Electron: Jurnal Ilmiah Teknik Elektro
p-ISSN 2830–523X | e-ISSN 2622–6588
Volume 6 Nomor 1 Bulan 2025

Tinjauan Komprehensif: Simulasi Sistem *Disk Scheduling* dengan Berbagai Algoritma Menggunakan OS Sim

Comprehensive Review: Disk Scheduling System Simulation with Various Algorithms Using OS Sim

Fidel Lusiana Putri[1], Djuniadi[2], Febry Putra Rochim[3]

[1,2,3] Teknik Elektro Fakultas Teknik Universitas Negeri Semarang;
email: [1]fidellusiana@students.unnes.ac.id, [2]djuniadi@mail.unnes.ac.id, [3]febry.putra@mail.unnes.ac.id

[Dikirimkan: hh Bulan 20xx, Direvisi: hh Bulan 20xx, Diterima: hh Bulan 20xx]
Corresponding Author: Fidel Lusiana Putri

INTISARI — Penjadwalan disk merupakan salah satu komponen penting dalam manajemen sumber daya sistem operasi untuk mengoptimalkan akses data pada perangkat penyimpanan. Sistem operasi menggunakan berbagai algoritma untuk mengatur permintaan input/output (I/O) agar lebih efisien, termasuk Shortest Seek Time First (SSTF), LOOK, dan Circular LOOK (C-LOOK). Penelitian ini bertujuan untuk mengevaluasi dan membandingkan kinerja algoritma-algoritma tersebut dalam mengelola permintaan disk melalui simulasi menggunakan OS-SIM. Metodologi yang diterapkan mencakup simulasi antrean permintaan I/O acak pada disk dengan konfigurasi spesifik. Data yang dianalisis meliputi Total Head Movement (THM) dan Average Seek Time (AST) untuk setiap algoritma. Simulasi dilakukan dengan posisi awal kepala disk yang telah ditentukan, dan setiap algoritma diuji berdasarkan cara masing-masing mengatur urutan akses disk. Hasil penelitian menunjukkan bahwa algoritma SSTF adalah yang paling efisien dengan nilai THM dan AST terendah, yaitu sebesar 208 track dan AST 29,71 ms, dibandingkan dengan algoritma LOOK dan C-LOOK. Algoritma LOOK memberikan keseimbangan antara efisiensi dan keadilan, karena mampu meminimalkan pergerakan kepala tanpa mengabaikan permintaan yang jauh. Sementara itu, C-LOOK lebih efektif dalam mengurangi kemungkinan starvation pada permintaan di ujung disk, namun dengan konsekuensi meningkatnya jumlah pergerakan kepala. Penelitian ini memberikan gambaran yang lebih jelas mengenai kelebihan dan kekurangan masing-masing algoritma, yang dapat dijadikan acuan dalam pemilihan algoritma penjadwalan disk pada sistem yang membutuhkan efisiensi dan keandalan tinggi dalam pengelolaan I/O.

KATA KUNCI — Penjadwalan Disk, OS Sim, SSTF, LOOK, C-LOOK

ABSTRACT — *Disk scheduling is one of the important components in operating system resource management to optimize data access on storage devices. Operating systems use various algorithms to manage input/output (I/O) requests more efficiently, including Shortest Seek Time First (SSTF), LOOK, and Circular LOOK (C-LOOK). This research aims to evaluate and compare the performance of these algorithms in managing disk requests through simulations using OS-SIM. The applied methodology includes simulating a queue of random I/O requests on disks with specific configurations. The data analyzed includes Total Head Movement (THM) and Average Seek Time (AST) for each algorithm. The simulation is performed with a predefined initial position of the disk head, and each algorithm is tested based on how each of them manages the disk access sequence. The results show that the SSTF algorithm is the most efficient with the lowest THM and AST values of 208 tracks and 29.71 ms AST, compared to the LOOK and C-LOOK algorithms. The LOOK algorithm provides a balance between efficiency and fairness, as it is able to minimize head movement without ignoring distant requests. Meanwhile, C-LOOK is more effective in reducing the possibility of starvation on requests at the end of the disk, but at the cost of increasing the number of head movements. This research provides a clearer picture of the advantages and disadvantages of each algorithm, which can be used as a reference in the selection of disk scheduling algorithms in systems that require high efficiency and reliability in I/O management.*

KEYWORDS — *Disk Scheduling, OS Sim, SSTF, LOOK, C-LOOK*

I. PENDAHULUAN

Sistem operasi adalah perangkat lunak terintegrasi yang sangat canggih, dirancang untuk menyediakan layanan manajemen sistem seperti manajemen proses, perangkat dan *file*, dengan mengelola semua sumber daya perangkat keras dan perangkat lunak [1], [2], [3]. Dalam upayanya mengelola berbagai tugas dan memastikan pemanfaatan sumber daya secara efisien, sistem operasi menerapkan berbagai algoritma penjadwalan, seperti penjadwalan CPU, penjadwalan *disk*, dan penggantian halaman [1], [2], [3], [4]. Algoritma-algoritma ini dirancang untuk mengatur dan mengoptimalkan kinerja sistem operasi dalam menangani berbagai jenis tugas.

