CPU Scheduling Alogorithms practice problems

·https://www.gatevidyalay.com/cpu-scheduling-practice-problems-numericals/

·https://www.javatpoint.com/os-scheduling-algorithms

·https://www.guru99.com/cpu-scheduling-algorithms.html

BANKERS ALGORITM

·https://www.guru99.com/bankers-algorithm-in-operating-system.html

·https://www.javatpoint.com/bankers-algorithm-in-operating-system

·https://www.gatevidyalay.com/bankers-algorithm-deadlock-avoidance-2/

PAGE REPLACEMENT ALGORITHM

- https://www.javatpoint.com/os-page-replacement-algorithms
- https://www.gatevidyalay.com/page-replacement-algorithms-page-fault/
- https://workat.tech/core-cs/tutorial/page-replacement-algorithms-in-operating-system-os-q0pioxh7iym5

PRACTICE PROBLEMS

CPU SCHEDULING ALGORITHMS

Problem-01:

Consider three process, all arriving at time zero, with total execution time of 10, 20 and 30 units respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of does the CPU remain idl

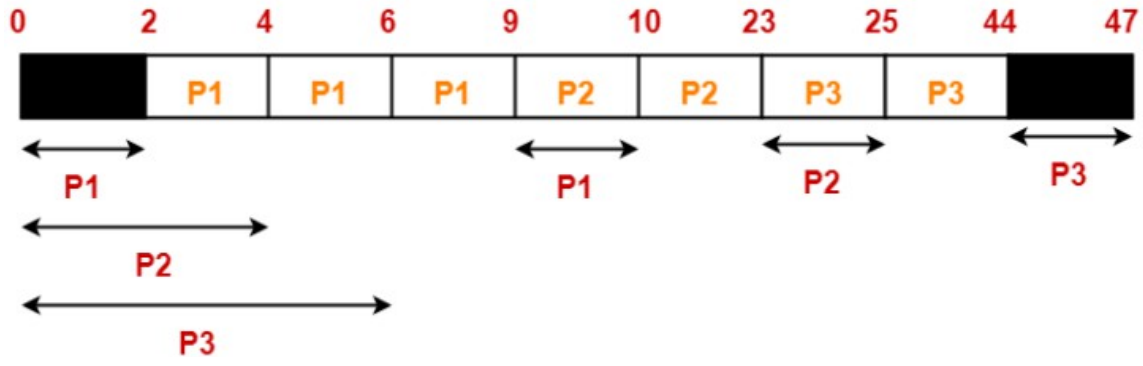
Solution-

According to question, we have-

	Total Burst Time	**I/O Burst**	**CPU Burst**	**I/O Burst**
Process P1	10	2	7	1
Process P2	20	4	14	2
Process P3	30	6	21	3

The scheduling algorithm used is Shortest Remaining Time First.

Gantt Chart-



Percentage of time CPU remains idle

= (5 / 47) x 100

= 10.638%

Problem-02:

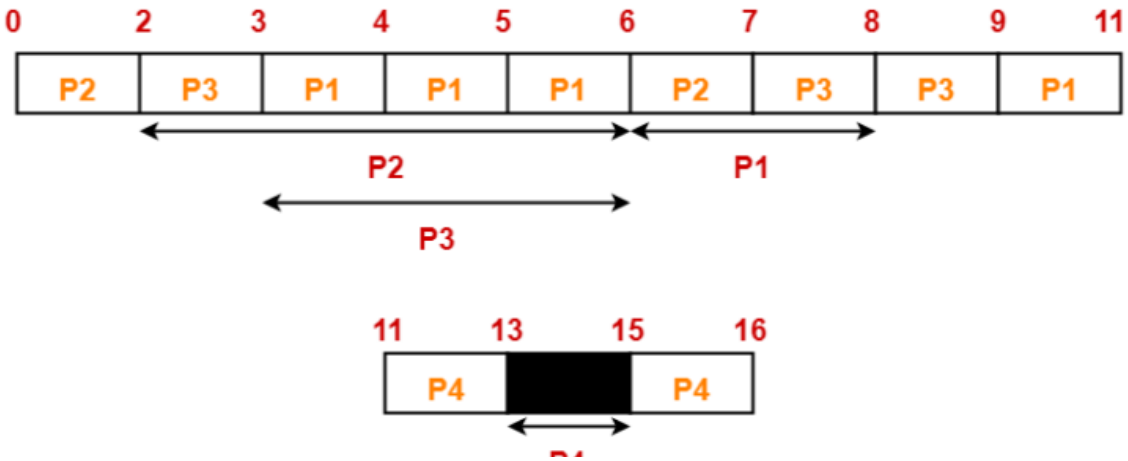
Consider the set of 4 processes whose arrival time and burst time are given below-

Process No.	**Arrival Time**	**Burst Time**		
		CPU Burst	**I/O Burst**	**CPU Burst**
P1	0	3	2	2
P2	0	2	4	1
P3	2	1	3	2
P4	5	2	2	1

If the CPU scheduling policy is Shortest Remaining Time First, calculate the average waiting time and average turn around time.

