JDMIS: Journal of Data Mining and Information Systems
Vol. 3, No. 1, 2025, p. 43~53
ISSN: ISSN: 2986-3473 | p-ISSN 2986-5271
DOI: 10.54259/jdmis.v3i1.4076

Analisis *Average Waiting Time* Penjadwalan CPU Menggunakan Algoritma *Shortest Remaining First* dan Algoritma Round Robin

Ronald Belferik[1], Evander Banjarnahor[2],
[1] Informatika, Fakultas Teknologi Informasi, Universitas Pelita Harapan, Indonesia
[2] Matematika, Fakultas Sains dan Teknologi, Universitas Pelita Harapan, Indonesia
Email: [1] ronald.belferik@gmail.com, [2] evander.banjarnahor@gmail.com,

ABSTRAK

Dalam sistem operasi, penjadwalan proses merupakan aspek kritis untuk menentukan urutan eksekusi proses oleh CPU. Penelitian ini melakukan perbandingan waktu tunggu rata – rata atau *average waiting time* (AWT) pada algoritma *Shortest Remaining First* (SRF) dan algoritma *Round Robin* (RR) dimana masalah yang hendak diselesaikan adalah penjadwalan pada CPU. Tujuan dari penelitian ini adalah mendapatkan algoritma yang memiliki rata – rata waktu tunggu yang singkat. Hasil pengujian diperoleh bahwa algoritma SRF memiliki waktu tunggu rata – rata yang sangat singkat dengan nilai 29.85 ms dibanding algoritma RR yang mendapatkan hasil AWT 65.6 ms.

Kata Kunci: *Shortest Remaining First, Round Robin, Average Waiting Time*

ABSTRACT

In operating systems, process scheduling is a critical aspect to determine the order of process execution by the CPU. This research compares the average waiting time (AWT) of Shortest Remaining First (SRF) algorithm and Round Robin (RR) algorithm where the problem to be solved is CPU scheduling. The purpose of this research is to get an algorithm that has a short average waiting time. The test results obtained that the SRF algorithm has a very short average waiting time with a value of 29.85 ms compared to the RR algorithm which gets an AWT result of 65.6 ms.

Keywords: *Shortest Remaining First, Round Robin, Average Waiting Time*

Penulis Korespondensi:

Ronald Belferik
Email: ronald.belferik@uph.edu

Article Info

Diterima: 19 Februari 2025
Direvisi: 23 Februari 2025
Disetujui: 25 Februari 2025

This is an open access article under the CC BY license.

1. PENDAHULUAN

Di era komputasi modern, penggunaan sumber daya komputasi yang efisien menjadi sangat penting untuk memastikan kinerja sistem yang optimal. Salah satu sumber daya yang penting di dalam sistem komputer adalah *Central Processing Unit* (CPU), yang bertanggung jawab untuk mengeksekusi instruksi-instruksi program. Dalam sistem komputer yang menjalankan banyak tugas *(multitasking)*, penjadwalan CPU memainkan peran penting dalam menentukan urutan proses-proses yang bersaing untuk mendapatkan akses ke CPU untuk dieksekusi [1]-[9].

Ketika prosesor tidak menjalankan tugas (kondisi *idle*), sistem operasi memilih tugas dari memori utama untuk dijalankan dan menugaskan prosesor untuk menjalankan tugas tersebut. Kebutuhan untuk memprogram prosesor untuk menjalankan tugas di memori utama untuk memastikan kelangsungan sistem operasi adalah masalah yang perlu ditangani dan diselesaikan. Ini adalah hasil dari tidak mengoptimalkan kinerja CPU. Tujuan dari *multiprograming* adalah untuk meminimalkan waktu tidak aktif *(idle)* CPU saat menjalankan beberapa proses [10]. Untuk memaksimalkan penggunaan CPU, beberapa proses disimpan dalam memori dan CPU menggunakan waktu yang dialokasikan untuk menjalankan satu proses [11]. Dengan cara ini, waktu menganggur dapat dikurangi dan sistem operasi dapat meningkatkan produktivitas komputer dengan menjadwalkan alokasi

waktu untuk setiap proses dalam sistem [8]. Dalam sebuah sistem operasi, terdapat banyak proses yang perlu dieksekusi, sehingga timbul masalah mengenai proses mana yang harus dieksekusi dalam sistem. Oleh karena itu, perlu adanya suatu pedoman dan mekanisme mengenai urutan kerja sistem komputer di dalam sistem operasi. Untuk mengatasi masalah ini, telah dikembangkan algoritma penjadwalan yang menentukan proses mana yang dieksekusi CPU terlebih dahulu. Proses yang belum menerima alokasi dari CPU ditempatkan dalam antrean. Proses tersebut kemudian dieksekusi.

Dalam konteks ini, penulis ingin melakukan penelitian yang berjudul "Analisis *Average Waiting Time* (AWT) Penjadwalan CPU Menggunakan Algoritma *Shortest Remaining First* dan Algoritma *Round Robin*". Penelitian ini menganalisis dan membandingkan kinerja algoritma prioritas waktu tunggu terpendek dan algoritma *Round Robin* dalam hal waktu tunggu rata-rata (AWT) di bawah beban CPU yang berbeda. Dengan membandingkan kedua algoritma ini, diharapkan dapat memberikan wawasan tentang *trade-off* antara efisiensi dan *fairness* dalam penjadwalan CPU.

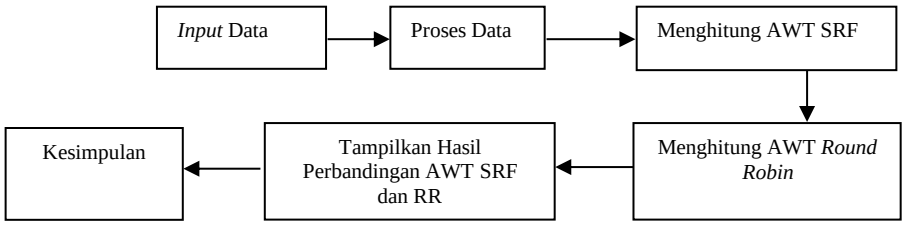
Penelitian ini akan melibatkan implementasi dan simulasi kedua algoritma penjadwalan CPU menggunakan data beban kerja sintetis. Hasil simulasi akan dianalisis untuk mengidentifikasi faktor-faktor yang mempengaruhi AWT pada masing-masing algoritma.