Salah satu komponen penting dalam manajemen sumber daya komputer adalah sistem penjadwalan *disk*. Penjadwalan *disk* dilakukan oleh sistem operasi untuk menangani permintaan *input* dan *output* (I/O) yang tertunda pada *disk* [5], [6], [7]. Proses ini

melibatkan pengaturan semua permintaan memori berdasarkan indeks memori, yang merupakan kombinasi nomor trek, sektor, dan *platter*, agar dapat diselesaikan dengan lebih cepat [1], [8], [9]. Kinerja algoritma penjadwalan disk diukur berdasarkan waktu respons *disk*, yang mencakup berbagai komponen seperti waktu pencarian *(seek time),* latensi rotasi, waktu pemilihan *platter*, dan waktu transfer data [1], [7], [8], [9], [10].

Manajemen kinerja *disk* penting dalam sistem operasi karena perbedaan signifikan antara kecepatan prosesor dan memori utama dengan kecepatan *disk*. Kecepatan *disk* yang lebih lambat menjadi hambatan utama, terutama pada sistem dengan kepala baca/tulis yang bergerak, di mana waktu pencarian antar silinder memakan waktu signifikan, sehingga efisiensi penjadwalan *disk* sangat diperlukan [8], [10]. Penjadwalan *disk* yang baik dapat mengurangi waktu tunggu, meningkatkan *throughput*, dan mengoptimalkan penggunaan sumber daya, sehingga memberikan pengalaman yang lebih responsif bagi pengguna [8], [9], [11]. *Disk scheduling* bertujuan untuk mengatur urutan akses permintaan data yang diterima oleh *disk*. Dalam prakteknya, sistem operasi harus memprioritaskan dan menentukan urutan permintaan *disk* untuk meningkatkan efisiensi [6], [7]. Berbagai algoritma telah dikembangkan untuk menyelesaikan masalah ini, dengan masing-masing memiliki keunggulan dan kekurangan dalam situasi yang berbeda.

Algoritma penjadwalan *disk* yang paling umum digunakan antara lain *First-In-First-Out* (FIFO), LIFO, *Shortest Seek Time First* (SSTF), SCAN, C-SCAN, LOOK, dan C-LOOK [1], [5], [7], [9], [10], [11], [12], [13], [14], [15], [16]. Setiap algoritma memiliki cara yang berbeda dalam memilih permintaan *disk* mana yang akan diproses terlebih dahulu. Algoritma LIFO *(Last-In-First-Out)* selalu memprioritaskan permintaan terakhir untuk diproses terlebih dahulu. Sementara itu, FIFO memproses permintaan berdasarkan urutan kedatangan tanpa memperhatikan jarak fisik kepala baca/tulis *disk*, yang bisa menyebabkan keterlambatan yang signifikan [1], [5], [9], [10], [11], [12], [13], [14]. SSTF memilih permintaan yang memiliki jarak terdekat dari posisi kepala, namun dapat menyebabkan *starvation* bagi permintaan yang lebih jauh [1], [5], [9], [10], [11], [12], [13]. SCAN dan C-SCAN lebih efisien dengan bergerak secara linier ke arah tertentu, kemudian membalik arah jika sudah mencapai ujung [1], [5], [9], [10], [11], [12], [13]. Algoritma LOOK adalah varian SCAN yang memeriksa apakah ada permintaan di arah pergerakan kepala *disk*, jika tidak ada, kepala *disk* dibalik ke arah sebaliknya. Sedangkan C-LOOK, mirip dengan C-SCAN, hanya bergerak sejauh yang diperlukan untuk melayani permintaan terakhir sebelum memulai putaran baru [1], [5], [9], [10], [11], [12], [13].

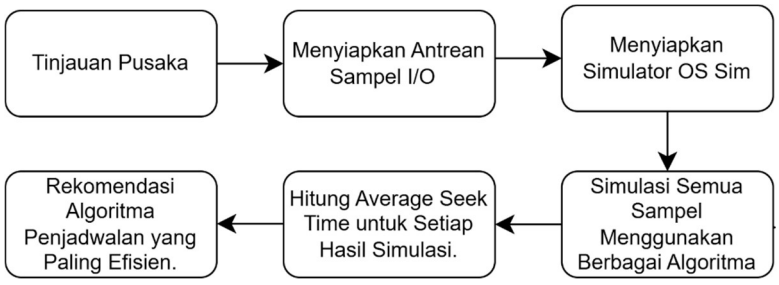
Simulasi sistem penjadwalan *disk* menggunakan berbagai algoritma penjadwalan ini dapat memberikan wawasan tentang kinerja masing-masing algoritma dalam kondisi yang berbeda [13], [17]. Melalui simulasi, kita dapat membandingkan efektivitas dan efisiensi masing-masing algoritma dalam mengelola permintaan akses *disk* pada sistem operasi. Penggunaan alat simulasi seperti OS Sim memungkinkan pengujian yang lebih mendalam dan pemahaman yang lebih baik tentang bagaimana berbagai algoritma bekerja dalam berbagai skenario.

Dalam simulasi ini, *disk* terdiri dari satu *platter*, 16 silinder, dan 12 sektor per silinder, yang totalnya mencapai 192 sektor. Setiap sektor diindeks mulai dari 0 di silinder terluar dan kepala *disk* bergerak untuk membaca atau menulis data pada sektor-sektor tersebut. Pengguna dapat menentukan posisi awal kepala *disk* dalam rentang 0 hingga 192 sektor.

