

Resource Management Technique (Operations Research) — Model Answers for Exam Revision

RESOURCE MANAGEMENT TECHNIQUE (OPERATIONS RESEARCH) — SEMESTER 5

[SECTION A — 2 Mark Questions](#)

[SECTION B — 5 Mark Questions](#)

[SECTION C — 10 Mark Questions](#)

RESOURCE MANAGEMENT TECHNIQUE (OPERATIONS RESEARCH) — SEMESTER 5

SECTION A — 2 Mark Questions

- 1. What is an Assignment Problem?** An assignment problem is a special type of Linear Programming Problem in which **n** resources (workers, machines) must be assigned to **n** tasks (jobs, machines) on a strict **one-to-one basis**, so that the total cost (or time) of completing all tasks is minimized (or total profit maximized). It is solved efficiently using the **Hungarian Method** rather than the general Simplex method, since its special structure (a balanced $n \times n$ cost matrix) allows a much faster solution technique.
- 2. What is PERT?** PERT (Programme Evaluation and Review Technique) is a network-based project-management technique used to plan, schedule, and control large, complex projects — particularly those involving **uncertain or variable** activity durations (e.g., research and development projects). It uses three time estimates for each activity (optimistic, most likely, pessimistic) to calculate an **expected time** and helps identify the critical path and the probability of completing the project by a target date.
- 3. What is / Define Transportation Problem** A transportation problem is a special type of LPP concerned with finding the most economical way of **transporting a single homogeneous commodity** from several supply points (sources/factories with fixed availability) to several demand points (destinations with fixed requirements), so as to **minimize total transportation cost** while satisfying all supply and demand constraints.
- 4. What is Game Theory?** Game Theory is a mathematical technique used to analyze **competitive situations (conflict of interest)** between two or more rational decision-makers (players), each trying to maximize their own gain (or minimize their loss), where the outcome for each player depends not only on their own strategy but also on the strategy chosen by the opponent(s).
- 5. What is the Monte Carlo Method?** The Monte Carlo method is a **simulation technique** that uses random numbers (typically generated using random number tables or computer-generated pseudo-random numbers, mapped to a probability distribution) to model and analyze systems that involve uncertainty or randomness, allowing an analyst to estimate the likely behaviour/performance of a real system by repeatedly simulating it, without needing to experiment on the real system itself.
- 6. What is the Simplex Method?** The Simplex Method is an iterative, algebraic algorithmic procedure used to solve **Linear Programming Problems** having more than two decision variables. Starting from an initial basic feasible solution (usually at the origin), it systematically moves from one corner point (vertex) of the feasible region to an adjacent one that improves the objective function value, until the optimal solution is reached.
- 7. What is the Graphical Method?** The graphical method is a technique for solving Linear Programming Problems that involve **only two decision variables**, by plotting the constraints as straight lines on a graph, identifying the feasible region (the area satisfying all constraints simultaneously), and then locating the corner point of that region which gives the maximum or minimum value of the objective function.
- 8. What is CPM?** CPM (Critical Path Method) is a network-based project-scheduling technique used mainly for projects with **well-known, fairly certain (deterministic)** activity durations, such as construction or repetitive engineering projects. It identifies the **critical path** — the longest sequence of dependent activities through the network — which determines the minimum total time required to complete the project, and highlights which activities have zero slack (float) and therefore cannot be delayed without delaying the whole project.
- 9. What is LPP? (Define Linear Programming Problem)** A Linear Programming Problem is a mathematical technique for determining the optimal allocation of limited resources among competing activities, in order to **maximize or minimize** a linear objective function, subject to a set of linear constraints (equalities/inequalities) and non-negativity restrictions on the decision variables.
- 10. What is a Sequencing Problem?** A sequencing problem deals with determining the **optimal order (sequence)** in which a given number of jobs should be processed through a given number of machines, so as to minimize the total elapsed time (makespan) or the idle time of the machines, given that each job must pass through the machines in a specified order.
- 11. What is Duality / the Dual Problem?** For every Linear Programming Problem (called the **Primal**),

there exists a related LPP called its **Dual**, constructed by transposing the roles of the primal's constraints and variables — the primal's objective coefficients become the dual's constraint RHS values and vice versa, and a maximization primal becomes a minimization dual (or vice versa). The Duality Theorem states that the optimal objective value of the primal always equals the optimal objective value of its dual, and the dual is often easier or more meaningful to solve/interpret (e.g., dual variables represent the “shadow price” of each primal resource).

