08. Operating Systems - PYQ Probability Matrix & Worked Traces

> **Target Assessment**: CoCubes / Capgemini / IT Corporate Placement Technical Modules
> **Weightage**: Operating Systems accounts for **20% to 25%** of all Technical MCQs (approx. 6–8 questions out of 30).

---

1. Operating Systems Topic Probability & Frequency Matrix

Rank	OS Sub-Topic	Probability / Frequency	Question Type	Primary Concepts & PYQ Patterns Tested
1	**CPU Scheduling**	**28% (VERY HIGH)**	Numerical / Gantt Chart	All 8 Schedulers: SRTF preemptive Gantt charts, Round Robin context switches & ready queue traces, SJF minimum waiting time, FCFS convoy effect, MLFQ queues.
2	**Page Replacement & Virtual Memory**	**24% (VERY HIGH)**	Numerical / Frame Trace	All 7 Page Replacements: FIFO, LRU, OPT, Second Chance Clock, Enhanced Second Chance (2-bit), LFU, MFU, Belady's Anomaly proof & frame allocation logic.
3	**Deadlocks & Banker's Algorithm**	**20% (HIGH)**	Matrix Calculation	$\text{Need} = \text{Max} - \text{Allocation}$, Safe State execution sequence verification, Resource Request Algorithm test, 4 Coffman conditions.
4	**Process Synchronization & Semaphores**	**14% (HIGH)**	Execution Trace / Code Analysis	Counting & Binary semaphores, `wait(S)` / `signal(S)` execution traces, Producer-Consumer, Readers-Writers, Dining Philosophers, Peterson's solution.
5	**Disk Scheduling**	**8% (MEDIUM)**	Numerical Track Distance	Total head movements for ALL 6 algorithms: FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK on identical track queues.
6	**Process States & Memory Layout**	**6% (MEDIUM)**	Conceptual MCQs	PCB contents, 5-state transition model, Stack vs Heap memory growth directions, system calls vs user mode.

---

2. Preemptive SRTF Scheduling (Gantt Chart & Metrics Trace)

Problem Setup

Consider 4 processes arriving at different time instances with their respective burst times:

Process	Arrival Time (AT)	Burst Time (BT)
P_1	0 ms	8 ms
P_2	1 ms	4 ms
P_3	2 ms	9 ms
P_4	3 ms	5 ms

Step-by-Step Execution Trace

1. **$t = 0 \text{ ms}$**:
2. P_1 arrives with $BT = 8$. P_1 starts execution.
3. **$t = 1 \text{ ms}$**:
4. P_2 arrives with $BT = 4$.
5. Remaining $BT(P_1) = 8 - 1 = 7$.
6. Compare $BT(P_2) = 4 < 7$. **Preempt P_1**, execute P_2.
7. **$t = 2 \text{ ms}$**:
8. P_3 arrives with $BT = 9$.
9. Currently running P_2 has remaining $BT = 3 < 9$. P_2 continues.
10. **$t = 3 \text{ ms}$**:
11. P_4 arrives with $BT = 5$.
12. Currently running P_2 has remaining $BT = 2 < 5$. P_2 continues.
13. **$t = 5 \text{ ms}$**:
14. P_2 completes ($CT(P_2) = 5$).
15. Ready Queue remaining burst times: $P_4 (5)$, $P_1 (7)$, $P_3 (9)$.
16. Smallest remaining burst time is $P_4 (5)$. Execute P_4.
17. **$t = 10 \text{ ms}$**:
18. P_4 completes ($CT(P_4) = 10$).
19. Ready Queue remaining: $P_1 (7)$, $P_3 (9)$. Execute P_1.
20. **$t = 17 \text{ ms}$**:
21. P_1 completes ($CT(P_1) = 17$).
22. Execute remaining $P_3 (9)$.
23. **$t = 26 \text{ ms}$**:
24. P_3 completes ($CT(P_3) = 26$).

