Unit - III CPU Scheduling

3.1 Scheduling Basic concept:

CPU Scheduling is a process used by the operating system to decide which task or process gets to use the CPU at a particular time. This is important because a CPU can only ~~hadle~~ handle one task at a time, but there are usually many task that need to be processed. The following are different purposes of a CPU scheduling time.

- Maximize the CPU utilization
- Minimize the response and waiting time of the process.

The main function of CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready queue.

CPU and I/O burst cycle:

A CPU and I/O Burst cycle describe the way a process alternates between using the CPU and performing I/O operations during its execution.

CPU Burst:

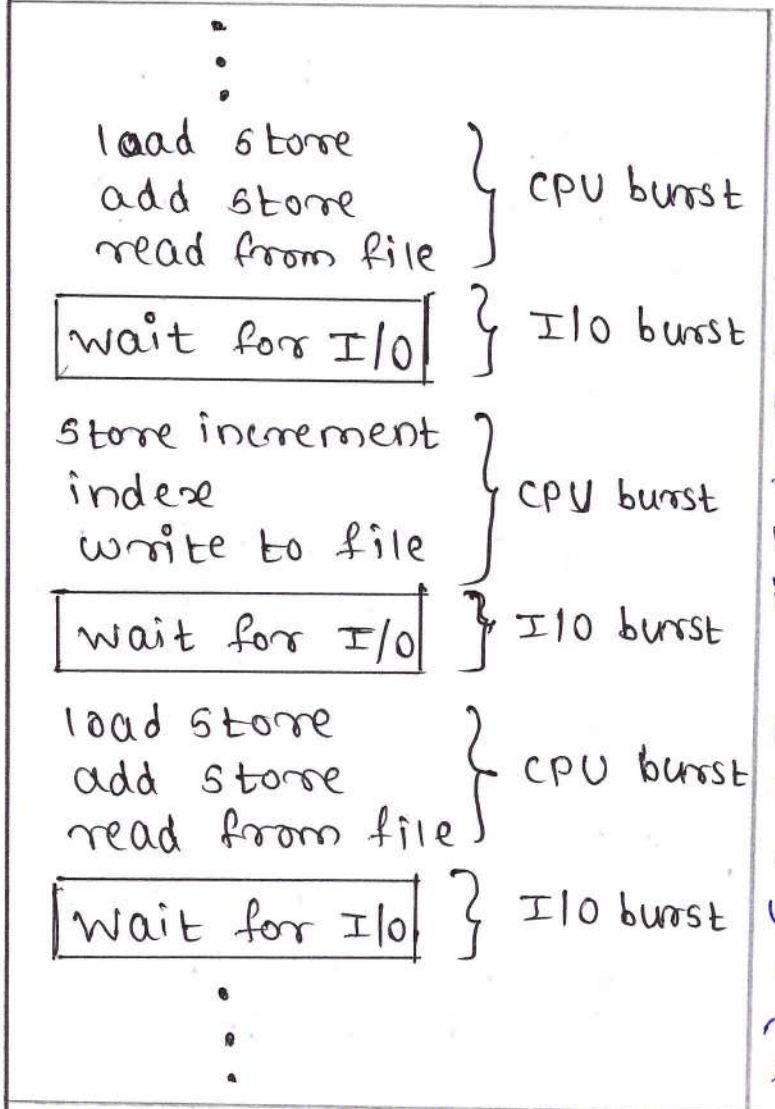
- The period during which a process is actively executing instructions on the CPU.
- The process performs calculations, data processing, or program logic.

I/O Burst:

- The period during which a process is waiting for an I/O operation to complete.
- Eg: Reading from a disk, receiving data from a network, or waiting for keyboard input.

The success of CPU scheduling depends on an observed property of processes:

- Process execution consists of a cycle of CPU execution and I/O wait.
- Processes alternate between these two states.



Process execution begins with a CPU burst. That is followed by an I/O burst, which is followed by another CPU burst, then another I/O burst and so on. Eventually, the final CPU burst ends with a system request to terminate execution.

Thus, any process would typically require both CPU time and I/O time. So, while it is using the I/O resources, CPU remains idle and vice versa. To utilize the resources efficiently we can schedule other processes to utilize out idle resources.

3.2 Pre-emptive and Non pre-emptive Scheduling:

These are the two techniques to schedule the incoming processes:

