

Student 1:

NAME: PUNNAM NAGA CHAITANYA

SECTION : K23CW

ROLL NUMBER : 42

REGISTRATION NUMBER : 12325544

Course Code: CSE316

**1. Project Overview**

Project Title:

The Advanced Disk Scheduling Simulator is designed to simulate and compare multiple disk scheduling algorithms to optimize seek time and reduce disk head movement. The project demonstrates the practical application of disk scheduling algorithms, providing a deeper understanding of their functionality and performance differences.

**Key Features:**

* Simulates the following scheduling algorithms: oFCFS (First

Come First Serve) oSJF (Shortest Job First - Non-preemptive) o

Round Robin (RR) with Time Quantum oPriority Scheduling (Non-preemptive)

* Generates a Gantt Chart to visualize process execution.
* Computes and displays key performance metrics:

oTurnaround Time (TAT) oWaiting Time (WT) o

Average TAT and WT

**2. Module-Wise Breakdown**

**Module 1**: Input and Setup

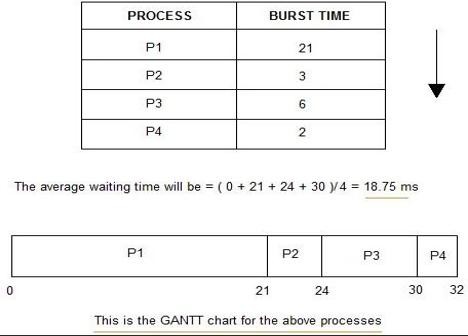
 Takes input for:

* Initial head position
* Sequence of disk requests
* Total number of requests

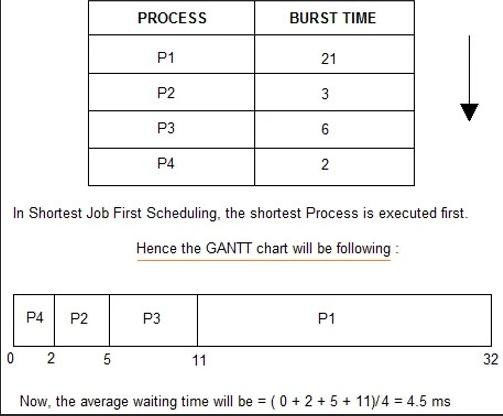
**Module 2:Disk Scheduling Algorithms**

FCFS (First Come First Serve):

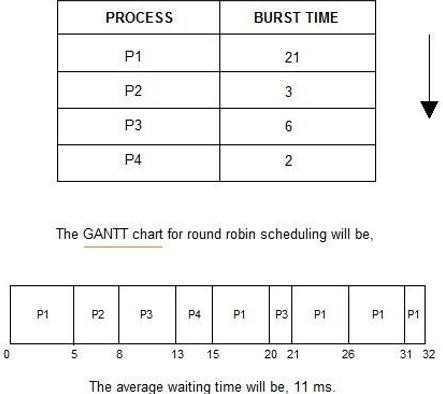
* + Processes executed in order of arrival.



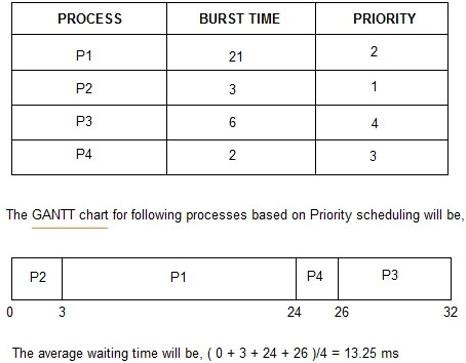
* SJF (Shortest Job First - Non-preemptive):
  + Selects the process with the shortest burst time.



* Round Robin (RR): oEach process is assigned a fixed time slice (quantum).



* Priority Scheduling:
  + Selects process based on priority (lower value = higher priority).



**Module 3:** Execution and Results

* Calculates total head movements for each algorithm.
* Displays the order of request processing.

**Module 4:** Output and Visualization

* + Displays simulation results with metrics.
* Provides a comparison of algorithms for performance analysis.

1. **Functionalities**

Core Functionalities:

* **Input Requests:**
  + Takes disk request sequence and initial head position.
* **Scheduler Selection:**
  + Allows the user to choose a scheduling algorithm.
* **Performance Analysis:**
  + Displays total head movements and order of requests.

**Additional Functionalities**:

* **Comparison of Algorithms:**
  + Compares results of different algorithms.
* **Repeat or Exit Option:**
  + Allows multiple executions of different algorithms.

1. **Technology Used**

**Programming Languages:**

* + C++ oPrimary language for project implementation.