Melalui penelitian ini, akan dilakukan simulasi untuk membandingkan kinerja algoritma penjadwalan *disk* menggunakan OS Sim, Algoritma yang akan diuji mencakup SSTF, LOOK, dan C-LOOK. Kinerja algoritma akan diuji berdasarkan berbagai metrik seperti waktu tunggu, waktu perputaran, dan kecepatan transfer data. Diharapkan, hasil simulasi ini dapat memberikan gambaran yang jelas mengenai algoritma mana yang paling sesuai digunakan untuk berbagai jenis aplikasi dan kondisi operasional pada sistem operasi.

II. METODE PENELITIAN

Metode penelitian ini adalah eksperimen simulasi berbagai algoritma penjadwalan *disk*. Ada berbagai simulator sistem komputer yang tersedia dalam bidang sistem operasi, salah satunya adalah OS-SIM. OS-SIM dapat mensimulasikan berbagai perilaku dalam desain sistem operasi ya0ng terdiri dari empat domain yaitu *Process Scheduling, Memory Management, File System*, dan *Disk Management* [18], [19], [20]. Sistem secara otomatis menghitung dan menunjukkan *Total Head Movement (THM) dan Average Seek Time (AST)* untuk setiap permintaan yang dilayani. OS-SIM dipilih sebagai simulator untuk menguji berbagai algoritma penjadwalan *disk* yang relevan. Selanjutnya, akan disiapkan antrean permintaan I/O yang disusun secara acak sebagai sampel. Semua sampel akan diuji menggunakan simulasi, dengan mencatat Total Pergerakan Kepala dan Waktu Pencarian Rata-rata, yang selanjutnya akan dianalisis lebih lanjut. Berdasarkan hasil pengujian berbagai algoritma, dilakukan perbandingan dan analisis mendalam terhadap data yang diperoleh, yang akhirnya akan menghasilkan rekomendasi berdasarkan temuan tersebut. Tahapan penelitian dapat dilihat pada gambar 1.



Gambar 1. Analisis Metode

A. TINJAUAN PUSTAKA

Tahap awal penelitian dilakukan dengan mengkaji teori dan konsep dasar terkait algoritma penjadwalan disk. Fokus kajian meliputi prinsip kerja algoritma Shortest Seek Time First (SSTF), LOOK, dan Circular LOOK (C-LOOK), serta analisis kelebihan dan kekurangan masing-masing algoritma berdasarkan studi literatur yang relevan.

B. MENYIAPKAN ANTREAN SAMPEL I/O

Setelah melakukan kajian teori, disusun data antrean permintaan I/O yang bersifat acak, mencakup sektor dan silinder pada disk. Sampel ini disiapkan untuk dijadikan input yang konsisten dalam pengujian semua algoritma.

C. MENYIAPKAN SIMULATOR OS SIM

Simulator OS-SIM dipilih untuk melakukan simulasi penjadwalan disk. OS-SIM memungkinkan input data antrean I/O, pengaturan posisi awal kepala disk, dan pemilihan algoritma penjadwalan. Sebelum simulasi, dilakukan konfigurasi awal sesuai kebutuhan penelitian.

D. SIMULASI SEMUA SAMPEL MENGGUNAKAN BERBAGAI ALGORITMA

Antrean I/O yang telah disiapkan kemudian disimulasikan menggunakan tiga algoritma berbeda untuk membadingkan, yaitu SSTF, LOOK, dan C-LOOK. Proses ini bertujuan mengukur kinerja masing-masing algoritma berdasarkan antrean yang sama. Sebelum simulasi dilakukan, berikut adalah penjelasan cara kerja masing-masing algoritma yang akan diuji:

1) SHORTEST SEEK TIME FIRST (SSTF)

Algoritma SSTF memilih permintaan I/O yang jaraknya paling dekat dengan posisi kepala disk saat ini. Dengan memprioritaskan permintaan terdekat, SSTF dapat mempercepat waktu rata-rata pencarian (seek time) dan mengurangi total pergerakan kepala disk (Total Head Movement). Namun, kekurangannya adalah potensi terjadinya starvation pada permintaan yang jauh dari posisi kepala disk.

2) LOOK

Algoritma LOOK menggerakkan kepala disk hanya sejauh permintaan I/O yang ada di satu arah. Ketika tidak ada lagi permintaan di arah tersebut, kepala disk akan membalik arah untuk melayani permintaan di sisi lain. LOOK lebih efisien dibandingkan algoritma SCAN karena tidak memaksa kepala disk untuk bergerak sampai batas ujung disk yang kosong. Algoritma ini menyeimbangkan antara efisiensi pergerakan dan keadilan pelayanan permintaan.

3) CIRCULAR LOOK (C-LOOK)

Algoritma C-LOOK mirip dengan LOOK, tetapi kepala disk hanya bergerak dalam satu arah hingga mencapai permintaan terjauh, lalu kembali ke permintaan terdekat berikutnya tanpa melayani permintaan di jalur balik. C-LOOK bertujuan untuk mengurangi kondisi starvation dengan mengutamakan pelayanan yang lebih merata, meskipun total pergerakan kepala disk bisa lebih tinggi dibandingkan SSTF.

E. MENGHITUNG AST DAN THM

Setelah simulasi, dilakukan penghitungan Total Head Movement (THM) dan Average Seek Time (AST) untuk masing-masing algoritma. Perhitungan ini bertujuan untuk mengevaluasi efisiensi pergerakan kepala disk dalam melayani permintaan.

