Modul Pembelajaran:
Algoritma Penjadwalan Proses – SJF/SRTF dan Round Robin

MATA KULIAH : SISTEM OPERASI

7 MEI 2025

JURUSAN TEKNIK INFORMATIKA

STMIK TAZKIA BOGOR

2025

Tujuan Pembelajaran

Setelah mempelajari materi dalam modul ini, mahasiswa diharapkan mampu:

- Memahami konsep dasar penjadwalan CPU dan tujuannya dalam sistem operasi.[1]
- Mengidentifikasi dan menjelaskan kriteria kinerja yang digunakan untuk mengevaluasi algoritma penjadwalan CPU.[1]
- Membedakan antara penjadwalan *preemptive* dan *non-preemptive*.[1]
- Menjelaskan prinsip kerja, aturan, kelebihan, dan kekurangan algoritma penjadwalan *Shortest Job First* (SJF) *non-preemptive*.[1]
- Menjelaskan prinsip kerja, aturan, kelebihan, dan kekurangan algoritma penjadwalan *Shortest Remaining Time First* (SRTF) atau SJF *preemptive*.[2]
- Menjelaskan prinsip kerja, aturan, kelebihan, dan kekurangan algoritma penjadwalan *Round Robin* (RR).[1]
- Menganalisis peran dan dampak ukuran *kuantum waktu* pada kinerja algoritma *Round Robin*.[4]
- Membuat diagram Gantt untuk memvisualisasikan eksekusi proses menggunakan algoritma SJF, SRTF, dan RR.[6]
- Menghitung metrik kinerja utama (waktu tunggu rata-rata, waktu penyelesaian rata-rata, waktu respons rata-rata) untuk algoritma SJF, SRTF, dan RR.[6]
- Membandingkan kinerja dan karakteristik algoritma SJF/SRTF dan *Round Robin* berdasarkan berbagai kriteria.[8]
- Menganalisis skenario di mana setiap algoritma penjadwalan lebih sesuai untuk digunakan.[10]

I. Pendahuluan Penjadwalan Proses

A. Definisi dan Tujuan Penjadwalan Proses

Penjadwalan proses merupakan salah satu konsep paling fundamental dan krusial dalam desain sistem operasi (SO) modern, khususnya pada sistem yang mendukung *multiprogramming* dan *multitasking*.[1] Secara esensial, penjadwalan proses adalah kumpulan kebijaksanaan dan mekanisme yang diimplementasikan dalam sistem operasi untuk mengatur urutan kerja serta menentukan alokasi waktu penggunaan *Central Processing Unit* (CPU) di antara berbagai proses yang aktif dan siap untuk dieksekusi.[3] Komponen sistem operasi yang bertanggung jawab atas tugas ini dikenal sebagai penjadwal (*scheduler*).[3] Penjadwal memilih proses dari sekumpulan proses yang berada dalam *ready queue* (antrian siap) untuk diberikan akses ke CPU.[3]

Kebutuhan akan penjadwalan muncul sebagai konsekuensi langsung dari arsitektur *multiprogramming*. Sistem *multiprogramming* dirancang untuk meningkatkan efisiensi penggunaan sumber daya komputasi dengan memungkinkan beberapa proses berada di memori utama secara bersamaan.[1] Namun, CPU sebagai unit pemrosesan sentral hanya mampu mengeksekusi satu instruksi dari satu proses pada satu waktu.[13] Jika proses yang sedang berjalan harus menunggu suatu kejadian, misalnya penyelesaian operasi *Input/Output* (I/O), maka CPU akan berada dalam kondisi menganggur (*idle*).[3] Untuk mencegah pemborosan siklus CPU yang berharga ini, sistem operasi memerlukan mekanisme untuk segera mengalihkan kontrol CPU ke proses lain yang sudah siap untuk dieksekusi di *ready queue*.[5] Mekanisme pengalihan inilah inti dari penjadwalan proses. Dengan demikian, penjadwalan proses menjadi kunci untuk mencapai tujuan utama *multiprogramming*, yaitu memaksimalkan utilisasi CPU dan menjaga agar sistem tetap produktif.[3]

Tujuan utama dari diterapkannya algoritma penjadwalan proses dalam sistem operasi adalah untuk mengoptimalkan kinerja sistem secara keseluruhan. Optimalisasi ini diukur melalui beberapa target spesifik [5]:

1. **Memaksimalkan Utilisasi CPU (***CPU Utilization***):** Menjaga agar CPU sesibuk mungkin dalam menjalankan tugas-tugas yang berguna, meminimalkan waktu *idle*.[5]
2. **Memaksimalkan** ***Throughput*****:** Meningkatkan jumlah total proses yang berhasil diselesaikan dalam satu satuan waktu.[5]
3. **Meminimalkan Waktu Tunggu (***Waiting Time***):** Mengurangi total waktu yang dihabiskan oleh sebuah proses menunggu di *ready queue* sebelum mendapatkan alokasi CPU.[13]
4. **Meminimalkan Waktu Penyelesaian (***Turnaround Time***):** Mengurangi total waktu

yang dibutuhkan oleh sebuah proses sejak pertama kali masuk ke sistem hingga selesai dieksekusi sepenuhnya.[13]

5. **Meminimalkan Waktu Respons (*Response Time*):** Mengurangi waktu yang dibutuhkan sejak sebuah permintaan (misalnya, perintah dari pengguna interaktif) dimasukkan hingga respons pertama dihasilkan oleh sistem.[13]
6. **Menjamin Keadilan (*Fairness*):** Memastikan bahwa setiap proses mendapatkan bagian sumber daya CPU yang adil dan tidak ada proses yang mengalami penundaan tak terbatas (*starvation*).[1]

Penting untuk dipahami bahwa tujuan-tujuan penjadwalan ini seringkali bersifat kontradiktif. Sebagai contoh, upaya untuk meminimalkan waktu respons, yang sangat krusial bagi sistem interaktif, seringkali dicapai melalui penggunaan *time quantum* yang kecil pada algoritma *Round Robin* atau melalui mekanisme *preemption* pada algoritma SRTF.[25] Namun, *time quantum* yang kecil atau frekuensi *preemption* yang tinggi akan meningkatkan jumlah peralihan konteks (*context switching*) antar proses.[1] Setiap *context switch* memerlukan *overhead* waktu sistem untuk menyimpan status proses saat ini dan memuat status proses berikutnya.[30] Peningkatan *overhead* ini akan mengurangi waktu efektif CPU yang tersedia untuk eksekusi instruksi proses, yang pada gilirannya dapat menurunkan *throughput* sistem secara keseluruhan.[30] Sebaliknya, algoritma seperti SJF *non-preemptive* yang dirancang untuk mengoptimalkan *turnaround time* rata-rata mungkin menghasilkan waktu respons yang sangat buruk untuk proses-proses panjang yang tiba lebih awal.[31] Hal ini menunjukkan bahwa tidak ada satu algoritma penjadwalan tunggal yang dapat dianggap "terbaik" secara universal. Pemilihan algoritma yang tepat selalu melibatkan pertimbangan *trade-off* berdasarkan prioritas dan tujuan spesifik dari sistem operasi yang dirancang.[30]

B. Kriteria Kinerja Penjadwalan

Untuk dapat mengevaluasi dan membandingkan efektivitas berbagai algoritma penjadwalan CPU, diperlukan serangkaian kriteria kinerja yang terukur. Kriteria-kriteria ini membantu dalam menentukan sejauh mana sebuah algoritma berhasil mencapai tujuan-tujuan penjadwalan yang telah ditetapkan.[1] Berikut adalah penjelasan rinci mengenai kriteria kinerja utama:

1. **Utilisasi CPU (*CPU Utilization*)**: Metrik ini mengukur seberapa sibuk CPU dalam menjalankan tugas-tugas yang produktif, dinyatakan dalam persentase waktu. Tujuannya adalah untuk memaksimalkan utilisasi CPU, idealnya mendekati 100%, meskipun pada sistem nyata biasanya berkisar antara 40% (beban ringan) hingga 90% (beban berat).[1] Utilisasi yang tinggi menunjukkan bahwa sumber daya CPU dimanfaatkan secara efisien.

2. **Throughput**: Metrik ini mengukur jumlah total proses atau pekerjaan (*job*) yang berhasil diselesaikan oleh sistem per satuan waktu (misalnya, proses per detik atau pekerjaan per jam).[3] *Throughput* yang tinggi menandakan efisiensi sistem dalam menyelesaikan tugas. Targetnya adalah memaksimalkan *throughput*.
3. **Waktu Penyelesaian (*Turnaround Time* - TAT)**: Ini adalah total waktu yang dibutuhkan oleh sebuah proses sejak saat ia masuk ke dalam sistem (*arrival time*) hingga proses tersebut selesai dieksekusi (*completion time*).[7] Waktu ini mencakup semua fase yang dilalui proses, termasuk waktu menunggu di memori, waktu menunggu di *ready queue*, waktu eksekusi di CPU, dan waktu yang dihabiskan untuk operasi I/O.[13] Secara matematis, TAT=Waktu Selesai−Waktu Kedatangan. Targetnya adalah meminimalkan rata-rata *turnaround time* (ATAT) untuk semua proses.[1]
4. **Waktu Tunggu (*Waiting Time* - WT)**: Metrik ini mengukur total waktu yang dihabiskan oleh sebuah proses dalam *ready queue*, menunggu giliran untuk mendapatkan alokasi CPU.[1] Algoritma penjadwalan CPU secara langsung mempengaruhi waktu tunggu ini. Secara matematis, WT=TAT−Burst Time (waktu eksekusi CPU).[7] Targetnya adalah meminimalkan rata-rata waktu tunggu (AWT).[13]
5. **Waktu Respons (*Response Time* - RT)**: Khususnya penting untuk sistem interaktif, waktu respons adalah interval waktu dari saat sebuah permintaan atau perintah dimasukkan oleh pengguna hingga respons pertama mulai ditampilkan di layar atau dihasilkan oleh sistem.[3] Waktu respons berbeda dari *turnaround time* karena hanya mengukur waktu hingga respons *pertama*, bukan hingga penyelesaian tugas secara keseluruhan. Secara matematis, RT=Waktu Mulai Eksekusi Pertama−Waktu Kedatangan. Targetnya adalah meminimalkan waktu respons.[13]
6. **Keadilan (*Fairness*)**: Kriteria ini bersifat lebih kualitatif, memastikan bahwa setiap proses mendapatkan porsi waktu CPU yang wajar dan tidak ada proses yang secara tidak adil ditunda eksekusinya untuk waktu yang sangat lama (*starvation*).[1]

Perlu dicatat bahwa metrik-metrik kinerja ini seringkali saling terkait dan kadang-kadang bertentangan. Sebagai contoh, algoritma penjadwalan seperti SJF yang dirancang untuk meminimalkan *Average Waiting Time* (AWT) terbukti optimal secara teoritis untuk metrik tersebut.[8] Namun, pencapaian optimalitas ini seringkali mengorbankan aspek keadilan. SJF dapat menyebabkan *starvation* bagi proses-proses yang memiliki *burst time* panjang, karena proses-proses dengan *burst time* pendek akan selalu didahulukan jika mereka terus menerus tiba di sistem.[33] *Starvation* merupakan bentuk ketidakadilan yang ekstrem. Selain itu, jika sebuah proses panjang dieksekusi terlebih dahulu (misalnya dalam FCFS atau SJF *non-preemptive* ketika proses panjang tersebut tiba pertama), maka proses-proses

pendek yang tiba kemudian akan mengalami waktu respons yang buruk.[10] Hal ini menunjukkan adanya hubungan terbalik antara optimalitas AWT (seperti pada SJF) dengan *fairness* dan *response time*.

Definisi "kinerja baik" sangat bergantung pada konteks dan tujuan dari sistem operasi itu sendiri. Sistem *batch*, yang fokus pada pemrosesan sejumlah besar data tanpa interaksi pengguna langsung, mungkin akan memprioritaskan *throughput* dan *turnaround time*.[22] Algoritma seperti SJF, yang unggul dalam meminimalkan *average turnaround time* [9], seringkali lebih cocok untuk lingkungan *batch*.[10] Sebaliknya, sistem interaktif atau *time-sharing*, seperti sistem operasi desktop modern, lebih menekankan pada pengalaman pengguna yang responsif.[22] Dalam konteks ini, *response time* dan *fairness* menjadi kriteria yang lebih dominan.[25] Algoritma *Round Robin*, yang secara eksplisit dirancang untuk *time-sharing* [1] dan dikenal memberikan *response time* yang baik [29], menjadi pilihan yang lebih sesuai untuk sistem interaktif.[10] Ini menegaskan bahwa tidak ada satu metrik tunggal atau algoritma tunggal yang superior dalam segala situasi; konteks operasional sistemlah yang menentukan kriteria kinerja mana yang paling relevan dan algoritma mana yang paling tepat untuk mencapai tujuan tersebut.

C. Tipe Penjadwalan: Preemptive vs. Non-Preemptive

Algoritma penjadwalan CPU dapat diklasifikasikan secara luas menjadi dua tipe utama berdasarkan apakah proses yang sedang berjalan dapat diinterupsi atau tidak: *preemptive* dan *non-preemptive*.[1] Pemahaman mengenai perbedaan fundamental antara kedua tipe ini penting karena mempengaruhi cara kerja algoritma dan karakteristik kinerja sistem.