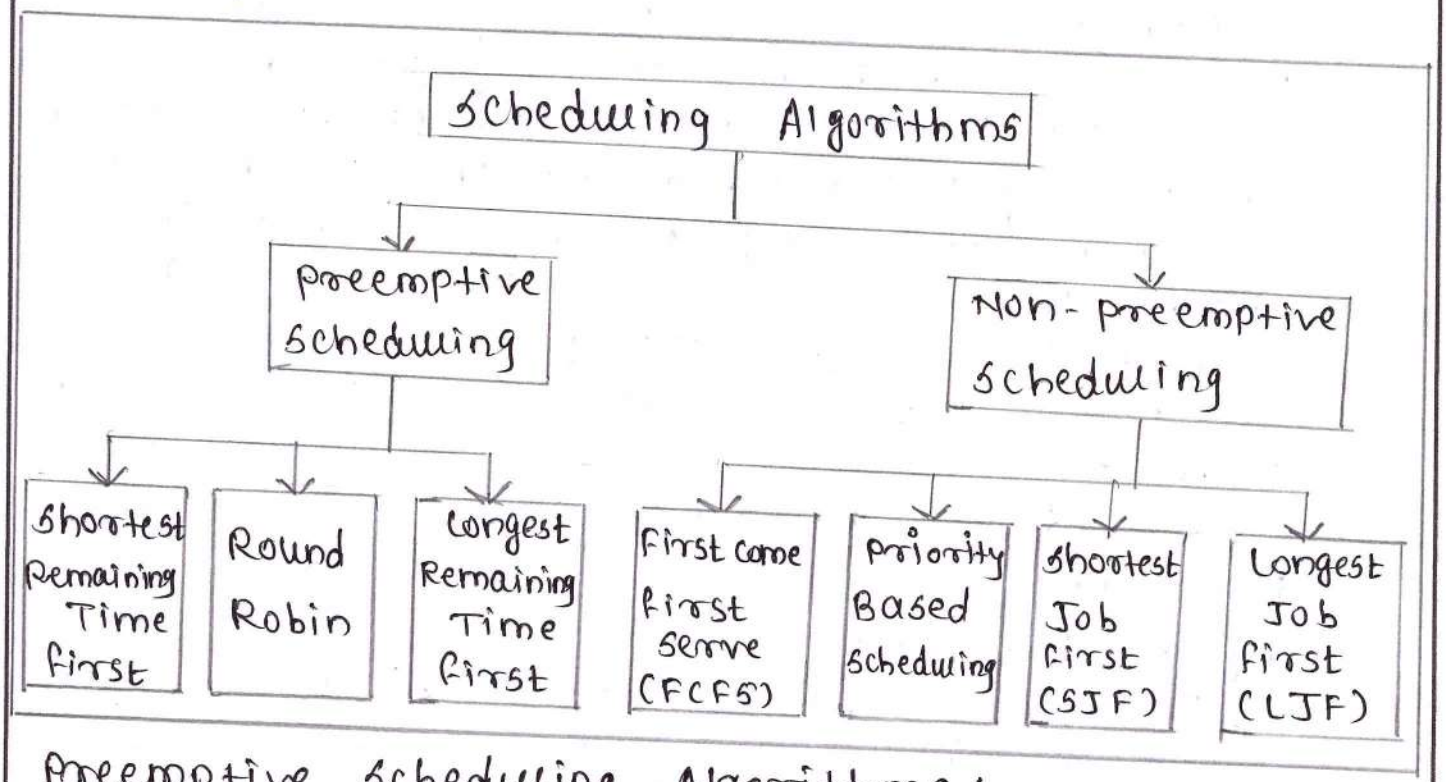
① Non pre-empting Scheduling:

Non-pre-empting scheduling algorithms refer to the class of CPU scheduling technique where once a process is allocated the CPU, it holds the CPU till the process gets terminated

or is pushed to the waiting state. No process is interrupted until it runs to completion. The scheduler allocates another process to the CPU only after the currently allocated process terminates and relinquishes control on the CPU.

② Pre-emptive Scheduling :

Preemptive scheduling algorithm fall under the category of process scheduling technique in which a running process can be interrupted by another process and sent to the ready queue even when it has not completed its entire execution in CPU.



Preemptive Scheduling Algorithms :

① Shortest Remaining Time First (SRTF):

Allot the processor to the process that has shortest remaining time first. Remaining time calculated as :

Remaining Time = Total Burst Time − CPU time already utilized by the process

② Round Robin :
Allot the processor to each of the process for a fixed time and then "Context Switch" to next processes in queue in a cyclic manner.

③ Longest Remaining Time first :
Allot the processor to the process that has longest remaining time.

Non - preemptive scheduling Algorithms :

① First come first serve (FCFS) :
Schedule the process on the basis of their arrival time.

② Priority Based Scheduling :
Each process has a priority assigned to it, processes with higher priority gets the processor first. If processes have different Arrival Time in Priority Based scheduling then we have to implement it using preemptive scheduling technique.

③ Shortest Job first (SJF) :
processes with shortest Burst Time gets the CPU first. If we have different Arival Time we will have to use preemptive scheduling ~~teq~~ technique.

④ Longest Job first :
processes with longest Burst Time gets the CPU first. This is similar to shortest Job first scheduling algorithm.

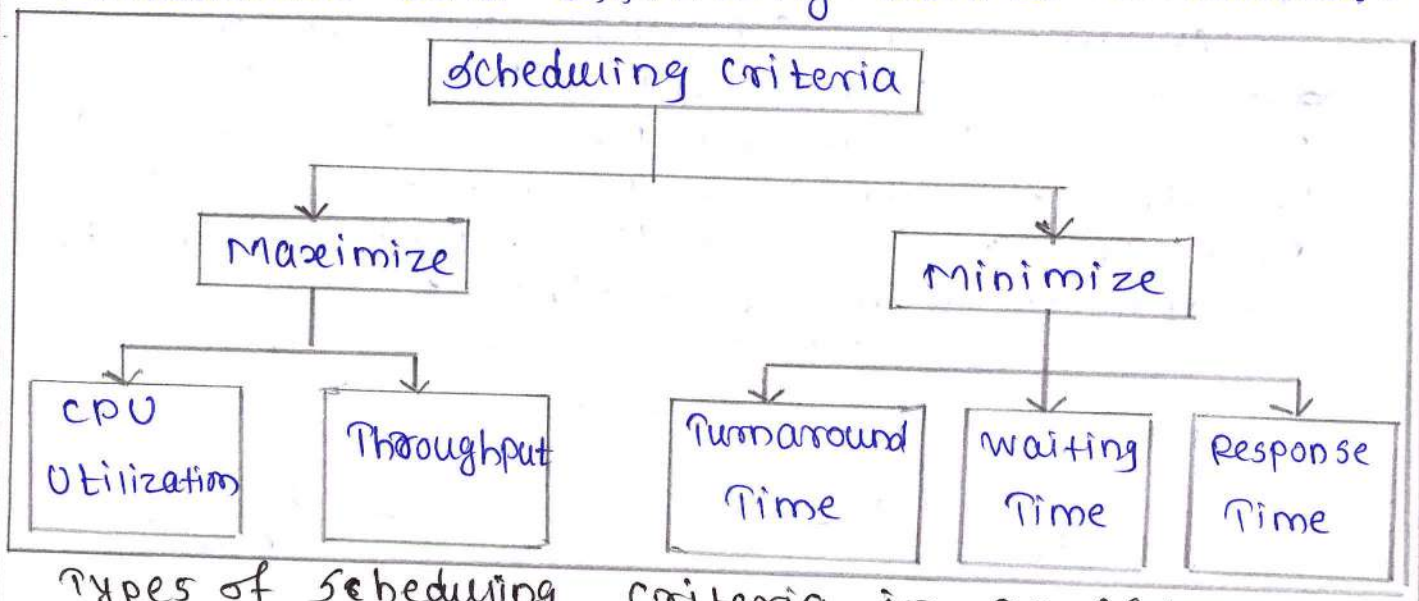
* Difference between preemptive and Non-preemptive:

Preemptive Scheduling	Non-preemptive Scheduling
① Resources are allocated according to the cycles for a limited time.	① Resources are used and then held by the process until it gets terminated.
② The process can be interrupted, even before the completion.	② The process is not interrupted until its life cycle is complete.

