SRM UNIVERSITY AP

Department Computer Science and Engineering
School of Engineering and Sciences

OPERATING SYSTEMS

PROJECT REPORT ON

SSTF DISC SCHEDULING

Submitted by
G.Ganga Srinivas(AP21110010262)
G.Madhukar(AP21110010277)
G.Maruti Mahesh(AP21110010289)
Y.Rohith Kamal Kumar(AP21110010303)
U.Bogesh(AP21110010309)
CSE-E

Submitted to
Dr.Manikandan V M

TABLE OF CONTENTS

OBJECTIVE

The objective of the SSTF (Shortest Seek Time First) disk scheduling project is to implement a disk scheduling algorithm that minimizes the seek time and maximizes the disk's overall performance. The SSTF algorithm selects the next request that is closest to the current head position, reducing the seek time required to access data.

Key objectives of an SSTF disk scheduling project are:

- **Minimizing seek time:** The primary goal is to reduce the time taken by the disk arm to move to the desired track, resulting in faster data retrieval and improved disk performance.
- **Maximizing throughput:** By optimizing the disk access patterns, the SSTF algorithm aims to increase the number of requests serviced per unit of time, thus improving the overall system throughput.
- **Fairness and responsiveness:** The SSTF algorithm should aim to provide fair access to all processes or requests, ensuring that no particular process is starved or delayed excessively.
- **Avoiding starvation:** The algorithm should prevent any request from being continuously neglected or bypassed indefinitely. All requests should eventually be serviced.
- **Implementing efficient data structures:** The project may involve designing and implementing data structures, such as queues or lists, to efficiently manage the pending disk requests and track the current head position.

Overall, the objective of an SSTF disk scheduling project is to improve disk performance by minimizing seek time and maximizing system throughput, while ensuring fairness and responsiveness to all requests.

TOOLS AND TECHNOLOGIES USED

IDE:

Visual Studio Code

Technologies Stack Used:

HTML, CSS, JavaScript.

SOURCE CODE

```
<!DOCTYPE html>
<html>
<head>
  <title>Shortest Seek Time First</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
      background-color: #000000;
    }
    h1 {
      text-align: center;
      font-size: 36px;
      color: #333;
      margin-bottom: 20px;
      color: rgb(0, 255, 55);
    }
    h2{
        color: rgb(0, 255, 55);
        font-size: 30px;
    }
    label {
      display: block;
      margin-bottom: 8px;
      color: white;
    }
    input[type="number"] {
      padding: 8px;
      border-radius: 4px;
      border: 1px solid #ccc;
      width: 100%;
      box-sizing: border-box;
```

```
    }
    #input-container
{
      margin-top: 20px;
    }
    #run-button
{
      display: block;
      margin: 10px auto;
      padding: 10px 20px;
      font-size: 16px;
      border-radius: 4px;
      background-color: #4CAF50;
      color: #fff;
      border: none;
      cursor: pointer;
    }
    #output
{
      margin-top: 20px;
      padding: 10px;
      background-color: #f4f4f4;
      border-radius: 4px;
      overflow-wrap: break-word;
    }
    .enter{
color: white;
    }
    .initial{
        color: white;
    }
    .white{
        color: white;
    }
    .white2{
```

```
        color: white;
    }
  </style>
</head>
<body>
    <u class="white">
        <h1>Operating System Project</h1>
    </u>
  <u class="white2">
    <center>
        <h2 class>Shortest Seek Time First</h2>
      </center>
  </u>
  <div id="input-container">
    <label for="request-count" class="enter">Enter the number of disk
requests:</label>
    <input type="number" id="request-count">
    <div id="request-inputs"></div>
    <label for="head-position" class="initial">Enter the initial position of the
head:</label>
    <input type="number" id="head-position">
    <button id="run-button">Run Disk Scheduling</button>
  </div>
  <div id="output"></div>
  <script>
    function calculatedifference(request, head, diff, n) {
      for (let i = 0; i < n; i++) {
        diff[i][0] = Math.abs(head - request[i]);
      } }
    function findMIN(diff, n) {
      let index = -1;
      let minimum = 1e9;
      for (let i = 0; i < n; i++) {
        if (!diff[i][1] && minimum > diff[i][0]) {
          minimum = diff[i][0];
```

```
          index = i;
        }  }
      return index;
    }
    function shortestSeekTimeFirst(request, head, n) {
      if (n == 0) {
        return;
      }
      let diff = new Array(n);
      for (let i = 0; i < n; i++) {
        diff[i] = new Array(2);
        diff[i][1] = 0;
      }
      let seekcount = 0;
      let seeksequence = new Array(n + 1);
      seeksequence[0] = head;
      for (let i = 0; i < n; i++) {
        calculatedifference(request, head, diff, n);
        let index = findMIN(diff, n);
        diff[index][1] = 1;
        seekcount += diff[index][0];
        head = request[index];
        seeksequence[i + 1] = head;
      }
      let outputElement = document.getElementById('output');
      outputElement.innerHTML = "Total number of seek operations = " +
seekcount + "<br>";
      outputElement.innerHTML += "Seek sequence is: " +
seeksequence.join(" -> ");
    }
    function createInputFields(count) {
      let requestInputsContainer =
document.getElementById('request-inputs');
      requestInputsContainer.innerHTML = '';
      for (let i = 1; i <= count; i++) {
```

```
        let label = document.createElement('label');
        label.setAttribute('for', 'request-' + i);
        label.innerText = 'Enter disk request ' + i + ':';
        let input = document.createElement('input');
        input.setAttribute('type', 'number');
        input.setAttribute('id', 'request-' + i);
        requestInputsContainer.appendChild(label);
        requestInputsContainer.appendChild(input);
      } }
    function runDiskScheduling() {
      let requestCountInput = document.getElementById('request-count');
      let requestCount = parseInt(requestCountInput.value);
      let proc = [];
      for (let i = 1; i <= requestCount; i++) {
        let requestInput = document.getElementById('request-' + i);
        proc.push(parseInt(requestInput.value));
      }
      let headInput = document.getElementById('head-position');
      let head = parseInt(headInput.value);
      shortestSeekTimeFirst(proc, head, requestCount);
    }
    document.getElementById('run-button').addEventListener('click',
runDiskScheduling);
    let initialRequestCount = 0;
    let requestCountInput = document.getElementById('request-count');
    requestCountInput.addEventListener('change', function() {
      let newRequestCount = parseInt(requestCountInput.value);
      if (newRequestCount !== initialRequestCount) {
        createInputFields(newRequestCount);
        initialRequestCount = newRequestCount;
      }
    });
  </script>
  </body>
  </html>
```

SAMPLE OUTPUT

FUTURE WORK

In future work for the SSTF (Shortest Seek Time First) disk scheduling project, several areas can be explored to further enhance the algorithm and its implementation. Advanced optimization algorithms can be developed to improve the efficiency of the SSTF disk scheduling.Expanding the project to handle multiple disks simultaneously can lead to more efficient utilization of resources.By exploring these areas of future work, the SSTF disk scheduling project can continue to evolve and deliver even better disk performance, responsiveness, and user experience.

REFERENCE LINK

https://sstf.netlify.app