F. MEMBERIKAN REKOMENDASI ALGORITMA PENJADWALAN DISK

Berdasarkan hasil analisis THM dan AST, ditentukan algoritma yang paling efisien. Rekomendasi ini diberikan berdasarkan kinerja terbaik dalam mengoptimalkan pergerakan kepala disk dan mempercepat waktu pelayanan permintaan I/O.

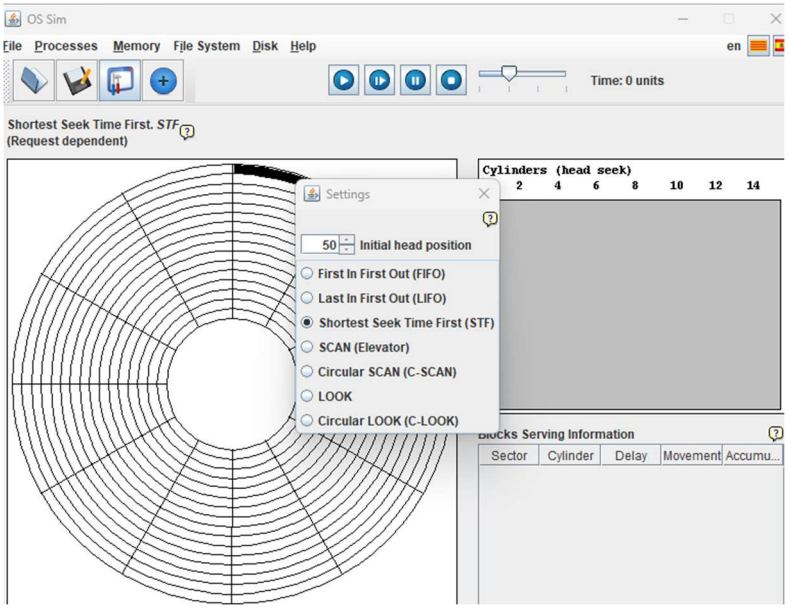
III. HASIL DAN PEMBAHASAN

Dalam sistem operasi, penjadwalan *disk* memainkan peran penting dalam mengelola akses ke *disk* dan menentukan urutan eksekusi permintaan I/O. Salah satu tantangan utama dalam manajemen *disk* adalah meminimalkan pergerakan kepala *disk*, yang mempengaruhi kinerja sistem secara keseluruhan. Oleh karena itu, penelitian ini berfokus pada simulasi sistem *disk scheduling* menggunakan berbagai algoritma penjadwalan yang diimplementasikan dalam OS-SIM. Berikut adalah data sampel antrean permintaan I/O yang akan digunakan untuk simulasi, perhatikan tabel I.

TABEL I
SAMPLE ANTREAN I/O

Sector	Cylinder	Delay
82	6	0
170	14	0
43	3	0
140	11	0
24	2	0
16	1	0
190	15	0

Berdasarkan tabel I, data dimasukkan ke dalam simulasi untuk menguji berbagai algoritma penjadwalan *disk.* Setiap algoritma, seperti *Shortest Seek Time First* (SSTF), LOOK dan C-LOOK akan memproses antrean ini dengan cara yang berbeda, mengutamakan permintaan-permintaan yang dipilih sesuai dengan kriteria algoritma. Simulasi ini untuk mengukur *Total Head Movement (THM) dan Average Seek Time (AST)* yang dihasilkan oleh setiap algoritma berdasarkan antrean permintaan ini.

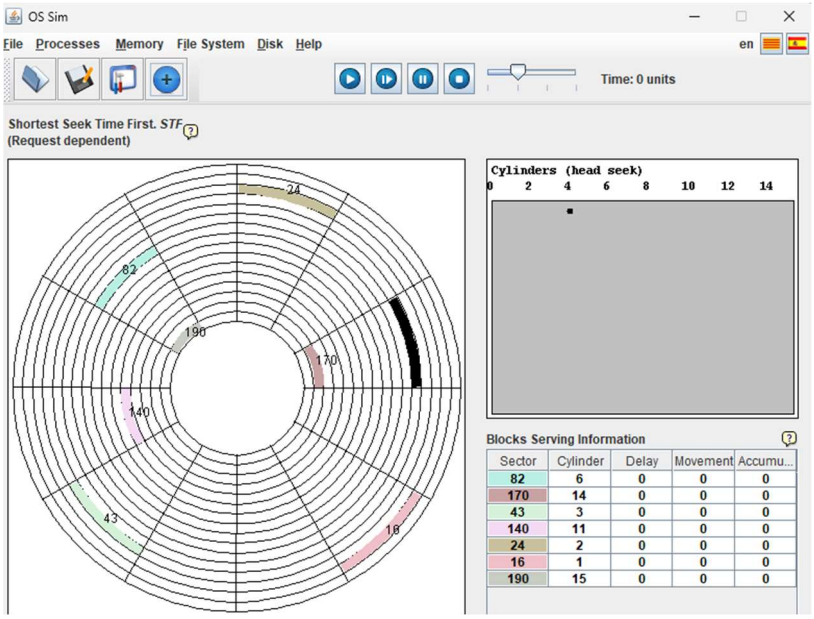


Gambar 2. Menentukan *Initial Head Position*

Dalam setiap simulasi yang dilakukan menggunakan OS-SIM, posisi awal kepala disk atau *initial head position* diatur pada sektor ke-50. Posisi awal ini berarti kepala *disk* mulai bergerak dari sektor 50 pada silinder yang sesuai, dalam hal ini sektor 50 terletak pada silinder yang sesuai dengan posisi sektor tersebut di *disk*, perhatikan gambar 2.

