*(1) Explain FCFS, Round Robin, Shortest Job First, Shortest Remaining Job First and Priority Scheduling algorithms with illustration.*

I FCFS (First Come First Serve):

- **Selection criteria :**

  The process that request first is served first. It means that processes are served in the exact order of their arrival.

- **Decision Mode :**

  Non preemptive: Once a process is selected, it runs until it is blocked for an I/O or some event, or it is terminated.

- **Implementation:**

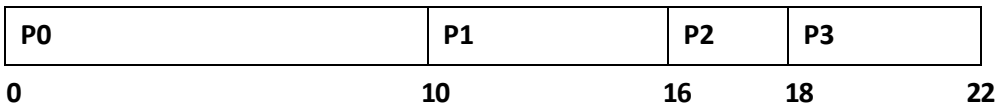
  This strategy can be easily implemented by using FIFO queue, FIFO means First In First Out. When CPU becomes free, a process from the first position in a queue is selected to run.

- **Example :**

  Consider the following set of four processes. Their arrival time and time required to complete the execution are given in following table. Consider all time values in milliseconds.

| Process | Arrival Time (T0) | Time required for completion (ΔT) (CPU Burst Time) |
|---|---|---|
| P0 | 0 | 10 |
| P1 | 1 | 6 |
| P2 | 3 | 2 |
| P3 | 5 | 4 |

- **Gantt Chart :**




- Initially only process P0 is present and it is allowed to run. But, when P0 completes, all other processes are present. So, next process P1 from ready queue is selected and allowed to run till it completes. This procedure is repeated till all processes completed their execution.

- **Statistics :**

| Process | Arrival Time (T0) | CPU Burst Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (WT=TAT-ΔT) |
|---|---|---|---|---|---|
| P0 | 0 | 10 | 10 | 10 | 0 |
| P1 | 1 | 6 | 16 | 15 | 9 |
| P2 | 3 | 2 | 18 | 15 | 13 |
| P3 | 5 | 4 | 22 | 17 | 13 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (10+15+15+17)/4 | = 57/4 | **= 14.25 ms.** |
| Average Waiting Time: | (0+9+13+13)/4 | = 35/4 | **= 8.75 ms.** |

- **Advantages:**
  - Simple, fair, no starvation.
  - Easy to understand, easy to implement.
- **Disadvantages :**
  - Not efficient. Average waiting time is too high.
  - Convoy effect is possible. All small I/O bound processes wait for one big CPU bound process to acquire CPU.
  - CPU utilization may be less efficient especially when a CPU bound process is running with many I/O bound processes.

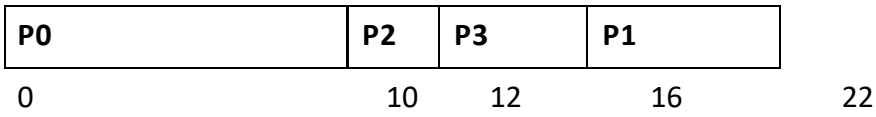
## II Shortest Job First (SJF):

- **Selection Criteria :**
  The process, that requires shortest time to complete execution, is served first.
- **Decision Mode :**
  Non preemptive: Once a process is selected, it runs until either it is blocked for an I/O or some event, or it is terminated.
- **Implementation :**
  This strategy can be implemented by using sorted FIFO queue. All processes in a queue are sorted in ascending order based on their required CPU bursts. When CPU becomes free, a process from the first position in a queue is selected to run.
- **Example :**

Consider the following set of four processes. Their arrival time and time required to complete the execution are given in following table. Consider all time values in milliseconds.

| Process | Arrival Time (T0) | Time required for completion (ΔT) (CPU Burst Time) |
|---|---|---|
| P0 | 0 | 10 |
| P1 | 1 | 6 |
| P2 | 3 | 2 |
| P3 | 5 | 4 |

- **Gantt Chart :**




- Initially only process P0 is present and it is allowed to run. But, when P0 completes, all other processes are present. So, process with shortest CPU burst P2 is selected and allowed to run till it completes. Whenever more than one process is available, such type of decision is taken. This procedure us repeated till all process complete their execution.