1. Penjadwalan Non-Preemptive (Non-Preemptive Scheduling):
   Dalam model penjadwalan non-preemptive, sekali CPU telah dialokasikan untuk sebuah proses, proses tersebut akan terus memegang kendali CPU hingga ia menyelesaikannya secara sukarela.1 Proses hanya akan melepaskan CPU dalam dua kondisi utama:
   - Ketika proses tersebut selesai dieksekusi (terminasi).
   - Ketika proses tersebut beralih ke status menunggu (*waiting state*), misalnya karena memerlukan operasi I/O. Dengan kata lain, sistem operasi tidak dapat secara paksa mengambil alih CPU dari proses yang sedang berjalan, meskipun ada proses lain dengan prioritas lebih tinggi atau kebutuhan mendesak yang menunggu di *ready queue*. Keputusan penjadwalan hanya terjadi ketika proses yang sedang berjalan melepaskan CPU.[3] Contoh algoritma yang umumnya bersifat *non-preemptive* adalah *First Come First Served* (FCFS) dan versi dasar dari *Shortest Job First* (SJF).[1]

2. Penjadwalan Preemptive (Preemptive Scheduling):
   Berbeda dengan non-preemptive, penjadwalan preemptive memungkinkan sistem operasi untuk menginterupsi (melakukan preempt) proses yang sedang berjalan dan mengalokasikan CPU ke proses lain.1 Interupsi ini dapat terjadi karena berbagai alasan, tergantung pada algoritma yang digunakan, seperti:
   - Habisnya jatah waktu (*time slice* atau *quantum*) yang dialokasikan untuk proses tersebut (seperti dalam algoritma *Round Robin*).
   - Kedatangan proses baru di *ready queue* yang memiliki prioritas lebih tinggi daripada proses yang sedang berjalan (seperti dalam algoritma *Priority Scheduling preemptive* atau SRTF).
   - Sebuah proses yang sebelumnya menunggu I/O selesai dan kembali ke *ready queue* dengan prioritas yang lebih tinggi. Ketika *preemption* terjadi, proses yang sedang berjalan dipindahkan dari status *running* ke status *ready*, dan penjadwal memilih proses lain (biasanya yang memiliki prioritas lebih tinggi atau yang berikutnya dalam antrian *Round Robin*) untuk dieksekusi.[3] Contoh algoritma yang bersifat *preemptive* adalah *Round Robin* (RR) dan *Shortest Remaining Time First* (SRTF, yang merupakan versi *preemptive* dari SJF).[1]

Pilihan antara penjadwalan *preemptive* dan *non-preemptive* memiliki implikasi mendasar terhadap *responsiveness* dan *fairness* sistem. Penjadwalan *non-preemptive* lebih sederhana karena proses berjalan hingga selesai atau memblokir dirinya sendiri. Namun, hal ini dapat menyebabkan masalah signifikan: jika sebuah proses dengan *burst time* yang sangat panjang mendapatkan CPU, semua proses lain, termasuk proses interaktif yang membutuhkan respons cepat atau proses pendek, harus menunggu hingga proses panjang tersebut selesai.[100] Ini secara drastis mengurangi *responsiveness* sistem dan dapat dianggap tidak adil.

Sebaliknya, penjadwalan *preemptive* memungkinkan sistem operasi untuk campur tangan dan mencegah satu proses memonopoli CPU.[101] Dengan menginterupsi proses yang berjalan lama untuk memberikan kesempatan kepada proses lain (seperti dalam RR [1] atau ketika proses berprioritas lebih tinggi tiba [101]), *preemption* secara signifikan meningkatkan *responsiveness* sistem [101] dan *fairness*.[101] Namun, keuntungan ini datang dengan biaya. Setiap kali *preemption* terjadi, sistem operasi harus melakukan *context switch*, yaitu menyimpan status (konteks) dari proses yang diinterupsi dan memuat status dari proses yang akan dijalankan berikutnya.[13] Operasi *context switch* ini memerlukan waktu dan sumber daya CPU, yang dikenal sebagai *overhead*.[35] Jika *preemption* terjadi terlalu sering (misalnya, dengan *time quantum* yang sangat kecil di RR), *overhead context switching* dapat menjadi signifikan, mengurangi efisiensi keseluruhan sistem. Jadi, terdapat *trade-off* yang jelas: *preemption* meningkatkan

responsiveness dan *fairness* dengan mengorbankan potensi efisiensi karena *overhead*.

Selain itu, implementasi penjadwalan *preemptive* lebih kompleks. Karena sebuah proses dapat diinterupsi pada *titik manapun* dalam eksekusinya, termasuk saat sedang mengakses atau memodifikasi data yang mungkin juga digunakan oleh proses lain (data bersama atau *shared data*), ada potensi terjadinya *race condition*.[13] *Race condition* terjadi ketika hasil eksekusi bergantung pada urutan penjadwalan yang tidak terduga dari proses-proses yang mengakses data bersama, yang dapat menyebabkan data menjadi tidak konsisten. Untuk mencegah hal ini, sistem operasi *preemptive* harus menyediakan mekanisme sinkronisasi, seperti *mutex* atau *semaphore*, untuk melindungi akses ke *critical sections* (bagian kode yang mengakses data bersama).[13] Penggunaan mekanisme sinkronisasi ini menambah kompleksitas baik pada desain sistem operasi maupun pada pengembangan aplikasi yang berjalan di atasnya.[102] Penjadwalan *non-preemptive*, di sisi lain, lebih sederhana dalam hal ini karena proses hanya melepaskan CPU pada titik-titik yang telah ditentukan (selesai atau blokir I/O) [101], sehingga pengembang dapat lebih mudah memastikan bahwa pelepasan CPU tidak terjadi di tengah operasi kritis. Ini menggarisbawahi bahwa kesederhanaan *non-preemptive* datang dengan potensi masalah kinerja (responsiveness/fairness), sementara fleksibilitas *preemptive* memerlukan penanganan kompleksitas tambahan terkait sinkronisasi.

II. Algoritma Shortest Job First (SJF) dan Shortest Remaining Time First (SRTF)

A. Konsep Dasar: Prioritas Berdasarkan Waktu Eksekusi

Algoritma *Shortest Job First* (SJF) adalah salah satu strategi penjadwalan CPU yang didasarkan pada prinsip sederhana: memberikan prioritas eksekusi kepada proses yang diperkirakan membutuhkan waktu CPU (*CPU burst time*) paling singkat.[1] Ide dasarnya adalah bahwa dengan menyelesaikan pekerjaan-pekerjaan terpendek terlebih dahulu, waktu tunggu rata-rata untuk semua proses dalam sistem dapat diminimalkan. SJF dapat dipandang sebagai kasus khusus dari algoritma penjadwalan berbasis prioritas, di mana nilai prioritas suatu proses berbanding terbalik dengan panjang *burst time* CPU berikutnya yang diprediksi; semakin pendek prediksi *burst time*, semakin tinggi prioritasnya.[2]

Motivasi utama di balik pengembangan SJF adalah untuk mengatasi salah satu kelemahan signifikan dari algoritma penjadwalan yang lebih sederhana seperti *First Come First Served* (FCFS), yaitu *convoy effect*.[10] Dalam FCFS, proses dijadwalkan

murni berdasarkan urutan kedatangan.[86] Jika sebuah proses dengan *burst time* yang sangat panjang (misalnya, P1) tiba lebih dahulu daripada beberapa proses dengan *burst time* yang pendek (misalnya, P2, P3), maka proses-proses pendek tersebut terpaksa harus menunggu hingga proses panjang P1 selesai dieksekusi.[86] Fenomena ini, di mana proses-proses pendek "terjebak" di belakang proses panjang, disebut *convoy effect* [10] dan dapat menyebabkan peningkatan drastis pada *average waiting time* (AWT).[86] Algoritma SJF [49] mencoba memperbaiki masalah ini dengan secara eksplisit memilih proses terpendek yang tersedia di *ready queue* untuk dieksekusi berikutnya. Dengan mendahulukan proses pendek (seperti P2 dan P3 dalam contoh di atas) daripada proses panjang (P1), waktu tunggu untuk proses-proses pendek tersebut menjadi minimal. Meskipun proses panjang P1 mungkin harus menunggu lebih lama, penurunan signifikan dalam waktu tunggu untuk proses-proses pendek biasanya lebih besar daripada peningkatan waktu tunggu P1, sehingga menghasilkan penurunan AWT secara keseluruhan.[75] Ini menunjukkan bahwa SJF merupakan upaya langsung untuk mengoptimalkan metrik AWT dengan mengatasi masalah spesifik yang muncul pada FCFS.

Terdapat dua varian utama dari algoritma SJF, yang dibedakan berdasarkan apakah proses yang sedang berjalan dapat diinterupsi atau tidak:

1. **SJF Non-Preemptive**: Ini adalah bentuk klasik SJF. Sekali sebuah proses dipilih dan mulai dieksekusi, ia akan terus berjalan hingga menyelesaikan *CPU burst*-nya atau secara sukarela melepaskan CPU (misalnya, untuk menunggu I/O).[1]
2. **SJF Preemptive**: Varian ini juga dikenal dengan nama *Shortest Remaining Time First* (SRTF).[2] Dalam SRTF, jika sebuah proses baru tiba di *ready queue* dan memiliki *burst time* yang lebih pendek daripada *sisa waktu eksekusi* dari proses yang sedang berjalan saat itu, maka proses yang sedang berjalan akan diinterupsi (*preempted*) dan CPU akan dialihkan ke proses baru yang lebih pendek tersebut.[2]

B. SJF Non-Preemptive

1. Prinsip Kerja dan Aturan

Algoritma SJF *non-preemptive* bekerja dengan memilih proses dari *ready queue* yang memiliki estimasi *CPU burst time* terpendek untuk dieksekusi berikutnya. Jika terdapat dua atau lebih proses dengan *burst time* terpendek yang sama, maka aturan *First Come First Served* (FCFS) biasanya diterapkan sebagai pemutus ikatan (*tie-breaker*), di mana proses yang tiba lebih dahulu di *ready queue* akan dipilih.[6] Karakteristik utama dari varian ini adalah sifat *non-preemptive*-nya: sekali CPU dialokasikan kepada sebuah proses, proses tersebut akan terus menggunakan CPU hingga ia menyelesaikan *CPU burst*-nya saat ini atau hingga ia secara eksplisit melepaskan

CPU, misalnya untuk melakukan operasi I/O atau karena terminasi.[1] Sistem operasi tidak dapat menginterupsi proses yang sedang berjalan ini, bahkan jika ada proses baru dengan *burst time* yang jauh lebih pendek tiba di *ready queue*.

2. Contoh Kasus dan Diagram Gantt

Untuk mengilustrasikan cara kerja SJF *non-preemptive*, mari kita pertimbangkan contoh berikut [1]:

Terdapat empat proses (P1, P2, P3, P4) dengan waktu kedatangan (*Arrival Time* - AT) dan *CPU Burst Time* (BT) dalam milidetik (ms) sebagai berikut:

Proses	Waktu Kedatangan (AT)	Burst Time (BT)
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Langkah-langkah penjadwalan:

- **Waktu 0:** Hanya P1 yang berada di *ready queue*. Maka, P1 dipilih untuk dieksekusi.
- **Waktu 7:** P1 selesai dieksekusi. Saat ini, proses P2 (tiba di 2), P3 (tiba di 4), dan P4 (tiba di 5) sudah berada di *ready queue*. Kita bandingkan *burst time* mereka: BT(P2)=4, BT(P3)=1, BT(P4)=4. Proses P3 memiliki BT terpendek (1 ms), sehingga P3 dipilih berikutnya.
- **Waktu 8:** P3 selesai dieksekusi (7 + 1 = 8). Proses yang tersisa di *ready queue* adalah P2 dan P4. Keduanya memiliki BT yang sama (4 ms). Kita gunakan FCFS sebagai *tie-breaker*. P2 tiba lebih dahulu (AT=2) daripada P4 (AT=5), sehingga P2 dipilih berikutnya.
- **Waktu 12:** P2 selesai dieksekusi (8 + 4 = 12). Hanya P4 yang tersisa di *ready queue*.
- **Waktu 16:** P4 selesai dieksekusi (12 + 4 = 16). Semua proses telah selesai.

Diagram Gantt:

Diagram Gantt berikut memvisualisasikan urutan eksekusi proses:

| P1 (7) | P3 (1) | P2 (4) | P4 (4) |
0 7 8 12 16

3. Perhitungan Metrik Kinerja (AWT, ATAT, ART)

Berdasarkan diagram Gantt di atas, kita dapat menghitung metrik kinerja untuk setiap proses:

- **Waktu Selesai (*Completion Time* - CT):**
  - CT(P1) = 7 ms
  - CT(P2) = 12 ms
  - CT(P3) = 8 ms
  - CT(P4) = 16 ms
- **Waktu Penyelesaian (*Turnaround Time* - TAT):** TAT=CT−AT
  - TAT(P1) = 7 - 0 = 7 ms
  - TAT(P2) = 12 - 2 = 10 ms
  - TAT(P3) = 8 - 4 = 4 ms
  - TAT(P4) = 16 - 5 = 11 ms
  - **Waktu Penyelesaian Rata-rata (ATAT):** (7 + 10 + 4 + 11) / 4 = 32 / 4 = **8 ms**
- **Waktu Tunggu (*Waiting Time* - WT):** WT=TAT−BT
  - WT(P1) = 7 - 7 = 0 ms
  - WT(P2) = 10 - 4 = 6 ms
  - WT(P3) = 4 - 1 = 3 ms
  - WT(P4) = 11 - 4 = 7 ms
  - **Waktu Tunggu Rata-rata (AWT):** (0 + 6 + 3 + 7) / 4 = 16 / 4 = **4 ms**
- **Waktu Respons (*Response Time* - ART):** ART=Waktu Mulai Eksekusi Pertama−AT. Untuk algoritma *non-preemptive*, waktu respons sama dengan waktu tunggu.
  - ART(P1) = 0 - 0 = 0 ms
  - ART(P2) = 8 - 2 = 6 ms
  - ART(P3) = 7 - 4 = 3 ms
  - ART(P4) = 12 - 5 = 7 ms
  - **Waktu Respons Rata-rata (AART):** (0 + 6 + 3 + 7) / 4 = 16 / 4 = **4 ms**

C. SJF Preemptive (SRTF)

1. Prinsip Kerja dan Aturan Preemption

Algoritma *Shortest Remaining Time First* (SRTF) adalah versi *preemptive* dari SJF.[2]

Prinsip utamanya adalah mengalokasikan CPU kepada proses yang memiliki *sisa waktu eksekusi* (*remaining burst time*) terpendek di antara semua proses yang ada di *ready queue* dan proses yang sedang berjalan.[51]

Aturan *preemption* berlaku sebagai berikut: Setiap kali ada proses baru yang tiba di *ready queue*, sisa waktu eksekusi dari proses yang *sedang berjalan* dibandingkan dengan *burst time* total dari proses yang baru tiba (atau sisa waktu eksekusinya jika proses tersebut pernah berjalan sebelumnya). Jika proses yang baru tiba memiliki sisa waktu eksekusi yang lebih pendek daripada sisa waktu eksekusi proses yang sedang berjalan, maka proses yang sedang berjalan akan diinterupsi (*preempted*).[2] Proses yang di-*preempt* ini akan dikembalikan ke *ready queue*, dan CPU akan dialokasikan kepada proses baru yang memiliki sisa waktu terpendek. Keputusan penjadwalan (perbandingan sisa waktu) dilakukan setiap kali ada proses baru tiba atau ketika proses yang sedang berjalan selesai.