Preemptive Scheduling	Non-preemptive Scheduling
③ Starvation may be caused, due to the insertion of priority process in the queue.	③ Starvation can occur when a process with large burst time occupies the system.
④ Maintaining queue and remaining time needs storage overhead.	④ No such overheads are required.
⑤ Fair scheduling can be applied where all the processes can get equal chance for CPU access.	⑤ A process may monopolize the CPU.
⑥ Deadlocks can be easily avoided.	⑥ Deadlocks may occur.

* Scheduling Criteria :

Scheduling criteria are the standards used to evaluate and compare CPU scheduling algorithms. A good scheduling algorithm should optimize system performance and efficiently utilize resources.



Types of Scheduling Criteria in an OS :

There are different CPU scheduling algorithms with different properties. There are many criteria suggested for comparing CPU schedule algorithms, some of which are :

- **CPU Utilization:**

The object of any CPU scheduling algorithm is to keep the CPU busy if possible and to maximize its usage. The range of CPU Utilization is in the range of 0 to 100 but in real-time, it is actually 50% to 90% which relies on the system's load.

$$\text{CPU Utilization} = \frac{\text{CPU Busy time}}{\text{Total Time}} \times 100$$

- **Throughput:**

It is a measure of the work that is done by the CPU which is directly proportional to the number of processes being executed and completed per unit of time. It keeps on varying which relies on the duration or length of process.

$$\text{Throughput} = \frac{\text{Number of completed processes}}{\text{Unit Time}}$$

- **Turnaround Time:**

An important scheduling criterion in OS for many process is how long it takes to execute a process. A turnaround time is the elapsed from the time of submission to that of completion. It is the summation of time spent waiting to get into the memory, waiting for a queue to be ready, for the I/O process, and for the execution of the CPU. The formula for calculating.

$$\text{Turnaround Time} = \text{Completion Time} - \text{Arrival Time}$$

- **Waiting Time:**

Once the execution starts, the scheduling process does not hinder the time that is required for the completion of the process. The only thing that is affected is the waiting time of the process i.e. the time that is spent by a process waiting in a queue. The formula for calculating

Waiting Time	=	Turnaround Time	−	Burst Time

- **Response Time :**

Turnaround time is not considered as the best criterion for comparing scheduling algorithms in an interactive system. Some outputs of the process might produce early while computing other results simultaneously. Another criterion is the time that is taken from process submission to generate the first response. This is called responce time and the formula for calculating

Response Time	=	First CPU Allocatim Time	−	Arrival Time

3.3 Types of Scheduling Algorithms :

A process scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms :

1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Shortest Remaining Time Next (SRTN)
4. Round Robin (RR)
5. Priority Scheduling
6. multilevel queue Scheduling

1] First Come First Serve (FCFS) :

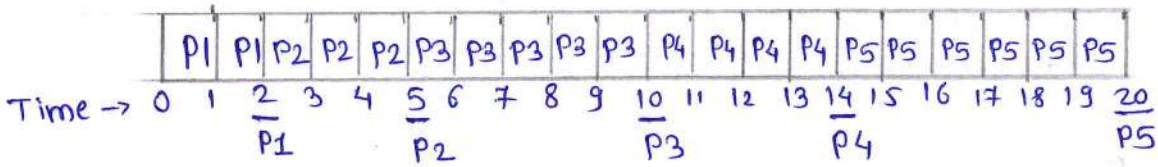
- The process which arrives first in the ready queue is firstly assigned the CPU.
- In case of a tie, process with smaller process id is executed first.
- It is always non-preemptive in nature.
- Jobs are executed on first come, first serve basis.
- Easy to understand and implement.
- poor in performance as average wait time is high.

9.① Consider the following processes with burst time, calculate the average waiting time and average turnaround time?

Process ID	Arrival Time	Burst Time / CPU Execution Time
P1	0	2
P2	1	3
P3	2	5
P4	3	4
P5	4	6

Solution :

Gantt Chart



Turnaround Time = Completion time − Arrival time

waiting Time = Turnaround time − Burst time

Process Id	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	2	2	2 − 0 = 2	2 − 2 = 0
P2	1	3	5	5 − 1 = 4	4 − 3 = 1
P3	2	5	10	10 − 2 = 8	8 − 5 = 3
P4	3	4	14	14 − 3 = 11	11 − 4 = 7
P5	4	6	20	20 − 4 = 16	16 − 6 = 10

$$\text{Average Turnaround Time} = \frac{2+4+8+11+16}{5} = \frac{41}{5} = 8.2$$

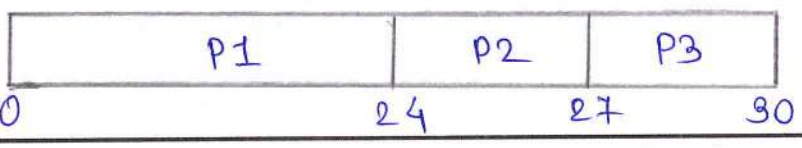
$$\text{Average waiting Time} = \frac{0+1+3+7+10}{5} = \frac{21}{5} = 4.2$$

9.② Consider the processes P1, P2, P3 given in the below table, arrives for execution in the same order, with Arrival Time 0 and given Burst Time.