- **Statistics :**

| Process | Arrival Time (T0) | CPU Burst Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (Wt=TAT-ΔT) |
|---|---|---|---|---|---|
| P0 | 0 | 10 | 10 | 10 | 0 |
| P1 | 1 | 6 | 22 | 21 | 15 |
| P2 | 3 | 2 | 12 | 9 | 7 |
| P3 | 5 | 4 | 16 | 11 | 7 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (10+21+9+11)/4 | = 51/4 | **= 12.75 ms.** |
| Average Waiting Time: | (0+15+7+7) / 4 | = 29 / 4 | **= 7.25 ms.** |

- **Advantages:**

- Less waiting time.
- Good response for short processes.

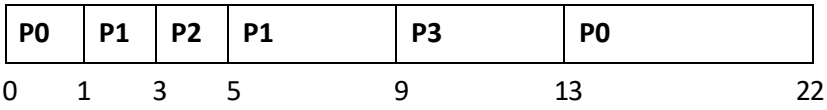
- **Disadvantages :**
  - It is difficult to estimate time required to complete execution.
  - Starvation is possible for long process. Long process may wait forever.

### III Shortest Remaining Time Next (SRTN):

- **Selection criteria :**
  The process, whose remaining run time is shortest, is served first. This is a preemptive version of SJF scheduling.
- **Decision Mode:**
  Preemptive: When a new process arrives, its total time is compared to the current process remaining run time. If the new job needs less time to finish than the current process, the current process is suspended and the new job is started.
- **Implementation :**
  This strategy can also be implemented by using sorted FIFO queue. All processes in a queue are sorted in ascending order on their remaining run time. When CPU becomes free, a process from the first position in a queue is selected to run.
- **Example :**
  Consider the following set of four processes. Their arrival time and time required to complete the execution are given in following table. Consider all time values in milliseconds.

| Process | Arrival Time (T0) | Time required for completion (ΔT) (CPU Burst Time) |
|---|---|---|
| P0 | 0 | 10 |
| P1 | 1 | 6 |
| P2 | 3 | 2 |
| P3 | 5 | 4 |

- **Gantt Chart :**



- Initially only process P0 is present and it is allowed to run. But, when P1 comes, it has shortest remaining run time. So, P0 is preempted and P1 is allowed to run. Whenever new process comes or current process blocks, such type of decision is taken. This procedure is repeated till all processes complete their execution.

- **Statistics :**

| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (WT=TAT-ΔT) |
|---|---|---|---|---|---|
| P0 | 0 | 10 | 22 | 22 | 12 |
| P1 | 1 | 6 | 9 | 8 | 2 |
| P2 | 3 | 2 | 5 | 2 | 0 |
| P3 | 5 | 4 | 13 | 8 | 4 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (22+8+2+8) / 4 | = 40/4 | **= 10 ms.** |
| Average Waiting Time: | (12+2+0+4)/4 | = 18 / 4 | **= 4.5 ms.** |

- **Advantages :**
  - Less waiting time.
  - Quite good response for short processes.
- **Disadvantages :**
  - Again it is difficult to estimate remaining time necessary to complete execution.
  - Starvation is possible for long process. Long process may wait forever.
  - Context switch overhead is there.

## IV Round Robin:

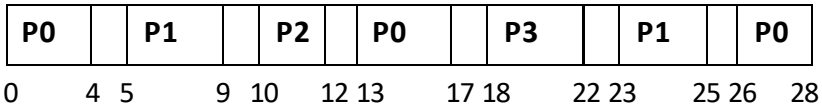
- **Selection Criteria:**
- Each selected process is assigned a time interval, called time quantum or time slice. Process is allowed to run only for this time interval. Here, two things are possible: First, Process is either blocked or terminated before the quantum has elapsed. In this case the CPU switching is done and another process is scheduled to run. Second, Process needs CPU burst longer than time quantum. In this case, process is running at the end of the time quantum. Now, it will be preempted and moved to the end of

the queue. CPU will be allocated to another process. Here, length of time quantum is critical to determine.