SECTION B — 5 Mark Questions

1. Solving an Assignment Problem (Numeric Method) The **Hungarian Method** is used to solve a balanced assignment problem (n workers, n jobs) in the following steps: 1. **Row Reduction:** Subtract the smallest element in each row from every element of that row. 2. **Column Reduction:** Subtract the smallest element in each column (of the row-reduced matrix) from every element of that column. 3. **Zero Covering:** Draw the minimum number of horizontal/vertical lines needed to cover all the zeros in the resulting matrix. 4. **Optimality Check:** If the number of lines equals n (the size of the matrix), an optimal assignment exists among the zeros — proceed to step 6. If fewer than n lines are needed, proceed to step 5. 5. **Matrix Adjustment:** Find the smallest uncovered element; subtract it from all uncovered elements, and add it to elements covered by **two** lines (intersections); elements covered by only one line remain unchanged. Return to step 3. 6. **Assignment:** Select zeros such that exactly one zero is chosen in each row and each column (no two chosen zeros share a row or column) — this gives the optimal assignment; sum the corresponding original costs for the minimum total cost.

Worked Example (frequently repeated data): Cost matrix: | | Job1 | Job2 | Job3 | Job4 | |—|—|—|—| | A | 42 | 35 | 28 | 21 | | B | 30 | 25 | 20 | 15 | | C | 30 | 25 | 20 | 15 | | D | 24 | 20 | 16 | 12 |

Applying the Hungarian steps above (row & column reduction, then covering/adjustment) leads to the optimal assignment: **A→Job4 (21), B→Job3 (20), C→Job2 (25), D→Job1 (24)** — Total Minimum Cost = **90**. (Note: since rows B and C are identical, B↔Job2/C↔Job3 swapped is an equally valid alternate optimal solution — also totaling 90 — a useful point to mention if the exam matrix has repeated rows.)

2. Dual of a Given LPP To write the dual of a primal LPP: 1. If the primal is a **maximization** problem with $\leq$ constraints, its dual will be a **minimization** problem with $\geq$ constraints (and vice versa for a minimization primal with $\geq$ constraints becoming a maximization dual with $\leq$ constraints). 2. The number of **variables** in the dual equals the number of **constraints** in the primal, and the number of **constraints** in the dual equals the number of **variables** in the primal. 3. The **RHS values** of the primal constraints become the **objective function coefficients** of the dual. 4. The **objective function coefficients** of the primal become the **RHS values** of the dual constraints. 5. The **coefficient matrix** of the dual is the **transpose** of the primal's coefficient matrix.

Example: Primal: Maximize $Z = 3x_1 + 5x_2$, subject to $x_1 + 2x_2 \leq 14$, $3x_1 - x_2 \leq 6$, $x_1, x_2 \geq 0$. Dual: Minimize $G = 14y_1 + 6y_2$, subject to $y_1 + 3y_2 \geq 3$, $2y_1 - y_2 \geq 5$, $y_1, y_2 \geq 0$.

Duality is useful because solving whichever of the two problems is computationally simpler (fewer constraints) automatically gives the optimal value of the other, and dual variable values represent the **marginal (shadow) value** of each primal resource.

3. Travelling Salesman Problem (TSP) The Travelling Salesman Problem asks for the sequence in which a salesman should visit n cities exactly once each, returning finally to the starting city, so that the **total distance/cost travelled is minimized**. It differs from the ordinary assignment problem because a simple assignment solution might create disconnected smaller loops (sub-cycles) rather than one single tour covering all cities.

Solution approach: TSP is commonly solved by first solving it **as an assignment problem** (using the Hungarian method) — if the resulting optimal assignment happens to form a single complete cycle through all cities, that is the TSP solution. If it instead breaks into two or more disjoint sub-cycles, an additional restriction is imposed: the cheapest link causing a sub-cycle is deleted/penalized (set to a very high cost, ∞) and the assignment problem is re-solved — this is repeated until a single continuous tour visiting every city exactly once is obtained.

4. Network Diagram / Critical Path (Part B version) A project network diagram represents a project's activities as arrows (or nodes, depending on convention) connected according to their logical sequence and dependency relationships. *Steps to find the critical path:* 1. Draw the network showing all activities and events (numbered nodes) in correct logical/precedence order. 2. Compute the **Earliest Start (ES)** and **Earliest Finish (EF)** time for every activity by a **forward pass** through the network ($EF = ES + \text{duration}$; ES of an activity = maximum EF of all its immediately preceding activities). 3. Compute the **Latest Start (LS)**

and **Latest Finish (LF)** time for every activity by a **backward pass**, starting from the project's final EF and working backward ($LS = LF - \text{duration}$). 4. Calculate the **Total Float/Slack** for each activity = $LS - ES$ (or $LF - EF$); activities with **zero float** lie on the **Critical Path**. 5. The Critical Path is the longest path (by duration) from the start node to the end node through the network, and its total length gives the **minimum possible project completion time**.