2. Contoh Kasus dan Diagram Gantt

Mari gunakan set proses yang sama seperti contoh SJF *non-preemptive* untuk melihat perbedaan akibat *preemption* [6]:

Proses	Waktu Kedatangan (AT)	Burst Time (BT)
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Langkah-langkah penjadwalan SRTF:

- **Waktu 0:** P1 tiba. P1 mulai dieksekusi. Sisa BT(P1) = 7.
- **Waktu 2:** P2 tiba (BT=4). Proses yang sedang berjalan adalah P1 dengan sisa BT = 7 - 2 = 5. Karena BT(P2) = 4 < Sisa BT(P1) = 5, maka P1 di-*preempt*. P2 mulai dieksekusi. Antrian: [P1(sisa 5)].
- **Waktu 4:** P3 tiba (BT=1). Proses yang sedang berjalan adalah P2 dengan sisa BT = 4 - 2 = 2. Karena BT(P3) = 1 < Sisa BT(P2) = 2, maka P2 di-*preempt*. P3 mulai dieksekusi. Antrian: [P1(sisa 5), P2(sisa 2)].
- **Waktu 5:** P4 tiba (BT=4). Proses yang sedang berjalan adalah P3 dengan sisa BT

= 1 - 1 = 0. P3 selesai dieksekusi. Sekarang kita bandingkan proses di antrian: P1(sisa 5), P2(sisa 2), P4(BT=4). P2 memiliki sisa BT terpendek (2). P2 melanjutkan eksekusi. Antrian:.

- **Waktu 7:** P2 selesai dieksekusi (5 + 2 = 7). Proses di antrian: P1(sisa 5), P4(BT=4). P4 memiliki BT terpendek (4). P4 mulai dieksekusi. Antrian: [P1(sisa 5)].
- **Waktu 11:** P4 selesai dieksekusi (7 + 4 = 11). Hanya P1 yang tersisa di antrian. P1 melanjutkan eksekusi. Antrian:.
- **Waktu 16:** P1 selesai dieksekusi (11 + 5 = 16). Semua proses telah selesai.

Diagram Gantt:

| P1(2) | P2(2) | P3(1) | P2(2) | P4(4) | P1(5) |
0 2 4 5 7 11 16

3. Perhitungan Metrik Kinerja (AWT, ATAT, ART)

Berdasarkan diagram Gantt SRTF:

- **Waktu Selesai (***Completion Time* **- CT):**
  - CT(P1) = 16 ms
  - CT(P2) = 7 ms
  - CT(P3) = 5 ms
  - CT(P4) = 11 ms
- **Waktu Penyelesaian (***Turnaround Time* **- TAT):** TAT=CT−AT
  - TAT(P1) = 16 - 0 = 16 ms
  - TAT(P2) = 7 - 2 = 5 ms
  - TAT(P3) = 5 - 4 = 1 ms
  - TAT(P4) = 11 - 5 = 6 ms
  - **Waktu Penyelesaian Rata-rata (ATAT):** (16 + 5 + 1 + 6) / 4 = 28 / 4 = **7 ms**
- **Waktu Tunggu (***Waiting Time* **- WT):** WT=TAT−BT
  - WT(P1) = 16 - 7 = 9 ms (Menunggu dari 2-4, 5-7, 7-11)
  - WT(P2) = 5 - 4 = 1 ms (Menunggu dari 4-5)
  - WT(P3) = 1 - 1 = 0 ms
  - WT(P4) = 6 - 4 = 2 ms (Menunggu dari 5-7)
  - **Waktu Tunggu Rata-rata (AWT):** (9 + 1 + 0 + 2) / 4 = 12 / 4 = **3 ms**
- **Waktu Respons (***Response Time* **- ART):** ART=Waktu Mulai Eksekusi Pertama−AT
  - ART(P1) = 0 - 0 = 0 ms

- ART(P2) = 2 - 2 = 0 ms
- ART(P3) = 4 - 4 = 0 ms
- ART(P4) = 7 - 5 = 2 ms
- **Waktu Respons Rata-rata (AART):** (0 + 0 + 0 + 2) / 4 = 2 / 4 = **0.5 ms**

Perhatikan bahwa AWT dan ATAT untuk SRTF (3 ms dan 7 ms) lebih rendah dibandingkan SJF *non-preemptive* (4 ms dan 8 ms) untuk contoh kasus ini, menunjukkan potensi peningkatan kinerja melalui *preemption*. Waktu respons rata-rata juga jauh lebih baik dengan SRTF.

D. Analisis SJF/SRTF: Kelebihan dan Kekurangan

Algoritma SJF dan varian *preemptive*-nya, SRTF, memiliki sejumlah kelebihan dan kekurangan yang perlu dipertimbangkan:

Kelebihan:

1. **Optimalitas Waktu Tunggu Rata-rata (AWT):** Keunggulan utama SJF/SRTF adalah kemampuannya untuk menghasilkan *average waiting time* (AWT) minimum secara teoritis di antara semua algoritma penjadwalan (SJF untuk *non-preemptive*, SRTF untuk *preemptive*) untuk sekumpulan proses yang diberikan.[6] Ini berarti secara rata-rata, proses menghabiskan waktu paling sedikit menunggu di *ready queue*.
2. **Optimalisasi Waktu Penyelesaian (TAT):** Sebagai konsekuensi dari AWT yang minimal, SJF/SRTF juga cenderung mengoptimalkan *average turnaround time* (ATAT).[6]
3. **Respons Cepat untuk Proses Pendek (Terutama SRTF):** Proses-proses dengan *burst time* pendek akan mendapatkan prioritas tinggi dan dieksekusi dengan cepat, menghasilkan waktu respons yang baik bagi mereka.[29] SRTF, karena sifat *preemptive*-nya, sangat efektif dalam memberikan respons cepat bahkan jika proses pendek tiba saat proses panjang sedang berjalan.
4. **Peningkatan Throughput (dalam kondisi tertentu):** Jika *workload* sistem didominasi oleh banyak proses pendek, SJF/SRTF dapat meningkatkan *throughput* karena proses-proses ini diselesaikan dengan cepat.[6]

Kekurangan:

1. **Kesulitan Memprediksi** ***Burst Time*****:** Kekurangan paling fundamental dan praktis dari SJF/SRTF adalah kebutuhan untuk mengetahui atau memprediksi secara akurat panjang *CPU burst time* berikutnya untuk setiap proses.[6] Dalam sistem nyata, panjang *burst* berikutnya seringkali tidak diketahui sebelumnya. Meskipun ada teknik estimasi (seperti rata-rata eksponensial dari *burst* sebelumnya [94]),

estimasi ini tidak selalu akurat dan dapat mengurangi efektivitas algoritma.[9] Optimalitas teoritis SJF/SRTF sangat bergantung pada asumsi pengetahuan *burst time* yang sempurna ini. Di dunia nyata, karena harus mengandalkan estimasi, kinerja aktualnya mungkin tidak mencapai batas bawah teoritis AWT.[75]

2. **Potensi** *Starvation* **untuk Proses Panjang:** SJF/SRTF dapat menyebabkan *starvation* (kelaparan) bagi proses-proses yang memiliki *burst time* panjang.[6] Jika ada aliran proses pendek yang terus-menerus masuk ke sistem, proses panjang mungkin tidak akan pernah mendapatkan kesempatan untuk dieksekusi karena selalu ada proses lain yang memiliki (sisa) *burst time* lebih pendek. Masalah *starvation* ini menyoroti konflik inheren antara upaya mencapai optimalitas kinerja (AWT minimum) dan menjamin *fairness* bagi semua proses. Algoritma yang murni fokus pada satu metrik dapat secara signifikan mengorbankan metrik penting lainnya. Fenomena ini mendorong pengembangan algoritma penjadwalan yang lebih canggih dan seimbang, seperti *Multilevel Feedback Queues* (MLFQ), yang mencoba mendekati efisiensi SJF sambil secara aktif mencegah *starvation* melalui mekanisme seperti *aging* (meningkatkan prioritas proses yang menunggu lama).[29]
3. **Overhead** *Context Switching* **(SRTF):** Varian *preemptive* (SRTF) dapat mengalami *overhead context switching* yang tinggi jika *preemption* sering terjadi.[32] Setiap kali proses baru yang lebih pendek tiba dan menyebabkan *preemption*, sistem harus menyimpan konteks proses yang berjalan dan memuat konteks proses baru, yang memakan waktu CPU dan dapat mengurangi *throughput* keseluruhan.

III. Algoritma Round Robin (RR)

A. Konsep Dasar: Time-Sharing dan Keadilan

Algoritma *Round Robin* (RR) adalah salah satu algoritma penjadwalan CPU yang paling umum digunakan, terutama dalam sistem operasi yang dirancang untuk lingkungan *time-sharing* dan interaktif.[1] Konsep inti di balik RR adalah pembagian waktu (*time-sharing*) yang adil di antara semua proses yang siap dieksekusi. Berbeda dengan SJF yang memprioritaskan berdasarkan estimasi waktu eksekusi, RR memperlakukan semua proses secara setara.[24]

Secara prinsip, RR bekerja mirip dengan algoritma *First Come First Served* (FCFS) dalam hal urutan pemrosesan awal (proses yang datang lebih dulu cenderung dilayani lebih dulu), namun dengan tambahan mekanisme *preemption*.[1] Setiap proses yang berada di *ready queue* tidak diizinkan berjalan hingga selesai, melainkan diberikan jatah waktu CPU yang terbatas, yang dikenal sebagai *time slice* atau *kuantum waktu* (*time quantum*).[2] *Ready queue* dalam RR biasanya diimplementasikan sebagai antrian

sirkular atau FIFO.[22]

Penjadwal mengambil proses dari depan *ready queue*, menjalankannya selama satu *time quantum*. Jika proses tersebut selesai sebelum *time quantum* habis, ia akan melepaskan CPU secara sukarela, dan penjadwal akan segera beralih ke proses berikutnya di antrian.[17] Namun, jika proses belum selesai setelah *time quantum*-nya habis, sistem operasi akan melakukan *preempt* pada proses tersebut, menyimpang konteksnya, dan memindahkannya kembali ke *ujung belakang ready queue*.[2] CPU kemudian dialokasikan ke proses yang sekarang berada di depan antrian. Proses ini terus berulang secara siklik, memastikan bahwa setiap proses mendapatkan kesempatan untuk menggunakan CPU secara bergantian.

Tujuan utama dari mekanisme ini adalah untuk mencapai *fairness* (keadilan) [7] dan mencegah terjadinya *starvation* [7], di mana suatu proses tidak pernah mendapatkan kesempatan untuk dieksekusi. Dengan memberikan jatah waktu yang sama kepada semua proses secara bergilir, RR menjamin bahwa setiap proses pada akhirnya akan mendapatkan akses ke CPU.

B. Peran Krusial Kuantum Waktu (*Time Quantum*)

Ukuran *kuantum waktu* (*time quantum* atau *time slice*) adalah parameter paling kritis yang menentukan perilaku dan kinerja algoritma *Round Robin*.[1] Pemilihan nilai kuantum yang tepat melibatkan *trade-off* penting antara *responsiveness* dan efisiensi sistem.

1. Dampak Ukuran Kuantum terhadap Kinerja

- **Kuantum Waktu Terlalu Kecil:** Jika nilai kuantum waktu dipilih sangat kecil (misalnya, mendekati waktu yang dibutuhkan untuk *context switch*), maka *preemption* akan terjadi sangat sering.[1] Di satu sisi, ini akan memberikan *response time* yang sangat baik, karena setiap proses akan mendapatkan giliran CPU dengan cepat, menciptakan ilusi pemrosesan paralel yang kuat, yang sangat diinginkan dalam sistem interaktif.[30] Namun, di sisi lain, frekuensi *context switching* yang tinggi akan menghasilkan *overhead* sistem yang signifikan.[1] Sebagian besar waktu CPU akan dihabiskan untuk menyimpan dan memuat konteks proses, bukan untuk melakukan pekerjaan yang sebenarnya, sehingga menurunkan efisiensi CPU dan *throughput* sistem secara keseluruhan.[8] AWT dan ATAT juga cenderung meningkat dalam kondisi ini.[24]
- **Kuantum Waktu Terlalu Besar:** Sebaliknya, jika nilai kuantum waktu dipilih sangat besar (misalnya, lebih besar dari *burst time* terpanjang dari semua proses), maka *preemption* akan jarang terjadi atau bahkan tidak terjadi sama sekali.[1] Dalam kasus ini, algoritma RR akan berperilaku sangat mirip, atau bahkan

identik, dengan algoritma FCFS.[1] Proses yang pertama kali masuk ke *ready queue* akan berjalan hingga selesai (atau hingga kuantum yang sangat besar itu habis), sementara proses lain menunggu. *Overhead context switching* akan minimal, tetapi keuntungan utama RR dalam hal *responsiveness* dan *fairness* untuk proses-proses pendek akan hilang.[30] Waktu respons untuk proses interaktif bisa menjadi sangat buruk.[66]

- **Dampak pada AWT dan ATAT:** Hubungan antara ukuran kuantum dan AWT/ATAT tidak selalu linear. Secara umum, *average turnaround time* (ATAT) dapat ditingkatkan jika sebagian besar proses dapat menyelesaikan *CPU burst* mereka dalam satu *time quantum*.[27] Penelitian menunjukkan bahwa dengan semakin besar kuantum yang dapat disediakan, AWT dan ATAT cenderung menjadi minimal, mendekati perilaku FCFS.[40] Namun, ini mengorbankan *response time*. Kuantum yang sangat kecil, meskipun memberikan *response time* yang baik, cenderung meningkatkan AWT dan ATAT karena *overhead*.[24]

Kesimpulannya, ukuran kuantum adalah parameter *tuning* paling krusial untuk RR. Tidak ada nilai "satu ukuran cocok untuk semua".[22] Nilai optimal sangat bergantung pada distribusi *burst time* dari proses-proses yang ada dalam *workload* sistem. Jika *workload* didominasi oleh proses interaktif pendek, kuantum kecil mungkin lebih disukai untuk *response time*.[30] Jika *workload* didominasi oleh proses *batch* panjang, kuantum besar mungkin lebih efisien untuk mengurangi *overhead*.[1] Karena *workload* sistem dapat bervariasi dari waktu ke waktu, kuantum waktu statis mungkin tidak selalu optimal.