- **Decision Mode:** Preemptive:
- **Implementation :**
  This strategy can be implemented by using circular FIFO queue. If any process comes, or process releases CPU, or process is preempted. It is moved to the end of the queue. When CPU becomes free, a process from the first position in a queue is selected to run.
- **Example :**
  Consider the following set of four processes. Their arrival time and time required to complete the execution are given in the following table. All time values are in milliseconds. Consider that time quantum is of 4 ms, and context switch overhead is of 1 ms.

| Process | Arrival Time (T0) | Time required for completion (ΔT) |
|---|---|---|
| P0 | 0 | 10 |
| P1 | 1 | 6 |
| P2 | 3 | 2 |
| P3 | 5 | 4 |

- **Gantt Chart :**




- At 4ms, process P0 completes its time quantum. So it preempted and another process P1 is allowed to run. At 12 ms, process P2 voluntarily releases CPU, and another process is selected to run. 1 ms is wasted on each context switch as overhead. This procedure is repeated till all process completes their execution.

- **Statistics:**

| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (WT=TAT-ΔT) |
|---|---|---|---|---|---|
| P0 | 0 | 10 | 28 | 28 | 18 |

| | | | | | |
|---|---|---|---|---|---|
| P1 | 1 | 6 | 25 | 24 | 18 |
| P2 | 3 | 2 | 12 | 9 | 7 |
| P3 | 5 | 4 | 22 | 17 | 13 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (28+24+9+17)/4 | = 78 / 4 | = 19.5 ms |
| Average Waiting Time: | (18+18+7+13)/4 | = 56 / 4 | = 14 ms |

- **Advantages:**
  - One of the oldest, simplest, fairest and most widely used algorithms.
- **Disadvantages:**
  - Context switch overhead is there.
  - Determination of time quantum is too critical. If it is too short, it causes frequent context switches and lowers CPU efficiency. If it is too long, it causes poor response for short interactive process.

## V Non Preemptive Priority Scheduling:

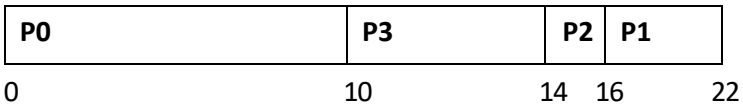
- **Selection criteria :**
  The process, that has highest priority, is served first.
- **Decision Mode:**
  Non Preemptive: Once a process is selected, it runs until it blocks for an I/O or some event, or it terminates.
- **Implementation :**
  This strategy can be implemented by using sorted FIFO queue. All processes in a queue are sorted based on their priority with highest priority process at front end. When CPU becomes free, a process from the first position in a queue is selected to run.
- **Example :**
  Consider the following set of four processes. Their arrival time, total time required completing the execution and priorities are given in following table. Consider all time values in millisecond and small values for priority means higher priority of a process.

| Process | Arrival Time (T0) | Time required for completion ( ΔT) | Priority |
|---|---|---|---|
| P0 | 0 | 10 | 5 |

| | | | |
|---|---|---|---|
| P1 | 1 | 6 | 4 |
| P2 | 3 | 2 | 2 |
| P3 | 5 | 4 | 0 |

Here, process priorities are in this order: P3 > P2 > P1 > P0.

- **Gantt Chart :**




- Initially only process P0 is present and it is allowed to run. But, when P0 completes, all other processes are present. So, process with highest priority P3 is selected and allowed to run till it completes. This procedure is repeated till all processes complete their execution.
- **Statistics :**

| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (TAT-ΔT) |
|---|---|---|---|---|---|
| P0 | 0 | 10 | 10 | 10 | 0 |
| P1 | 1 | 6 | 22 | 21 | 15 |
| P2 | 3 | 2 | 16 | 13 | 11 |
| P3 | 5 | 4 | 14 | 9 | 5 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (10+21+13+9) / 4 | = 53 / 4 | = 13.25 ms |
| Average Waiting Time: | (0+15+11+5) / 4 | = 31 / 4 | = 7.75 ms |

- **Advantages:**
  - Priority is considered. Critical processes can get even better response time.
- **Disadvantages:**
  - Starvation is possible for low priority processes. It can be overcome by using technique called 'Aging'.
  - Aging*:* gradually increases the priority of processes that wait in the system for a long time.