Solution-

Gantt Chart-



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	**Exit time**	**Turn Around time**	**Waiting time**
P1	11	11 – 0 = 11	11 – (3+2) = 6
P2	7	7 – 0 = 7	7 – (2+1) = 4
P3	9	9 – 2 = 7	7 – (1+2) = 4
P4	16	16 – 5 = 11	11 – (2+1) = 8

Now,

- Average Turn Around time = (11 + 7 + 7 + 11) / 4 = 36 / 4 = 9 units
- Average waiting time = (6 + 4 + 4 + 8) / 4 = 22 / 5 = 4.4 units

Problem-03:

Consider the set of 4 processes whose arrival time and burst time are given below-

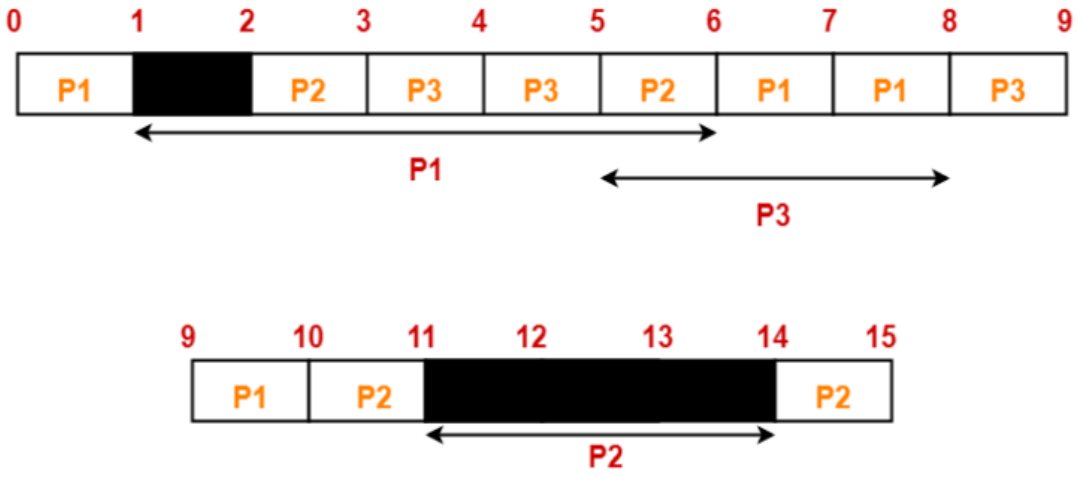
Process No.	Arrival Time	Priority	Burst Time		
			CPU Burst	I/O Burst	CPU Burst
P1	0	2	1	5	3
P2	2	3	3	3	1
P3	3	1	2	3	1

If the CPU scheduling policy is Priority Scheduling, calculate the average waiting time and average turn around time. (Lower number means higher priority)

Solution-

The scheduling algorithm used is Priority Scheduling.

Gantt Chart-



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	10	10 – 0 = 10	10 – (1+3) = 6
P2	15	15 – 2 = 13	13 – (3+1) = 9
P3	9	9 – 3 = 6	6 – (2+1) = 3

Now,

- Average Turn Around time = (10 + 13 + 6) / 3 = 29 / 3 = 9.67 units
- Average waiting time = (6 + 9 + 3) / 3 = 18 / 3 = 6 units

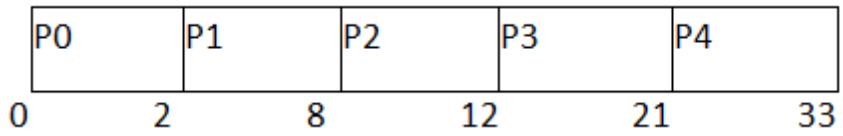
Problem-04:

In the Following schedule, there are 5 processes with process ID **P0, P1, P2, P3 and P4**. P0 arrives at time 0, P1 at time 1, P2 at time 2, P3 arrives at time 3 and Process P4 arrives at time 4 in the ready queue. The processes and their respective Arrival and Burst time are given in the following table. calculate the average waiting time and average turn around time.(FCFS)

Process ID	Arrival Time	Burst Time
0	0	2
1	1	6
2	2	4
3	3	9
4	6	12

Solution-

Gantt chart



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	10	6
3	3	9	21	18	9
4	6	12	33	29	17

Avg Waiting Time=31/5

Problem-05:

In the following Example, we are considering five processes P1, P2, P3, P4, P5 and P6. Their arrival time and Burst time are given below.

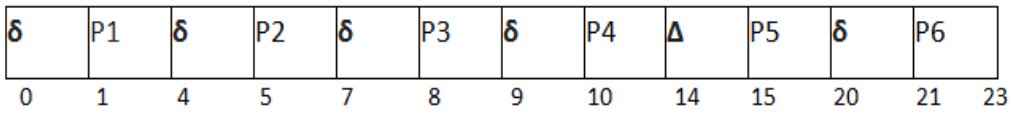
Process ID	Arrival Time	Burst Time
1	0	3
2	1	2

3	2	1
4	3	4
5	4	5
6	5	2

If the context switching time of the system is 1 unit then calculate the efficency.