2. Menentukan Ukuran Kuantum yang Ideal

Dalam praktiknya, ukuran kuantum waktu untuk RR biasanya dipilih dalam rentang antara 10 hingga 100 milidetik.[1] Waktu *context switch* sendiri biasanya jauh lebih kecil, misalnya sekitar 10 mikrosekon, sehingga *overhead*-nya relatif kecil dibandingkan kuantum.[22]

Sebuah aturan praktis (*rule of thumb*) yang sering dikutip adalah mencoba mengatur kuantum waktu sedemikian rupa sehingga sekitar 80% dari *CPU burst* proses lebih pendek dari satu *time quantum*.[35] Tujuannya adalah agar sebagian besar proses dapat menyelesaikan eksekusi CPU mereka dalam satu giliran tanpa perlu di-*preempt* dan kembali ke antrian, sehingga meminimalkan *context switching* yang tidak perlu sambil tetap mempertahankan sifat *time-sharing*.

Namun, karena sifat *workload* yang dinamis, banyak penelitian telah mengeksplorasi penggunaan *kuantum waktu dinamis* (*dynamic time quantum*).[5] Pendekatan ini mencoba menyesuaikan ukuran kuantum secara otomatis berdasarkan kondisi sistem

saat ini atau karakteristik proses di *ready queue*. Misalnya, kuantum dapat dihitung berdasarkan rata-rata *burst time* [5], median *burst time*, atau menggunakan metode statistik yang lebih canggih seperti persentil (misalnya, mengambil *burst time* proses ke-80% setelah diurutkan [61]) atau *interquartile range* (IQR) untuk menangani *outlier*.[67] Tujuannya adalah untuk mencapai kinerja yang lebih baik (AWT/ATAT lebih rendah) dibandingkan RR dengan kuantum statis, dengan beradaptasi terhadap *workload* yang berubah.

Upaya optimasi kuantum waktu ini, terutama melalui pendekatan dinamis, menunjukkan pergeseran paradigma. Algoritma RR klasik dengan kuantum statis mengutamakan *fairness* murni (setiap proses mendapat jatah waktu yang persis sama). Namun, pendekatan kuantum dinamis mencoba meningkatkan *efisiensi* (mengurangi AWT/ATAT, mendekati kinerja SJF) dengan memberikan kuantum yang lebih sesuai dengan kebutuhan proses (misalnya, kuantum lebih besar untuk proses yang tampak lebih panjang), sambil tetap berusaha mempertahankan *responsiveness* yang wajar dan mencegah *starvation* dengan memastikan semua proses tetap mendapatkan giliran. Ini merupakan upaya untuk mencari titik tengah, menggabungkan keunggulan efisiensi SJF dengan keunggulan *fairness* dan *responsiveness* RR.[14]

C. Contoh Kasus dan Diagram Gantt

Untuk memahami penerapan algoritma RR secara praktis, mari kita lihat dua contoh: satu dengan waktu kedatangan proses yang sama, dan satu lagi dengan waktu kedatangan yang berbeda.

1. Contoh dengan Waktu Kedatangan Sama

Misalkan terdapat 3 proses (P1, P2, P3) yang semuanya tiba pada waktu 0 (AT=0) dengan *burst time* (BT) sebagai berikut, dan *time quantum* (TQ) = 4 ms [1]:

Proses	Burst Time (BT)
P1	24
P2	3
P3	3

Langkah-langkah penjadwalan RR (TQ=4):

- **Waktu 0-4:** P1 dieksekusi. Sisa BT(P1) = 20. Antrian: [P2, P3, P1].
- **Waktu 4-7:** P2 dieksekusi. BT(P2) = 3 <= TQ = 4. P2 selesai. Antrian: [P3, P1].
- **Waktu 7-10:** P3 dieksekusi. BT(P3) = 3 <= TQ = 4. P3 selesai. Antrian: [P1].
- **Waktu 10-14:** P1 dieksekusi. Sisa BT(P1) = 16. Antrian: [P1].
- **Waktu 14-18:** P1 dieksekusi. Sisa BT(P1) = 12. Antrian: [P1].
- **Waktu 18-22:** P1 dieksekusi. Sisa BT(P1) = 8. Antrian: [P1].
- **Waktu 22-26:** P1 dieksekusi. Sisa BT(P1) = 4. Antrian: [P1].
- **Waktu 26-30:** P1 dieksekusi. Sisa BT(P1) = 0. P1 selesai. Antrian:.

Diagram Gantt (AT=0, TQ=4):

| P1(4) | P2(3) | P3(3) | P1(4) | P1(4) | P1(4) | P1(4) |
0 4 7 10 14 18 22 26 30

2. Contoh dengan Waktu Kedatangan Berbeda

Sekarang, mari kita gunakan contoh dari [54] dengan waktu kedatangan yang berbeda dan TQ = 2:

Proses	Waktu Kedatangan (AT)	Burst Time (BT)
P1	0	5
P2	1	4
P3	2	2
P4	3	1

Langkah-langkah penjadwalan RR (TQ=2):

- **Waktu 0:** P1 tiba. Antrian: [P1].
- **Waktu 0-2:** P1 dieksekusi. Sisa BT(P1) = 3. P2 tiba (Waktu 1). P3 tiba (Waktu 2). P1 kembali ke antrian. Antrian: [P2, P3, P1].
- **Waktu 2-4:** P2 dieksekusi. Sisa BT(P2) = 2. P4 tiba (Waktu 3). P2 kembali ke antrian. Antrian: [P3, P1, P4, P2].

- **Waktu 4-6:** P3 dieksekusi. BT(P3) = 2 <= TQ = 2. P3 selesai. Antrian: [P1, P4, P2].
- **Waktu 6-8:** P1 dieksekusi. Sisa BT(P1) = 1. P1 kembali ke antrian. Antrian: [P4, P2, P1].
- **Waktu 8-9:** P4 dieksekusi. BT(P4) = 1 <= TQ = 2. P4 selesai. Antrian: [P2, P1].
- **Waktu 9-11:** P2 dieksekusi. Sisa BT(P2) = 0. P2 selesai. Antrian: [P1].
- **Waktu 11-12:** P1 dieksekusi. Sisa BT(P1) = 0. P1 selesai. Antrian:.

Diagram Gantt (AT berbeda, TQ=2):

| P1(2) | P2(2) | P3(2) | P1(2) | P4(1) | P2(2) | P1(1) |
0 2 4 6 8 9 11 12

D. Perhitungan Metrik Kinerja (AWT, ATAT, ART)

Mari kita hitung metrik kinerja untuk kedua contoh di atas.

Contoh 1 (AT=0, TQ=4):

- **CT:** P1=30, P2=7, P3=10
- **TAT (**CT−AT**):** P1=30, P2=7, P3=10
  - ATAT = (30 + 7 + 10) / 3 = 47 / 3 = **15.67 ms**
- **WT (**TAT−BT**):** P1=30-24=6, P2=7-3=4, P3=10-3=7
  - AWT = (6 + 4 + 7) / 3 = 17 / 3 = **5.67 ms** [1]
- **ART (**Waktu Mulai Pertama−AT**):** P1=0, P2=4, P3=7
  - AART = (0 + 4 + 7) / 3 = 11 / 3 = **3.67 ms**

Contoh 2 (AT berbeda, TQ=2):

- **CT:** P1=12, P2=11, P3=6, P4=9
- **TAT (**CT−AT**):** P1=12-0=12, P2=11-1=10, P3=6-2=4, P4=9-3=6
  - ATAT = (12 + 10 + 4 + 6) / 4 = 32 / 4 = **8 unit waktu** [54]
- **WT (**TAT−BT**):** P1=12-5=7, P2=10-4=6, P3=4-2=2, P4=6-1=5
  - AWT = (7 + 6 + 2 + 5) / 4 = 20 / 4 = **5 unit waktu** [54]
- **ART (**Waktu Mulai Pertama−AT**):** P1=0-0=0, P2=2-1=1, P3=4-2=2, P4=8-3=5
  - AART = (0 + 1 + 2 + 5) / 4 = 8 / 4 = **2 unit waktu**

Perhitungan ini menunjukkan bagaimana metrik kinerja dapat bervariasi tergantung pada *burst time*, waktu kedatangan, dan *time quantum* yang digunakan.

E. Analisis RR: Kelebihan dan Kekurangan

Algoritma *Round Robin* memiliki karakteristik unik yang membawa sejumlah kelebihan dan kekurangan:

Kelebihan:

1. **Keadilan (*Fairness*):** Ini adalah keunggulan utama RR. Setiap proses dijamin mendapatkan porsi waktu CPU yang sama dalam setiap siklus, mencegah monopoli CPU oleh satu proses.[7]
2. **Pencegahan *Starvation*:** Sebagai konsekuensi langsung dari keadilan, RR secara inheren bebas dari *starvation*. Setiap proses dalam *ready queue* pada akhirnya akan mendapatkan giliran untuk dieksekusi.[7] Jika ada *n* proses dan kuantum *q*, proses tidak akan menunggu lebih dari (n–1)q unit waktu.[5]
3. **Responsif:** RR sangat cocok untuk sistem interaktif dan *time-sharing* karena memberikan waktu respons yang baik. Pengguna merasakan sistem yang tanggap karena proses mereka mendapatkan kesempatan berjalan secara berkala, bahkan jika belum selesai.[24]
4. **Kesederhanaan Implementasi:** Logika dasar RR relatif sederhana dibandingkan beberapa algoritma lain, membuatnya mudah untuk dipahami dan diimplementasikan dalam sistem operasi.[7]

Kekurangan:

1. **Kinerja Sangat Bergantung pada Ukuran Kuantum:** Efektivitas RR sangat sensitif terhadap pemilihan *time quantum*. Nilai yang tidak tepat dapat menurunkan kinerja secara signifikan.[1]
2. **Overhead *Context Switching* Tinggi:** Jika *time quantum* terlalu kecil, frekuensi *context switching* menjadi sangat tinggi. *Overhead* yang terkait dengan penyimpanan dan pemulihan konteks proses dapat menghabiskan sebagian besar waktu CPU, mengurangi efisiensi dan *throughput*.[1]
3. **AWT dan ATAT Cenderung Lebih Tinggi:** Dibandingkan dengan algoritma seperti SJF yang mengoptimalkan metrik ini, RR seringkali menghasilkan *average waiting time* dan *average turnaround time* yang lebih lama, terutama untuk proses-proses pendek yang mungkin dapat selesai lebih cepat tanpa *preemption*.[8]
4. **Tidak Memperhitungkan Prioritas:** RR memperlakukan semua proses dengan kepentingan yang sama. Ini menjadi kekurangan jika sistem memerlukan penanganan berbeda untuk proses-proses dengan tingkat urgensi atau prioritas yang bervariasi.[24]

Kesederhanaan algoritma RR merupakan pedang bermata dua. Di satu sisi, ia mudah

diimplementasikan dan menjamin keadilan.[7] Di sisi lain, ketidakmampuannya untuk mempertimbangkan karakteristik proses seperti *burst time* (seperti yang dilakukan SJF) berarti ia mengorbankan potensi optimasi AWT dan ATAT.[8] Proses pendek mungkin harus mengalami beberapa *context switch* yang tidak perlu, sementara proses panjang terus menerus diinterupsi, yang mungkin kurang efisien daripada membiarkannya berjalan lebih lama dalam satu giliran.

Meskipun demikian, RR tetap menjadi algoritma yang sangat penting. Ia tidak hanya banyak digunakan dalam implementasi praktis tetapi juga menjadi fondasi konseptual untuk algoritma penjadwalan yang lebih canggih. Contohnya adalah *Multilevel Feedback Queues* (MLFQ).[29] MLFQ mencoba mengatasi keterbatasan RR dan SJF dengan menggabungkan ide-ide dari keduanya. MLFQ menggunakan beberapa antrian prioritas, seringkali dengan RR diterapkan pada antrian berprioritas tinggi untuk menjaga *responsiveness*.[29] Namun, MLFQ juga secara dinamis menyesuaikan prioritas proses berdasarkan perilakunya: proses yang cenderung *CPU-bound* (menggunakan seluruh kuantum) diturunkan prioritasnya, sementara proses yang *I/O-bound* (melepaskan CPU sebelum kuantum habis) dipertahankan atau dinaikkan prioritasnya.[29] Ini secara efektif meniru ide SJF (mendahulukan proses pendek/interaktif) tanpa memerlukan prediksi *burst time* eksplisit, sambil tetap mencegah *starvation* melalui mekanisme *aging*. Ini menunjukkan bagaimana RR, meskipun memiliki keterbatasan, berfungsi sebagai blok bangunan esensial dalam evolusi desain algoritma penjadwalan yang berusaha menyeimbangkan berbagai tujuan kinerja yang seringkali bertentangan.

IV. Studi Komparatif: SJF/SRTF vs. Round Robin

Membandingkan algoritma SJF/SRTF dan *Round Robin* (RR) secara langsung membantu memahami *trade-off* yang terlibat dalam pemilihan strategi penjadwalan CPU. Perbandingan ini akan mencakup metrik kinerja kuantitatif, aspek kualitatif seperti keadilan, serta pertimbangan praktis seperti *overhead*.