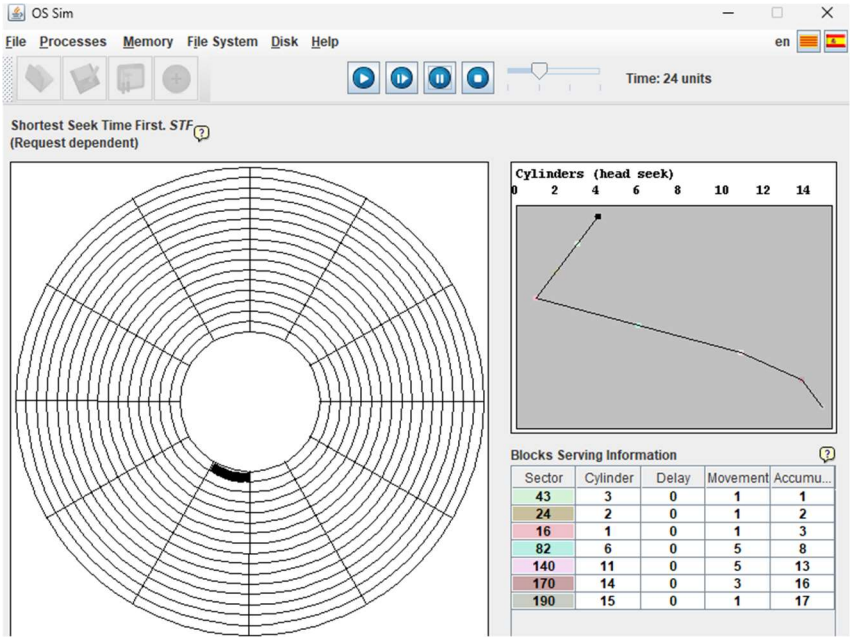
A. SHORTEST SEEK TIME FIRST (SSTF)

Algoritma SSTF mulai dengan mengurutkan seluruh permintaan I/O berdasarkan urutan yang meningkat, kemudian menghitung waktu pencarian kepala *disk* untuk setiap permintaan, dan memilih permintaan dengan waktu pencarian paling singkat. Dengan demikian, SSTF dapat mempercepat waktu respons rata-rata, yang pada gilirannya meningkatkan tingkat *throughput* [8]. Namun, kelemahan utama dari SSTF adalah kecenderungannya untuk selalu memilih permintaan dengan waktu pencarian terpendek, yang bisa menyebabkan permintaan dengan waktu pencarian lebih lama tidak terlayani tepat waktu, atau yang dikenal dengan istilah *starvation* [10]. Dalam simulasi ini kepala *disk* memulai pergerakan dari sektor 50 dan kemudian bergerak sesuai dengan algoritma SSTF, yang memilih permintaan dengan waktu pencarian terpendek dari posisi kepala *disk* saat ini.



Gambar 3. Mengisi *Request*

Pada grafik *disk* yang ditampilkan, setiap sektor yang terlibat dalam simulasi ditandai dengan warna yang berbeda. Posisi sektor-sektor ini terlihat pada lingkaran-lingkaran yang mewakili silinder di *disk*, perhatikan gambar 3.



Gambar 4. Hasil Simulasi SSTF

Hasil simulasi *Shortest Seek Time First* (SSTF) yang ditampilkan menunjukkan bagaimana algoritma ini melayani permintaan I/O dengan memilih permintaan yang memiliki waktu pencarian terpendek dari posisi kepala *disk* saat ini, perhatikan gambar 4.

TABEL II
HASIL SIMULASI SSTF

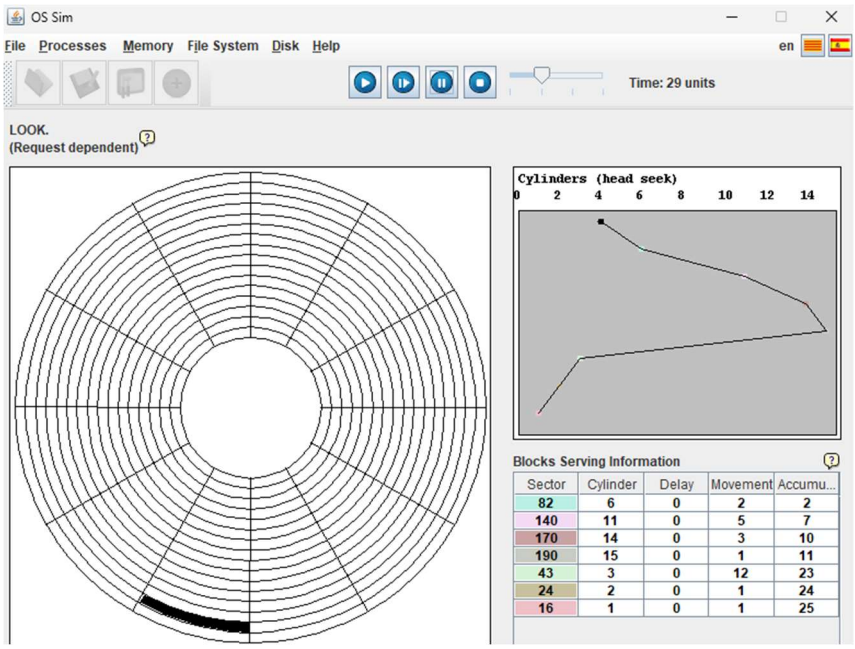
Shortest Seek Time First		
Served Sector	**Movement**	**Accumulated Movement**
43	1	1
24	1	2
16	1	3
82	5	8
140	5	13
170	3	16
190	1	17