5. Solve / Analyze a Game Pay-off Matrix General procedure for solving a two-person zero-sum game: 1. **Check for a Saddle Point:** find the minimum value in each row (row minima) and take the **maximum** of these (maximin); find the maximum value in each column (column maxima) and take the **minimum** of these (minimax). If $\text{maximin} = \text{minimax}$, a **saddle point** exists — this common value is the **value of the game**, and each player's optimal (pure) strategy is the row/column at which it occurs. 2. **If No Saddle Point:** the game has no stable pure-strategy solution, so players must adopt **mixed strategies** (assigning probabilities to each of their available pure strategies). For a 2×2 game, a direct formula gives the optimal probabilities and value. For a $2 \times n$ or $m \times 2$ game, the **graphical method** is used: plotting each pure strategy of the larger player as a line (function of the opposing player's mixed probability p) and finding the point that maximizes the lower envelope (for the maximizing player) or minimizes the upper envelope (for the minimizing player). 3. **Dominance Rule** can often simplify a larger matrix first — if every payoff in one row is $\leq$ the corresponding payoff in another row, the smaller row is dominated and can be deleted (similarly for columns), before applying steps 1–2.

Worked Example (saddle point case — frequently repeated matrix):

	B1	B2	B3	B4	Row Min
A1	1	7	3	4	1
A2	5	6	4	5	4
A3	7	2	0	3	0
Col Max	7	7	4	5	

$\text{Maximin} = 4$ (Row A2), $\text{Minimax} = 4$ (Column B3) → **Maximin = Minimax = 4**, so a **saddle point exists** at (A2, B3). Optimal strategy: Player A always plays A2, Player B always plays B3; **Value of the Game = 4**.

6. Transportation Problem — Initial Solution (Vogel's Approximation Method) Vogel's Approximation Method (VAM) generally gives a better (often optimal or near-optimal) initial basic feasible solution than simpler methods like North-West Corner or Least Cost. Steps: 1. For each **row**, calculate the **penalty** = difference between the smallest and the second-smallest cost in that row. Do the same for each **column**. 2. Identify the row or column with the **largest** penalty; within that row/column, allocate the maximum possible quantity to the cell having the **lowest cost**, fully satisfying whichever of the corresponding supply or demand is smaller. 3. Cross out (eliminate) the row or column whose supply/demand has become zero, and recompute the penalties for the remaining reduced table. 4. Repeat steps 1–3 until all supplies and demands are exhausted. 5. The result is the initial feasible solution, which can then be tested and improved for optimality using the **Modified Distribution Method (MODI)**, if required.

7. Monte-Carlo Simulation / Techniques Monte Carlo simulation models a real system that involves an element of chance/uncertainty by using random numbers to generate representative sample outcomes, repeated over many trials to study its overall probable behaviour. (See Section C, Q on Monte Carlo Simulation Steps for the full detailed procedure, since this identical question also appears as a 10-mark question in several papers.)

SECTION C — 10 Mark Questions

1. Duality — Use Duality to Solve an LPP *Rules for writing the dual (recap):* the number of dual variables equals the number of primal constraints; the primal's RHS values become the dual's objective coefficients; the primal's objective coefficients become the dual's RHS values; the constraint matrix is transposed; and a $\text{max}/\leq$ primal converts to a $\text{min}/\geq$ dual (and vice versa).

Worked Example: Primal: Maximize $Z = 3x_1 + 5x_2$ subject to: $x_1 + 2x_2 \leq 14$; $3x_1 - x_2 \leq 6$; $x_1, x_2 \geq 0$

Dual: Minimize $G = 14y_1 + 6y_2$ subject to: $y_1 + 3y_2 \geq 3$; $2y_1 - y_2 \geq 5$; $y_1, y_2 \geq 0$

Solving the dual by the Simplex Method (converting $\geq$ constraints using surplus variables and the Big-M or Two-Phase method as required) will, by the **Fundamental Duality Theorem**, yield an optimal value $G^* = Z^*$ (equal to the primal's optimal value), and the optimal dual variable values (y_1^*, y_2^*) can be read directly from the final primal Simplex table's z-row under the slack-variable columns (they represent the **shadow prices** — the marginal value of one additional unit of each primal resource). Because the dual here has only 2 constraints (versus the primal's 2 constraints but different variable structure), whichever formulation has fewer constraints/simpler $\geq$ requirements is generally chosen to reduce the computational effort of applying Simplex.