Gantt Chart

```
+------+------+-------+-------+---------+
|  P1  |  P2  |   P4  |   P1  |    P3   |
```

```
+------+------+-------+-------+---------+
0      1      5       10      17        26
```

Complete Computation Matrix

- **Turnaround Time (TAT)**: $TAT = CT - AT$
- **Waiting Time (WT)**: $WT = TAT - BT$
- **Response Time (RT)**: $RT = \text{First Executed Time} - AT$

Process	AT	BT	Completion Time (CT)	Turnaround Time (TAT)	Waiting Time (WT)	Response Time (RT)
P_1	0	8	17	$17 - 0 = 17 \text{ ms}$	$17 - 8 = 9 \text{ ms}$	$0 - 0 = 0 \text{ ms}$
P_2	1	4	5	$5 - 1 = 4 \text{ ms}$	$4 - 4 = 0 \text{ ms}$	$1 - 1 = 0 \text{ ms}$
P_3	2	9	26	$26 - 2 = 24 \text{ ms}$	$24 - 9 = 15 \text{ ms}$	$17 - 2 = 15 \text{ ms}$
P_4	3	5	10	$10 - 3 = 7 \text{ ms}$	$7 - 5 = 2 \text{ ms}$	$5 - 3 = 2 \text{ ms}$
Average	--	--	--	**13.00 ms**	**6.50 ms**	**4.25 ms**

---

3. Round Robin (RR) Quantum Trace & Context Switching

Problem Setup

- **Time Quantum (Q)**: 2 ms
- Processes arriving over time:

Process	Arrival Time (AT)	Burst Time (BT)
P_1	0 ms	5 ms
P_2	1 ms	4 ms
P_3	2 ms	2 ms
P_4	4 ms	1 ms

Ready Queue Evolution & Timeline Trace

Time Window	Active Process	Remaining BT before slice	Execution Time	Remaining BT after slice	Ready Queue at End of Slice
$0 - 2 \text{ ms}$	P_1	5	2 ms	3	

Time Window	Active Process	Remaining BT before slice	Execution Time	Remaining BT after slice	Ready Queue at End of Slice
					`[P2, P3, P1]` *(P2 arrived at t=1, P3 at t=2)*
$2 - 4 \text{ ms}$	P_2	4	2 ms	2	`[P3, P1, P4, P2]` *(P4 arrived at t=4)*
$4 - 6 \text{ ms}$	P_3	2	2 ms	**0 (Terminates)**	`[P1, P4, P2]`
$6 - 8 \text{ ms}$	P_1	3	2 ms	1	`[P4, P2, P1]`
$8 - 9 \text{ ms}$	P_4	1	1 ms	**0 (Terminates)**	`[P2, P1]`
$9 - 11 \text{ ms}$	P_2	2	2 ms	**0 (Terminates)**	`[P1]`
$11 - 12 \text{ ms}$	P_1	1	1 ms	**0 (Terminates)**	`[]`

Gantt Chart

```
+------+------+------+------+----+------+----+
|  P1  |  P2  |  P3  |  P1  | P4 |  P2  | P1 |
+------+------+------+------+----+------+----+
0      2      4      6      8    9      11   12
```

Complete Computation Matrix

Process	AT	BT	Completion Time (CT)	Turnaround Time (TAT)	Waiting Time (WT)	Response Time (RT)
P_1	0	5	12	$12 - 0 = 12 \text{ ms}$	$12 - 5 = 7 \text{ ms}$	$0 - 0 = 0 \text{ ms}$
P_2	1	4	11	$11 - 1 = 10 \text{ ms}$	$10 - 4 = 6 \text{ ms}$	$2 - 1 = 1 \text{ ms}$
P_3	2	2	6	$6 - 2 = 4 \text{ ms}$	$4 - 2 = 2 \text{ ms}$	$4 - 2 = 2 \text{ ms}$
P_4	4	1	9	$9 - 4 = 5 \text{ ms}$	$5 - 1 = 4 \text{ ms}$	$8 - 4 = 4 \text{ ms}$

Process	AT	BT	Completion Time (CT)	Turnaround Time (TAT)	Waiting Time (WT)	Response Time (RT)
Average	--	--	--	**7.75 ms**	**4.75 ms**	**1.75 ms**

- **Total Context Switches**: **6 switches** (at $t = 2, 4, 6, 8, 9, 11$).