**Libraries and Tools:**

* **Standard I/O Library:** #include<iostream>
* **Algorithm Library:** #include<algorithm>

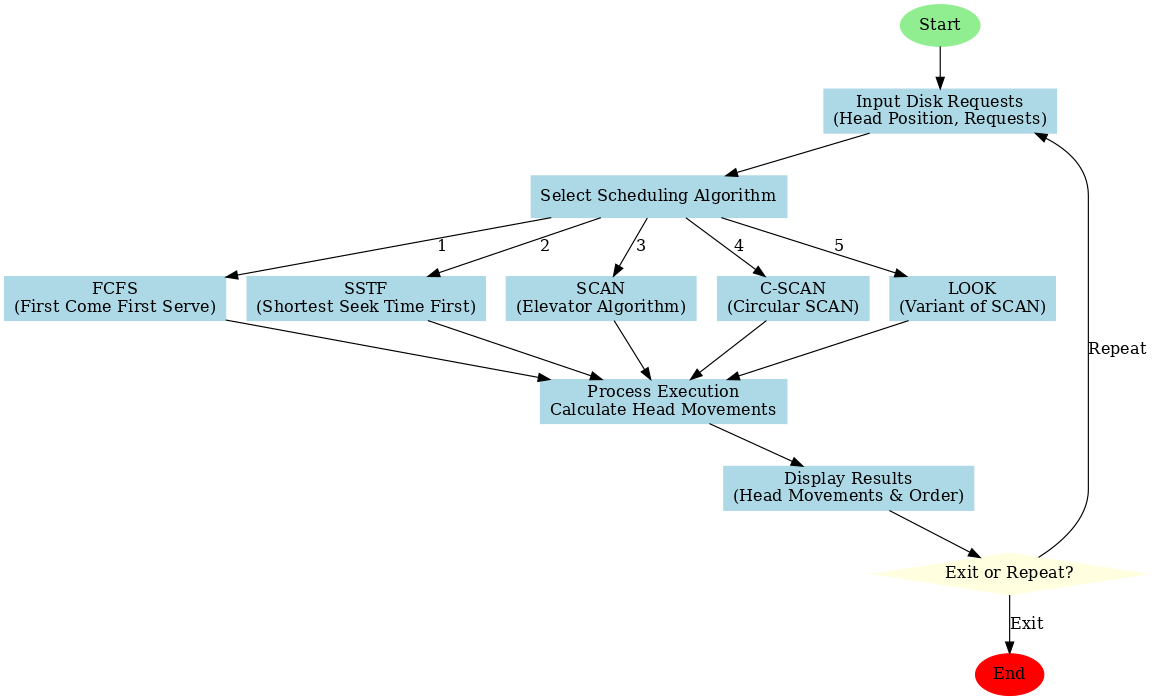
**Other Tools:** •GitHub for

Version Control:

* + - Maintains version history and commits for tracking project development.
  + Gantt Chart Visualization:
    - Displayed using console-based textual representation.

**Flowchart Overview:**

* 1. Start
  2. Input Process Details
  3. Select Scheduling Algorithm oFCFS oSJF oRound Robin oPriority Scheduling
  4. Process Execution Based on Selection
  5. Calculate Completion, TAT, and WT
  6. Display Results and Gantt Chart
  7. Repeat or Exit



1. **Revision Tracking on GitHub**

Repository Name: [**Operating-System**](https://github.com/Anmol339607/Operating-System)

Repository link: [https://github.com/Anmol339607/OperatingSystem](https://github.com/Anmol339607/Operating-System)

Github profile: https://github.com/Anmol339607

**Conclusion:**

The Advanced Disk Scheduling Simulator successfully simulates and compares multiple disk scheduling algorithms. It highlights the efficiency, advantages, and limitations of each algorithm, providing valuable insights into their behavior and practical implications.

**Result Analysis:**

**Performance Comparison**

* **FCFS: Higher average seek time due to linear processing.**
* **SSTF: Minimizes head movement by choosing the nearest request.**
* **SCAN: Provides better overall performance for larger request sequences.**
* **C-SCAN: Improves fairness by circular scanning.**
* **LOOK: Similar to SCAN but avoids unnecessary movement.**

**Key Observations**

* **SSTF provides the lowest seek time in most cases but may lead to starvation.**
* **SCAN and C-SCAN reduce head movements effectively in dense request patterns.**

Future Scope:

* **Real-Time Disk Simulation:** Implementing a real-time simulator to analyze real-world disk performance.
* **Hybrid Algorithms:** Introducing hybrid approaches to combine the strengths of different algorithms.
* **Graphical Visualization:** Adding graphical representations to improve result analysis.

**8. References**

* Silberschatz, A., Galvin, P. B., & Gagne, G. (2020). Operating System Concepts.
* Tanenbaum, A. S., & Bos, H. (2015). Modern Operating Systems.

# Appendix

6. **A. AI-Generated Project Elaboration/Breakdown Report**

**Project Breakdown Report**

**Objective:**

**The project breakdown includes detailed explanations of each module, including input setup, algorithm execution, and performance comparison. The generated results validate the correctness and efficiency of the implemented disk scheduling algorithms.**

**Module-Wise Breakdown:**

**Module 1: Input and Setup**

* **Takes input for:**
  + **Initial Head Position**
  + **Sequence of Disk Requests**
  + **Total Number of Requests**