Process	Arrival Time	Burst Time
P1	0	24
P2	0	3
P3	0	3

Solution:

Gantt chart



Process	Wait Time	Turnaround Time
P1	0	24
P2	24	27
P3	27	30

Average waiting Time = $\frac{0+24+27}{3} = \frac{51}{3} = 17$ ms

Average Turnaround Time = $\frac{24+27+30}{3} = \frac{81}{3} = 27$ ms

Throughput = 3 Jobs / 30 sec = 0.1 jobs/sec

2] Shortest Job First (SJF):

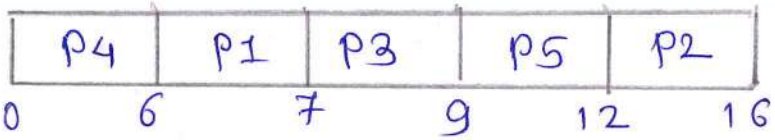
- process which have the shortest burst time are scheduled first.
- If two processes have the same burst time, then FCFS is used to break the tie.
- This is a non-preemptive, preemptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- The processer should know in advance how much time process will take.
- pre-emptive mode of Shortest Job First is called as Shortest Remaining Time First (SRTF).

Q.1) consider the set of 5 processes whose arrival time and burst time are given below:

Process Id	Arrival Time	Burst Time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

If the CPU scheduling policy is SJF non-pre-emptive, calculate the average waiting time and average turnaround time.

Gantt chart:



Process Id	Exit Time	Turnaround Time	Waiting Time
P1	7	7 - 3 = 4	4 - 1 = 3
P2	16	16 - 1 = 15	15 - 4 = 11
P3	9	9 - 4 = 5	5 - 2 = 3
P4	6	6 - 0 = 6	6 - 6 = 0
P5	12	12 - 2 = 10	10 - 3 = 7

Average Turnaround Time $= \frac{4+15+5+6+10}{5} = \frac{40}{5} = \underline{8 \text{ unit}}$

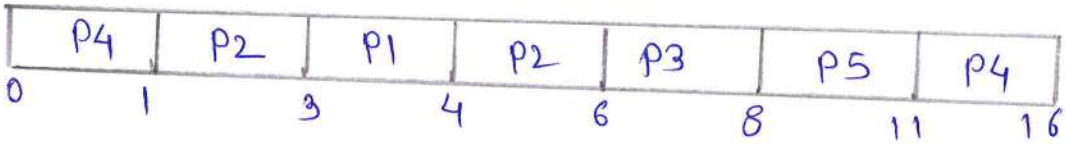
Average waiting Time $= \frac{3+11+3+0+7}{5} = \frac{24}{5} = \underline{4.8 \text{ unit}}$

Q.2) consider the set of 5 processes whose arrival time and burst time are given below:

process Id	Arrival Time	Burst Time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

If the CPU scheduling policy is SJF pre-emptive, calculate the average waiting time and average turnaround time.

Gantt chart:



Process Id	Exit Time	Turnaround Time	Waiting Time
P1	4	4 - 3 = 1	1 - 1 = 0
P2	6	6 - 1 = 5	5 - 4 = 1
P3	8	8 - 4 = 4	4 - 2 = 2
P4	16	16 - 0 = 16	16 - 6 = 10
P5	11	11 - 2 = 9	9 - 3 = 6

Average Turnaround Time $= \frac{1+5+4+16+9}{5} = \frac{35}{5} = \underline{7 \text{ units}}$

Average waiting Time $= \frac{0+1+2+10+6}{5} = \frac{19}{5} = \underline{3.8 \text{ units}}$

3] Shortest Remaining Time Next (SRTN):

Shortest Remaining Time Next (SRTN) is a preemptive CPU scheduling algorithm and the preemptive version of Shortest Job first (SJF) scheduling.

In SRTN, the process with the smallest remaining burst time is always selected for execution. If a new process arrives with a burst time shorter than the remaining time of the currently running process, the CPU is immediately assigned to the new process.

Working of SRTN:

1. Select the process with the shortest remaining execution time.
2. Execute it for a short period.
3. When a new process arrives:
   - Compare its burst time with the remaining time of the current process.
   - If the new process has a shorter remaining time, preempt the current process.
4. Repeat until all processes are completed.

Example: Consider the following processes:

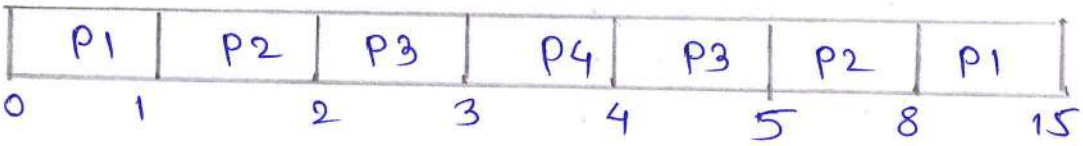
Process	Arrival Time (AT)	Burst Time (BT)
P1	0	8
P2	1	4
P3	2	2
P4	3	1

Step by step execution:

- Time 0: P1 starts execution
- Time 1: P2 arrives. Remaining time of P1 = 7, BT of P2 = 4. Since 4 < 7, P1 is preempted and P2 starts.
- Time 2: P3 arrives. Remaining time of P2 = 3, BT of P3 = 2. Since 2 < 3, P2 is preempted and P3 starts.
- Time 3: P4 arrives. Remaining time of P3 = 1, BT of P4 = 1. Execute P4 first.

- Time 4 : P4 completes.
- Time 4-5 : P3 completes.
- Time 5-8 : P2 completes.
- Time 8-15 : P1 completes.