2. METODE PENELITIAN

Pada penelitian ini dilakukan perbandingan *Average Waiting Time* (AWT) Pada algoritma *Shortest Remaining First* dan algoritma *Round Robin*. Dalam Algoritma *Shortest Remaining First*, proses dengan sisa waktu paling sedikit untuk diselesaikan dipilih untuk dijalankan. Proses yang sedang berjalan akan terus berjalan hingga selesai atau proses baru dengan sisa waktu yang lebih pendek tiba. Dengan cara ini, proses yang dapat menyelesaikan paling cepat akan selalu diprioritaskan [12]. Algoritma *Round Robin* merupakan penjadwalan yang paling tua, sederhana, adil, banyak digunakan algoritmanya dan mudah diimplementasikan. Penjadwalan ini bukan dijalankan oleh proses lain tetapi oleh penjadwalan berdasarkan lama waktu berjalannya proses *(preempt by time).* Penjadwalan tanpa prioritas berasumsi bahwa semua proses memiliki kepentingan yang sama, sehingga tidak ada prioritas tertentu. Semua proses dianggap penting sehingga diberi sejumlah waktu oleh pemroses yang disebut kuanta *(quantum)* atau *time slice* dimana proses itu berjalan. Jika proses masih *running* sampai akhir *quantum*, maka CPU akan mem-*preempt* proses itu dan memberikannya ke proses lain [13]. Pada penelitian ini metode penelitian yang digunakan ialah *library search* dan *experiment*. Pada *library search*, maka akan dicari penelitian terkait untuk dikembangkan dan *experiment* yaitu menguji coba algoritma penjadwalan dengan mencari AWT pada tiap algoritma dan lalu membandingkannya.

2.1. Bagan Penelitian

Pada bagan penelitian, *input* data berupa contoh data penjadwalan yang akan diuji. Selanjutnya data tersebut diolah dengan menggunakan algoritma SRF dan RR. Hasil proses adalah AWT masing-masing algoritma. Setelah itu diambil kesimpulan AWT yang terbaik dari algoritma yang digunakan.



Gambar 1. Bagan penelitian

2.2. Alur Penelitian

Alur penelitian yang dilakukan adalah mengumpulkan *literature*. Melakukan uji coba pada algoritma tersebut untuk menghasilkan *average waiting time* pada algoritma penjadwalan *Shortest Remaining First* dan *Round Robin*. Jumlah proses data antrean dilakukan secara dinamis yang akan diproses oleh kedua algoritma tersebut, data antrean yang akan diproses oleh masing-masing kedua algoritma tersebut ditentukan berdasarkan *waiting time* yang telah ditentukan, sehingga proses data antrean diproses secara dinamis yang akan diharapkan mendapatkan nilai *average waiting time* antar kedua algoritma tersebut.

2.3. Penentuan Proses Data Antrian

Proses data antrean akan dilakukan secara berurutan oleh kedua algoritma penjadwalan tersebut yaitu algoritma *Shortest Remaining First* dan *Round Robin*. Berikut dapat dijelaskan proses data antrean yang dilakukan oleh kedua algoritma penjadwalan tersebut:

1. Proses Data *Shortest Remaining First* (SRF)
   Pada algoritma ini seluruh proses yang berada pada *ready queue* akan diproses berdasarkan *burst time* yang terkecil. Hal ini menyebabkan setiap proses akan memiliki *waiting time* yang singkat. Dengan setiap proses yang

memiliki *waiting time* yang singkat, maka *average waiting time* juga akan menjadi semakin pendek. Oleh karena itu, algoritma ini dapat dikatakan sebagai algoritma yang optimal [10].

2. Proses Data Antrean *Round Robin* (RR)
   Proses data antrean pada algoritma penjadwalan *Round Robin* merupakan algoritma yang menggunakan sistem *time sharing*, dimana setiap proses data yang mengantre pada algoritma ini akan mendapatkan bagian di proses berdasarkan *quantum time* yaitu jika proses data antrean melewati dari *quantum time* maka proses tersebut akan melakukan antrean lagi dari awal [14].
3. Menghitung *Average Waiting Time* (AWT)
   Pada proses perhitungan *waiting time* ini, setiap proses data antrean yang diproses oleh ketiga algoritma tersebut akan dihitung masing-masing *waiting time* dan total *waiting time*-nya yang dipengaruhi oleh *arrival time* dan *burst time* pada suatu data antrean tersebut. Proses dalam mencari *waiting time* pada suatu data antrean adalah waktu eksekusi dikurangi dengan waktu tiba, sedangkan dalam mencari *average waiting time* pada suatu data antrean adalah dengan cara menjumlahkan seluruh *waiting time* yang ada pada seluruh proses data antrean dan dibagi dengan seluruh jumlah proses data antrean.

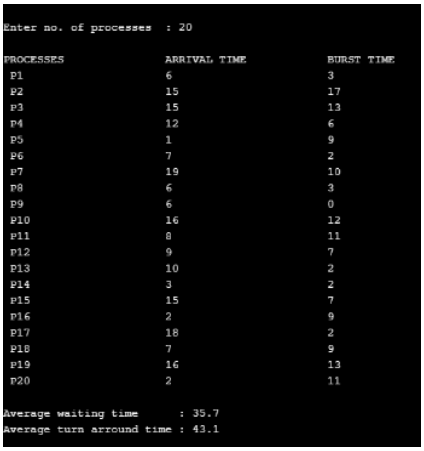
2.4. *Pseudocode* Algoritma SRF

Langkah – langkah *source code* dalam mencari AWT pada algoritma SRF yaitu:

** Traverse until all process gets completely executed.*
> Find process with minimum remaining time at every single time lap.
> Reduce its time by 1.
> Check if its remaining time becomes 0
> Increment the counter of process completion.
> Completion time of current process = current_time +1;
> Calculate waiting time for each completed process.
wt[i]= Completion time - arrival_time-burst_time
>Increment time lap by one.
** Find turnaround time (waiting_time+burst_time).*

2.5. Tampilan Program Algoritma SRF

Pada tampilan program Algoritma RR menampilkan data 20 proses, yaitu dari P1 hingga P20, beserta waktu kedatangan *(Arrival Time)* dan waktu burst *(Burst Time)* masing-masing proses. Waktu kedatangan menunjukkan kapan proses masuk ke dalam sistem, sedangkan waktu *burst* menunjukkan durasi waktu yang dibutuhkan proses untuk dieksekusi oleh CPU. Setelah data proses diolah menggunakan suatu algoritma SRF, diperoleh hasil perhitungan *Average Waiting Time* (AWT) atau rata-rata waktu tunggu: 35.7 satuan waktu. AWT adalah rata-rata waktu yang dihabiskan setiap proses dalam antrean sebelum dieksekusi. Semakin kecil nilai AWT, semakin efisien algoritma penjadwalan dalam meminimalkan waktu tunggu proses. Kemudian juga diperoleh hasil *Average Turnaround Time* (ATT): 43.1 satuan waktu. ATT adalah rata-rata total waktu yang dibutuhkan setiap proses, mulai dari saat kedatangan hingga selesai dieksekusi. AAT mencakup waktu tunggu dan waktu *burst*. Semakin kecil nilai AAT, semakin efisien algoritma penjadwalan dalam menyelesaikan proses secara keseluruhan.



```
Enter no. of processes  : 20

PROCESSES               ARRIVAL TIME            BURST TIME
 P1                     6                       3
 P2                     15                      17
 P3                     15                      13
 P4                     12                      6
 P5                     1                       9
 P6                     7                       2
 P7                     19                      10
 P8                     6                       3
 P9                     6                       0
 P10                    16                      12
 P11                    8                       11
 P12                    9                       7
 P13                    10                      2
 P14                    3                       2
 P15                    15                      7
 P16                    2                       9
 P17                    18                      2
 P18                    7                       9
 P19                    16                      13
 P20                    2                       11

Average waiting time      : 35.7
Average turn arround time : 43.1
```

Gambar 2. Tampilan program algoritma SRF

2.6. *Pseudocode* Algoritma RR

Langkah-langkah *source code* dalam mencari AWT pada algoritma RR yaitu:

CPU scheduler picks the process from the circular/ready queue, set a timer to interrupt it after 1 time slice / quantum and dispatches it .

** If process has burst time less than 1 time slice/quantum*
> Process will leave the CPU after the completion
> CPU will proceed with the next process in the ready queue / circular queue .
else If process has burst time longer than 1 time slice/quantum
> Timer will be stopped. It cause interruption to the OS .
> Executed process is then placed at the tail of the circular / ready querue by applying the context switch.
> CPU scheduler then proceeds by selecting the next process in the ready queue.

2.7. Tampilan Program Algoritma RR

Pada tampilan program Algoritma RR menampilkan data 20 proses, yaitu dari P1 hingga P20, beserta waktu kedatangan *(Arrival Time)* dan waktu burst *(Burst Time)* masing-masing proses. Waktu kedatangan menunjukkan kapan proses masuk ke dalam sistem, sedangkan waktu *burst* menunjukkan durasi waktu yang dibutuhkan proses untuk dieksekusi oleh CPU. Setelah data proses diolah menggunakan algoritma RR, diperoleh hasil perhitungan *Average Waiting Time* (AWT) atau rata-rata waktu tunggu: 112.55 satuan waktu. AWT adalah rata-rata waktu yang dihabiskan setiap proses dalam antrean sebelum dieksekusi. Semakin kecil nilai AWT, semakin efisien algoritma penjadwalan dalam meminimalkan waktu tunggu proses. Kemudian juga diperoleh hasil *Average Turnaround Time* (ATT): 122.2 satuan waktu. ATT adalah rata-rata total waktu yang dibutuhkan setiap proses, mulai dari saat kedatangan hingga selesai dieksekusi. AAT mencakup waktu tunggu dan waktu *burst*. Semakin kecil nilai AAT, semakin efisien algoritma penjadwalan dalam menyelesaikan proses secara keseluruhan.

```
Enter Number Of Process: 20
Quantum: 3
Processes  Burst time  Waiting time  Turn around time
 1              6          53             59
 2              15         151            166
 3              15         154            169
 4              12         135            147
 5              1          12             13
 6              7          107            114
 7              19         174            193
 8              6          71             77
 9              6          74             80
 10             16         172            188
 11             8          114            122
 12             9          116            125
 13             10         144            154
 14             3          37             40
 15             15         163            178
 16             2          43             45
 17             18         173            191
 18             7          128            135
 19             16         176            192
 20             2          54             56
Average waiting time = 112.55
Average turn around time = 122.2
```

Gambar 3. Tampilan program algoritma RR

3. HASIL DAN PEMBAHASAN

Pada penelitian ini yang menjadi *input* adalah jumlah proses, *arrival time* dan *burst time* dari tiap proses. Cara *input* proses adalah dengan melakukan pemanggilan *file* data antrean yang sudah ditentukan proses_id, *arrival time* dan *burst time* yang telah di simpan dalam *storage*. Jika *file* proses telah di *input*, maka proses-proses tersebut akan dikerjakan dengan menggunakan algoritma *Shortest Remaining First* dan didapatkan *waiting time* dari setiap proses yang dikerjakan. Setelah itu data tersebut digunakan kembali dengan menggunakan algoritma *Round Robin* untuk menghitung *waiting time* dari setiap proses. Hal ini dilakukan sampai semua proses benar-benar selesai dilayani oleh CPU. Setelah semua proses selesai dilayani oleh CPU, maka akan ditampilkan proses data-data antrian yang dikerjakan dari kedua algoritma penjadwalan tersebut, *waiting time* pada setiap proses dan AWT seluruh proses.