Solution-

The system will take extra 1 unit of time (overhead) after the execution of every process to schedule the next process.



Inefficiency=(6/23)X100%

Efficiency=(1-6/23)X100%

Shortest Job First (SJF) Scheduling

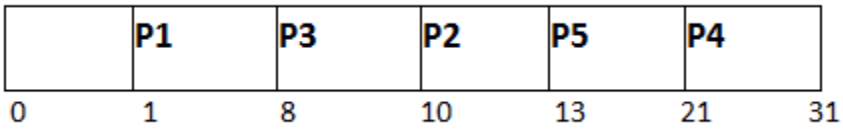
Problem-01:

In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

PID	Arrival Time	Burst Time
1	1	7
2	3	3
3	6	2
4	7	10
5	9	8

Solution-

Gantt chart



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

Avg Waiting Time = 27/5

Shortest Remaining Time First (SRTF) Scheduling Algorithm

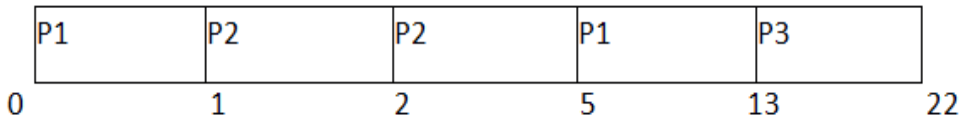
Problem-01:

Given the arrival time and burst time of 3 jobs in the table below. Calculate the Average waiting time of the system.

Process ID	Arrival Time	Burst Time
1	0	9
2	1	4
3	2	9

Solution-

Gantt chart



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	9	13	13	4
2	1	4	5	4	0
3	2	9	22	20	11

Avg Waiting Time = (4+0+11)/3 = 5 units

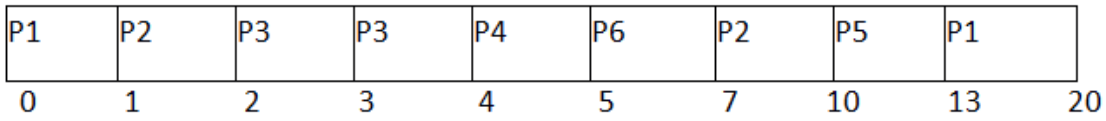
Problem02:

In this Example, there are five jobs P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table.

Process ID	Arrival Time	Burst Time
1	0	8
2	1	4
3	2	2
4	3	1
5	4	3
6	5	2

Solution:

Gantt Chart:



Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
1	0	8	20	20	12	0
2	1	4	10	9	5	1
3	2	2	4	2	0	2
4	3	1	5	2	1	4
5	4	3	13	9	6	10
6	5	2	7	2	0	5

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Avg Waiting Time = 24/6

Round Robin Scheduling Algorithm

Problem-01:

In the following example, there are six processes named as P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table. The time quantum of the system is 4 units.

Process ID	Arrival Time	Burst Time
1	0	5
2	1	6
3	2	3
4	3	1
5	4	5
6	6	4

According to the algorithm, we have to maintain the ready queue and the Gantt chart. The structure of both the data structures will be changed after every scheduling.

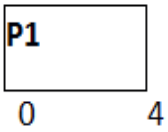
Solution-:

Ready Queue:

Initially, at time 0, process P1 arrives which will be scheduled for the time slice 4 units. Hence in the ready queue, there will be only one process P1 at starting with CPU burst time 5 units.

P1
5

GANTT chart



The P1 will be executed for 4 units first.

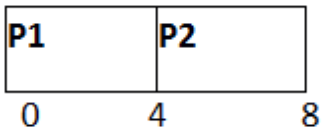
Ready Queue

Meanwhile the execution of P1, four more processes P2, P3, P4 and P5 arrives in the ready queue. P1 has not completed yet, it needs another 1 unit of time hence it will also be added back to the ready queue.

P2	P3	P4	P5	P1
6	3	1	5	1

GANTT chart

After P1, P2 will be executed for 4 units of time which is shown in the Gantt chart.



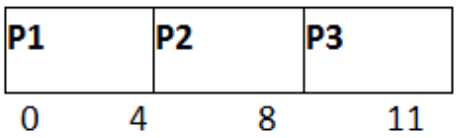
Ready Queue

During the execution of P2, one more process P6 is arrived in the ready queue. Since P2 has not completed yet hence, P2 will also be added back to the ready queue with the remaining burst time 2 units.

P3	P4	P5	P1	P6	P2
3	1	5	1	4	2

GANTT chart

After P1 and P2, P3 will get executed for 3 units of time since its CPU burst time is only 3 seconds.