**Module 2: Scheduling Algorithms**

** FCFS (First Come First Serve):**

* **Serves requests in order of arrival.**

** SSTF (Shortest Seek Time First):**

* **Serves the request closest to the current head position.**

** SCAN (Elevator Algorithm):**

* **Moves head towards the end and then reverses direction.**

** C-SCAN (Circular SCAN):**

* **Moves head in one direction and jumps to the start.**

** LOOK:**

* **Similar to SCAN but only moves to the last request before reversing.**

**Module 3: Execution and Result Generation**

* **Calculates total head movements for each algorithm.**
* **Displays order of request servicing.**

**Module 4: Gantt Chart and Visualization**

** Shows simulation results with metrics.**

** Displays comparison of algorithms for performance analysis.**

**Problem Statement:**

* Advanced Disk Scheduling Simulator
* **Solution/code**

**#include<iostream>**

**#include<algorithm>**

**#include<vector>**

**#include<cmath>**

**using namespace std;**

**vector<int> requests;**

**int head, numRequests;**

**void inputRequests() {**

**cout << "Enter the number of disk requests: ";**

**cin >> numRequests;**

**requests.resize(numRequests);**

**cout << "Enter the disk request sequence: ";**

**for (int i = 0; i < numRequests; i++)**

**cin >> requests[i];**

**cout << "Enter the initial head position: ";**

**cin >> head;**

**}**

**void fcfs() {**

**int totalMovement = 0, currentPos = head;**

**cout << "\nFCFS Order of Request Processing: ";**

**for (int i = 0; i < numRequests; i++) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(requests[i] - currentPos);**

**currentPos = requests[i];**

**}**

**cout << requests[numRequests - 1] << "\nTotal Head Movement: " << totalMovement << " cylinders\n";**

**}**

**void sstf() {**

**vector<int> temp = requests;**

**int totalMovement = 0, currentPos = head;**

**cout << "\nSSTF Order of Request Processing: ";**

**while (!temp.empty()) {**

**auto it = min\_element(temp.begin(), temp.end(), [&](int a, int b) {**

**return abs(a - currentPos) < abs(b - currentPos);**

**});**

**cout << currentPos << " -> ";**

**totalMovement += abs(\*it - currentPos);**

**currentPos = \*it;**

**temp.erase(it);**

**}**

**cout << currentPos << "\nTotal Head Movement: " << totalMovement << " cylinders\n";**

**}**

**void scan(bool direction) {**

**vector<int> temp = requests;**

**temp.push\_back(head);**

**temp.push\_back(0);**

**sort(temp.begin(), temp.end());**

**int pos = find(temp.begin(), temp.end(), head) - temp.begin();**

**int totalMovement = 0, currentPos = head;**

**cout << "\nSCAN Order of Request Processing: ";**

**if (direction) {**

**for (int i = pos; i < temp.size(); i++) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**for (int i = pos - 1; i >= 0; i--) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**} else {**

**for (int i = pos; i >= 0; i--) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**for (int i = pos + 1; i < temp.size(); i++) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**}**

**cout << currentPos << "\nTotal Head Movement: " << totalMovement << " cylinders\n";**

**}**

**void cscan() {**

**vector<int> temp = requests;**

**temp.push\_back(head);**

**temp.push\_back(0);**

**temp.push\_back(199); // Assuming disk size of 200 cylinders**

**sort(temp.begin(), temp.end());**

**int pos = find(temp.begin(), temp.end(), head) - temp.begin();**

**int totalMovement = 0, currentPos = head;**

**cout << "\nC-SCAN Order of Request Processing: ";**

**for (int i = pos; i < temp.size(); i++) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**for (int i = 0; i < pos; i++) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**cout << currentPos << "\nTotal Head Movement: " << totalMovement << " cylinders\n";**

**}**

**void look(bool direction) {**

**vector<int> temp = requests;**

**temp.push\_back(head);**

**sort(temp.begin(), temp.end());**

**int pos = find(temp.begin(), temp.end(), head) - temp.begin();**

**int totalMovement = 0, currentPos = head;**

**cout << "\nLOOK Order of Request Processing: ";**

**if (direction) {**

**for (int i = pos; i < temp.size(); i++) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**for (int i = pos - 1; i >= 0; i--) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**} else {**

**for (int i = pos; i >= 0; i--) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**for (int i = pos + 1; i < temp.size(); i++) {**

**cout << currentPos << " -> ";**

**totalMovement += abs(temp[i] - currentPos);**

**currentPos = temp[i];**

**}**

**}**

**cout << currentPos << "\nTotal Head Movement: " << totalMovement << " cylinders\n";**

**}**

**int main() {**

**inputRequests();**

**int choice;**

**do {**

**cout << "\nSelect Disk Scheduling Algorithm:\n";**

**cout << "1. FCFS\n2. SSTF\n3. SCAN\n4. C-SCAN\n5. LOOK\n6. Exit\n";**

**cin >> choice;**

**switch (choice) {**

**case 1:**

**fcfs();**

**break;**

**case 2:**

**sstf();**

**break;**

**case 3:**

**scan(true);**

**break;**

**case 4:**

**cscan();**

**break;**

**case 5:**

**look(true);**

**break;**

**case 6:**

**cout << "Exiting...\n";**

**break;**

**default:**

**cout << "Invalid choice! Try again.\n";**

**}**

**} while (choice != 6);**

**return 0;**

**}**