A. Perbandingan Metrik Kinerja (AWT, ATAT, ART, Throughput)

1. **Waktu Tunggu Rata-rata (AWT) & Waktu Penyelesaian Rata-rata (ATAT):**
    - **SJF/SRTF:** Secara teoritis, algoritma ini memberikan AWT dan ATAT minimum.[8] Dengan memprioritaskan pekerjaan terpendek, proses-proses ini selesai lebih cepat, mengurangi waktu tunggu keseluruhan.
    - **Round Robin (RR):** RR umumnya menghasilkan AWT dan ATAT yang lebih tinggi dibandingkan SJF/SRTF.[8] Hal ini disebabkan oleh *overhead context switching* dan fakta bahwa proses pendek mungkin harus menunggu beberapa putaran untuk selesai. Kinerja RR dalam hal ini sangat bergantung

pada ukuran *time quantum*.[27]

2. **Waktu Respons Rata-rata (ART):**
    - **SJF/SRTF:** SRTF (versi *preemptive*) memberikan ART yang sangat baik karena proses terpendek (atau dengan sisa waktu terpendek) segera dijalankan.[29] SJF *non-preemptive* bisa memiliki ART yang buruk jika terjadi *convoy effect*.[10]
    - **Round Robin (RR):** RR unggul dalam memberikan ART yang rendah dan konsisten.[8] Setiap proses mendapatkan kesempatan CPU dalam interval waktu yang relatif singkat, membuatnya ideal untuk sistem interaktif.
3. **Throughput:**
    - **SJF/SRTF:** Dapat mencapai *throughput* tinggi jika *workload* didominasi oleh proses-proses pendek yang dapat diselesaikan dengan cepat.[6]
    - **Round Robin (RR):** *Throughput* RR bisa lebih rendah dibandingkan SJF, terutama jika *time quantum* terlalu kecil, karena waktu CPU terbuang untuk *overhead context switching* yang sering.[8]

B. Perbandingan Aspek Keadilan dan Potensi Starvation

- **SJF/SRTF:** Algoritma ini secara inheren **tidak adil** terhadap proses-proses panjang. Prioritas yang diberikan kepada proses terpendek dapat menyebabkan proses panjang menunggu tanpa batas waktu jika ada aliran proses pendek yang konstan. Oleh karena itu, SJF/SRTF **rentan terhadap** *starvation*.[6]
- **Round Robin (RR):** RR dirancang dengan prinsip **keadilan**. Setiap proses mendapatkan kesempatan yang sama untuk menggunakan CPU dalam setiap siklus putaran. Akibatnya, RR **tidak rentan terhadap** *starvation*.[7]

C. Perbandingan Overhead Context Switching

- **SJF (Non-Preemptive):** Memiliki *overhead context switching* yang relatif rendah karena peralihan hanya terjadi ketika proses selesai atau memblokir dirinya sendiri.[10]
- **SRTF (Preemptive SJF):** Dapat memiliki *overhead* yang lebih tinggi daripada SJF *non-preemptive* jika *preemption* sering terjadi akibat kedatangan proses-proses yang lebih pendek.[29]
- **Round Robin (RR):** Cenderung memiliki *overhead context switching* tertinggi, terutama jika *time quantum* dipilih sangat kecil.[1] Setiap akhir *quantum* (jika proses belum selesai) memicu *context switch*.

D. Skenario Penggunaan: Kapan Memilih SJF/SRTF atau RR?

Pemilihan antara SJF/SRTF dan RR bergantung pada prioritas dan karakteristik sistem:

- **Gunakan SJF/SRTF jika:**

    - Prioritas utama adalah **meminimalkan AWT dan ATAT**.[9]
    - Sistem adalah **sistem** *batch* di mana *throughput* tinggi lebih penting daripada *response time* interaktif.[9]
    - Panjang *CPU burst* proses dapat **diprediksi atau diestimasi** dengan cukup akurat.[9]
    - Potensi ***starvation*** untuk proses panjang dapat ditoleransi atau diatasi dengan mekanisme tambahan (seperti *aging*).[10]
- **Gunakan Round Robin (RR) jika:**
    - Prioritas utama adalah **keadilan (***fairness***)** dan **waktu respons (***response time***)** yang baik untuk semua pengguna/proses.[9]
    - Sistem adalah **sistem interaktif** atau ***time-sharing***.[1]
    - **Pencegahan** *starvation* adalah keharusan.[9]
    - *Overhead context switching* yang mungkin lebih tinggi dapat diterima demi *responsiveness*.

Secara filosofis, pilihan antara SJF/SRTF dan RR mencerminkan penekanan yang berbeda dalam desain sistem operasi. SJF/SRTF mewakili pendekatan yang berorientasi pada **efisiensi sistem** secara keseluruhan, dengan fokus pada penyelesaian pekerjaan secepat mungkin secara rata-rata (mengoptimalkan AWT/ATAT).[75] Namun, efisiensi ini dicapai dengan mengorbankan keadilan bagi proses-proses individual, terutama yang berdurasi panjang. Sebaliknya, RR mewakili pendekatan yang lebih **berorientasi pada pengguna**, dengan fokus pada penyediaan pengalaman interaktif yang responsif dan perlakuan yang adil bagi semua proses yang berjalan bersamaan.[29] Keadilan dan *responsiveness* ini seringkali dicapai dengan mengorbankan sedikit efisiensi AWT/ATAT dibandingkan SJF.

Karena banyak sistem operasi modern harus menangani campuran *workload* yang beragam (proses *batch* yang intensif CPU berjalan bersamaan dengan proses interaktif pengguna) [10], algoritma dasar seperti SJF atau RR murni mungkin tidak memadai. Hal ini mendorong pengembangan algoritma penjadwalan hibrida atau yang lebih kompleks, seperti *Multilevel Feedback Queues* (MLFQ).[10] Algoritma semacam ini mencoba untuk menyeimbangkan berbagai tujuan penjadwalan: memberikan respons cepat (seperti RR) untuk tugas interaktif, memastikan *throughput* yang baik (seperti SJF/FCFS) untuk tugas *batch*, dan mencegah *starvation* melalui mekanisme penyesuaian prioritas dinamis.[33] Ini menunjukkan bahwa desain penjadwalan praktis seringkali merupakan sintesis cerdas dari prinsip-prinsip dasar yang diwakili oleh SJF dan RR.

E. Tabel Perbandingan Komprehensif

Tabel berikut menyajikan ringkasan perbandingan antara algoritma SJF (*Non-Preemptive*), SRTF (*Preemptive* SJF), dan *Round Robin* (RR) berdasarkan kriteria-kriteria utama yang telah dibahas.

Tabel 1: Perbandingan Algoritma Penjadwalan SJF/SRTF vs. Round Robin

Fitur	SJF (Non-Preemptive)	SRTF (Preemptive SJF)	Round Robin (RR)
Pendekatan Seleksi	*Burst time* total terpendek	Sisa *burst time* terpendek	Urutan kedatangan & *time quantum* tetap
Tipe	*Non-Preemptive*	*Preemptive*	*Preemptive*
AWT (Rata-rata)	Minimal (Teoritis)	Minimal (Teoritis)	Cenderung lebih tinggi dari SJF/SRTF, bergantung pada *quantum*
ATAT (Rata-rata)	Minimal (Teoritis)	Minimal (Teoritis)	Cenderung lebih tinggi dari SJF/SRTF, bergantung pada *quantum*
ART (Rata-rata)	Bisa buruk (*convoy effect*)	Sangat baik	Sangat baik, konsisten
Throughput	Bisa tinggi (jika banyak proses pendek)	Bisa tinggi (jika banyak proses pendek)	Bisa lebih rendah (karena *overhead context switch*)
Fairness (Keadilan)	Tidak adil	Tidak adil	Adil
Potensi Starvation	Ya (untuk proses panjang)	Ya (untuk proses panjang)	Tidak
Context Switch Overhead	Rendah	Bisa tinggi (jika *preemption* sering)	Bisa sangat tinggi (jika *quantum* kecil)
Kebutuhan Prediksi BT	Ya (estimasi)	Ya (estimasi sisa waktu)	Tidak

Kompleksitas Implementasi	Relatif sedang (perlu estimasi)	Lebih kompleks (perlu cek sisa waktu & *preemption*)	Relatif sederhana
Skenario Penggunaan Ideal	Sistem *batch*, optimalisasi AWT/ATAT	Sistem *batch*/interaktif (jika ART penting), optim. AWT/ATAT	Sistem interaktif/*time-sharing*, *fairness* & ART penting, cegah *starvation*

Tabel ini memberikan gambaran ringkas mengenai *trade-off* yang ada antara ketiga algoritma penjadwalan ini. Pemilihan algoritma yang paling sesuai akan sangat bergantung pada tujuan spesifik dan karakteristik beban kerja dari sistem operasi yang ditargetkan.

V. Rangkuman

Modul ini telah membahas secara mendalam dua kelompok algoritma penjadwalan CPU yang fundamental dalam sistem operasi: *Shortest Job First* (SJF) beserta varian *preemptive*-nya, *Shortest Remaining Time First* (SRTF), dan *Round Robin* (RR).

A. Poin Kunci Algoritma SJF/SRTF:

- Beroperasi berdasarkan prinsip memprioritaskan proses dengan (sisa) waktu eksekusi CPU terpendek.
- Secara teoritis, menawarkan *average waiting time* (AWT) dan *average turnaround time* (ATAT) yang optimal.
- SRTF (versi *preemptive*) memberikan *response time* yang sangat baik untuk proses-proses pendek.
- Kelemahan utama terletak pada kesulitan praktis untuk memprediksi secara akurat *burst time* proses di masa depan.
- Rentan terhadap masalah *starvation*, di mana proses-proses dengan *burst time* panjang dapat terus menerus ditunda jika ada aliran proses pendek yang konstan.

B. Poin Kunci Algoritma Round Robin:

- Dirancang untuk sistem *time-sharing*, bekerja dengan memberikan jatah waktu CPU (*time quantum*) yang sama kepada setiap proses secara bergilir.
- Bersifat *preemptive;* proses diinterupsi jika *time quantum*-nya habis dan dipindahkan ke akhir *ready queue*.
- Menjamin *fairness* (keadilan) dan secara efektif mencegah terjadinya *starvation*.
- Memberikan *response time* yang baik dan konsisten, cocok untuk sistem interaktif.

- Kinerjanya sangat bergantung pada pemilihan ukuran *time quantum*. Kuantum yang terlalu kecil meningkatkan *overhead context switching*, sementara kuantum yang terlalu besar mengurangi *responsiveness* dan membuat RR berperilaku seperti FCFS.
- Umumnya menghasilkan AWT dan ATAT yang lebih tinggi dibandingkan SJF/SRTF.

C. Pertimbangan dalam Memilih Algoritma Penjadwalan:
Pemilihan algoritma penjadwalan CPU bukanlah keputusan yang sederhana dan tidak ada satu algoritma yang "terbaik" untuk semua situasi. Keputusan harus didasarkan pada pemahaman mendalam tentang:

- **Tujuan Utama Sistem:** Apakah prioritasnya adalah memaksimalkan *throughput* dan efisiensi (seperti dalam sistem *batch*), atau memaksimalkan *responsiveness* dan *fairness* (seperti dalam sistem interaktif)?
- **Karakteristik Beban Kerja (*Workload*):** Apakah sistem cenderung menangani proses-proses pendek dan interaktif, proses-proses *batch* yang panjang dan intensif CPU, atau campuran keduanya?
- **Kemampuan Prediksi:** Seberapa akurat sistem dapat memprediksi *burst time* proses? Jika prediksi sulit dilakukan, algoritma seperti SJF menjadi kurang praktis.
- **Toleransi terhadap *Starvation*:** Apakah *starvation* dapat diterima untuk beberapa jenis proses, atau apakah *fairness* mutlak diperlukan?
- **Overhead yang Dapat Diterima:** Seberapa besar *overhead context switching* yang dapat ditoleransi oleh sistem?

Dalam banyak sistem operasi modern, algoritma penjadwalan yang digunakan seringkali merupakan kombinasi atau varian yang lebih canggih dari algoritma-algoritma dasar ini (misalnya, *Multilevel Feedback Queues*) untuk mencoba mencapai keseimbangan terbaik antara berbagai kriteria kinerja yang seringkali bertentangan.

VI. Latihan Soal dan Pembahasan

Bagian ini menyajikan soal latihan untuk menguji pemahaman mengenai algoritma SJF/SRTF dan *Round Robin*, beserta pembahasan langkah-demi-langkah.

A. Soal Latihan SJF/SRTF

Soal 1: Diberikan lima proses dengan waktu kedatangan (AT) dan *burst time* (BT) dalam milidetik sebagai berikut:

Proses	AT	BT
P1	0	8

P2	1	4
P3	2	9
P4	3	5
P5	4	2

a. Gambarkan diagram Gantt dan hitung AWT, ATAT, serta AART untuk algoritma SJF Non-Preemptive.
b. Gambarkan diagram Gantt dan hitung AWT, ATAT, serta AART untuk algoritma SRTF (SJF Preemptive).

B. Soal Latihan Round Robin

Soal 2: Diberikan empat proses dengan waktu kedatangan (AT) dan *burst time* (BT) dalam milidetik sebagai berikut:

Proses	AT	BT
P1	0	10
P2	2	6
P3	3	2
P4	5	4

a. Gambarkan diagram Gantt dan hitung AWT, ATAT, serta AART untuk algoritma Round Robin dengan Time Quantum (TQ) = 3 ms.
b. Gambarkan diagram Gantt dan hitung AWT, ATAT, serta AART untuk algoritma Round Robin dengan Time Quantum (TQ) = 5 ms.

C. Pembahasan Langkah-demi-Langkah

Pembahasan Soal 1:

a. SJF Non-Preemptive

- **Waktu 0:** P1 tiba. P1 dieksekusi (BT=8).
- **Waktu 8:** P1 selesai. Proses di *ready queue*: P2(AT=1, BT=4), P3(AT=2, BT=9), P4(AT=3, BT=5), P5(AT=4, BT=2). Proses dengan BT terpendek adalah P5 (BT=2). P5 dieksekusi.
- **Waktu 10:** P5 selesai (8+2=10). Proses di *ready queue*: P2(BT=4), P3(BT=9), P4(BT=5). Proses dengan BT terpendek adalah P2 (BT=4). P2 dieksekusi.
- **Waktu 14:** P2 selesai (10+4=14). Proses di *ready queue*: P3(BT=9), P4(BT=5). Proses dengan BT terpendek adalah P4 (BT=5). P4 dieksekusi.
- **Waktu 19:** P4 selesai (14+5=19). Hanya P3 (BT=9) tersisa. P3 dieksekusi.
- **Waktu 28:** P3 selesai (19+9=28).