Gantt chart :

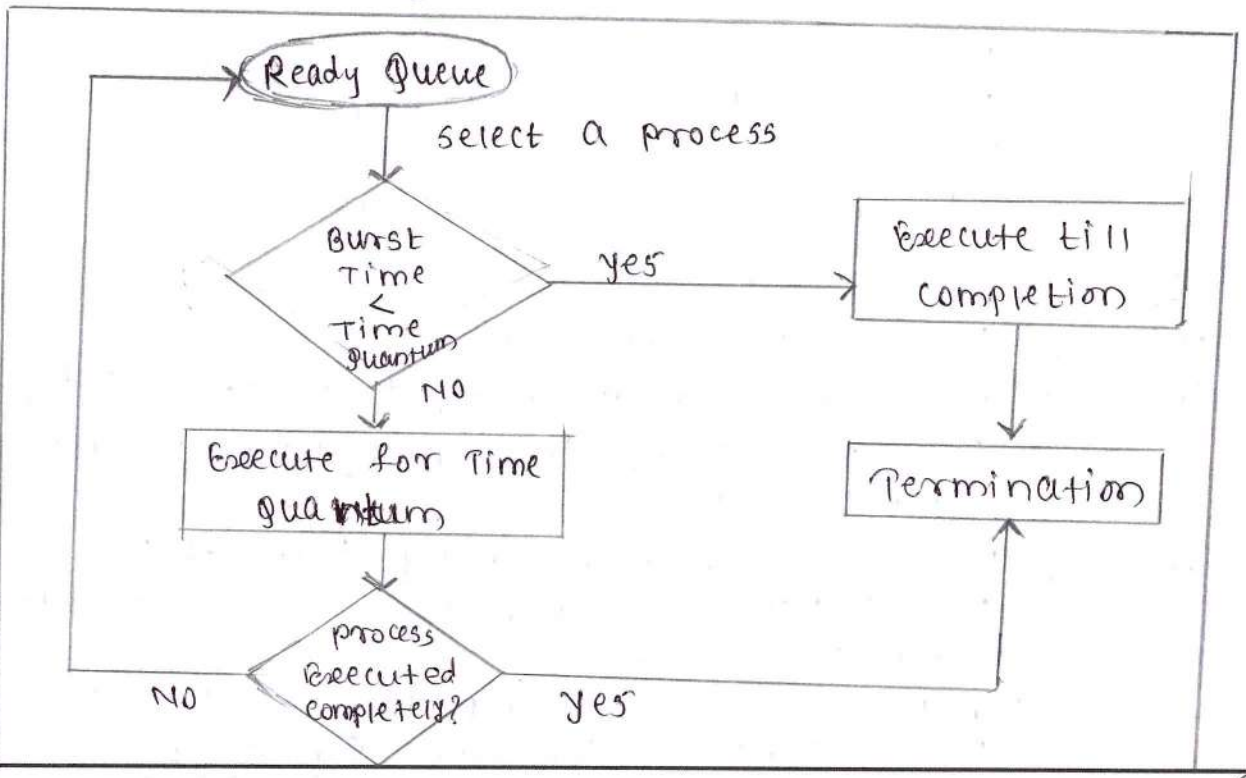


Average Turnaround Time $= \frac{15+7+3+1}{4} = 6.5$ units

Average waiting Time $= \frac{7+3+1+0}{4} = 2.75$ units

4] Round Robin (RR) :

- CPU is assigned to the process on the basis of FCFS for a fixed amount of time.
- This fixed amount of time is called as time quantum or time slice.
- After the time quantum expires, the running process is preemptive and sent to the ready queue.
- Then the processor is assigned to the next arrived process.
- It is always preemptive in nature.



Q.1) consider the set of 5 processes whose arrival time and burst time are given below:

process Id	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turnaround time.

Solution:

Ready Queue: ~~P1~~, ~~P2~~, ~~P3~~, ~~P1~~, ~~P4~~, ~~P5~~, ~~P2~~, ~~P1~~, ~~P5~~

Gantt Chart:

P1	P2	P3	P1	P4	P5	P2	P1	P5

0 2 4 5 7 9 11 12 13 14

Turn Around Time = completion Time − Arrival Time

Waiting Time = Turn Around Time − Burst Time

Process Id	completion Time	Turn Around Time	waiting Time
P1	13	13-0 = 13	13-5 = 8
P2	12	12-1 = 11	11-3 = 8
P3	5	5-2 = 3	3-1 = 2
P4	9	9-3 = 6	6-2 = 4
P5	14	14-4 = 10	10-3 = 7

Average Turn Around Time $= \frac{13+11+3+6+10}{5} = \frac{43}{5} = 8.6$ unit

Average waiting Time $= \frac{8+8+2+4+7}{5} = \frac{29}{5} = 5.8$ unit

Q.2) consider the set of 6 processes whose Arrival time and burst time are given below:

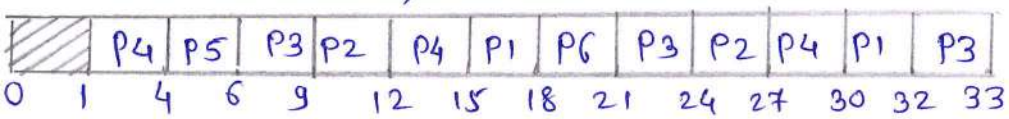
process Id	Arrival Time	Burst Time
P1	5	5
P2	4	6
P3	3	7
P4	1	9
P5	2	2
P6	6	3

If the CPU scheduling policy is Round Robin with time quantum = 3, calculate the average waiting time and average Turnaround Time.