3.1. Uji Coba Dengan 20 Data

Pada pengujian yang dilakukan penulis dengan parameter *input* adalah jumlah proses sebanyak 20 proses, *arrival time* dan *burst time* dari tiap proses dengan menggunakan proses antrean *single queue* yang dikerjakan oleh kedua algoritma penjadwalan. Proses yang akan dieksekusi oleh CPU adalah seperti pada Tabel 1 berikut ini :

Tabel 1. Data antrean dengan jumlah 20 proses

Proses_ID	***Arrival Time***	***Burst Time***
P_1	0	8
P_2	2	7
P_3	5	10
P_4	11	6

Proses_ID	*Arrival Time*	*Burst Time*
P_5	16	4
P_6	19	9
P_7	25	5
P_8	30	9
P_9	30	7
P_10	30	5
P_11	30	4
P_12	34	9
P_13	36	11
P_14	39	5
P_15	43	8
P_16	51	9
P_17	58	6
P_18	58	8
P_19	58	9
P_20	58	12

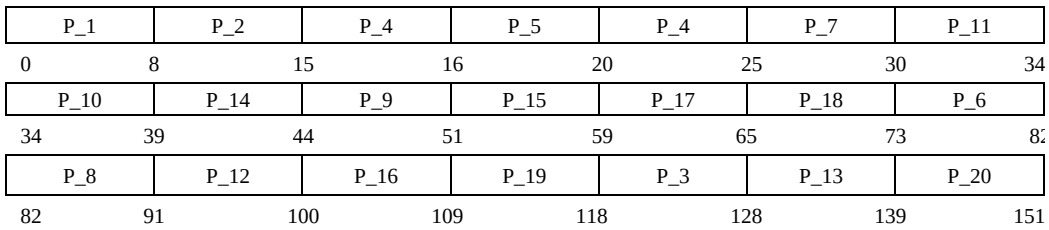
Dari tabel di atas, akan dilakukan proses data antrean untuk mencari *average waiting time* pada setiap masing-masing atau ketiga algoritma penjadwalan tersebut dengan menggunakan *single queue*. Disini dapat dijelaskan beberapa proses data antrean, dimana *arrival time* adalah waktu ketika proses berada di memori utama sebelum proses tersebut dikerjakan oleh CPU. *Burst Time* mengacu pada alokasi lamanya waktu eksekusi yang telah dialokasikan kepada masing-masing proses sejak proses itu dibuat. Dalam sistem komputer, setiap proses memerlukan waktu untuk dieksekusi oleh CPU. *Bust Time* mengukur lamanya waktu tersebut [15]. Setelah dari penjelasan di atas, maka proses data antrean dari masing-masing kedua algoritma penjadwalan tersebut.

3.2. Hasil Pengujian 20 Proses Dengan Algoritma *Shortest Remaining First*

Pada algoritma SRF, seluruh proses yang berada pada *ready queue* akan diselesaikan berdasarkan proses yang memiliki *burst time* terkecil. Dikarenakan algoritma SRF menyelesaikan suatu proses berdasarkan proses yang memiliki *burst time* terkecil, maka hal ini akan berdampak pada *waiting time* yang singkat pada seluruh prosesnya. Dengan *waiting time* yang singkat dari setiap proses, maka *average waiting time* pada algoritma SRF juga menjadi singkat. Hal ini membuat algoritma SRF dapat dikatakan salah satu algoritma yang optimal.

Hal yang pertama yang harus dilakukan adalah menggambarkan kronologi eksekusi proses pada Table 1 di atas dengan *Gantt Chart*. Mengenai hasil *Gant Chart* dari Tabel 1 di atas dapat dijelaskan proses data antrean satu per satu. Pada saat CPU tidak mengerjakan sesuatu atau dalam posisi 0 datang sebuah proses P1 yang membutuhkan waktu penyelesaian atau disebut *burst time* yang berjumlah 8, karena algoritma penjadwalan SRF merupakan algoritma yang melalukan proses berdasarkan *burst time* terkecil, maka proses selanjutnya ialah mengecek *burst time* P2, P3, P4 dan seterusnya jika sudah tiba. Jika *burst time* pada proses P2, P3, P4 dan seterusnya lebih kecil dari *burst time* P1 yang sedang diproses, maka proses yang memiliki *burst time* terkecil harus diproses terlebih dahulu dan P1 masuk ke status menunggu sampai dilayani. Jika semua proses telah selesai dikerjakan, maka langkah selanjutnya ialah menghitung waktu *average waiting time* dari seluruh proses yang telah dikerjakan.

1. *Gantt Chart*



Gambar 4. *Gantt chart* 20 data dengan algoritma *shortest remaining first*

2. *Average Waiting Time*

Hal yang dilakukan untuk menghitung *average waiting time* adalah dengan menghitung berapa lama waktu tunggu atau *waiting time* yang dihasilkan oleh proses data antrean tersebut dan dibagi dari jumlah seluruh proses. Untuk menghitung *waiting time* dapat dilakukan dengan cara mengurangi waktu eksekusi dengan waktu datang. Hasil *waiting time* dari setiap proses data antrean pada Gambar 4 adalah seperti tabel di bawah ini:

Tabel 2. *Waiting time* dengan *shortest remaining first*

Proses	*Waiting Time*(ms)
P_1	0
P_2	6
P_3	113
P_4	8
P_5	0
P_6	54
P_7	0
P_8	52
P_9	14
P_10	4
P_11	0
P_12	57
P_13	92
P_14	0
P_15	8
P_16	49
P_17	1
P_18	7
P_19	51
P_20	81

Dari tabel 2. di atas maka *average waiting time* (AWT) dapat dihitung yaitu :

AWT = (WT P_1 + WT P_2 + WT P_3 + WT P_4 + WT P_5 + WT P_6 + WT P_7 + WT P_8 + WT P_9 + WT P_10 + WT P_11 + WT P_12 + WT P_13 + WT P_14 + WT P_15 + WT P_16 + WT P_17 + WT P_18 + WT P_19 + WT P_20) / 20