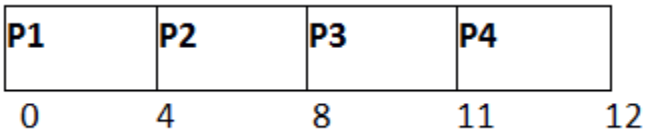


Ready Queue

Since P3 has been completed, hence it will be terminated and not be added to the ready queue. The next process will be executed is P4.

P4	P5	P1	P6	P2
1	5	1	4	2

GANTT chart



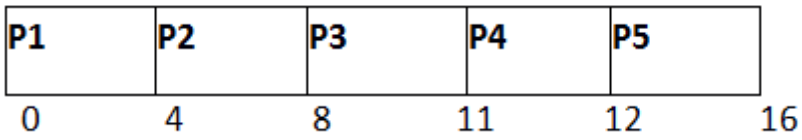
After, P1, P2 and P3, P4 will get executed. Its burst time is only 1 unit which is lesser then the time quantum hence it will be completed.

Ready Queue

The next process in the ready queue is P5 with 5 units of burst time. Since P4 is completed hence it will not be added back to the queue.

P5	P1	P6	P2
5	1	4	2

GANTT chart



P5 will be executed for the whole time slice because it requires 5 units of burst time which is higher than the time slice.

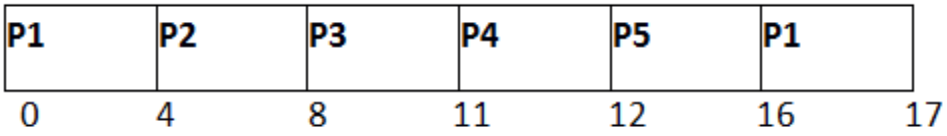
Ready Queue

P5 has not been completed yet; it will be added back to the queue with the remaining burst time of 1 unit.

P1	P6	P2	P5
1	4	2	1

GANTT Chart

The process P1 will be given the next turn to complete its execution. Since it only requires 1 unit of burst time hence it will be completed.

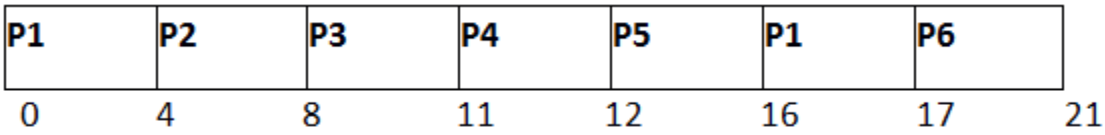


Ready Queue

P1 is completed and will not be added back to the ready queue. The next process P6 requires only 4 units of burst time and it will be executed next.

P6	P2	P5
4	2	1

GANTT chart



P6 will be executed for 4 units of time till completion.

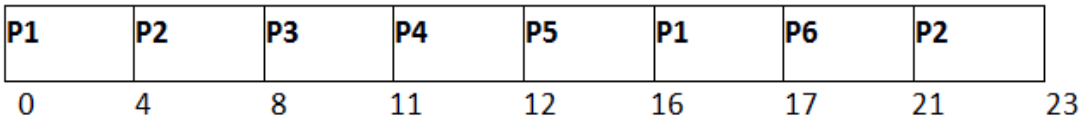
Ready Queue

Since P6 is completed, hence it will not be added again to the queue. There are only two processes present in the ready queue. The Next process P2 requires only 2 units of time.

P2	P5
2	1

GANTT Chart

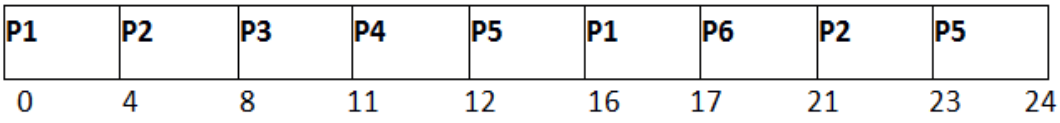
P2 will get executed again, since it only requires only 2 units of time hence this will be completed.



Ready Queue

Now, the only available process in the queue is P5 which requires 1 unit of burst time. Since the time slice is of 4 units hence it will be completed in the next burst.

P5
1



GANTT chart

P5 will get executed till completion

The completion time, Turnaround time and waiting time will be calculated as shown in the table below.

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	5	17	17	12
2	1	6	23	22	16
3	2	3	11	9	6
4	3	1	12	9	8
5	4	5	24	20	15
6	6	4	21	15	11

Avg Waiting Time = (12+16+6+8+15+11)/6 = 76/6 units

Priority Scheduling

Problem 01:-

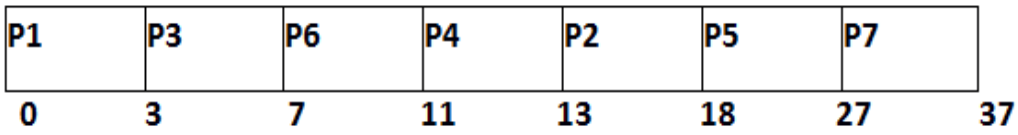
In the Example, there are 7 processes P1, P2, P3, P4, P5, P6 and P7. Their priorities, Arrival Time and burst time are given in the table.