Solution :

Ready Queue : P4, P5, P3, P2, P4, P1, P6, P3, P2, P4, P1, P3,

Gantt Chart :



we know,

Turnaround Time = Completion Time − Arrival Time

waiting Time = TurnAround Time − Burst Time

Process Id	completion Time	Turn Around Time	Waiting Time
P1	32	32 - 5 = 27	27 - 5 = 22
P2	27	24 - 4 = 23	23 - 6 = 17
P3	33	33 - 3 = 30	30 - 7 = 23
P4	30	30 - 1 = 29	29 - 9 = 20
P5	6	6 - 2 = 4	4 - 2 = 2
P6	21	21 - 6 = 15	15 - 3 = 12

Average TurnAround Time = $\frac{27+23+30+29+4+15}{6} = \frac{128}{6} = 21.33$

Average waiting Time = $\frac{22+17+23+20+2+12}{6} = \frac{96}{6} = 16$ units

5] priority Scheduling :

- Out of all the available processes, CPU is assigned to the process having the highest priority.
- In case of a tie, it is broken by FCFS scheduling.
- priority scheduling can be used in both preemptive and non-preemptive mode.
- The waiting time for the process having the highest priority will always be zero in pre-emptive mode.
- The waiting time for the process having the highest priority may not be zero in non-preemptive mode.
- priority scheduling in preemptive and non-preemptive mode behaves exactly same under following conditions −

- The arrival time of all the process is same
- All the processes become available.

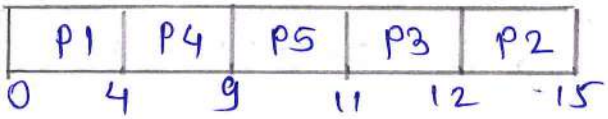
Q.1] Consider the set of 5 processes whose arrival time and burst time are given below:

Process Id	Arrival Time	Burst Time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turnaround time. (Higher number represents higher priority).

Solution:

Gantt chart:



we know,

Turnaround Time = completion Time − Arrival Time

Waiting Time = Turnaround Time − Burst Time

Process Id	completion Time	Turnaround Time	Waiting Time
P1	4	4 − 0 = 4	4 − 4 = 0
P2	15	15 − 1 = 14	14 − 3 = 11
P3	12	12 − 2 = 10	10 − 1 = 9
P4	9	9 − 3 = 6	6 − 5 = 1
P5	11	11 − 4 = 7	7 − 2 = 5

$$\text{Average Turnaround Time} = \frac{4+14+10+6+7}{5} = \frac{41}{5} = 8.2 \text{ unit}$$

$$\text{Average waiting Time} = \frac{0+11+9+1+5}{5} = \frac{26}{5} = 5.2 \text{ unit}$$

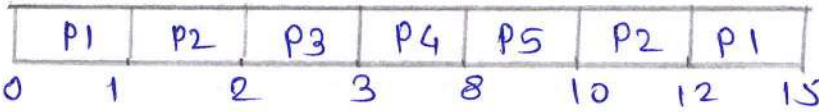
Q.2] Consider the set of 5 processes whose AT & BT are given below-

Process Id	Arrival Time	Burst Time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turn around time.

Solution:

Gantt Chart:



we know,

- Turnaround Time = completion Time − Arrival Time
- waiting Time = Turnaround Time − Burst Time

Process Id	completion Time	Turnaround Time	Waiting Time
P1	15	15-0 = 15	15-4 = 11
P2	12	12-1 = 11	11-3 = 8
P3	3	3-2 = 1	1-1 = 0
P4	8	8-3 = 5	5-5 = 0
P5	10	10-4 = 6	6-2 = 4

Average Turnaround Time $= \frac{15+11+1+5+6}{5} = \frac{38}{5} = 7.6$ unit

Average waiting Time $= \frac{11+8+0+0+4}{5} = \frac{23}{5} = 4.6$ unit

6] multilevel queue scheduling:

A multilevel queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm.

For example:

① System processes queue − contains operating system tasks.

② Interactive Processes queue − contains user-interactive program.

③ Batch processes queue − contains background Jobs.

Suppose the system has three queues:

Queue	Processes	Scheduling Algorithm	Priority
Q1	System processes	Round Robin	Highest
Q2	Interactive Processes	Round Robin	Medium
Q3	Batch processes	FCFS	Lowest

Processes:

process	Queue	Burst Time
P1	Q1	4
P2	Q1	3
P3	Q2	5
P4	Q2	2
P5	Q3	6

Scheduling: The CPU always executes processes from the highest – priority queue first.

Gantt chart:

P1	P2	P3	P4	P5
0	4	7	12	14	20

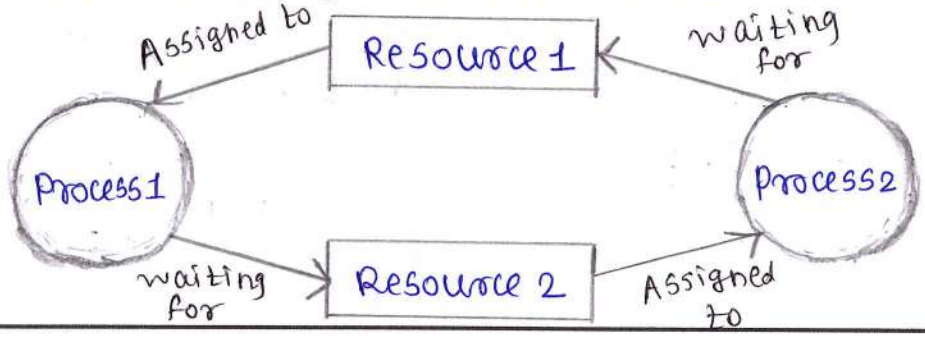
Execution order:

1. P1 (Q1) executes first.
2. P2 (Q1) executes next.
3. After Q1 becomes empty, Q2 processes are executed.
4. P3 and P4 executes
5. finally, Q3 processes P5 executes.

3.4 Deadlock:

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

~~For example, in the below~~



Necessary conditions leading to Deadlock:

Deadlock can arise if following four necessary conditions hold simultaneously.

1) Mutual Exclusion:
One or more than one resource are non-sharable means only one process can use at a time.

2) Hold and wait:
A process is holding at least one resource and waiting for another resources.

3) No pre-emption:
A resource cannot be taken from a process unless the process releases the resource means the process which once scheduled will be executed till the completion and no other process can be scheduled by the scheduler meanwhile.

4) Circular wait:
A set of processes are waiting for each other in circular form means all the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resources which is being held by the first process.