= (0 + 6 + 113 + 8 + 0 + 54 + 0 + 52 + 14 + 4 + 0 + 57 + 92 + 0 + 8 + 49 + 1 + 7 + 51 + 81) / 20

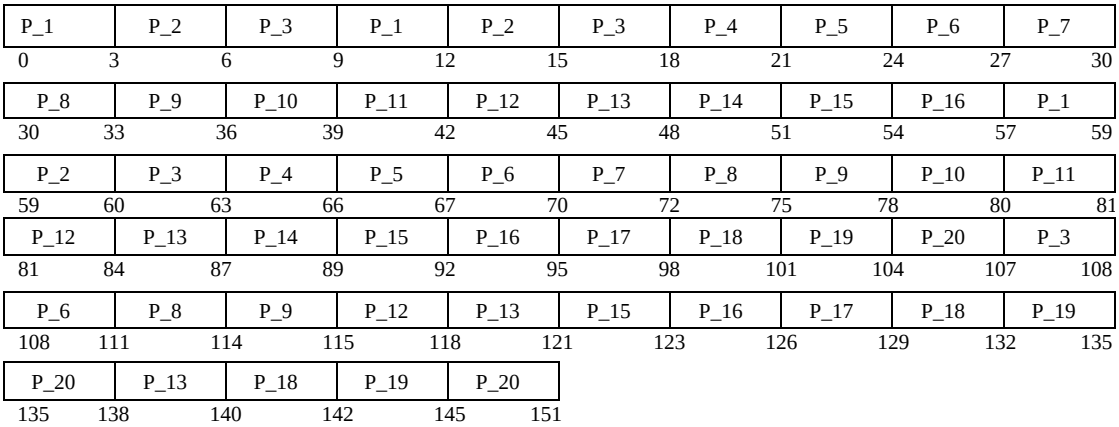
= **29.85ms**

Pada hasil waktu tunggu rata-rata atau *average waiting time* di atas didapat dengan cara menjumlahkan seluruh *waiting time* pada proses P1 sampai proses P20 dan dibagikan dengan jumlah seluruh proses data antrean.

3.3. Hasil Pengujian 20 Proses Dengan Algoritma RR

Pada metode algoritma *Round Robin* melakukan proses antrean yaitu metode yang setiap prosesnya memperoleh alokasi waktu CPU dalam *quantum time* yang telah diketahui. Pada pengujian algoritma penjadwalan *Round Robin* ini *quantum time* yang digunakan sebesar 3 ms. Hal yang pertama yang harus dilakukan adalah menggambarkan kronologi eksekusi proses pada Tabel 2 di atas dengan *Gantt Chart*. Berikut adalah hasil pengujian dari parameter pengujian *Round Robin* dengan *quantum time* yang telah diketahui 3 ms berdasarkan Tabel 2 di atas:

1. Gantt Chart



Gambar 5. *Gantt chart* 20 data dengan algoritma *Round Robin*

Mengenai hasil *gantt chart* di atas dapat dijelaskan proses data antrean satu per satu. Dari *Gantt Chart* di atas terlihat bahwa setiap proses dikerjakan berdasarkan *quantum time* yang telah ditentukan sebesar 3 ms. Pertama P1 akan dikerjakan sebanyak 3 langkah, kemudian P2 sebanyak 3 langkah dan begitu selanjutnya sampai P20. Proses data antrean yang sudah diproses menurut waktu yang diberikan, maka akan kembali menunggu dan berada pada proses data antrean paling belakang. Pada P1 *start time* adalah 0 dan *arrival time* 0, setelah itu dikerjakan proses data antrean P1 dengan nilai *quantum time* sebesar 3 ms dan *burst time* 8. Selanjutnya, *burst time* pada proses data antrean P1 dikurangkan sebesar 3 ms berdasarkan *quantum time* yang telah ditentukan dan *burst time* pada P1 menjadi sebesar 5,maka proses P1 berhenti pada posisi 3. Selanjutnya, akan dikerjakan proses data antrean berikutnya berdasarkan nilai *arrival time* harus lebih kecil atau sama dengan nilai posisi P1,

maka proses data antrean yang dikerjakan berikutnya adalah P2 dengan *start time* 3, *arrival time* 2 dan *burst time* 7. Setelah itu *burst time* pada P2 dikurangkan sebesar 3 ms dan *burst time* pada P2 menjadi sebesar 4 ms, maka proses P2 berhenti pada posisi 6 dan begitu seterusnya proses yang akan dikerjakan pada P3 sampai P20 selesai di proses. Selanjutnya, hal yang dilakukan adalah menghitung *average waiting time*.

2. *Average Waiting Time*(AWT)

Hal yang dilakukan untuk menghitung *average waiting time* adalah dengan menghitung berapa lama waktu tunggu atau *waiting time* yang dihasilkan oleh proses data antrian tersebut dan dibagi dari jumlah seluruh proses. Untuk menghitung *waiting time* dapat dilakukan dengan cara mengurangi waktu eksekusi dengan waktu datang. Hasil *waiting time* dari setiap proses data antrian pada Gambar 5 adalah seperti tabel berikut ini :

Tabel 3. *Waiting time* dengan *round robin*

Proses	Waiting Time(ms)
P_1	51
P_2	51
P_3	93
P_4	49
P_5	47
P_6	83
P_7	42
P_8	75
P_9	78
P_10	45
P_11	47
P_12	75
P_13	93
P_14	45
P_15	72
P_16	66
P_17	65
P_18	76
P_19	78
P_20	81

Dari Tabel 3 di atas maka *average waiting time* (AWT) dapat dihitung yaitu :

AWT = (WT P_1 + WT P_2 + WT P_3 + WT P_4 + WT P_5 + WT P_6 + WT P_7 + WT P_8 + WT P_9 + WT P_10 + WT P_11 + WT P_12 + WT P_13 + WT P_14 + WT P_15 + WT P_16 + WT P_17 + WT P_18 + WT P_19 + WT P_20) / 20

= (51 + 51 + 93 + 49 + 47 + 83 + 42 + 75 + 78 + 45 + 47 + 75 + 93 + 45 + 72 + 66 + 65 + 76 + 78 + 81) / 20

= 1312 / 20

= **65.6ms**

Pada hasil waktu tunggu rata-rata atau *average waiting time* di atas didapat dengan cara menjumlahkan seluruh *waiting time* pada proses P1 sampai proses P20 dan dibagikan dengan jumlah seluruh proses data antrean.