Process ID	Priority	Arrival Time	Burst Time
1	2	0	3
2	6	2	5
3	3	1	4
4	5	4	2
5	7	6	9
6	4	5	4

7	10	7	10

Solution-

Gantt Chart



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
1	2	0	3	3	3	0	0
2	6	2	5	18	16	11	13
3	3	1	4	7	6	2	3
4	5	4	2	13	9	7	11
5	7	6	9	27	21	12	18
6	4	5	4	11	6	2	7
7	10	7	10	37	30	18	27

Avg Waiting Time = (0+11+2+7+12+2+18)/7 = 52/7 units

Problem 02-

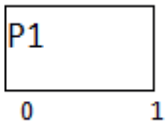
There are 7 processes P1, P2, P3, P4, P5, P6 and P7 given. Their respective priorities, Arrival Times and Burst times are given in the table below.

Process Id	Priority	Arrival Time	Burst Time
1	2(L)	0	1
2	6	1	7
3	3	2	3
4	5	3	6
5	4	4	5
6	10(H)	5	15
7	9	15	8

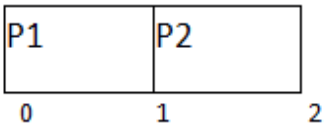
Solution-

GANTT chart Preparation

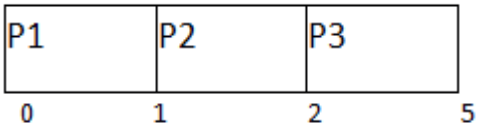
At time 0, P1 arrives with the burst time of 1 units and priority 2. Since no other process is available hence this will be scheduled till next job arrives or its completion (whichever is lesser).



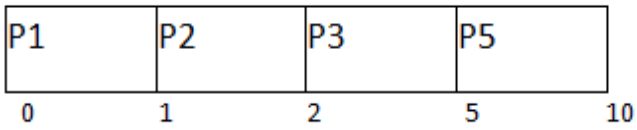
At time 1, P2 arrives. P1 has completed its execution and no other process is available at this time hence the Operating system has to schedule it regardless of the priority assigned to it.



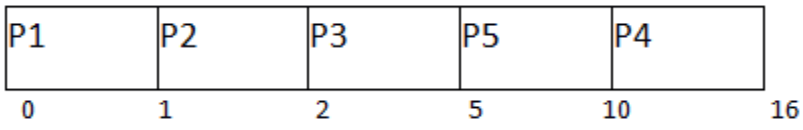
The Next process P3 arrives at time unit 2, the priority of P3 is higher to P2. Hence the execution of P2 will be stopped and P3 will be scheduled on the CPU.



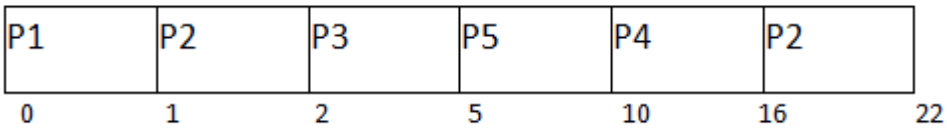
During the execution of P3, three more processes P4, P5 and P6 becomes available. Since, all these three have the priority lower to the process in execution so PS can't preempt the process. P3 will complete its execution and then P5 will be scheduled with the priority highest among the available processes.



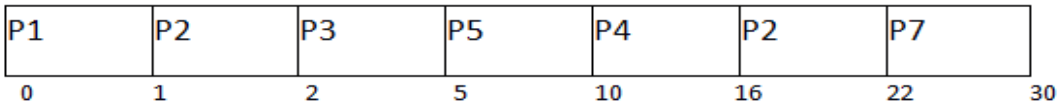
Meanwhile the execution of P5, all the processes got available in the ready queue. At this point, the algorithm will start behaving as Non Preemptive Priority Scheduling. Hence now, once all the processes get available in the ready queue, the OS just took the process with the highest priority and execute that process till completion. In this case, P4 will be scheduled and will be executed till the completion.



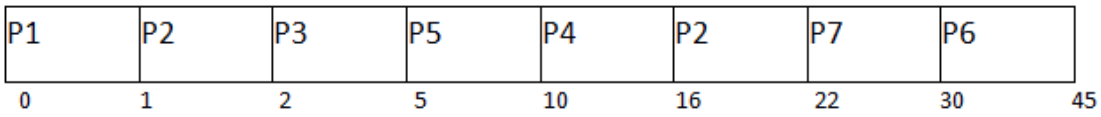
Since P4 is completed, the other process with the highest priority available in the ready queue is P2. Hence P2 will be scheduled next.