Exam tip: always double check by solving the smaller/simpler of the primal-dual pair directly (e.g. via Simplex or graphically if it has only 2 variables), then verify that $Z^* = G^*$ as required by the theorem —

examiners frequently award marks for correctly stating and using this equality check.

2. Sequencing Problems A sequencing problem finds the order of processing **n jobs through m machines** (each job passing through the machines in the same fixed order) that minimizes total elapsed time. For **n jobs and 2 machines**, **Johnson's Rule** gives the optimal sequence directly: 1. List the processing time of every job on Machine 1 and Machine 2. 2. Find the smallest processing time in the whole table. If it belongs to Machine 1, schedule that job as early as possible (leftmost available position); if it belongs to Machine 2, schedule it as late as possible (rightmost available position). 3. Remove that job from consideration and repeat until all jobs are placed.

For **n jobs and m > 2 machines**, this can only be reduced to an equivalent 2-machine problem (and Johnson's Rule applied) if either: **min(Machine 1 times) ≥ max(times on all intermediate machines)**, OR **min(Machine m times) ≥ max(times on all intermediate machines)**. If so, two fictitious machines G and H are created: G = sum of times on Machine 1 through Machine (m-1); H = sum of times on Machine 2 through Machine m — then Johnson's Rule is applied to (G, H).

Worked Example (4 Jobs, 6 Machines — frequently repeated data): | Job | M1 | M2 | M3 | M4 | M5 | M6 | |—|—| —|—|—|—|—| | A | 19 | 8 | 8 | 3 | 11 | 24 | | B | 18 | 6 | 9 | 6 | 9 | 18 | | C | 12 | 5 | 8 | 5 | 7 | 15 | | D | 20 | 5 | 3 | 4 | 8 | 11 |

Check: min(M1) = 12 (Job C); max(M2,M3,M4,M5 across all jobs) = 11 (Job A's M5). Since $12 \geq 11$, the condition holds — reduce to two machines: **G = M1+M2+M3+M4+M5**, **H = M2+M3+M4+M5+M6**

Job	G	H
A	49	54
B	48	48
C	37	40
D	40	31

Applying Johnson's Rule: smallest value overall is D's H = 31 → place D **last**. Next smallest is C's G = 37 → place C **first**. Next is B (G=H=48, tie) → place next available front slot. Finally A → remaining slot.

Optimal Sequence: C → B → A → D

Computing actual completion times across all 6 machines for this sequence gives a **Total Elapsed Time (Makespan) = 114 units**, with Machine 6 (the last/bottleneck machine) having an **idle time = 114 – (15+18+24+11) = 46 units** (total elapsed time minus the sum of Job processing times on that machine).

3. Critical Path / Network Diagrams Method (applicable to any project network given in the exam): 1. Draw the network of activities in their correct logical (precedence) order, with numbered event nodes and each activity shown as an arrow labelled with its duration. 2. **Forward Pass** — compute Earliest Start (ES) and Earliest Finish (EF) for every activity, moving from the start node to the end node: EF = ES + duration; the ES of an activity = the **maximum** EF among all activities immediately preceding it (this handles merging paths correctly). 3. **Backward Pass** — compute Latest Finish (LF) and Latest Start (LS) for every activity, moving from the end node back to the start node: LS = LF – duration; the LF of an activity = the **minimum** LS among all activities immediately following it. 4. **Total Float (Slack)** for each activity = LS – ES (equivalently LF – EF). Activities with **zero float** cannot be delayed at all without delaying the whole project — the chain of such zero-float activities from start to end is the **Critical Path**. 5. The length of the critical path = the **minimum total project duration**.

Illustrative worked example (since the exact diagram in the question paper isn't reproduced here, apply the identical 5-step method above to whatever specific network is given — the procedure never changes): Suppose a project has activities with durations 10, 8, 9 leading into a stage with durations 8, 16, 7, which then lead into a final stage with durations 7, 6, 12, 13, 5, 15. Applying the forward pass along every path from start to finish, summing durations along each distinct path, and taking the **longest** path length gives the project duration; the backward pass then confirms which specific activities have zero slack and therefore make up that critical path. (Exam tip: always redraw the exact network exactly as given in your question paper and mechanically apply steps 2-4 above — the numeric answer is entirely determined by the specific diagram provided.)