3.4. Hasil Pengujian 35 Proses

Pada pengujian berikutnya peneliti melakukan pengujian kembali dengan jumlah proses data antrean 35 proses. Pada pengujian yang dilakukan penulis dengan parameter *input* adalah jumlah proses sebanyak 35 proses, *arrival time* dan *burst time* dari tiap proses dengan menggunakan proses antrian *single queue* yang dikerjakan oleh ketiga algoritma penjadwalan. Proses yang akan dieksekusi oleh CPU adalah seperti pada Tabel 4 berikut ini :

Tabel 4. Uji coba 35 data antrean

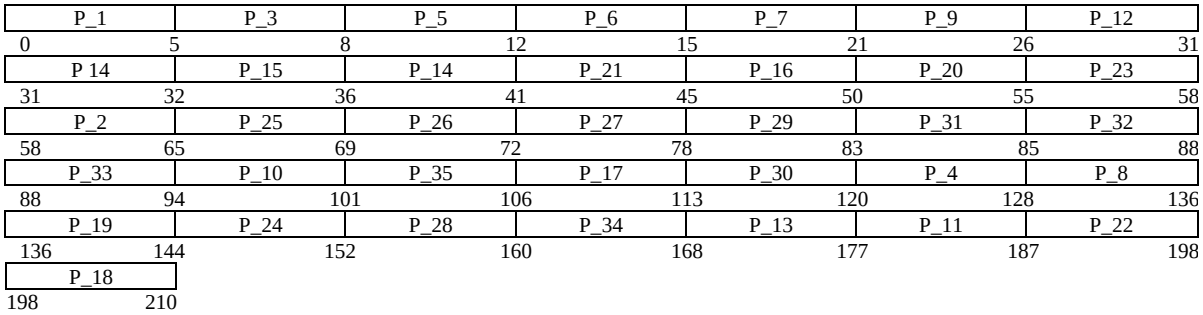
Proses ID	Arrival Time	BurstTime
P_1	0	5
P_2	2	7
P_3	4	3
P_4	6	8
P_5	7	4
P_6	9	3
P_7	12	6
P_8	15	8
P_9	17	5
P_10	19	7
P_11	21	10
P_12	25	5
P_13	27	9
P_14	29	6
P_15	32	4
P_16	33	5
P_17	38	7

Proses ID	Arrival Time	BurstTime
P_18	40	12
P_19	40	8
P_20	40	5
P_21	40	4
P_22	44	11
P_23	55	3
P_24	58	8
P_25	64	4
P_26	68	3
P_27	68	6
P_28	68	8
P_29	75	5
P_30	80	7
P_31	81	2
P_32	83	3
P_33	84	6
P_34	95	8
P_35	100	5

3.5. Hasil Pengujian 35 Proses Dengan Algoritma *Shortest Remaining First*

Pada algoritma SRF, seluruh proses yang berada pada *ready queue* akan diselesaikan berdasarkan proses yang memiliki *burst time* terkecil. Dikarenakan setiap proses diselesaikan berdasarkan proses yang memiliki *burst time* terkecil, maka *waiting time* setiap proses akan menjadi singkat dan hal ini juga menyebabkan *average waiting time* menjadi singkat. Oleh karena itu, algoritma ini dapat dikatakan salah satu algoritma yang optimal. Hal yang pertama yang harus dilakukan adalah menggambarkan kronologi eksekusi proses pada Tabel 4 di atas dengan *Gantt Chart*. Berikut adalah hasil pengujian dari parameter pengujian berdasarkan Tabel 4 di atas :

1. *Gant Chart*



Gambar 6. *Gant chart* 35 data dengan algoritma *shortest remaining first*

Mengenai hasil *gantt chart* di atas dapat dijelaskan proses data antrean satu per satu. Pada saat CPU tidak mengerjakan sesuatu atau dalam posisi 0 datang sebuah proses P1 yang membutuhkan waktu penyelesaian atau disebut *burst time* yang berjumlah 8, karena algoritma penjadwalan *SRF* merupakan algoritma yang melakukan proses berdasarkan *burst time* terkecil, maka proses selanjutnya ialah mengecek *burst time* P2, P3, P4 dan seterusnya jika sudah tiba. Jika *burst time* pada proses P2, P3, P4 dan seterusnya lebih kecil dari *burst time* P1 yang sedang diproses, maka proses yang memiliki *burst time* terkecil harus diproses terlebih dahulu dan P1 masuk ke status menunggu sampai dilayani. Jika semua proses telah selesai dikerjakan, maka langkah selanjutnya ialah menghitung waktu *average waiting time* dari seluruh proses yang telah dikerjakan.

2. *Average Waiting Time*

Hal yang dilakukan untuk menghitung *average waiting time* adalah dengan menghitung berapa lama waktu tunggu atau *waiting time* yang dihasilkan oleh proses data antrean tersebut dan dibagi dari jumlah seluruh proses. Untuk menghitung *waiting time* dapat dilakukan dengan cara mengurangi waktu eksekusi dengan waktu datang. Hasil *waiting time* dari setiap proses data antrean pada gambar 6 adalah seperti tabel di bawah ini :

Tabel 5. *Waiting time* dengan *shortest remaining first*

Proses	*Waiting Time*(ms)
P_1	0
P_2	56
P_3	1
P_4	114
P_5	1
P_6	3
P_7	3
P_8	113
P_9	4
P_10	75

Proses	*Waiting Time*(ms)
P_11	156
P_12	1
P_13	141
P_14	6
P_15	0
P_16	12
P_17	68
P_18	158
P_19	96
P_20	10
P_21	1
P_22	1
P_23	4
P_24	84
P_25	3
P_26	33
P_27	2
P_28	2
P_29	3
P_30	33
P_31	2
P_32	2
P_33	4
P_34	65
P_35	1