---

4. LRU Page Fault & Frame Trace Matrix

Problem Setup

- **Reference String**: `7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2`
- **Frame Allocation**: **3 Frames** (Initially empty)
- **Algorithm**: Least Recently Used (LRU) - Replaces the page that has not been used for the longest time in the past.

Step-by-Step Frame State Trace

Step	Page Reference	Frame 1	Frame 2	Frame 3	Status	Page Fault Count	Explanation / Recency Order
1	**7**	**7**	-	-	**Page Fault**	1	Initial load
2	**0**	7	**0**	-	**Page Fault**	2	Initial load
3	**1**	7	0	**1**	**Page Fault**	3	Initial load (Frames full)
4	**2**	**2**	0	1	**Page Fault**	4	Replaces `7` (LRU page)
5	**0**	2	0	1	**HIT**	4	Page `0` present. Recency: `0 > 2 > 1`
6	**3**	2	0	**3**	**Page Fault**	5	Replaces `1` (LRU page). Recency: `3 > 0 > 2`
7	**0**	2	0	3	**HIT**	5	Page `0` present. Recency: `0 > 3 > 2`
8	**4**	**4**	0	3	**Page Fault**	6	Replaces `2` (LRU page). Recency: `4 > 0 > 3`
9	**2**	4	0	**2**	**Page Fault**	7	Replaces `3` (LRU page). Recency: `2 > 4 > 0`
10	**3**	4	**3**	2		8	

Step	Page Reference	Frame 1	Frame 2	Frame 3	Status	Page Fault Count	Explanation / Recency Order
					Page Fault		Replaces `0` (LRU page). Recency: `3 > 2 > 4`
11	**0**	**0**	3	2	**Page Fault**	9	Replaces `4` (LRU page). Recency: `0 > 3 > 2`
12	**3**	0	3	2	**HIT**	9	Page `3` present. Recency: `3 > 0 > 2`
13	**2**	0	3	2	**HIT**	9	Page `2` present. Recency: `2 > 3 > 0`

Summary Statistics

- **Total References**: 13
- **Total Page Faults**: **9**
- **Total Page Hits**: **4**
- **Hit Ratio**: $\frac{4}{13} \approx \mathbf{30.77\%}$
- **Page Fault Ratio**: $\frac{9}{13} \approx \mathbf{69.23\%}$
- **Belady's Anomaly**: LRU is a **Stack Algorithm**; it is **strictly immune** to Belady's Anomaly (increasing frames will never increase page faults).

---

5. Banker's Algorithm Matrix & Safe State Verification

Problem Setup

- 5 Processes (P_0, P_1, P_2, P_3, P_4)
- 3 Resource Types (A, B, C) with total system instances: $A=10, B=5, C=7$.

Given Matrices:

Process	Allocation Matrix (A, B, C)	Max Matrix (A, B, C)	Need Matrix ($\text{Max} - \text{Allocation}$)
P_0	$(0, 1, 0)$	$(7, 5, 3)$	$(7, 4, 3)$
P_1	$(2, 0, 0)$	$(3, 2, 2)$	$(1, 2, 2)$
P_2	$(3, 0, 2)$	$(9, 0, 2)$	$(6, 0, 0)$
P_3	$(2, 1, 1)$	$(2, 2, 2)$	$(0, 1, 1)$
P_4	$(0, 0, 2)$	$(4, 3, 3)$	$(4, 3, 1)$