Difference between Deadlock and Starvation:

Sr.	Deadlock	Starvation
1	Deadlock is a situation where no process got blocked and no process proceeds.	Starvation is a situation where the low priority process got blocked and the high priority process proceed.
2	Deadlock is an infinite waiting.	Starvation is a long waiting but not infinite.
3	Every Deadlock is always a starvation.	Every starvation need not be deadlock.
4	The requested resource is blocked by the other process.	The requested resource is continously be used by the higher priority processes.

Deadlock Handling:

The various strategies for handling deadlock are-

1. Deadlock Prevention
2. Deadlock Avoidance
3. Deadlock Detection and Recovery
4. Deadlock Ignorance

1. Deadlock prevention:

Deadlocks can be prevented at least one of the four required conditions:

Mutual Exclusion:

- Shared resources such as read-only files do not lead to deadlocks.
- Unfortunately, some resources, such as printers and tape drives, require exclusive access by a single process.

Hold and wait:

To prevent this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

No preemption:

Preemption of process resources allocations can prevent this condition of deadlocks, when it is possible.

circular wait:

One way to avoid circular wait is to number all resources, and to require that processes request resources only in strictly increasing (or decreasing) order.

② Deadlock Avoidance:

- In deadlock avoidance, the operating system checks whether the system is in safe state or in unsafe state at every step which the operating system performs.
- The process continues until the system is in safe state.
- Once the system moves to unsafe state, the OS has to backtrack one step.
- In simple words, The OS reviews each allocation so that the allocation doesn't cause the deadlock in the system.

③ Deadlock detection and recovery:

- This strategy involves waiting until a deadlock occurs.
- After deadlock occurs, the system state is recovered.
- The main challenge with this approach is detecting the deadlock.

④ Deadlock Ignorance:

- This strategy involves ignoring the concept of deadlock and assuming as if it does not exist.
- This strategy helps to avoid the extra overhead of handling deadlock.
- Windows and Linux use this strategy and it is the most widely used method.

Deadlock avoidance - Banker's Algorithm :

Bankers algorithm is a deadlock avoidance algorithm developed by Edsger Dijkstra. It is used by an operating system to allocate resources safely so that the system never enters a deadlock state.

The Banker's algorithm that shows how to allocate resources to processes in a way that ensures that the system remains in a safe state. This works like a banker who allocates money to customers. Before allocating money to a customer, the banker must ensure that there is enough money in the bank to satisfy the maximum possible requests of all customers.

The Bankers Algorithm works in this way-

- When a new process enters the system, it must declare the maximum number of instances of each resources type that it may need.
- The OS then calculates whether allowing the request will leave the system in a safe state. If it does, the resources are allocated to the process.
- If the request would lead to an unsafe state, it is not allowed to take any other resources too. The process must wait until some resources are released by other processes.

Key concepts :

① Available :
Number of available instances of each resource type.

② Max :
Maximum number of resources each process may request.

③ Allocation :
Resources currently allocated to each process.

④ Need :
Resources still needed by a process.

$$\text{Need} = \text{Max} - \text{Allocation}$$

Steps for Banker's Algorithm :

The Bankers Algorithm works in two steps:

① Safety Algorithm - safe state check
② Resources Request Handling.

① Safety Algorithm :

The safety algorithm is one that determines whether or not the system is in a safe state. A safe state is one where there is at least one sequence of processes such that each process can obtain its maximum resources request and complete its execution without leading to a deadlock. The safety algorithm works as follows -

steps :
① Calculate Need = Max - Allocation
② Set work = available

③ Find a process whose:
- Need ≤ Work
- Not yet finished

④ Assume the process completes:
- Work = Work + Allocation

⑤ Repeat until all processes finish.

⑥ If all processes can finish, the system is safe; otherwise, it is unsafe.

Example:

Allocation matrix

Process	Allocation	Max
P0	1	3
P1	2	4
P2	1	2

Need matrix

Process	Need = Max - Allocation
P0	2
P1	2
P2	1

Available Resources = 2

Checking safe sequence: Work = 2
- P0 needs 2 → execute P0
  - Work = 2 + 1 = 3
- P2 needs 1 → execute P2
  - Work = 3 + 1 = 4
- P1 needs 2 → executed P1
  - Work = 4 + 2 = 6

Safe sequence: <P0, P2, P1>
Hence the system is in a safe state.

② Resources Request Handling:
When a process makes a request for resources, the Bankers Algorithm check whether granting the request will leave the system in a safe state. The steps for handling a resources request are as follows-

When a process request resources:

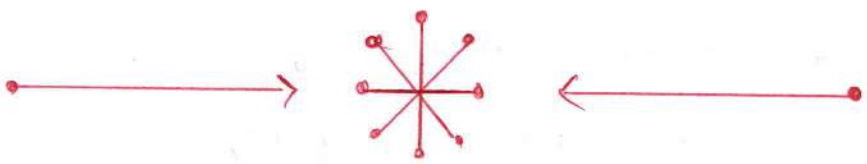
① Check if Request ≤ Need.
② Check if Request ≤ Available.
③ Temporarily allocate resources.
④ Run the safety Algorithm.
⑤ If the system remains safe, grant the request.
⑥ Otherwise, rollback and make the process wait.

Advantages:

① prevents deadlocks before they occur.
② Ensures the system always remains in a safe state.
③ Efficient for systems with fixed number of resources.

Disadvantages:

① Requires advance knowledge of maximum resources needs.
② complex for large systems.
③ Resources count must remain constant.