Dari tabel 4.5 diatas maka *average waiting time* (AWT) dapat dihitung yaitu :
AWT = (WT P_1 + WT P_2 + WT P_3 + WT P_4 + WT P_5 + WT P_6 + WT P_7 + WT P_8 + WT P_9 + WT P_10 + WT P_11 + WT P_12 + WT P_13 + WT P_14 + WT P_15 + WT P_16 + WT P_17 + WT P_18 + WT P_19 +WT P_20) / 20
= (0 + 56 + 1 + 114 + 1+ 3 + 3 + 113 + 4 + 75 + 156 + 1 + 141 + 6 + 0 + 49 + 12 + 68 + 158 + 96 + 10 + 1 + 143 + 0 + 86 + 1 + 1 + 4 +84 + 3 + 33 + 2 + 2 + 4 + 65 + 1) / 35
= **41.3714 ms**

Pada hasil waktu tunggu rata-rata atau *average waiting time* di atas didapat dengan cara menjumlahkan seluruh *waiting time* pada proses P1 sampai proses P35 dan dibagikan dengan jumlah seluruh proses data antrean.

3.6 Hasil Pengujian 35 Proses dengan algoritma Round Robin

Pada metode algoritma *Round Robin* melakukan proses antrean yaitu metode yang setiap prosesnya memperoleh alokasi waktu CPU dalam *quantum time* yang telah diketahui. Pada pengujian algoritma penjadwalan *Round Robin* ini *quantum time* yang digunakan sebesar 3 ms. Hal yang pertama yang harus dilakukan adalah menggambarkan kronologi eksekusi proses pada Tabel 5 di atas dengan *Gant Chart*. Berikut adalah hasil pengujian dari parameter pengujian *Round Robin* dengan *quantum time* yang telah diketahui 3 ms berdasarkan Tabel 5 di atas :

1. *Gant Chart*

P_1	P_2	P_3	P_4	P_5	P_6	P_7
0	3	6	9	12	15	18	21

P_8	P_9	P_10	P_11	P_12	P_13	P_14
21	24	27	30	33	36	39	42

P_15	P_16	P_17	P_18	P_19	P_20	P_21
42	45	48	51	54	57	60	63

P_22	P_23	P_24	P_25	P_26	P_27	P_28
63	66	69	72	75	78	81	84

P_29	P_30	P_31	P_32	P_33	P_34	P_35
84	87	90	92	95	98	101	104

P_1	P_2	P_4	P_5	P_7	P_8	P_9
104	106	109	112	113	116	119	121

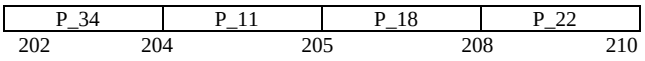
P_10	P_11	P_12	P_13	P_14	P_15	P_16
121	124	127	129	132	135	136	138

P_17	P_18	P_19	P_20	P_21	P_22	P_24
138	141	144	147	149	150	153	156

P_25	P_27	P_28	P_29	P_30	P_33	P_34
156	157	160	163	165	168	171	174

P_35	P_2	P_4	P_8	P_10	P_11	P_13
174	176	177	179	181	182	185	188

P_17	P_18	P_19	P_22	P_24	P_28	P_30
188	189	192	194	197	199	201	202



Gambar 7. *Gant chart* 35 data dengan algoritma *round robin*

Mengenai hasil *Gant Chart* diatas dapat dijelaskan proses data antrian satu per satu. Dari *Gant chart* diatas terlihat bahwa setiap proses dikerjakan berdasarkan *quantum time* yang telah ditentukan sebesar 3 ms. Pertama P_1 akan dikerjakan sebanyak 3 langkah, kemudian P_2 sebanyak 3 langkah dan begitu selanjutnya sampai P_20. Proses data antrian yang sudah diproses menurut waktu yang diberikan, maka akan kembali menunggu dan berada pada proses data antrian paling belakang. Pada P_1 *start time* adalah 0 dan *arrival time* 0, setelah itu dikerjakan proses data antrian P_1 dengan nilai *quantum time* sebesar 3 ms dan *burst time* 5 dan selanjutnya *burst time* pada proses data antrian P_1 dikurangkan sebesar 3 ms berdasarkan *quantum time* yang telah ditentukan dan *burst time* pada P_1 menjadi sebesar 2, maka proses P_1 berhenti pada posisi 3. Selanjutnya, akan dikerjakan proses data antrian berikutnya berdasarkan nilai *arrival time* harus lebih kecil atau sama dengan nilai posisi P_1, maka proses data antrian yang dikerjakan berikutnya adalah P_2 dengan *start time* 3, *arrival time* 2 dan *burst time* 7. Setelah itu *burst time* pada P_2 dikurangkan sebesar 3 ms dan *burst time* pada P_2 menjadi sebesar 4 ms, maka proses P_2 berhenti pada posisi 6 dan begitu seterusnya proses yang akan dikerjakan pada P_3 sampai P_35 selesai di proses. Selanjutnya, hal yang dilakukan adalah menghitung *average waiting time*.

2. ***Average Waiting Time* (AWT)**

Hal yang dilakukan untuk menghitung *average waiting time* adalah dengan menghitung berapa lama waktu tunggu atau *waiting time* yang dihasilkan oleh proses data antrean tersebut dan dibagi dari jumlah seluruh proses. Untuk menghitung *waiting time* dapat dilakukan dengan cara mengurangi waktu eksekusi dengan waktu datang. Hasil *waiting time* dari setiap proses data antrean pada Gambar 7 adalah seperti tabel berikut ini :

Tabel 6. *Waiting time* dengan *round robin*

Proses	***Waiting Time* (ms)**
P_1	101
P_2	106
P_3	2
P_4	165
P_5	102
P_6	6
P_7	98
P_8	158
P_9	99
P_10	156
P_11	174
P_12	99
P_13	152
P_14	100
P_15	100
P_16	100
P_17	144
P_18	156
P_19	146
P_20	104
P_21	106
P_22	155
P_23	11
P_24	133
P_25	89
P_26	7
P_27	86
P_28	125
P_29	85
P_30	115
P_31	9
P_32	9
P_33	81
P_34	101
P_35	71