Initial Available Vector Calculation:

$$\text{Sum of Allocation} = (0+2+3+2+0, 1+0+0+1+0, 0+0+2+1+2) = (7, 2, 5)$$ $$\text{Available} = \text{Total} - \text{Sum of Allocation} = (10-7, 5-2, 7-5) = \mathbf{(3, 3, 2)}$$

Step-by-Step Safety Algorithm Execution Trace

1. **Initialize**: `Work = Available = (3, 3, 2)`, `Finish[i] = False` for all i.
2. **Step 1**: Check P_0: $\text{Need}(P_0) = (7, 4, 3) \le (3, 3, 2)$? **False**. Skip P_0.
3. **Step 2**: Check P_1: $\text{Need}(P_1) = (1, 2, 2) \le (3, 3, 2)$? **True!**
4. P_1 executes to completion and releases resources.
5. $\text{Work} = (3, 3, 2) + \text{Allocation}(P_1) = (3, 3, 2) + (2, 0, 0) = \mathbf{(5, 3, 2)}$.
6. `Finish[P1] = True`.
7. **Step 3**: Check P_3: $\text{Need}(P_3) = (0, 1, 1) \le (5, 3, 2)$? **True!**
8. P_3 executes to completion and releases resources.
9. $\text{Work} = (5, 3, 2) + \text{Allocation}(P_3) = (5, 3, 2) + (2, 1, 1) = \mathbf{(7, 4, 3)}$.
10. `Finish[P3] = True`.
11. **Step 4**: Check P_4: $\text{Need}(P_4) = (4, 3, 1) \le (7, 4, 3)$? **True!**
12. P_4 executes to completion and releases resources.
13. $\text{Work} = (7, 4, 3) + \text{Allocation}(P_4) = (7, 4, 3) + (0, 0, 2) = \mathbf{(7, 4, 5)}$.
14. `Finish[P4] = True`.
15. **Step 5**: Check P_0: $\text{Need}(P_0) = (7, 4, 3) \le (7, 4, 5)$? **True!**
16. P_0 executes to completion and releases resources.
17. $\text{Work} = (7, 4, 5) + \text{Allocation}(P_0) = (7, 4, 5) + (0, 1, 0) = \mathbf{(7, 5, 5)}$.
18. `Finish[P0] = True`.
19. **Step 6**: Check P_2: $\text{Need}(P_2) = (6, 0, 0) \le (7, 5, 5)$? **True!**
20. P_2 executes to completion and releases resources.
21. $\text{Work} = (7, 5, 5) + \text{Allocation}(P_2) = (7, 5, 5) + (3, 0, 2) = \mathbf{(10, 5, 7)}$.
22. `Finish[P2] = True`.

Result

The system is in a **SAFE STATE**.
Safe Sequence: $\mathbf{\langle P_1, P_3, P_4, P_0, P_2 \rangle}$ (or $\langle P_1, P_3, P_0, P_2, P_4 \rangle$).

Resource Request Algorithm Test Case

Question: If Process P_1 requests $Request(P_1) = (1, 0, 2)$, can the request be granted immediately?

1. **Condition 1**: $Request(P_1) = (1, 0, 2) \le \text{Need}(P_1) = (1, 2, 2)$? **True**.
2. **Condition 2**: $Request(P_1) = (1, 0, 2) \le \text{Available} = (3, 3, 2)$? **True**.
3. **Pretend Allocation**:
4. $\text{New Available} = (3, 3, 2) - (1, 0, 2) = \mathbf{(2, 3, 0)}$
5. $\text{New Allocation}(P_1) = (2, 0, 0) + (1, 0, 2) = \mathbf{(3, 0, 2)}$