4. Transportation Problem (Minimum Cost) Steps to solve a transportation problem for minimum total cost: 1. Check that the problem is **balanced** (total supply = total demand); if not, add a dummy row/column with zero cost to balance it. 2. Find an **Initial Feasible Solution** using North-West Corner Rule, Least Cost Method, or (preferably) **Vogel's Approximation Method (VAM)** for a solution closer to optimal. 3. Check that the number of allocated (basic) cells equals **(rows + columns – 1)**; if fewer, the solution is degenerate and a very small allocation (ε) must be added to an appropriate independent cell. 4. Test the current solution for optimality using the **Modified Distribution Method (MODI)**: compute row (ui) and column (vj) numbers

such that $u_i + v_j = \text{cost}$ for every allocated cell (setting one $u_i = 0$ to start), then calculate the **opportunity cost** for every unallocated cell as $(c_{ij} - u_i - v_j)$. 5. If all opportunity costs are ≥ 0 , the current solution is optimal — stop. If any cell has a **negative** opportunity cost, the solution can be improved: trace the **closed loop (stepping-stone path)** through that cell, reallocate units around the loop (adding/subtracting alternately) by the maximum amount permitted, and repeat from step 4 with the new solution. 6. When all opportunity costs are non-negative, the total cost of the current allocation is the **minimum transportation cost**.

5. Game Theory (Solve Pay-off Matrix) (See Section B, Q5 for the general saddle-point/dominance/graphical procedure.)

Worked Example — No Saddle Point, solved via the Graphical Method (frequently repeated 2×3 matrix): | |
 B1 | B2 | B3 | |—|—|—| | A1 | 3 | -3 | 4 | | A2 | -1 | 1 | -3 |

Row minima: $A1 = -3, A2 = -3 \rightarrow \text{Maximin} = -3$. Column maxima: $B1 = 3, B2 = 1, B3 = 4 \rightarrow \text{Minimax} = 1$. Since $\text{Maximin} (-3) \neq \text{Minimax} (1)$, **no saddle point exists** — a mixed strategy is required.

Let Player A play $A1$ with probability p and $A2$ with probability $(1-p)$. The expected payoff to A against each of B's pure strategies is: - $E1(p) = 3p - 1(1-p) = 4p - 1$ - $E2(p) = -3p + 1(1-p) = -4p + 1$ - $E3(p) = 4p - 3(1-p) = 7p - 3$

Plotting these three lines against $p \in [0,1]$ and finding the **highest point of the lower envelope** (Player A wants to maximize the minimum of these), the binding pair is $E2$ and $E3$. Setting them equal: $7p - 3 = -4p + 1 \rightarrow 11p = 4 \rightarrow p = 4/11$ (so A plays $A1$ w.p. $4/11$, $A2$ w.p. $7/11$)

Value of the game at this point: $v = 7(4/11) - 3 = 28/11 - 33/11 = -5/11$

Since column $B1$'s line is never part of the lower envelope at the optimal point, **$B1$ is not used** by Player B. Reducing to the 2×2 sub-game of columns $B2, B3$ and solving for B's mixed strategy $(q, 1-q)$ by equalizing A's expected payoffs: $4 - 7q = 4q - 3 \rightarrow q = 7/11$

Optimal Strategies: Player A $\rightarrow (A1: 4/11, A2: 7/11)$; Player B $\rightarrow (B1: 0, B2: 7/11, B3: 4/11)$; **Value of Game** = **$-5/11$** (negative, indicating the game favours Player B).

6. Simplex Method (Solve LPP) Steps: 1. **Standard Form:** Convert all inequality constraints to equations by adding **slack variables** (for $\leq$ constraints) or subtracting **surplus variables** (for $\geq$ constraints); for a minimization problem or $\geq$ constraints, artificial variables and the **Big-M** (or Two-Phase) method may be needed. 2. Set up the **initial Simplex table**, with the slack/artificial variables forming the initial basic feasible solution (typically all decision variables = 0 to start). 3. Compute the **net evaluation row ($z_j - c_j$)**. If, for a maximization problem, all values in this row are ≥ 0 , the current solution is already optimal — stop. Otherwise, select the most negative entry to identify the **entering variable** (the column to bring into the basis). 4. Perform the **minimum ratio test** ($\text{RHS} \div \text{corresponding positive coefficient in the entering column}$) to identify the **leaving variable** (the row/pivot row). 5. Perform a **pivot operation** (Gauss-Jordan elimination) to make the entering variable's column a unit column, updating the entire table. 6. Repeat steps 3-5 until the optimality condition is satisfied; the values in the RHS column then give the optimal values of the basic variables, and the value in the Z-row/RHS cell gives the optimal objective value.