Diagram Gantt (SJF Non-Preemptive):

| P1(8) | P5(2) | P2(4) | P4(5) | P3(9) |
0 8 10 14 19 28

Tabel Hasil Perhitungan (SJF Non-Preemptive):

Proses	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)	ART (WT)
P1	0	8	8	8	0	0
P2	1	4	14	13	9	9
P3	2	9	28	26	17	17
P4	3	5	19	16	11	11
P5	4	2	10	6	4	4
Rata-rata				13.8	8.2	8.2

b. SRTF (SJF Preemptive)

- **Waktu 0:** P1 tiba (BT=8). P1 mulai eksekusi. Sisa BT(P1)=8.
- **Waktu 1:** P2 tiba (BT=4). Sisa BT(P1)=7. BT(P2)=4 < Sisa BT(P1)=7. P1 di-*preempt*. P2 mulai eksekusi. Antrian: [P1(7)]. Sisa BT(P2)=4.
- **Waktu 2:** P3 tiba (BT=9). Sisa BT(P2)=3. BT(P3)=9 > Sisa BT(P2)=3. P2 lanjut. Antrian: [P1(7), P3(9)].
- **Waktu 3:** P4 tiba (BT=5). Sisa BT(P2)=2. BT(P4)=5 > Sisa BT(P2)=2. P2 lanjut. Antrian: [P1(7), P3(9), P4(5)].
- **Waktu 4:** P5 tiba (BT=2). Sisa BT(P2)=1. BT(P5)=2 > Sisa BT(P2)=1. P2 lanjut. Antrian: [P1(7), P3(9), P4(5), P5(2)].
- **Waktu 5:** P2 selesai (1+4=5). Proses di antrian: P1(7), P3(9), P4(5), P5(2). Sisa BT terpendek adalah P5(2). P5 mulai eksekusi. Antrian: [P1(7), P3(9), P4(5)]. Sisa BT(P5)=2.
- **Waktu 7:** P5 selesai (5+2=7). Proses di antrian: P1(7), P3(9), P4(5). Sisa BT terpendek adalah P4(5). P4 mulai eksekusi. Antrian: [P1(7), P3(9)]. Sisa BT(P4)=5.
- **Waktu 12:** P4 selesai (7+5=12). Proses di antrian: P1(7), P3(9). Sisa BT terpendek adalah P1(7). P1 mulai eksekusi. Antrian: [P3(9)]. Sisa BT(P1)=7.
- **Waktu 19:** P1 selesai (12+7=19). Hanya P3(9) tersisa. P3 mulai eksekusi. Antrian:. Sisa BT(P3)=9.
- **Waktu 28:** P3 selesai (19+9=28).

Diagram Gantt (SRTF):

| P1(1) | P2(4) | P5(2) | P4(5) | P1(7) | P3(9) |
0 1 5 7 12 19 28

Tabel Hasil Perhitungan (SRTF):

Proses	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)	ART (Mulai-AT)
P1	0	8	19	19	11	0
P2	1	4	5	4	0	0
P3	2	9	28	26	17	17
P4	3	5	12	9	4	4
P5	4	2	7	3	1	1
Rata-rata				12.2	6.6	4.4

Pembahasan Soal 2:

a. Round Robin (TQ = 3 ms)

- **Waktu 0:** P1 tiba. Antrian: [P1].
- **Waktu 0-3:** P1 dieksekusi. Sisa BT(P1)=7. P2 tiba(2), P3 tiba(3). P1 kembali ke antrian. Antrian: [P2, P3, P1].
- **Waktu 3-6:** P2 dieksekusi. Sisa BT(P2)=3. P4 tiba(5). P2 kembali ke antrian. Antrian: [P3, P1, P4, P2].
- **Waktu 6-8:** P3 dieksekusi. BT(P3)=2 <= TQ=3. P3 selesai. Antrian: [P1, P4, P2].
- **Waktu 8-11:** P1 dieksekusi. Sisa BT(P1)=4. P1 kembali ke antrian. Antrian: [P4, P2, P1].
- **Waktu 11-14:** P4 dieksekusi. Sisa BT(P4)=1. P4 kembali ke antrian. Antrian: [P2, P1, P4].
- **Waktu 14-17:** P2 dieksekusi. Sisa BT(P2)=0. P2 selesai. Antrian: [P1, P4].
- **Waktu 17-20:** P1 dieksekusi. Sisa BT(P1)=1. P1 kembali ke antrian. Antrian: [P4, P1].
- **Waktu 20-21:** P4 dieksekusi. Sisa BT(P4)=0. P4 selesai. Antrian: [P1].
- **Waktu 21-22:** P1 dieksekusi. Sisa BT(P1)=0. P1 selesai. Antrian:.

Diagram Gantt (RR, TQ=3):

| P1(3) | P2(3) | P3(2) | P1(3) | P4(3) | P2(3) | P1(3) | P4(1) | P1(1) |

0 3 6 8 11 14 17 20 21 22

Tabel Hasil Perhitungan (RR, TQ=3):

Proses	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)	ART (Mulai-AT)
P1	0	10	22	22	12	0
P2	2	6	17	15	9	1 (3-2)
P3	3	2	8	5	3	3 (6-3)
P4	5	4	21	16	12	6 (11-5)
Rata-rata				14.5	9.0	2.5

b. Round Robin (TQ = 5 ms)

- **Waktu 0:** P1 tiba. Antrian: [P1].
- **Waktu 0-5:** P1 dieksekusi. Sisa BT(P1)=5. P2 tiba(2), P3 tiba(3), P4 tiba(5). P1 kembali ke antrian. Antrian: [P2, P3, P4, P1].
- **Waktu 5-10:** P2 dieksekusi. Sisa BT(P2)=1. P2 kembali ke antrian. Antrian: [P3, P4, P1, P2].
- **Waktu 10-12:** P3 dieksekusi. BT(P3)=2 <= TQ=5. P3 selesai. Antrian: [P4, P1, P2].
- **Waktu 12-16:** P4 dieksekusi. BT(P4)=4 <= TQ=5. P4 selesai. Antrian: [P1, P2].
- **Waktu 16-21:** P1 dieksekusi. Sisa BT(P1)=0. P1 selesai. Antrian: [P2].
- **Waktu 21-22:** P2 dieksekusi. Sisa BT(P2)=0. P2 selesai. Antrian:.

Diagram Gantt (RR, TQ=5):

| P1(5) | P2(5) | P3(2) | P4(4) | P1(5) | P2(1) |
0 5 10 12 16 21 22

Tabel Hasil Perhitungan (RR, TQ=5):

Proses	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)	ART (Mulai-AT)
P1	0	10	21	21	11	0
P2	2	6	22	20	14	3 (5-2)
P3	3	2	12	9	7	7 (10-3)
P4	5	4	16	11	7	7 (12-5)
Rata-rata				15.25	9.75	4.25

Analisis singkat: Dengan TQ=3, AWT=9.0 dan ATAT=14.5. Dengan TQ=5, AWT=9.75 dan

ATAT=15.25. Dalam kasus ini, TQ yang lebih kecil (3) memberikan AWT dan ATAT yang sedikit lebih baik, meskipun dengan lebih banyak *context switch*. Namun, ART rata-rata lebih baik dengan TQ=3. Ini mengilustrasikan *trade-off* dalam pemilihan *time quantum*.

IX. Referensi

4 ejurnal.umri.ac.id
23 slideshare.net
1 arna.lecturer.pens.ac.id
2 spada.uns.ac.id
12 repository.bsi.ac.id
17 repository.unikom.ac.id
18 id.scribd.com
24 id.scribd.com
3 cerryzhang.files.wordpress.com
25 socs.binus.ac.id
38 slideshare.net
21 online.flipbuilder.com
19 byjus.com
13 geeksforgeeks.org
119 biglysales.com
9 redwood.com
96 en.wikipedia.org
49 takeuforward.org
6 tutorialspoint.com
80 studytonight.com
73 takeuforward.org
74 studytonight.com
51 geeksforgeeks.org
104 geeksforgeeks.org
111 en.wikipedia.org
75 geeksforgeeks.org
105 sitesbay.com
76 data-flair.training
33 geeksforgeeks.org
77 unstop.com
78 lass.cs.umass.edu
79 youtube.com
37 oodlestechnologies.com
100 turing.com
101 geeksforgeeks.org
102 cs.stackexchange.com

106 geeksforgeeks.org
56 youtube.com
80 studytonight.com
50 geeksforgeeks.org
120 codex.cs.yale.edu
65 repository.dinus.ac.id
57 spada.uns.ac.id
58 id.scribd.com
81 slideshare.net
62 youtube.com
82 youtube.com
51 geeksforgeeks.org
83 youtube.com
59 condor.depaul.edu
121 youtube.com
26 cs.columbia.edu
113 eprints.untirta.ac.id
97 id.scribd.com
114 media.neliti.com
1 arna.lecturer.pens.ac.id
84 binus.ac.id
85 cerryzhang.files.wordpress.com
23 slideshare.net
122 m.youtube.com
117 researchgate.net
4 ejurnal.umri.ac.id
115 ro.scribd.com
14 repository.uinsa.ac.id
98 eprints.umpo.ac.id
25 socs.binus.ac.id
123 j-ptiik.ub.ac.id
45 ejurnal.seminar-id.com
39 repository.uksw.edu
124 researchgate.net
86 id.scribd.com
116 media.neliti.com
60 download.garuda.kemdikbud.go.id
27 ejurnal.methodist.ac.id
5 ejournal-medan.uph.edu
87 researchgate.net
61 download.garuda.kemdikbud.go.id
7 studytonight.com
62 youtube.com

63 youtube.com
52 stackoverflow.com
53 geeksforgeeks.org
64 cs.stackexchange.com
57 spada.uns.ac.id
65 repository.dinus.ac.id
23 slideshare.net
84 binus.ac.id
4 ejurnal.umri.ac.id
24 id.scribd.com
46 digilibadmin.unismuh.ac.id
125 jurnal.amikom.ac.id
88 download.garuda.kemdikbud.go.id
5 ejournal-medan.uph.edu
117 researchgate.net
40 researchgate.net
126 id.scribd.com
28 ccbp.in
41 citeseerx.ist.psu.edu
30 read.seas.harvard.edu
66 blog.heycoach.in
54 geeksforgeeks.org
89 geeksforgeeks.org
67 pmc.ncbi.nlm.nih.gov
34 ijseas.com
93 cs.utexas.edu
35 cs.stackexchange.com
8 geeksforgeeks.org
9 redwood.com
36 ijarcce.com
45 ejurnal.seminar-id.com
39 repository.uksw.edu
90 journal.aptii.or.id
65 repository.dinus.ac.id
68 researchgate.net
40 researchgate.net
127 ojs.stmik-banjarbaru.ac.id
10 designgurus.io
29 cs162.org
112 lass.cs.umass.edu
33 geeksforgeeks.org
31 stackoverflow.com
118 cs.unc.edu

32 eng.libretexts.org
94 cs.cornell.edu
11 jurnal.polgan.ac.id
69 researchgate.net
99 id.scribd.com
103 pdfs.semanticscholar.org
107 ijsret.com
91 researchgate.net
128 courses.cs.washington.edu
92 geeksforgeeks.org
129 drpress.org
13 geeksforgeeks.org
70 stackoverflow.com
42 cs.odu.edu
22 cs.uic.edu
130 cs.cornell.edu
55 afteracademy.com
64 cs.stackexchange.com
71 youtube.com
1 arna.lecturer.pens.ac.id
23 slideshare.net
85 cerryzhang.files.wordpress.com
131 youtube.com
108 id.scribd.com
109 scribd.com
57 spada.uns.ac.id
132 researchgate.net
72 slideshare.net
133 id.scribd.com
134 ru.scribd.com
43 slideshare.net
47 repository.dinus.ac.id
122 m.youtube.com
95 researchgate.net
12 repository.bsi.ac.id
135 id.scribd.com
15 osf.io
136 arna.lecturer.pens.ac.id
48 slideshare.net
137 repository.upy.ac.id
16 id.scribd.com
44 slideshare.net
110 repository.bsi.ac.id

20 tif.unusida.ac.id
138 digilib.esaunggul.ac.id
139 eprints.uad.ac.id
140 id.scribd.com
141 tif.unusida.ac.id
142 repository.unimal.ac.id
143 eprints.unm.ac.id
144 slideshare.net
6 tutorialspoint.com
80 studytonight.com
74 studytonight.com
51 geeksforgeeks.org
97 id.scribd.com
1 arna.lecturer.pens.ac.id
84 binus.ac.id
85 cerryzhang.files.wordpress.com
4 ejurnal.umri.ac.id
60 download.garuda.kemdikbud.go.id
27 ejurnal.methodist.ac.id
7 studytonight.com
28 ccbp.in
30 read.seas.harvard.edu
66 blog.heycoach.in
54 geeksforgeeks.org
89 geeksforgeeks.org
8 geeksforgeeks.org
9 redwood.com
10 designgurus.io
29 cs162.org
33 geeksforgeeks.org
31 stackoverflow.com
32 eng.libretexts.org
11 jurnal.polgan.ac.id
1 arna.lecturer.pens.ac.id
23 slideshare.net
57 spada.uns.ac.id
72 slideshare.net
110 repository.bsi.ac.id
20 tif.unusida.ac.id
15 osf.io
135 id.scribd.com