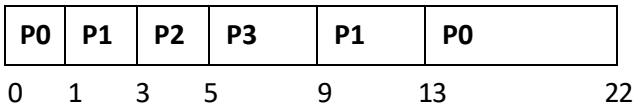
**Preemptive Priority Scheduling:**

- **Selection criteria :**
  The process, that has highest priority, is served first.
- **Decision Mode:**
  Preemptive: When a new process arrives, its priority is compared with current process priority. If the new job has higher priority than the current, the current process is suspended and new job is started.
- **Implementation :**
  This strategy can be implemented by using sorted FIFO queue. All processes in a queue are sorted based on priority with highest priority process at front end. When CPU becomes free, a process from the first position in a queue is selected to run.
- **Example :**
  Consider the following set of four processes. Their arrival time, time required completing the execution and priorities are given in following table. Consider all time values in milliseconds and small value of priority means higher priority of the process.

| Process | Arrival Time (T0) | Time required for completion ( ΔT) | Priority |
|---|---|---|---|
| P0 | 0 | 10 | 5 |
| P1 | 1 | 6 | 4 |
| P2 | 3 | 2 | 2 |
| P3 | 5 | 4 | 0 |

Here process priorities are in this order: P3>P2>P1>P0

- **Gantt chart:**




- Initially only process P0 is present and it is allowed to run. But when P1 comes, it has higher priority. So, P0 is preempted and P1 is allowed to run. This process is repeated till all processes complete their execution.
- **Statistics:**

| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (TAT-ΔT) |
|---|---|---|---|---|---|
| P0 | 0 | 10 | 22 | 22 | 12 |
| P1 | 1 | 6 | 13 | 12 | 6 |
| P2 | 3 | 2 | 5 | 2 | 0 |
| P3 | 5 | 4 | 9 | 4 | 0 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (22+12+2+4) / 4 | = 40 / 4 | = 10 ms |
| Average Waiting Time: | (12+6+0+0) / 4 | = 18 / 4 | = 4.5 ms |

- **Advantages:**
  - Priority is considered. Critical processes can get even better response time.
- **Disadvantages:**
  - Starvation is possible for low priority processes. It can be overcome by using technique called 'Aging'.
  - Aging: gradually increases the priority of processes that wait in the system for a long time.
  - Context switch overhead is there.

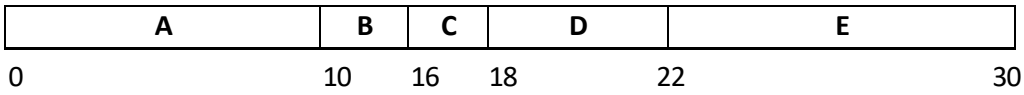
(2) *Five batch jobs A to E arrive at same time. They have estimated running times 10,6,2,4 and 8 minutes. Their priorities are 3,5,2,1 and 4 respectively with 5 being highest priority. For each of the following algorithm determine mean process turnaround time. Ignore process swapping overhead.*

*Round Robin, Priority Scheduling, FCFS, SJF.*

| Process | Running Time<br>Time required for completion ( ΔT) | Priority |
|---|---|---|
| A | 10 | 3 |
| B | 6 | 5 |
| C | 2 | 2 |
| D | 4 | 1 |

| | | |
|---|---|---|
| E | 8 | 4 |

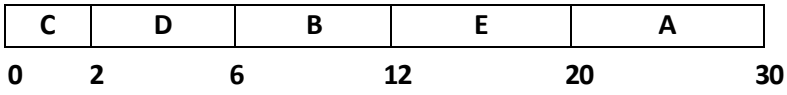
First Come First Served:




| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (TAT-ΔT) |
|---|---|---|---|---|---|
| A | 0 | 10 | 10 | 10 | 0 |
| B | 0 | 6 | 16 | 16 | 10 |
| C | 0 | 2 | 18 | 18 | 16 |
| D | 0 | 4 | 22 | 22 | 18 |
| E | 0 | 8 | 30 | 30 | 22 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (10+16+18+22+30) / 5 | = 96 / 5 | = 19.2 ms |
| Average Waiting Time: | (0+10+16+18+22) / 5 | = 56 / 5 | = 13.2 ms |

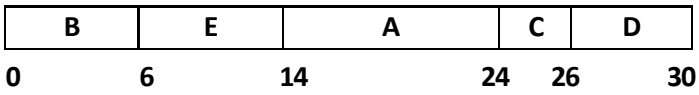
Shortest Job First:




| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (TAT-ΔT) |
|---|---|---|---|---|---|
| A | 0 | 10 | 30 | 30 | 20 |
| B | 0 | 6 | 12 | 12 | 6 |
| C | 0 | 2 | 2 | 2 | 0 |
| D | 0 | 4 | 6 | 6 | 2 |
| E | 0 | 8 | 20 | 20 | 12 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (30+12+2+6+20) / 5 | = 70 / 5 | = 14 ms |
| Average Waiting Time: | (20+6+0+2+12) / 5 | = 40 / 5 | = 8 ms |

Priority:

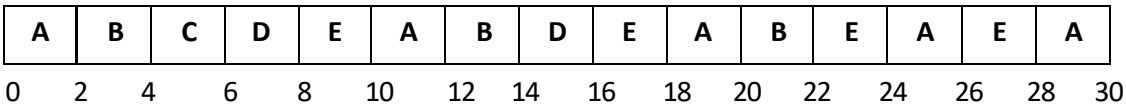



| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (TAT-ΔT) |
|---|---|---|---|---|---|
| A | 0 | 10 | 24 | 24 | 14 |
| B | 0 | 6 | 6 | 6 | 0 |
| C | 0 | 2 | 26 | 26 | 24 |
| D | 0 | 4 | 30 | 30 | 26 |
| E | 0 | 8 | 14 | 14 | 6 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (24+6+26+30+14) / 5 | = 100 / 5 | = 20 ms |
| Average Waiting Time: | (14+0+24+26+6) / 5 | = 70 / 5 | = 14 ms |

Round Robin:

Time slice OR Quantum time= 2min.



| Process | Arrival time (T0) | Completion Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT=T1-T0) | Waiting Time (TAT-ΔT) |
|---|---|---|---|---|---|
| A | 0 | 10 | 30 | 30 | 20 |
| B | 0 | 6 | 22 | 22 | 16 |
| C | 0 | 2 | 6 | 6 | 4 |
| D | 0 | 4 | 16 | 16 | 12 |
| E | 0 | 8 | 28 | 28 | 20 |

| | | | |
|---|---|---|---|
| Average Turnaround Time: | (30+22+6+16+28) / 5 | = 102 / 5 | = 20.4 ms |
| Average Waiting Time: | (20+16+4+12+20) / 5 | = 72 / 5 | = 14.4 ms |

(3) Suppose that the following processes arrive for the execution at the times indicated. Each process will run the listed amount of time. Assume preemptive scheduling.

| Process | Arrival Time (ms) | Burst Time (ms) |
|---|---|---|
| P1 | 0.0 | 8 |
| P2 | 0.4 | 4 |
| P3 | 1.0 | 1 |

What is the turnaround time for these processes with Shortest Job First scheduling algorithm?

(4) Consider the following set of processes with length of CPU burst time given in milliseconds.

| Process | Burst Time | Priority |
|---|---|---|
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

Assume arrival order is: P1, P2, P3, P4, P5 all at time 0 and a smaller priority number implies a higher priority. Draw the Gantt charts illustrating the execution of these processes using preemptive priority scheduling.