Worked Example: Maximize $Z = 3x_1 + 5x_2$, subject to: $x_1 + 2x_2 \leq 14$; $3x_1 - x_2 \leq 6$; $x_1, x_2 \geq 0$. Adding slack variables s_1, s_2 : $x_1 + 2x_2 + s_1 = 14$; $3x_1 - x_2 + s_2 = 6$. Starting with $x_1 = x_2 = 0$ ($s_1=14, s_2=6$), successive Simplex iterations (selecting entering/leaving variables per steps 3-4 above) lead to the optimal corner point at **$x_1 = 26/7, x_2 = 40/7$** , giving **$Z^* = 278/7 \approx 39.71$** (exact values depend on the specific coefficients given in the exam's version of the problem — always re-run the same 6 steps mechanically on the given data).

7. PERT Estimation / Project Duration PERT handles activities whose duration is **uncertain**, by requiring three time estimates for every activity: - **Optimistic time (t_o)** — the shortest possible time if everything goes exceptionally well. - **Most likely time (t_m)** — the normal, most probable duration. - **Pessimistic time (t_p)** — the longest time if significant difficulties are encountered.

Expected time for each activity is calculated using the Beta-distribution-based weighted average formula: **$t_e = (t_o + 4t_m + t_p) / 6$**

Variance of each activity's duration (indicating uncertainty) is calculated as: **$\sigma^2 = ((t_p - t_o) / 6)^2$**

Procedure to estimate overall project duration: 1. Calculate t_e for every activity using the formula above. 2. Construct the project network using these expected times, exactly as in CPM, and perform the forward/backward pass to find the **Critical Path** and hence the **Expected Project Duration (T_e)** — the sum of t_e values along the critical path. 3. Sum the **variances** of only the critical-path activities to get the overall **project variance**, and take its square root to get the project's standard deviation (σ). 4. Using the

Normal distribution (since the sum of many independent activity times tends to be normally distributed by the Central Limit Theorem), calculate the **probability of completing the project by a given target date (Ts)** using the Z-score formula: $Z = (Ts - Te) / \sigma$, then look up the corresponding probability from the standard normal table.

This lets project managers not only estimate when a project is *likely* to finish, but also state a **confidence level/probability** of meeting any particular target completion date.

8. Assignment Problem (Solve) (Method as detailed in Section B, Q1 — the Hungarian Algorithm.)

Fully Worked Example (frequently repeated 4x4 data): | | Job1 | Job2 | Job3 | Job4 | |—|—|—|—| | A | 42 | 35 | 28 | 21 | | B | 30 | 25 | 20 | 15 | | C | 30 | 25 | 20 | 15 | | D | 24 | 20 | 16 | 12 |

Step 1 — Row reduction (subtract row minimum):

21	14	7	0
15	10	5	0
15	10	5	0
12	8	4	0

Step 2 — Column reduction (subtract column minimum):

9	6	3	0
3	2	1	0
3	2	1	0
0	0	0	0

Step 3 — Cover zeros: only 2 lines (Row D and Column 4) are needed to cover all zeros — since this is fewer than $n = 4$, the solution is not yet optimal.

Step 4 — Adjust: subtract the smallest uncovered value (=1) from all uncovered cells, add it to doubly-covered cells, giving:

8	5	2	0
2	1	0	0
2	1	0	0
0	0	0	1

Re-covering still needs only 3 lines (max matching = 3), so one more adjustment round (subtract smallest uncovered value = 1 again) gives:

7	4	2	0
1	0	0	0
1	0	0	0
0	0	2	2

Step 5 — Optimal Assignment found (4 lines now needed): A → Job4, B → Job2, C → Job3, D → Job1

Total Minimum Cost = 21 + 25 + 20 + 24 = 90

(Note: because Rows B and C have identical costs, swapping B ↔ Job3 and C ↔ Job2 gives an alternative, equally optimal solution also totalling 90 — an important point to mention in the exam answer when a matrix has duplicate rows/columns, as it shows awareness of multiple optimal solutions.)