Dari tabel 1.6 diatas maka *average waiting time* (AWT) dapat dihitung yaitu :

AWT = (WT P_1 + WT P_2 + WT P_3 + WT P_4 + WT P_5 + WT P_6 + WT P_7 + WT P_8 + WT P_9 + WT P_10 + WT P_11 + WT P_12 + WT P_13 + WT P_14 + WT P_15 +WT P_16 + WT P_17 + WT P_21 + WT P_20 + WT P_19 + WT P_18 + WT P_22 + WT P_23 + WT P_24 + WT P_25 + WT P_26 + WT P_27 + WT P_28 + WT P_29 + WT P_30 + WT P_31 + WT P_32 + WT P_33 + WT P_34 + WT P_35) / 35

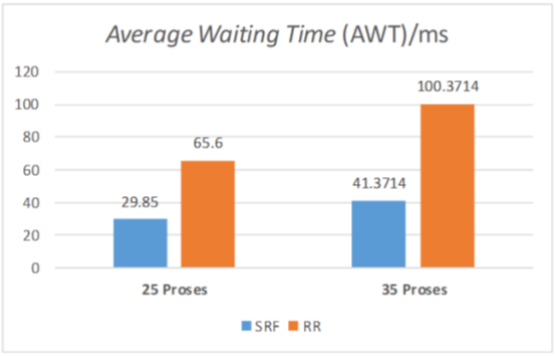
= (101 + 168 + 2 + 165 + 102 + 6 + 98 + 158 + 99 + 156 + 174 + 99 + 152 + 100 + 100 + 100 + 144 + 156 + 146 + 104 + 106 + 155 + 11 + 133 + 89 + 7

+ 86 + 125 + 85 + 115 + 9 + 9 + 81 + 101 + 71) / 35
= 3513 / 35
= 100.3714 ms

Pada hasil waktu tunggu rata-rata atau *average waiting time* diatas didapat dengan cara menjumlahkan seluruh *waiting time* pada proses P_1 sampai proses P_35 dan dibagikan dengan jumlah seluruh proses data antrian.

4. KESIMPULAN

Dari hasil percobaan menunjukkan bahwa AWT pada algoritma SRF lebih baik dibandingkan algoritma RR. Hal ini dibuktikan dengan hasil percobaan dimana algoritma SRF mendapatkan hasil AWT 29.85 ms dan algoritma RR mendapatkan hasil AWT 65.6 ms.



Gambar 8. Hasil AWT Pada Algoritma SRF dan RR

REFERENSI

[1] P. Tri Dharma, P. Rakhmat, "Analisis Algoritma Round Robin pada Penjadwalan CPU", Jurnal Ilmiah Teknologi Informasi Asia, Vol. 15, No. 2 Tahun 2021.
[2] N. Nuraini and I. Ahmad, "Sistem Informasi Manajemen Kepegawaian Menggunakan Metode Key Performance Indicator Untuk Rekomendasi Kenaikan Jabatan (Studi Kasus: Kejaksaan Tinggi Lampung),"J. Teknol. dan Sist. Inf., vol. 2, no. 3, p. 81, 2021.
[3] I. Tasya Fitria., "Analisis Performa Algoritma Penjadwalan CPU Dalam Sistem Operasi Komputer Untuk Pengoptimalan Responsivitas", Jurnal Teknologi Pintar, Vol. 3, November 2023.
[4] O. Hoger K, J. Kamal H, H. Shalau F, Comperative, "Comparative Analysis of The Essential CPU Scheduling Algorithms", Journal Bulletin of Electrical Engineering and Informatics, Vol. 10, No. 5, PP. 2742-2750, October 2021.
[5] P. Rakhmat, P. Tri Dharma, "Median-Average Round Robin (MARR) Algorithm for Optimal CPU Task Scheduling", Jurnal dan Penelitian Teknik Informatika (Sinkron), Vol. 9, No. 1, January 2025.
[6] S. Siti, S. M Yogie, Rismayanti, "Penerapan Algoritma Shortest Job First (SJF) dan Priority Scheduling (PS) pada Maintanance Mesin ATM, Jurnal UINSU, Vol. 07, No. 1, April 2023.
[7] M. Mr. Rohit, Operating System, EduGorilla Prep Experts.
[8] H. Richki, Arsitektur dan Organisasi Komputer, Ponorogo : Uwais Inspirasi Indonesia, 2021.
[9] A. Kande, dkk, Introduction to Operating Systems, India : Nitya Publications, 2020.
[10] A. Mahdi S, A. Firas Sabah, M. Tariq A, "Reducing Waiting and Idle Time for A Group of Jobs in The Grid Computing", Journal Bulleting of Electrical Engineering and Informatics, Vol. 12, No. 5, PP. 3115-3123, October 2023.
[11] N. Ahmad Mursyidun, dkk Kernel System, Malang : Ahlimedia Press, 2021.
[12] J. Sandeep, "Shortest Remaining Time First (Preemptive SJF) Scheduling Algorithm", 03 Februari 2025, [Online]. Tersedia : https://www.geeksforgeeks.org/shortest-remaining-time-first-preemptive-sjf-scheduling-algorithm [Diakses : 25 Februari 2025].
[13] S. Agung, J. Erfian, "Sistem Penjadwalan Produksi Makanan Sei Menggunakan Algoritma Round Robin di CV Gyumbox, eProsiding Teknik Informatika (PROTEKTIF), Juni 2021, Vol. 2, No. 1.
[14] T. Raj Gaurang, A. Ambuj Kumar, K. Vikas, S. Durgesh, Design Concepts of Operating System, Delhi : Namya Press.Com, 2021.
[15] W. Wisnu, M. Desinta, A. Kezia Jazzlyn, "Implementasi Algoritma Round Robin dan Priority Pada Sistem Antrian Rumah Sakit", Jurnal Fasilkom, Vol 14, No. 2, PP. 507-513, Agustus 2024.