Works cited

1. arna.lecturer.pens.ac.id, accessed May 6, 2025, https://arna.lecturer.pens.ac.id/Diktat_SO/4.Penjadwalan%20CPU.pdf
2. Praktikum 10 Penjadwalan CPU 2 - SPADA UNS, accessed May 6, 2025, https://spada.uns.ac.id/mod/resource/view.php?id=216919
3. Penjadwalan CPU - WordPress.com, accessed May 6, 2025, https://cerryzhang.files.wordpress.com/2018/07/pertemuan-ke-91.pdf
4. Implementasi Algoritma Round Robin dan Priority Pada Sistem Antrian Rumah Sakit - Jurnal UMRI, accessed May 6, 2025, https://ejurnal.umri.ac.id/index.php/JIK/article/download/7334/3079/
5. MODIFIKASI ALGORITMA ROUND ROBIN DENGAN DYNAMIC QUANTUM TIME DAN PENGURUTAN PROSES SECARA ASCENDING, accessed May 6, 2025, https://ejournal-medan.uph.edu/isd/article/download/79/9/73
6. Shortest Job First Scheduling in Operating Systems - Tutorialspoint, accessed May 6, 2025, https://www.tutorialspoint.com/operating_system/os_shortest_job_first_scheduling.htm
7. Round Robin Scheduling Algorithm | Studytonight, accessed May 6, 2025, https://www.studytonight.com/operating-system/round-robin-scheduling
8. Difference between Shortest Job First (SJF) and Round-Robin (RR ..., accessed May 6, 2025, https://www.geeksforgeeks.org/difference-between-shortest-job-first-sjf-and-round-robin-rr-scheduling-algorithms/
9. Job Scheduling Algorithms: Which Is Best For Your Workflow?, accessed May 6, 2025, https://www.redwood.com/article/job-scheduling-algorithms/
10. Comparing different scheduling algorithms for scenario-based design, accessed May 6, 2025, https://www.designgurus.io/answers/detail/comparing-different-scheduling-algorithms-for-scenario-based-design
11. jurnal.polgan.ac.id, accessed May 6, 2025, https://jurnal.polgan.ac.id/index.php/sinkron/article/download/12525/2409/17098
12. MODUL SISTEM OPERASI - UNIVERSITAS BINA SARANA INFORMATIKA, accessed May 6, 2025, https://repository.bsi.ac.id/repo/files/338798/download/Modul-SO.pdf
13. CPU Scheduling in Operating Systems - GeeksforGeeks, accessed May 6, 2025, https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/
14. SUPER SMART OPTIMIZED ROUND ROBIN - Repository UIN Sunan Ampel Surabaya, accessed May 6, 2025, https://repository.uinsa.ac.id/605/1/Achmad%20Teguh%20Wibowo_%20Super%20smart%20optimized%20round%20robin.pdf
15. osf.io, accessed May 6, 2025, https://osf.io/6yh4e/download
16. Modul 5 - Penjadwalan Proses | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/395281184/Modul-5-Penjadwalan-Proses-docx
17. Jurnal Teknik Komputer Unikom – Komputika – Volume 3, No. 2 – 2014 ! 7

IMPLEMENTASI PENGEMBANGAN PENJADWALAN ROUND-ROBBIN, accessed May 6, 2025, https://repository.unikom.ac.id/30346/1/2-folkes.pdf
18. Konsep Penjadwalan Proses Sistem Operasi - Kelompok 6 | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/549874904/Konsep-Penjadwalan-Proses-Sistem-Operasi-Kelompok-6
19. Process Scheduler in Operating System - BYJU'S, accessed May 6, 2025, https://byjus.com/gate/process-scheduler-in-operating-system-notes/
20. tif.unusida.ac.id, accessed May 6, 2025, https://tif.unusida.ac.id/wp-content/uploads/2022/11/Modul-Sistem-Operasi.pdf
21. Page 1 - Konsep Penjadwalan CPU - Flipbuilder, accessed May 6, 2025, https://online.flipbuilder.com/uvief/rgvp/files/basic-html/page1.html
22. Operating Systems: CPU Scheduling, accessed May 6, 2025, https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html
23. Algoritma penjadwalan proses | PPT - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/algoritma-penjadwalan-proses-150977782/150977782
24. Makalah Penjadwalan Proses | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/691021097/MAKALAH-PENJADWALAN-PROSES
25. Penjadwalan / Scheduling – School of Computer Science - SoCS BINUS University, accessed May 6, 2025, https://socs.binus.ac.id/2020/11/16/penjadwalan-scheduling/
26. Scheduling I, accessed May 6, 2025, https://www.cs.columbia.edu/~jae/4118-LAST/L15-scheduling-1.pdf
27. ejurnal.methodist.ac.id, accessed May 6, 2025, https://ejurnal.methodist.ac.id/index.php/methodika/article/download/8/1
28. Round Robin Program in C - NxtWave, accessed May 6, 2025, https://www.ccbp.in/blog/articles/round-robin-program-in-c
29. cs162.org, accessed May 6, 2025, https://cs162.org/static/dis/4.pdf
30. Scheduler, accessed May 6, 2025, https://www.read.seas.harvard.edu/~kohler/class/05s-osp/notes/notes5.html
31. c - FCFS versus SJF versus RR - Stack Overflow, accessed May 6, 2025, https://stackoverflow.com/questions/24654297/fcfs-versus-sjf-versus-rr
32. 9.2: Scheduling Algorithms - Engineering LibreTexts, accessed May 6, 2025, https://eng.libretexts.org/Courses/Delta_College/Operating_System%3A_The_Basics/09%3A_CPU_Scheduling/9.2%3A_Scheduling_Algorithms
33. Advantages and Disadvantages of various CPU scheduling ..., accessed May 6, 2025, https://www.geeksforgeeks.org/advantages-and-disadvantages-of-various-cpu-scheduling-algorithms/
34. BEST TIME QUANTUM ROUND ROBIN CPU SCHEDULING ALGORITHM - IJSEAS, accessed May 6, 2025, https://ijseas.com/volume3/v3i5/ijseas20170529.pdf
35. Round Robin time slice(quantum time) - Computer Science Stack Exchange, accessed May 6, 2025,

https://cs.stackexchange.com/questions/120094/round-robin-time-slicequantum-time
36. Comparison of Round Robin Based CPU Scheduling Algorithms - ijarcce, accessed May 6, 2025, https://ijarcce.com/wp-content/uploads/2015/10/IJARCCE-7.pdf
37. Basic of Scheduler And Its Advantage And Disadvantages - Oodles Technologies, accessed May 6, 2025, https://www.oodlestechnologies.com/blogs/basic-of-scheduler-and-its-advantage-and-disadvantages/
38. Materi Penjadwalan CPU & Algoritma Penjadwalan CPU | PPT - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/materi-penjadwalan-cpu-algoritma-penjadwalan-cpu/267234208
39. Analisis Perbandingan Algoritma Penjadwalan CPU A New Improved Round Robin dan A Dynamic Time Quantum Shortest Job Round Robin, accessed May 6, 2025, https://repository.uksw.edu/bitstream/123456789/11319/2/T1_672010239_Full%20text.pdf
40. Round Robin Time Quantum vs. Performance Measures - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/figure/Round-Robin-Time-Quantum-vs-Performance-Measures_tbl3_345932391
41. Calculating Dynamic Time Quantum for Round Robin Process Scheduling Algorithm - CiteSeerX, accessed May 6, 2025, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a1630bb661332e09c156ae1f494963dd6023ce50
42. CPU Scheduling, accessed May 6, 2025, https://www.cs.odu.edu/~price/cs471/public_html/spring17/lectures/Scheduling.htm
43. Penjadualan CPU | PPT - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/sli-it-012336416-penjadualan-cpu/40241271
44. Penjadwalan-Proses.ppt - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/penjadwalanprosesppt/263272211
45. Analisis Perbandingan Algoritma Penjadwalan CPU First Come First Serve (FCFS) Dan Round Robin - OJS Seminar Indonesia Journal, accessed May 6, 2025, https://ejurnal.seminar-id.com/index.php/bits/article/download/1047/725
46. PENERAPAN METODE ROUND ROBIN DALAM EFEKTIVITAS LOAD BALANCER PADA PENDAFTARAN BEASISWA DI UNIVERSITAS MUHAMMADIYAH MAKASSAR PROP, accessed May 6, 2025, https://digilibadmin.unismuh.ac.id/upload/42230-Full_Text.pdf
47. Penjadwalan Proses, accessed May 6, 2025, https://repository.dinus.ac.id/docs/ajar/Penjadwalan_Proses_I.pptx
48. Proses Penjadwalan Pada Sistem Operasi - Copy.pptx - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/proses-penjadwalan-pada-sistem-operasi-copypptx/266896445

49. SJF Scheduling : Explained - Tutorial - takeUforward, accessed May 6, 2025, https://takeuforward.org/operating-system/sjf-scheduling-explained/
50. Program for Shortest Job First (or SJF) CPU Scheduling | Set 1 (Non- preemptive), accessed May 6, 2025, https://www.geeksforgeeks.org/program-for-shortest-job-first-or-sjf-cpu-scheduling-set-1-non-preemptive/
51. Shortest Remaining Time First (Preemptive SJF) Scheduling Algorithm | GeeksforGeeks, accessed May 6, 2025, https://www.geeksforgeeks.org/shortest-remaining-time-first-preemptive-sjf-scheduling-algorithm/
52. How to calculate average waiting time of Round robin scheduling? - Stack Overflow, accessed May 6, 2025, https://stackoverflow.com/questions/11644659/how-to-calculate-average-waiting-time-of-round-robin-scheduling
53. Program for Round Robin Scheduling for the Same Arrival Time - GeeksforGeeks, accessed May 6, 2025, https://www.geeksforgeeks.org/program-for-round-robin-scheduling-for-the-same-arrival-time/?ref=leftbar-rightbar
54. Round Robin Scheduling Algorithm with Different Arrival Time ..., accessed May 6, 2025, https://www.geeksforgeeks.org/round-robin-scheduling-with-different-arrival-times/
55. What is Burst time, Arrival time, Exit time, Response time, Waiting time, Turnaround time, and Throughput? - AfterAcademy, accessed May 6, 2025, https://afteracademy.com/blog/what-is-burst-arrival-exit-response-waiting-turnaround-time-and-throughput/
56. OS16a - Non-preemptive Shortest Job First (SJF) | Solved Example - YouTube, accessed May 6, 2025, https://www.youtube.com/watch?v=TC6CcmwRph8
57. spada.uns.ac.id, accessed May 6, 2025, https://spada.uns.ac.id/mod/resource/view.php?id=11725
58. Algoritma Penjadwalan Proses | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/491655800/Algoritma-Penjadwalan-Proses
59. CSC443/343 Oct12, accessed May 6, 2025, https://condor.depaul.edu/glancast/443class/docs/lecOct12.html
60. download.garuda.kemdikbud.go.id, accessed May 6, 2025, http://download.garuda.kemdikbud.go.id/article.php?article=1275777&val=14795&title=Super%20Smart%20Optimized%20Round%20Robin
61. Vol. 2 No. 2 Oktober 2014 Jurnal TEKNOIF ISSN : 2338-2724 1 - METODE ALTERNATIF PENETAPAN TIME QUANTUM DINAMIS UNTUK PERBAIKAN KINERJA PENJADWALAN ROUND ROBIN PADA PROSESSOR TUNGGAL, accessed May 6, 2025, http://download.garuda.kemdikbud.go.id/article.php?article=2820636&val=25245&title=METODE%20ALTERNATIF%20PENETAPAN%20TIME%20QUANTUM%20DINAMIS%20UNTUK%20PERBAIKAN%20KINERJA%20PENJADWALAN%20ROUND%20ROBIN%20PADA%20PROSESSOR%20TUNGGAL

62. Round Robin Scheduling, Gantt Chart, Avg. Turnaround Time, Avg. Waiting, Response Time: Numerical - YouTube, accessed May 6, 2025, https://www.youtube.com/watch?v=XoWI0r6S-Vk&pp=0gcJCdgAo7VqN5tD
63. Round Robin Scheduling (Turnaround Time & Waiting Time) - YouTube, accessed May 6, 2025, https://www.youtube.com/watch?v=7TpxxTNrcTg
64. Operating Systems - SJN/Round Robin - turn around time + waiting time, accessed May 6, 2025, https://cs.stackexchange.com/questions/85717/operating-systems-sjn-round-robin-turn-around-time-waiting-time
65. Penjadwalan, accessed May 6, 2025, https://repository.dinus.ac.id/docs/ajar/Penjadwalan.pdf
66. Round Robin Scheduling Using Queues - HeyCoach | Blogs, accessed May 6, 2025, https://blog.heycoach.in/round-robin-scheduling-using-queues/
67. An enhanced round robin using dynamic time quantum for real-time asymmetric burst length processes in cloud computing environment - PubMed Central, accessed May 6, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11326590/
68. (PDF) ANALISIS PERBANDINGAN ALGORITMA PENJADWALAN ROUND ROBIN DAN SHORTEST JOB FIRST UNTUK MANAJEMEN PROSES DALAM SINGLE PROCESSING - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/publication/366345386_ANALISIS_PERBANDINGAN_ALGORITMA_PENJADWALAN_ROUND_ROBIN_DAN_SHORTEST_JOB_FIRST_UNTUK_MANAJEMEN_PROSES_DALAM_SINGLE_PROCESSING
69. Comparison Analysis of CPU Scheduling : FCFS, SJF and Round Robin - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/publication/305492984_Comparison_Analysis_of_CPU_Scheduling_FCFS_SJF_and_Round_Robin
70. Calculating turnaround time for SJF and Round Robin - Stack Overflow, accessed May 6, 2025, https://stackoverflow.com/questions/22340528/calculating-turnaround-time-for-sjf-and-round-robin
71. ROUND ROBIN | SJF | SOLUTIONS ADDA | GATE TEST SERIES | EXPLAINED BY VIVEK, accessed May 6, 2025, https://www.youtube.com/watch?v=Jjxkfw9iTHU
72. Penjadwalan.pdf - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/penjadwalanpdf/253982691
73. SRTF Scheduling : Explained - Tutorial - takeUforward, accessed May 6, 2025, https://takeuforward.org/operating-system/srtf-scheduling-explained/
74. Shortest Remaining Time First Scheduling Algorithm - Studytonight, accessed May 6, 2025, https://www.studytonight.com/operating-system/shortest-remaining-time-first-scheduling-algorithm
75. Shortest Job First or SJF CPU Scheduling - GeeksforGeeks, accessed May 6, 2025, https://www.geeksforgeeks.org/shortest-job-first-or-sjf-cpu-scheduling/
76. Shortest Job First (SJF) Scheduling in Operating System - DataFlair, accessed May 6, 2025, https://data-flair.training/blogs/shortest-job-first-sjf-scheduling-in-operating-sys