P2 is given the CPU till the completion. Since its remaining burst time is 6 units hence P7 will be scheduled after this.



The only remaining process is P6 with the least priority, the Operating System has no choice unless of executing it. This will be executed at the last.



The Completion Time of each process is determined with the help of GANTT chart. The turnaround time and the waiting time can be calculated by the following formula.

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turn around Time	Waiting Time
1	2	0	1	1	1	0
2	6	1	7	22	21	14
3	3	2	3	5	3	0
4	5	3	6	16	13	7
5	4	4	5	10	6	1
6	10	5	15	45	40	25
7	9	6	8	30	24	16

Avg Waiting Time = (0+14+0+7+1+25+16)/7 = 63/7 = 9 units

Example of Banker's algorithm

Problem01-

Consider a system that contains five processes P1, P2, P3, P4, P5 and the three resource types A, B and C. Following are the resources types: A has 10, B has 5 and the resource type C has 7 instances.

Process	Allocation A B C	Max A B C	Available A B C
P1	0 1 0	7 5 3	3 3 2
P2	2 0 0	3 2 2	
P3	3 0 2	9 0 2	
P4	2 1 1	2 2 2	
P5	0 0 2	4 3 3	

Answer the following questions using the banker's algorithm:

1. What is the reference of the need matrix?
2. Determine if the system is safe or not.
3. What will happen if the resource request (1, 0, 0) for process P1 can the system accept this request immediately?

Ans. 1: Context of the need matrix is as follows:

Need [i] = Max [i] - Allocation [i]
Need for P1: (7, 5, 3) - (0, 1, 0) = 7, 4, 3
Need for P2: (3, 2, 2) - (2, 0, 0) = 1, 2, 2
Need for P3: (9, 0, 2) - (3, 0, 2) = 6, 0, 0
Need for P4: (2, 2, 2) - (2, 1, 1) = 0, 1, 1
Need for P5: (4, 3, 3) - (0, 0, 2) = 4, 3, 1

Process	Need A B C
P1	7 4 3
P2	1 2 2
P3	6 0 0
P4	0 1 1
P5	4 3 1

Hence, we created the context of need matrix.

Ans. 2: Apply the Banker's Algorithm:

Available Resources of A, B and C are 3, 3, and 2.

Now we check if each type of resource request is available for each process.

Step 1: For Process P1:

Need <= Available

7, 4, 3 <= 3, 3, 2 condition is **false**.

So, we examine another process, P2.

Step 2: For Process P2:

Need <= Available

1, 2, 2 <= 3, 3, 2 condition **true**

New available = available + Allocation

(3, 3, 2) + (2, 0, 0) => 5, 3, 2

Similarly, we examine another process P3.

Step 3: For Process P3:

P3 Need <= Available

6, 0, 0 < = 5, 3, 2 condition is **false**.

Similarly, we examine another process, P4.

Step 4: For Process P4:

P4 Need <= Available

0, 1, 1 <= 5, 3, 2 condition is **true**

New Available resource = Available + Allocation

5, 3, 2 + 2, 1, 1 => 7, 4, 3

Similarly, we examine another process P5.

Step 5: For Process P5:

P5 Need <= Available

4, 3, 1 <= 7, 4, 3 condition is **true**

New available resource = Available + Allocation

7, 4, 3 + 0, 0, 2 => 7, 4, 5

Now, we again examine each type of resource request for processes P1 and P3.

Step 6: For Process P1:

P1 Need <= Available

7, 4, 3 <= 7, 4, 5 condition is **true**

New Available Resource = Available + Allocation

7, 4, 5 + 0, 1, 0 => 7, 5, 5

So, we examine another process P2.

Step 7: For Process P3:

P3 Need <= Available

6, 0, 0 <= 7, 5, 5 condition is true

New Available Resource = Available + Allocation

7, 5, 5 + 3, 0, 2 => 10, 5, 7

Hence, we execute the banker's algorithm to find the safe state and the safe sequence like P2, P4, P5, P1 and P3.

Ans. 3: For granting the Request (1, 0, 2), first we have to check that **Request <= Available**, that is (1, 0, 2) <= (3, 3, 2), since the condition is true. So the process P1 gets the request immediately

Problem-02:

A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST?

	Allocation			Request		
	X	**Y**	**Z**	**X**	**Y**	**Z**
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

Solution-

According to question-

- Total = [X Y Z] = [5 5 5]
- Total _Alloc = [X Y Z] = [5 4 3]

Now,

Available

= Total – Total_Alloc

= [5 5 5] – [5 4 3]

= [0 1 2]

Step-01:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.

•It completes its execution and then free up the instances of resources held by it.

Then,

Available

= [0 1 2] + [2 0 1]

= [2 1 3]

Step-02:

•With the instances available currently, only the requirement of the process P0 can be satisfied.

•So, process P0 is allocated the requested resources.

•It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [2 1 3] + [1 2 1]

= [3 3 4]

Step-03:

•With the instances available currently, the requirement of the process P2 can be satisfied.