Berdasarkan pergerakan sektor yang ditunjukkan pada tabel II, dapat dihitung THM dengan menjumlahkan jarak yang ditempuh untuk setiap permintaan:

THM = (43 - 50) + (24 - 43) + (16 - 24) + (82 - 16) + (140 - 82) + (170 - 140) + (190 - 170)
THM = 7 + 19 + 8 + 66 + 58 + 30 + 20 = 208 *track*
Dalam hal ini, ada 7 permintaan yang dilayani. Jadi, AST dapat dihitung sebagai berikut:
AST = 208 / 7 ≈ 29.71 ms

B. LOOK

LOOK merupakan pengembangan dari algoritma SCAN yang menawarkan waktu pencarian lebih efisien, di mana kepala *disk* hanya bergerak sejauh permintaan I/O yang ada tanpa melampaui batas tersebut. Algoritma ini mampu mengurangi kondisi *starvation*, mengoptimalkan waktu tunggu, dan memberikan respons yang lebih konsisten [8], [10].



Gambar 5. Hasil Simulasi LOOK

Hasil simulasi menggunakan algoritma LOOK yang ditampilkan pada gambar di atas menunjukkan bagaimana algoritma ini mengelola pergerakan kepala *disk* dengan cara melayani permintaan I/O berdasarkan arah pergerakan kepala *disk* tanpa melampaui permintaan yang ada dalam antrean, perhatikan gambar 5.

TABEL III
HASIL SIMULASI LOOK

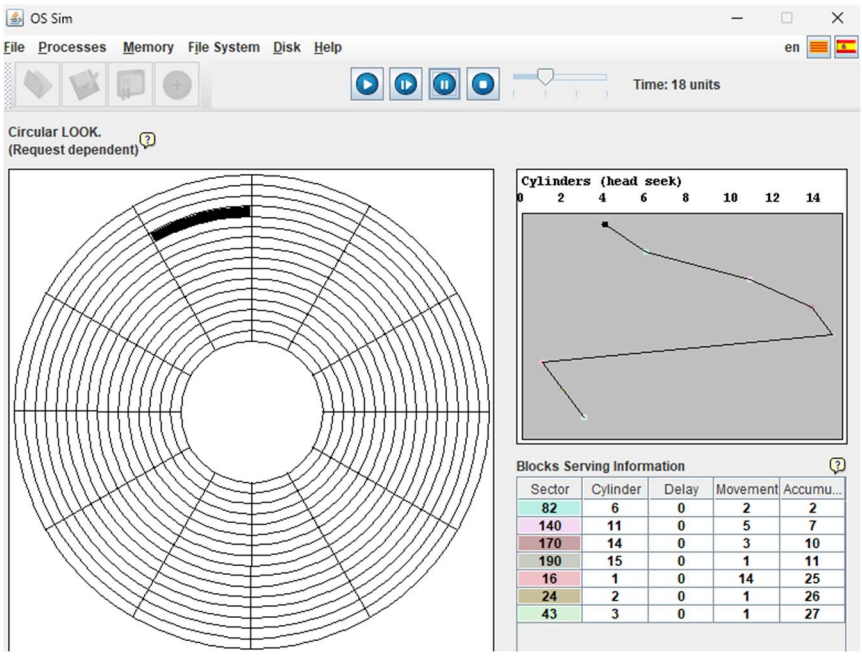
LOOK		
Served sector	**Movement**	**Accumulated Movement**
82	2	2
140	5	7
170	3	10
190	1	11
43	12	23
24	1	24
16	1	25

Berdasarkan pergerakan sektor yang ditunjukkan pada tabel III, dapat dihitung THM dengan menjumlahkan jarak yang ditempuh untuk setiap permintaan:

THM = (82 - 50) + (140 - 82) + (170 - 140) + (190 - 170) + (1900 - 43) + (43 - 24) + (24 - 16)
THM = 32 + 58 + 30 + 20 + 147 + 19 + 8 = 314 *track*
Dalam hal ini, ada 7 permintaan yang dilayani. Jadi, AST dapat dihitung sebagai berikut:
AST = 314 / 7 ≈ 44.85 ms

C. *CIRCULAR LOOK*

C-LOOK adalah penyempurnaan dari algoritma C-SCAN yang menawarkan waktu pencarian sedikit lebih efisien. Meskipun cara kerjanya mirip dengan C-SCAN, di mana kepala *disk* tidak melayani permintaan saat bergerak kembali dari ujung, C-LOOK berhenti hanya setelah mencapai permintaan terakhir dalam antrean [8].



Gambar 6. Hasil Simulasi C-LOOK

Hasil simulasi menggunakan algoritma *Circular* LOOK (C-LOOK) yang ditampilkan pada gambar di atas menggambarkan bagaimana algoritma ini mengelola pergerakan kepala *disk* dengan cara yang lebih efisien dibandingkan algoritma SCAN dan LOOK. Berikut penjelasan mengenai hasil simulasi, termasuk *Total Head Movement (THM)* dan *Average Seek Time (AST),* perhatikan gambar 6.