22/07/26

22/7/26

Unit - III CPU Scheduling

<table>
<tr><th>S.N.</th><th>MSBTE Board Asked Questions</th><th>Exam Year</th><th>Marks</th></tr>
<tr><td>1)</td><td>Define i)Turnaround time ii) Throughput</td><td>Winter-2025</td><td>2</td></tr>
<tr><td>2)</td><td>Describe CPU and I/O burst cycle with suitable diagram.</td><td>Winter-2024</td><td>2</td></tr>
<tr><td>3)</td><td>State any four criteria in CPU scheduling.</td><td>Winter-2024</td><td>2</td></tr>
<tr><td>4)</td><td>State any two features of non premptive scheduling.</td><td>Winter-2025
Summer-2024</td><td>2</td></tr>
<tr><td>5)</td><td>Write the difference between pre-emptive and non-preemptive scheduling.</td><td>Summer-2023</td><td>2</td></tr>
<tr><td>6)</td><td>Define Deadlock.</td><td>Summer-2023</td><td>2</td></tr>
<tr><td>7)</td><td>Describe necessary conditions leading to deadlock.</td><td>Winter-2025
Winter-2024
Summer-2023</td><td>4</td></tr>
<tr><td>8)</td><td>State the meaning of scheduler. Explain any one type of scheduler in detail.</td><td>Winter-2025</td><td>4</td></tr>
<tr><td>9)</td><td>Describe First Come First Served (FCFS) scheduling algorithm with example.</td><td>Winter-2024</td><td>4</td></tr>
<tr><td>10)</td><td>Describe the concept of scheduling queues with diagram.</td><td>Winter-2024</td><td>4</td></tr>
<tr><td>11)</td><td>Write steps of Banker's algorithm to avoid deadlock.</td><td>Winter-2024</td><td>4</td></tr>
<tr><td>12)</td><td>Define deadlock. State the conditions necessary for deadlock.</td><td>Summer-2024</td><td>4</td></tr>
<tr><td>13)</td><td>Explain following terms with respect to scheduling
i) CPU utilization ii) Throughput
iii) Turnaround time iv) Waiting time</td><td>Winter-2023</td><td>4</td></tr>
<tr><td>14)</td><td>What is deadlock? Discuss any one method of deadlock prevention.</td><td>Winter-2023</td><td>4</td></tr>
<tr><td>15)</td><td>Explain I/O burst and CPU burst.</td><td>Winter-2025</td><td>4</td></tr>
<tr><td>16)</td><td>Compare FCFS with SJF (Any 4 points).</td><td>Winter-2025</td><td>4</td></tr>
<tr><td>17)</td><td>Compare Short Job First (SJF) and Shortest Remaining Time (SRTN) scheduling algorithm (any four points).</td><td>Summer-2024</td><td>4</td></tr>
<tr><td>18)</td><td>State and explain four scheduling criteria.</td><td>Summer-2023</td><td>4</td></tr>
<tr><td>19)</td><td>Solve given problem by using FCFS scheduling algorithm. Draw correct Gantt chart and calculate average waiting time and average turnaround time-
<table>
<tr><th>Process</th><th>Arrival time</th><th>Burst time</th></tr>
<tr><td>P0</td><td>0</td><td>10</td></tr>
<tr><td>P1</td><td>1</td><td>29</td></tr>
<tr><td>P2</td><td>2</td><td>3</td></tr>
<tr><td>P3</td><td>3</td><td>7</td></tr>
<tr><td>P4</td><td>4</td><td>12</td></tr>
</table></td><td>Winter-2023</td><td>4</td></tr>
</table>

22/7/26

<table>
<tr><td>20)</td><td>How pre-emptive scheduling is better than non pre-emptive scheduling by solving following problem using SJF (Solve it by using pre-emptive SJF and non-pre-emptive SJF also).
<table>
<tr><th>Process</th><th>Arrival time</th><th>Burst time</th></tr>
<tr><td>P1</td><td>0</td><td>8</td></tr>
<tr><td>P2</td><td>1</td><td>4</td></tr>
<tr><td>P3</td><td>2</td><td>9</td></tr>
<tr><td>P4</td><td>3</td><td>5</td></tr>
</table></td><td>Winter-2023</td><td>4</td></tr>
<tr><td>21)</td><td>Consider the following processes.
<table>
<tr><th>Process</th><th>Burst time</th><th>Priority</th></tr>
<tr><td>01</td><td>8</td><td>1</td></tr>
<tr><td>02</td><td>5</td><td>0</td></tr>
<tr><td>03</td><td>5</td><td>2</td></tr>
<tr><td>04</td><td>13</td><td>1</td></tr>
</table>
Find out average waiting time for the following algorithms:
i) SJF ii) FCFS
iii) Priority scheduling (Low number → High Priority)</td><td>Winter-2025</td><td>6</td></tr>
<tr><td>22)</td><td>Describe Shortest Remaining Time Next (SRTN) and Round Robin Scheduling algorithms with suitable example.</td><td>Winter-2025</td><td>6</td></tr>
<tr><td>23)</td><td>Explain FCFS in detail with example showing how to calculate turnaround time and average waiting time.</td><td>Winter-2025</td><td>6</td></tr>
<tr><td>24)</td><td>With neat diagram explain multilevel queue scheduling.</td><td>Summer-2023</td><td>6</td></tr>
<tr><td>25)</td><td>Calculate average waiting time for SJF (Shortest Job First) and Round Robin (RR) algorithm table: (Time slice 4 ms)
<table>
<tr><th>Process</th><th>Burst Time</th></tr>
<tr><td>P1</td><td>10</td></tr>
<tr><td>P2</td><td>4</td></tr>
<tr><td>P3</td><td>9</td></tr>
<tr><td>P4</td><td>6</td></tr>
</table></td><td>Winter-2024</td><td>6</td></tr>
<tr><td>26)</td><td>What is the average turnaround time for the following process using :
i) FCFS scheduling algorithm
ii) SJF non-preemptive scheduling algorithm
iii) Round Robin Scheduling algorithm
<table>
<tr><th>Process</th><th>Arrival Time</th><th>Burst time</th></tr>
<tr><td>P1</td><td>0</td><td>8</td></tr>
<tr><td>P2</td><td>1</td><td>4</td></tr>
<tr><td>P3</td><td>2</td><td>1</td></tr>
</table></td><td>Summer-2024</td><td>6</td></tr>
</table>