•So, process P2 is allocated the requested resources.

•It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [3 3 4] + [2 2 1]

= [5 5 5]

Thus,

•There exists a safe sequence P1, P0, P2 in which all the processes can be executed.

•So, the system is in a safe state.

•Process P2 will be executed at last.

Problem-03:

An operating system uses the banker's algorithm for deadlock avoidance when managing the allocation of three resource types X, Y and Z to three processes P0, P1 and P2. The table given below presents the current system state. Here, the Allocation matrix shows the current number of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution.

	Allocation			**Max**		
	X	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in safe state. Consider the following independent requests for additional resources in the current state-

REQ1: P0 requests 0 units of X, 0 units of Y and 2 units of Z

REQ2: P1 requests 2 units of X, 0 units of Y and 0 units of Z

Which of the following is TRUE?

1. Only REQ1 can be permitted
2. Only REQ2 can be permitted
3. Both REQ1 and REQ2 can be permitted
4. Neither REQ1 nor REQ2 can be permitted

Solution-

According to question,

Available = [X Y Z] = [3 2 2]

Now,

Need = Max – Allocation

So, we have-

	Allocation			Max			Need		
	X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	1	8	4	3	8	4	2
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

Currently, the system is in safe state.

(It is given in question. If we want, we can check)

Checking Whether REQ1 Can Be Entertained-

- Need of P0 = [0 0 2]
- Available = [3 2 2]

Clearly,

- With the instances available currently, the requirement of REQ1 can be satisfied.
- So, banker's algorithm assumes that the request REQ1 is entertained.
- It then modifies its data structures as-

	Allocation			Max			Need		
	X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	**3**	8	4	3	8	4	**0**
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

Available

= [3 2 2] – [0 0 2]

= [3 2 0]

- Now, it follows the safety algorithm to check whether this resulting state is a safe state or not.
- If it is a safe state, then REQ1 can be permitted otherwise not.

Step-01

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [3 2 0] + [3 2 0]

= [6 4 0]

Now,

- It is not possible to entertain any process.
- The system has entered the deadlock state which is an unsafe state.
- Thus, REQ1 will not be permitted.

Checking Whether REQ2 Can Be Entertained-

- Need of P1 = [2 0 0]
- Available = [3 2 2]

Clearly,

- With the instances available currently, the requirement of REQ1 can be satisfied.
- So, banker's algorithm assumes the request REQ2 is entertained.
- It then modifies its data structures as-

	Allocation			Max			Need		
	X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	1	8	4	3	8	4	2
P1	**5**	2	0	6	2	0	**1**	0	0
P2	2	1	1	3	3	3	1	2	2

Available

= [3 2 2] – [2 0 0]

= [1 2 2]

- Now, it follows the safety algorithm to check whether this resulting state is a safe state or not.
- If it is a safe state, then REQ2 can be permitted otherwise not.

Step-01:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [1 2 2] + [5 2 0]

= [6 4 2]

Step-02:

- With the instances available currently, only the requirement of the process P2 can be satisfied.

- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [6 4 2] + [2 1 1]

= [8 5 3]

Step-03:

- With the instances available currently, the requirement of the process P0 can be satisfied.
- So, process P0 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [8 5 3] + [0 0 1]

= [8 5 4]

Thus,

- There exists a safe sequence P1, P2, P0 in which all the processes can be executed.
- So, the system is in a safe state.
- Thus, REQ2 can be permitted.

Thus, Correct Option is (B).

Problem-04:

A system has 4 processes and 5 allocatable resource. The current allocation and maximum needs are as follows-

	Allocated					Maximum				
A	1	0	2	1	1	1	1	2	1	3
B	2	0	1	1	0	2	2	2	1	0
C	1	1	0	1	1	2	1	3	1	1
D	1	1	1	1	0	1	1	2	2	0

If Available = [0 0 X 1 1], what is the smallest value of x for which this is a safe state?

Solution-

Let us calculate the additional instances of each resource type needed by each process.

We know,

Need = Maximum – Allocation

So, we have-

	Need				
A	0	1	0	0	2
B	0	2	1	0	0
C	1	0	3	0	0
D	0	0	1	1	0

Case-01: For X = 0

If X = 0, then-

Available

= [0 0 0 1 1]

- With the instances available currently, the requirement of any process can not be satisfied.
- So, for X = 0, system remains in a deadlock which is an unsafe state.

Case-02: For X = 1

If X = 1, then-

Available

= [0 0 1 1 1]

Step-01:

- With the instances available currently, only the requirement of the process D can be satisfied.
- So, process D is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 0 1 1 1] + [1 1 1 1 0]

= [1 1 2 2 1]

- With the instances available currently, the requirement of any process can not be satisfied.
- So, for X = 1, system remains in a deadlock which is an unsafe state.

