systems.

FUTURISTIC SCOPE

The **Disk Scheduling Visualization Tool** serves as a foundational platform for understanding and analysing disk scheduling algorithms. As technology evolves, there are several avenues for enhancing this tool to align with contemporary advancements in operating systems and storage technologies.

1. Integration with Solid-State Drives (SSDs)

Traditional disk scheduling algorithms were primarily designed for Hard Disk Drives (HDDs). With the widespread adoption of SSDs, which have different performance characteristics, there is a need to adapt or develop new algorithms tailored for SSDs. Future versions of the tool could simulate SSD-specific scheduling strategies, providing users with insights into optimizing I/O operations for modern storage devices.

2. Incorporation of Machine Learning Techniques

The application of machine learning in disk scheduling is an emerging field. Algorithms can be developed to learn from historical I/O patterns and adapt scheduling strategies accordingly. Implementing such intelligent scheduling mechanisms could enhance the tool's capability to simulate and visualize adaptive disk scheduling approaches.

3. Real-Time Disk Scheduling Simulations

Integrating real-time constraints into disk scheduling is crucial for applications requiring stringent timing guarantees. Future enhancements could include the simulation of real-time disk scheduling algorithms, allowing users to visualize how deadlines and priorities affect disk I/O operations.

4. Visualization of Advanced Scheduling Metrics

Beyond basic performance metrics, future versions of the tool could incorporate advanced metrics such as Quality of Service (QoS) parameters, fairness indices, and energy consumption estimates. Visualizing these metrics would provide a more comprehensive understanding of the trade-offs involved in different disk scheduling strategies.

5. Cross-Platform Compatibility and Cloud Integration

Expanding the tool's accessibility by developing mobile applications and integrating cloud-based simulations could broaden its user base. This would enable users to access and interact with the tool across various devices and platforms, facilitating learning and experimentation in diverse environments.

6. Educational Modules and Interactive Tutorials

To enhance the learning experience, the tool could include structured educational modules and interactive tutorials. These resources would guide users through the concepts of disk scheduling, providing step-by-step explanations and hands-on exercises to reinforce learning.

REFERENCES

1. **Disk Scheduling: Selection of Algorithm** by S. Yashvir and Om Prakash.
Explores various disk scheduling algorithms and their selection criteria, offering insights into algorithmic efficiency.

2. Disk Scheduling Revisited by John K. Ousterhout, M. Satyanarayana, and David C. Anderson.
Discusses the evolution of disk scheduling algorithms and introduces new approaches to improve disk utilization.

3. **GeeksforGeeks – Disk Scheduling Algorithms**
This tutorial offers an in-depth explanation of various disk scheduling algorithms, including FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK. It provides examples, advantages, disadvantages, and performance comparisons for each algorithm.
https://www.geeksforgeeks.org/disk-scheduling-algorithms/

4. **Scaler Topics – Disk Scheduling Algorithms**
Scaler Topics provides a comprehensive guide on disk scheduling algorithms, covering their working principles, use cases, and performance metrics. The article includes visual aids to enhance understanding.
https://www.scaler.com/topics/disk-scheduling-algorithms/

5. TakeUForward – Disk Scheduling Algorithms
This article explains various disk scheduling algorithms with clear examples and illustrations. It discusses the pros and cons of each algorithm and provides insights into their practical applications.
https://takeuforward.org/operating-system/disk-scheduling-algorithms/

6. **W3Schools** HTML, CSS, and JavaScript References. Retrieved from:
https://www.w3schools.com/