tem/
77. Scheduling Algorithms In OS (Operating System) Explained +Examples - Unstop, accessed May 6, 2025, https://unstop.com/blog/scheduling-algorithms-in-operating-system
78. Strategy Description Advantages Disadvantages - LASS, accessed May 6, 2025, https://lass.cs.umass.edu/~shenoy/courses/fall14/discussions/discussion3/concepts.pdf
79. 3.12- Shortest Remaining Time First Advantages & Disadvantages In Operating System In Hindi - YouTube, accessed May 6, 2025, https://www.youtube.com/watch?v=qRPxrqP8J6g
80. Shortest Job First Scheduling Algorithm | Studytonight, accessed May 6, 2025, https://www.studytonight.com/operating-system/shortest-job-first
81. Ch 10 | PPT - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/ch-10-85899115/85899115
82. 1.11 Shortest Remaining Time First (SRTF) Scheduling Algorithm - YouTube, accessed May 6, 2025, https://www.youtube.com/watch?v=bMcYsy7l1Go
83. Shortest Remaining Time First Algorithm - Details including Gantt Chart, Sample GATE Qs Solved, accessed May 6, 2025, https://www.youtube.com/watch?v=gw4OQnDUmFg
84. ROUND-ROBIN SCHEDULING - Malang - BINUS UNIVERSITY, accessed May 6, 2025, https://binus.ac.id/malang/2021/11/round-robin-scheduling/
85. 12 Algoritma Penjadwalan CPU - WordPress.com, accessed May 6, 2025, https://cerryzhang.files.wordpress.com/2018/07/pertemuan-ke-121.pdf
86. Algo Penjadwalan | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/744064143/algo-penjadwalan
87. (PDF) Implementasi Algoritma Round Robin dan Priority Pada Sistem Antrian Rumah Sakit, accessed May 6, 2025, https://www.researchgate.net/publication/387651588_Implementasi_Algoritma_Round_Robin_dan_Priority_Pada_Sistem_Antrian_Rumah_Sakit
88. PENERAPAN ALGORITMA ROUND ROBIN DAN MODULO PADA LOAD BALANCING WEB SERVER, accessed May 6, 2025, http://download.garuda.kemdikbud.go.id/article.php?article=3358786&val=29468&title=PENERAPAN%20ALGORITMA%20ROUND%20ROBIN%20DAN%20MODULO%20PADA%20LOAD%20BALANCING%20WEB%20SERVER
89. Program for Round Robin Scheduling for the Same Arrival Time ..., accessed May 6, 2025, https://www.geeksforgeeks.org/program-for-round-robin-scheduling-for-the-same-arrival-time/
90. Meningkatkan Responsivitas Pada Sistem Operasi Android Melalui Implementasi Algoritma Penjadwalan Mutakhir, accessed May 6, 2025, https://journal.aptii.or.id/index.php/Router/article/download/146/229
91. Comparison of CPU Scheduling Algorithms: FCFS, SJF, SRTF, Round Robin, Priority Based, and Multilevel Queuing - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/publication/365100564_Comparison_of_CPU_Scheduling_Algorithms_FCFS_SJF_SRTF_Round_Robin_Priority_Based_and_Multilevel_

Queuing
92. Comparison of Different CPU Scheduling Algorithms in OS - GeeksforGeeks, accessed May 6, 2025, https://www.geeksforgeeks.org/comparison-of-different-cpu-scheduling-algorithms-in-os/
93. CS372: Solutions for Homework 9 - Texas Computer Science, accessed May 6, 2025, https://www.cs.utexas.edu/~lorenzo/corsi/cs372/06F/hw/9sol.html
94. CPU Scheduling - Computer Science Cornell, accessed May 6, 2025, https://www.cs.cornell.edu/courses/cs4410/2018fa/schedule/slides/04-scheduling.pdf
95. APLIKASI SIMULASI ALGORITMA PENJADWALAN SISTEM OPERASI - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/profile/Raissa-Amanda/publication/358020062_Aplikasi_Simulasi_Algoritma_Penjadwalan_Sistem_Operasi/links/61fa4f9faad5781d41c6ea25/Aplikasi-Simulasi-Algoritma-Penjadwalan-Sistem-Operasi.pdf
96. Shortest job next - Wikipedia, accessed May 6, 2025, https://en.wikipedia.org/wiki/Shortest_job_next
97. Round Robin Merupakan Salah Satu Algoritma Penjadwalan Yang ..., accessed May 6, 2025, https://id.scribd.com/doc/312434277/Round-Robin-Merupakan-Salah-Satu-Algoritma-Penjadwalan-Yang-Paling-Sederhana-Untuk-Proses-Dalam-Sistem-Operasi
98. BAB II TINJAUAN PUSTAKA 2.1 Algortima Round Robin Konsep dasar dari algoritma ini adalah dengan menggunakan time- sharing. Pada, accessed May 6, 2025, https://eprints.umpo.ac.id/7171/3/BAB%20II.pdf
99. Pengantar Sistem Operasi Komputer | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/612092551/Pengantar-Sistem-Operasi-Komputer
100. Different Types of Non-Preemptive CPU Scheduling Algorithms - Turing, accessed May 6, 2025, https://www.turing.com/kb/different-types-of-non-preemptive-cpu-scheduling-algorithms
101. Preemptive and Non-Preemptive Scheduling | GeeksforGeeks, accessed May 6, 2025, https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/
102. Difficulty understanding pre-emptive vs non-preemptive CPU scheduling, accessed May 6, 2025, https://cs.stackexchange.com/questions/47926/difficulty-understanding-pre-emptive-vs-non-preemptive-cpu-scheduling
103. Comparative analysis of the essential CPU scheduling algorithms - Semantic Scholar, accessed May 6, 2025, https://pdfs.semanticscholar.org/0a24/8215a55473199cc1254c3f9c32ae2510e541.pdf
104. Introduction of Shortest Remaining Time First (SRTF) algorithm - GeeksforGeeks, accessed May 6, 2025, https://www.geeksforgeeks.org/introduction-of-shortest-remaining-time-first-srtf-algorithm/

105. Shortest Job First (SJF) Scheduling - Operating System Tutorial - Sitesbay, accessed May 6, 2025, https://www.sitesbay.com/os/os-sjf-scheduling-advantage-disadvantage
106. Shortest Job First (or SJF) CPU Scheduling Non-preemptive algorithm using Segment Tree, accessed May 6, 2025, https://www.geeksforgeeks.org/shortest-job-first-or-sjf-cpu-scheduling-non-preemptive-algorithm-using-segment-tree/
107. A Comparative Review of CPU Scheduling Algorithms - Journal of Scientific Research & Engineering Trends, accessed May 6, 2025, https://ijsret.com/wp-content/uploads/2021/07/IJSRET_V7_issue4_555.pdf
108. Algoritma Penjadwalan Proses | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/presentation/336340997/ALGORITMA-PENJADWALAN-PROSES
109. Algoritma Penjadwalan Proses | PDF - Scribd, accessed May 6, 2025, https://www.scribd.com/document/493884319/algoritma-penjadwalan-proses
110. repository.bsi.ac.id, accessed May 6, 2025, https://repository.bsi.ac.id/repo/files/373275/download/Modul-Sistem-Operasi.pdf
111. Shortest remaining time - Wikipedia, accessed May 6, 2025, https://en.wikipedia.org/wiki/Shortest_remaining_time
112. on Scheduling Algorithms! - LASS, accessed May 6, 2025, https://lass.cs.umass.edu/~shenoy/courses/fall10/lectures/Lec06.pdf
113. eprints.untirta.ac.id, accessed May 6, 2025, https://eprints.untirta.ac.id/20390/3/Hanif%20Anggit%20Wicaksono_3332150017_02.pdf
114. ANALISIS ALGORITMA ROUND ROBIN, LEAST CONNECTION, DAN RATIO PADA LOAD BALANCNG MENGGUNAKAN OPNET MODELER - Neliti, accessed May 6, 2025, https://media.neliti.com/media/publications/67705-ID-analisis-algoritma-round-robin-least-con.pdf
115. Round Robin Merupakan Salah Satu Algoritma Penjadwalan Yang Paling Sederhana Untuk Proses Dalam Sistem Operasi - Scribd, accessed May 6, 2025, https://ro.scribd.com/doc/312434277/Round-Robin-Merupakan-Salah-Satu-Algoritma-Penjadwalan-Yang-Paling-Sederhana-Untuk-Proses-Dalam-Sistem-Operasi
116. Implementasi Algoritma Round Robin Pada Sistem Penjadwalan Mata Kuliah - Neliti, accessed May 6, 2025, https://media.neliti.com/media/publications/465400-none-08b6f2ef.pdf
117. (PDF) Analisis Algoritma Round Robin pada Penjadwalan CPU - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/publication/357774719_Analisis_Algoritma_Round_Robin_pada_Penjadwalan_CPU
118. Scheduling Processes, accessed May 6, 2025, https://www.cs.unc.edu/~porter/courses/comp530/f18/slides/scheduling-intro.pdf
119. How do Modern Operating Systems Handle Process Scheduling? - Bigly Sales, accessed May 6, 2025,

https://biglysales.com/how-do-modern-operating-systems-handle-process-scheduling/
120. CPU Scheduling, accessed May 6, 2025, https://codex.cs.yale.edu/avi/os-book/OS10/practice-exercises/PDF-practice-solu-dir/5.pdf
121. SRTF algorithm - an example - YouTube, accessed May 6, 2025, https://www.youtube.com/watch?v=wDmLvYiXzXI
122. Sistem Operasi-Penjadwalan Proses dengan Algoritma Round Robin dan Priority Scheduling. - YouTube, accessed May 6, 2025, https://m.youtube.com/watch?v=88tRZh4Gk2U
123. Analisis Perbandingan Performa Algoritma Round Robin dan Least Connection untuk Load Balancing pada Software Defined Network, accessed May 6, 2025, https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/542/226/3173
124. Analisis Perbandingan Algoritma Penjadwalan CPU First Come First Serve (FCFS) Dan Round Robin - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/publication/358113599_Analisis_Perbandingan_Algoritma_Penjadwalan_CPU_First_Come_First_Serve_FCFS_Dan_Round_Robin
125. ANALISIS KINERJA LOAD BALANCING ROUND ROBIN PADA WEBSITE SKALABEL - JURNAL UNIVERSITAS AMIKOM YOGYAKARTA, accessed May 6, 2025, https://jurnal.amikom.ac.id/index.php/joism/article/download/1441/513
126. Modul Perkuliahan Sistem Operasi | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/440265391/Modul-Perkuliahan-Sistem-Operasi
127. Perbandingan Kinerja dan Efektivitas Algoritma FCFS dan SJF pada, accessed May 6, 2025, https://ojs.stmik-banjarbaru.ac.id/index.php/progresif/article/view/2450
128. CPU Scheduling - Washington, accessed May 6, 2025, https://courses.cs.washington.edu/courses/cse451/12au/l11.pdf
129. Comparison and Analysis of Scheduling Algorithms: Exploring Performance, Time, Fairness, and Applicable Scenarios | Highlights in Science, Engineering and Technology - Darcy & Roy Press, accessed May 6, 2025, https://drpress.org/ojs/index.php/HSET/article/view/28734
130. CPU Scheduling - Computer Science Cornell, accessed May 6, 2025, https://www.cs.cornell.edu/courses/cs4410/2018sp/schedule/slides/04-scheduling.pdf
131. Round Robin Scheduling - Solved Problem (Part 1) - YouTube, accessed May 6, 2025, https://www.youtube.com/watch?v=QICmgBOMjII&pp=ygUGI3JyZGU0
132. Implementasi Algoritma Shortest Job First dan Round Robin pada Sistem Penjadwalan Pengiriman Barang - ResearchGate, accessed May 6, 2025, https://www.researchgate.net/publication/301548316_Implementasi_Algoritma_Shortest_Job_First_dan_Round_Robin_pada_Sistem_Penjadwalan_Pengiriman_Barang
133. SJF Dan SRF | PDF - Scribd, accessed May 6, 2025, https://id.scribd.com/document/662363703/SJF-dan-SRF
134. 6.penjadwalan CPU | PDF - Scribd, accessed May 6, 2025, https://ru.scribd.com/document/428387399/6-Penjadwalan-CPU

135. Modul 5 - Penjadwalan Proses | PDF | Komputer - Scribd, accessed May 6, 2025, https://id.scribd.com/doc/114600794/Modul-5-Penjadwalan-Proses-1
136. Struktur Sistem Operasi, accessed May 6, 2025, https://arna.lecturer.pens.ac.id/Diktat_SO/2.Struktur%20Sistem%20Operasi.pdf
137. sistem operasi - Repository UPY, accessed May 6, 2025, http://repository.upy.ac.id/7118/1/11.%20BUKU%20AJAR%20SISTEM%20OPERASI%202018_compressed.pdf
138. modul praktikum - sistem operasi - Universitas Esa Unggul, accessed May 6, 2025, https://digilib.esaunggul.ac.id/public/UEU-Course-9291-7_0142.pdf
139. MODUL PRAKTIKUM - eprints UAD, accessed May 6, 2025, https://eprints.uad.ac.id/29907/1/Modul%20SO%20Genap%202019%202020%20New.pdf
140. Materi Kuliah Sistem Operasi Teknik Informatika | PDF | Bisnis | Komputer - Scribd, accessed May 6, 2025, https://id.scribd.com/document/369438953/Materi-Kuliah-Sistem-Operasi-Teknik-Informatika
141. Modul Praktikum Sistem Operasi - Tentang TIF UNUSIDA, accessed May 6, 2025, https://tif.unusida.ac.id/wp-content/uploads/2022/11/MODUL-PRAKTIKUM-SISTEM-OPERASI.pdf
142. Sistem Operasi - Digital Library Universitas Malikussaleh, accessed May 6, 2025, https://repository.unimal.ac.id/4073/1/BUKU%20sistem%20operasi%20PDF.pdf
143. PENGEMBANGAN MODUL PEMBELAJARAN SISTEM OPERASI BERBASIS JARINGAN PADA PROGRAM STUDI TEKNIK INFORMATIKA DAN KOMPUTER UNIVERSITAS, accessed May 6, 2025, https://eprints.unm.ac.id/26371/1/10.%20Pengembangan%20Modul%20Pembelajaran%20Sistem%20Operasi%20Berbasis%20Jaringan%20pada%20Program%20Studi%20Teknik%20Informatika%20dan%20Komputer%20Universitas%20Negeri%20Makassar.pdf
144. MODUL AJAR INFORMATIKA 3 - SISTEM KOMPUTER.docx - SlideShare, accessed May 6, 2025, https://www.slideshare.net/slideshow/modul-ajar-informatika-3-sistem-komputer-docx/257183456