Case-02: For X = 2

If X = 2, then-

Available

= [0 0 2 1 1]

Step-01:

- With the instances available currently, only the requirement of the process D can be satisfied.
- So, process D is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 0 2 1 1] + [1 1 1 1 0]

= [1 1 3 2 1]

Step-02:

- With the instances available currently, only the requirement of the process C can be satisfied.
- So, process C is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [1 1 3 2 1] + [1 1 0 1 1]

= [2 2 3 3 2]

Step-03:

- With the instances available currently, the requirement of both the processes A and B can be satisfied.
- So, processes A and B are allocated the requested resources one by one.
- They complete their execution and then free up the instances of resources held by it.

Then-

Available

= [2 2 3 3 2] + [1 0 2 1 1] + [2 0 1 1 0]

= [5 2 6 5 3]

Page Replacement Algorithms Examples

Types of Page Replacement Algorithms

There are various page replacement algorithms. Each algorithm has a different method by which the pages can be replaced.

1.**Optimal Page Replacement algorithm** → this algorithms replaces the page which will not be referred for so long in future. Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimal.

2.**Least recent used (LRU) page replacement algorithm** → this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.

3.**FIFO** → in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

Problem01:

Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. which one of the following is true with respect to page replacement policies First-In-First-out (FIFO) and Least Recently Used (LRU)?

Solution:

FIFO

Request	3	8	2	3	9	1	6	3	8	9	3	6	2	1	3
Frame 5						1	1	1	1	1	1	1	1	1	1
Frame 4					9	9	9	9	9	9	9	9	2	2	2
Frame 3			2	2	2	2	2	2	8	8	8	8	8	8	8
Frame 2		8	8	8	8	8	8	3	3	3	3	3	3	3	3
Frame 1	3	3	3	3	3	3	6	6	6	6	6	6	6	6	6
Miss/Hit	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

According to FIFO, the page which first comes in the memory will first goes out.

Number of Page Faults = 9
Number of hits = 6

LRU

According to LRU, the page which has not been requested for a long time will get replaced with the new one.

Number of Page Faults = 9
Number of Hits = 6

Request	3	8	2	3	9	1	6	3	8	9	3	6	2	1	3
Frame 5						1	1	1	1	1	1	1	2	2	2
Frame 4					9	9	9	9	9	9	9	9	9	9	9
Frame 3			2	2	2	2	2	2	8	8	8	8	8	1	1
Frame 2		8	8	8	8	8	6	6	6	6	6	6	6	6	6
Frame 1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Miss/Hit	Miss	Miss	Miss	Hit	Miss	Miss	Hit	Hit	Miss	Hit	Miss	Hit	Miss	Miss	Hit

Problem02-

Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. the number of frames in the memory is 3. Find out the number of page faults respective to:

1.Optimal Page Replacement Algorithm

2.FIFO Page Replacement Algorithm

3.LRU Page Replacement Algorithm

Solution:

Optimal Page Replacement Algorithm

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	2	2	2
Frame 2		7	7	7	7	7	7	7	7	7
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

Number of Page Faults in Optimal Page Replacement Algorithm = 5

LRU Page Replacement Algorithm

Request	**4**	**7**	**6**	**1**	**7**	**6**	**1**	**2**	**7**	**2**
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	**Miss**	**Miss**	**Miss**	**Miss**	**Hit**	**Hit**	**Hit**	**Miss**	**Miss**	**Hit**

Number of Page Faults in LRU = 6

FIFO Page Replacement Algorithm

Request	**4**	**7**	**6**	**1**	**7**	**6**	**1**	**2**	**7**	**2**
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

Number of Page Faults in FIFO = 6

Belady'sAnomaly

In the case of LRU and optimal page replacement algorithms, it is seen that the number of page faults will be reduced if we increase the number of frames. However, Balady found that, In FIFO page replacement algorithm, the number of page faults will get increased with the increment in number of frames.

This is the strange behavior shown by FIFO algorithm in some of the cases. This is an Anomaly called as Belady'sAnomaly.

Let's examine such example :

The reference String is given as 0 1 5 3 0 1 4 0 1 5 3 4. Let's analyze the behavior of FIFO algorithm in two cases.

Case 1: Number of frames = 3

Request	0	1	5	3	0	1	4	0	1	5	3	4
Frame 3			5	5	5	1	1	1	1	1	3	3
Frame 2		1	1	1	0	0	0	0	0	5	5	5
Frame 1	0	0	0	3	3	3	4	4	4	4	4	4
Miss/Hit	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

Number of Page Faults = 9

Case 2: Number of frames = 4

Request	0	1	5	3	0	1	4	0	1	5	3	4
Frame 4				3	3	3	3	3	3	5	5	5
Frame 3			5	5	5	5	5	5	1	1	1	1
Frame 2		1	1	1	1	1	1	0	0	0	0	4
Frame 1	0	0	0	0	0	0	4	4	4	4	3	3
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Miss	Miss

Number of Page Faults = 10

Therefore, in this example, the number of page faults is increasing by increasing the number of frames hence this suffers from Belady'sAnomaly.