6. $\text{New Need}(P_1) = (1, 2, 2) - (1, 0, 2) = \mathbf{(0, 2, 0)}$
7. **Re-test Safety**:
8. P_1 has $\text{Need} (0, 2, 0) \le (2, 3, 0)$ -> Runs! New Available = $(2, 3, 0) + (3, 0, 2) = (5, 3, 2)$.
9. P_3 runs next -> New Available = $(5, 3, 2) + (2, 1, 1) = (7, 4, 3)$.
10. P_4 runs next -> New Available = $(7, 4, 3) + (0, 0, 2) = (7, 4, 5)$.
11. P_0 runs next -> New Available = $(7, 4, 5) + (0, 1, 0) = (7, 5, 5)$.
12. P_2 runs next -> New Available = $(7, 5, 5) + (3, 0, 2) = (10, 5, 7)$.
13. **Conclusion**: System remains in a safe state. **Request can be granted IMMEDIATELY.**

---

6. Counting & Binary Semaphore Execution Trace

6.1 Binary Semaphore / Mutex Operation Mechanics

- `wait(S)` / `down(S)` / `P(S)`: $$S = S - 1$$ $$\text{if } S < 0 \implies \text{Block calling process, add to Semaphore Queue } Q$$
- `signal(S)` / `up(S)` / `V(S)`: $$S = S + 1$$ $$\text{if } S \le 0 \implies \text{Remove a process from Semaphore Queue } Q, \text{wake it to Ready state}$$

6.2 Step-by-Step Execution Trace of 3 Concurrent Processes

- Initial state: Binary Semaphore $S = 1$, Blocked Queue $Q = []$.

Event #	Process & Operation	Semaphore S Value	Action Taken	Active in CS	Blocked Queue (Q)
1	P_1 calls `wait(S)`	$1 - 1 = \mathbf{0}$	$S \ge 0 \implies$ Access Granted	P_1	`[]`
2	P_2 calls `wait(S)`	$0 - 1 = \mathbf{-1}$	$S < 0 \implies$ P_2 Blocked	P_1	`[P2]`
3	P_3 calls `wait(S)`	$-1 - 1 = \mathbf{-2}$	$S < 0 \implies$ P_3 Blocked	P_1	`[P2, P3]`
4	P_1 finishes CS, calls `signal(S)`	$-2 + 1 = \mathbf{-1}$	$S \le 0 \implies$ Unblock P_2	P_2	`[P3]`
5	P_2 finishes CS, calls `signal(S)`	$-1 + 1 = \mathbf{0}$	$S \le 0 \implies$ Unblock P_3	P_3	`[]`
6	P_3 finishes CS, calls `signal(S)`	$0 + 1 = \mathbf{1}$	$S > 0 \implies$ Queue empty	None	`[]`

Key Formula Insights for MCQs

- If semaphore value is **positive ($S > 0$)**: S represents the **number of available resources**.
- If semaphore value is **negative ($S < 0$)**: $|S|$ represents the **exact count of processes blocked in queue Q**.

7. All 6 Disk Scheduling Algorithms Benchmark

Benchmark Problem Setup

- **Disk Range**: Tracks `0` to `199` (Total 200 tracks).
- **Initial Head Position**: Track **53**
- **Pending Track Request Queue**: `98, 183, 37, 122, 14, 124, 65, 67`
- **Head Movement Direction**: Moving towards **larger track numbers (Right / towards 199)** for SCAN, C-SCAN, LOOK, and C-LOOK.

7.1 FCFS (First-Come, First-Served)

Services requests strictly in order of arrival.

- **Track Traversal Sequence**:
  `53 -> 98 -> 183 -> 37 -> 122 -> 14 -> 124 -> 65 -> 67`
- **Head Distance Calculations**: $$|98 - 53| = 45$$ $$|183 - 98| = 85$$ $$|37 - 183| = 146$$ $$|122 - 37| = 85$$ $$|14 - 122| = 108$$ $$|124 - 14| = 110$$ $$|65 - 124| = 59$$ $$|67 - 65| = 2$$
- **Total Head Movement**: $45 + 85 + 146 + 85 + 108 + 110 + 59 + 2 = \mathbf{640 \text{ tracks}}$

7.2 SSTF (Shortest Seek Time First)

Services the request closest to the current head position.