TABEL IV
HASIL SIMULASI C-LOOK

C-LOOK		
Served sector	**Movement**	**Accumulated Movement**
82	2	2
140	5	7
170	3	10
190	1	11
16	14	25
24	1	26
43	1	27

Berdasarkan pergerakan sektor yang ditunjukkan pada tabel IV, dapat dihitung THM dengan menjumlahkan jarak yang ditempuh untuk setiap permintaan:

THM = (82 - 50) + (140 - 82) + (170 - 140) + (190 - 170) + (190 - 16) + (24 - 16) + (43 - 24)
THM = 32 + 58 + 30 + 20 + 174 + 8 + 19 = 341 *track*
Dalam hal ini, ada 7 permintaan yang dilayani. Jadi, AST dapat dihitung sebagai berikut:
AST = 341 / 7 ≈ 48.71 ms

Setelah dilakukan simulasi diatas, berikut akan dilakukan perbandingkan efisiensi kinerja algoritma diatas berdasarkan dua parameter utama, yaitu *Total Head Movement* (THM) dan *Average Seek Time* (AST). Algoritma-algoritma ini digunakan untuk mengelola permintaan I/O pada disk dengan cara yang berbeda, yang mempengaruhi jumlah waktu yang dibutuhkan kepala *disk* untuk melayani permintaan.

TABEL V
PERBANDINGAN ALGORITMA

Comparison of Disk Scheduling Algorithms		
Algorithm	**THM**	**AST**
SSTF	208	29.71
LOOK	314	44.85
C-LOOK	341	48.71

Berdasarkan tabel V, perbandingan antara algoritma SSTF, LOOK, dan C-LOOK berdasarkan efisiensi, SSTF muncul sebagai yang paling efisien. Algoritma ini memiliki THM terendah, yaitu 208 *track*, dan AST yang paling cepat, yaitu 29.71 unit waktu. SSTF mencapai efisiensi maksimal dengan selalu memilih permintaan terdekat dari posisi kepala *disk*, yang mengurangi pergerakan kepala *disk* secara signifikan. Namun, kelemahannya adalah kemungkinan terjadinya kondisi *starvation* bagi permintaan yang lebih jauh, karena permintaan yang lebih dekat cenderung dilayani lebih dulu.

Di sisi lain, LOOK memiliki THM yang lebih tinggi, yaitu 314 *track*, dan AST sebesar 44.85 unit waktu, menjadikannya sedikit kurang efisien dibandingkan SSTF. Meskipun demikian, LOOK masih lebih efisien dibandingkan dengan SCAN, karena tidak melayani permintaan saat kepala *disk* kembali ke ujung *disk*. Algoritma ini memberikan keseimbangan yang lebih baik antara efisiensi dan keadilan dalam pelayanan permintaan, meskipun waktu pencarian dan pergerakan kepala *disk* masih lebih besar dibandingkan SSTF.

C-LOOK, meskipun lebih efisien dibandingkan SCAN, memiliki THM tertinggi di antara ketiganya, yaitu 341 *track*, dengan AST sebesar 48.71 unit waktu. Meskipun C-LOOK menghindari pergerakan kepala *disk* yang tidak perlu saat kembali ke awal, algoritma ini tetap memiliki THM dan AST yang lebih tinggi dibandingkan dengan LOOK dan SSTF. Meskipun memberikan pelayanan yang lebih adil dan mengurangi *starvation*, pergerakan kepala *disk* dan waktu pencarian yang lebih besar membuatnya kurang efisien dibandingkan kedua algoritma lainnya.

Secara keseluruhan, SSTF adalah algoritma yang paling efisien dalam hal waktu pencarian dan pergerakan kepala *disk*, meskipun dapat menghadapi masalah *starvation*. LOOK lebih adil dan sedikit lebih efisien dibandingkan dengan C-LOOK, namun masih memiliki waktu pencarian yang lebih tinggi. C-LOOK, meskipun menghindari beberapa kelemahan SCAN, memiliki THM dan AST yang lebih tinggi, menjadikannya kurang efisien dalam konteks ini.

IV. KESIMPULAN

Penelitian ini menunjukkan bahwa algoritma penjadwalan disk memainkan peran penting dalam mengoptimalkan kinerja akses data pada sistem operasi. Penelitian ini membandingkan kinerja tiga algoritma penjadwalan *disk*, yaitu *Shortest Seek Time First* (SSTF), LOOK, dan *Circular* LOOK (C-LOOK), dengan menggunakan simulator OS-SIM. Berdasarkan hasil simulasi, algoritma SSTF

menunjukkan efisiensi tertinggi dengan *Total Head Movement* (THM) terendah sebesar 208 *track* dan *Average Seek Time* (AST) sebesar 29,71 ms. Meskipun demikian, kelemahan SSTF adalah potensi terjadinya *starvation* pada permintaan yang lebih jauh. Algoritma LOOK memberikan keseimbangan antara efisiensi dan keadilan dengan THM sebesar 314 track dan AST sebesar 44,85 ms, namun masih kurang efisien dibandingkan SSTF. Algoritma C-LOOK, meskipun mampu mengurangi kondisi *starvation*, memiliki THM tertinggi sebesar 341 track dan AST sebesar 48,71 ms, menjadikannya algoritma dengan efisiensi terendah di antara ketiga algoritma yang diuji.Secara keseluruhan, penelitian ini memberikan panduan tentang algoritma penjadwalan disk yang optimal untuk berbagai kondisi operasional, dengan SSTF sebagai pilihan utama untuk efisiensi tinggi, LOOK untuk keseimbangan, dan C-LOOK untuk mengurangi kondisi *starvation*.