- **Track Traversal Sequence**:
  `53 -> 65 -> 67 -> 37 -> 14 -> 98 -> 122 -> 124 -> 183`
- **Head Distance Calculations**: $$|65 - 53| = 12$$ $$|67 - 65| = 2$$ $$|37 - 67| = 30$$ $$|14 - 37| = 23$$ $$|98 - 14| = 84$$ $$|122 - 98| = 24$$ $$|124 - 122| = 2$$ $$|183 - 124| = 59$$
- **Total Head Movement**: $12 + 2 + 30 + 23 + 84 + 24 + 2 + 59 = \mathbf{236 \text{ tracks}}$

7.3 SCAN (Elevator Algorithm)

Moves towards one end of the disk (199), servicing requests along the way, then reverses direction and services remaining requests.

- **Track Traversal Sequence**:
  `53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 199 (Boundary) -> 37 -> 14`
- **Head Distance Calculations**:
- Distance to right disk boundary (199): $|199 - 53| = 146$
- Return distance to lowest remaining request (14): $|199 - 14| = 185$
- **Total Head Movement**: $146 + 185 = \mathbf{331 \text{ tracks}}$

7.4 C-SCAN (Circular SCAN)

Moves from one end to the other end (199) servicing requests. When it reaches the end, it immediately returns to the opposite end (0) without servicing any requests on the return trip.

- **Track Traversal Sequence**:
  `53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 199 (Boundary) --(Full Return Jump)--> 0 (Boundary) -> 14 -> 37`
- **Head Distance Calculations**:
- Distance to right boundary: $|199 - 53| = 146$
- Jump distance across disk: $|199 - 0| = 199$
- Distance from 0 to 37: $|37 - 0| = 37$
- **Total Head Movement**: $146 + 199 + 37 = \mathbf{382 \text{ tracks}}$

7.5 LOOK

Similar to SCAN, but only goes as far as the **final request** in each direction (does NOT waste seek time going to the disk boundary `199`).

- **Track Traversal Sequence**:
  `53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 (Highest Req) -> 37 -> 14`
- **Head Distance Calculations**:
- Distance to highest request (183): $|183 - 53| = 130$
- Distance from 183 to lowest request (14): $|183 - 14| = 169$
- **Total Head Movement**: $130 + 169 = \mathbf{299 \text{ tracks}}$

7.6 C-LOOK (Circular LOOK)

Similar to C-SCAN, but only goes as far as the final request in the primary direction, then jumps directly to the lowest request without going to track `0` or `199`.

- **Track Traversal Sequence**:
  `53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 (Highest Req) --(Return Jump)--> 14 (Lowest Req) -> 37`
- **Head Distance Calculations**:
- Distance to highest request (183): $|183 - 53| = 130$
- Jump distance from 183 to 14: $|183 - 14| = 169$
- Distance from 14 to 37: $|37 - 14| = 23$
- **Total Head Movement**: $130 + 169 + 23 = \mathbf{322 \text{ tracks}}$

7.7 Unified Disk Scheduling Comparative Benchmark Table

Algorithm	Direction Sensitive?	Visits Disk Ends (0/199)?	Servicing on Return?	Total Head Movement	Rank / Efficiency
SSTF	No	No	Yes	**236 Tracks**	**1st (Minimum Seek Time)**
LOOK	Yes	**No**	Yes	**299 Tracks**	**2nd**
C-LOOK	Yes	**No**	No (Jump only)	**322 Tracks**	**3rd**
SCAN	Yes	**Yes**	Yes	**331 Tracks**	**4th**
C-SCAN	Yes	**Yes**	No (Jump only)	**382 Tracks**	**5th**
FCFS	No	No	N/A	**640 Tracks**	**6th (Worst)**