KONFLIK KEPENTINGAN

Penulis menyatakan bahwa tidak terdapat konflik kepentingan.

REFERENSI

[1] A. R. Dash, B. Dash, S. K. Sahu, N. K. Sahu, and S. Panda, "Optimized universal disk scheduling algorithm," *Journal of The Institution of Engineers (India): Series B*, pp. 1–71, 2024, doi: 10.1007/s40031-024-01036-9.

[2] R. A. Putri, "Aplikasi simulasi algoritma penjadwalan sistem operasi," *Jurnal Teknologi Informasi*, vol. 5, no. 1, pp. 98–102, Jul. 2021, doi: 10.36294/jurti.v5i1.2215.

[3] D. T. Putra and R. Purnomo, "Analisis algoritma round robin pada penjadwalan cpu," *Jurnal Ilmiah Teknologi Informasi Asia*, vol. 15, no. 2, pp. 85–90, 2021.

[4] M. rajasekhar Reddy, V. V. D. S. S. Ganesh, S. Lakshmi, and Y. Sireesha, "Comparative analysis of cpu scheduling algorithms and their optimal solutions," in *Proceedings of the Third International Conference on Computing Methodologies and Communication (ICCMC 2019)*, IEEE, 2019, pp. 255–260.

[5] A. Shankar, A. Ravat, and A. K. Pandey, "Comparative study of disk scheduling algorithms and proposal of a new algorithm for better efficiency," in *2ndINTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND SOFTW ARE ENGINEERING (ICACSE-2019*, 2019, pp. 21–24. [Online]. Available: https://ssrn.com/abstract=3349013

[6] E. G and S. M, "Proposing disk scheduling algorithm to enhance the efficiency of disk performance," *SPECIAL ISSUE For IJEIT ON ENGINEERING AND INFORMATION TECHNOLOGY. ,* , vol. 12, no. 1, pp. 111–119, 2024, [Online]. Available: www.ijeit.misuratau.edu.ly

[7] M. K. M R, "An efficient disk scheduling algorithm using binary search tree," in *2019 1st International Conference on Advances in Information Technology* , IEEE, 2019, pp. 184–192.

[8] B. P. Pokharel, "A Comparative analysis of disk scheduling algorithms," *International Journal for Research in Applied Sciences and Biotechnology*, vol. 8, no. 2, pp. 184–195, Apr. 2021, doi: 10.31033/ijrasb.8.2.24.

[9] R. A. Dash, S. K. Sahu, and B. Kewal, "An optimized disk scheduling algorithm with bad-sector management," *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, vol. 9, no. 3, pp. 1–21, 2019, doi: 10.5012/ijcsea.2019.9301.

[10] M. A. Khuhro, M. A. Bamboat, K. Kumar, I. A. Halepoto, N. Mirbahar, and G. S. Khan, "Analysis of Traditional Hard Disk Scheduling Algorithms: A Review," *Quaid-e-Awam University Research Journal of Engineering Science & Technology*, vol. 19, no. 1, pp. 51–58, Jun. 2021, doi: 10.52584/qrj.1901.07.

[11] S. Shastri, A. Sharma, and V. Mansotra, "A Comparative analysis of disk scheduling algorithms," *An International Multidisciplinary Research e-Journal*, vol. 2, no. 4, pp. 16–25, 2016, [Online]. Available: www.indianscholar.co.in

[12] H. Rahmani, S. Arshad, and M. E. Moghaddan, "A disk scheduling algorithm based on ANT colony optimization," pp. 37–42, 2020, doi: 10.48550/arXiv.2003.00926.

[13] S. Suranauwarat, "A disk scheduling algorithm simulator," *Computers in Education Journal*, vol. 8, no. 3, pp. 1–9, 2017.

[14] Hayatunnufus, M. Riasetiawan, and A. Ashari, "Performance analysis of fifo and round robin scheduling process algorithm in iot operating system for collecting landslide data," in *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*, IEEE, 2020, pp. 63–68.

[15] S. S. Myint, K. N. Myint, and S. K. Lwin, "Practical performance comparison of disk scheduling algorithms," *Journal of Computer Applications and Research*, vol. 1, no. 1, pp. 1–5, 2020.

[16] S. Ratna, *Sistem operasi*, 1st ed., vol. 18. Yayasan Kita Menulis, 2023.

[17] S. M. Ali, R. F. Alshahrani, A. H. Hadadi, T. A. Alghamdi, F. H. Almuhsin, and E. E. El-Sharawy, "A review on the cpu scheduling algorithms: comparative study," *IJCSNS International Journal of Computer Science and Network Security*, vol. 21, no. 1, pp. 19–26, 2021, doi: 10.22937/IJCSNS.2021.21.1.4.

[18] T. D. Putra and R. Purnomo, "Simulation and modelling of pre-emptive priority cpu scheduling algorithm," *Jurnal dan Penelitian Teknik Informatika*, vol. 8, no. 3, pp. 1631–1640, 2024, doi: 10.33395/sinkron.v8i3.13352.

[19] T. D. Putra and R. Purnomo, "Comparison between simple round robin and improved round robin algorithms," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 10, no. 1, pp. 266–272, 2023.

[20] R. Purnomo and T. D. Putra, "Simulation of preemptive shortest job first algorithm," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 11, no. 5, pp. 1–12, May 2022, doi: 10.17148/ijarcce.2022.11501.