inkMaths

ANSWER BOOK

Allocation and the Hungarian algorithm

Decision Mathematics 2 · Year FM

6 questions · 25 marks

NAVY EDITION

inkphysics.co.uk/maths

SECTION A · Warm-up

A1 Every complete allocation uses exactly one cell from each row. Subtracting the same amount from a whole row therefore reduces the total of every allocation by that same amount, so their order is unchanged and the cheapest one before the subtraction is still the cheapest after it. M1 for the observation that every allocation uses one cell from each row, A1 for the conclusion that the ordering is unchanged. The same argument applies to columns, since every allocation uses one cell from each column too. [2]

A2 Four is fewer than five, so no allocation using only zeros exists yet. Find the smallest uncovered entry, subtract it from every uncovered element, and add it to every element covered by two lines. Then count the minimum covering lines again. M1 for the smallest uncovered entry, A1 for the augmenting step described in full. [2]

SECTION B · Standard

B1 Row minima are 9, 8 and 9, so the row-reduced matrix is **5,** 0, 3; 3, 0, 7; 4, 1, 0. [6]
Its column minima are 3, 0 and 0, so the column-reduced matrix is **2,** 0, 3; 0, 0, 7; 1, 1, 0.
Write both matrices down; the method marks live in them, and an answer with only the final allocation scores almost nothing.
The zeros need **3** lines to cover, which matches the size of the matrix, so a complete allocation of zeros exists: **P** to 2, Q to 1, R to 3.
Read the cost off the original table, never the reduced one: 9 + 11 + 9 = **29**.
M1 A1 for the row reduction, M1 A1 for the column reduction, A1 for the allocation, A1 for the cost.

B2 The algorithm only minimises, so first turn the profits into costs. The largest entry is **15**, so subtract every entry from 15 to get 1, 6, 3; 4, 7, 0; 2, 5, 6. Subtracting from anything smaller than the largest entry leaves negatives and loses the first method mark. [5]
Row minima 1, 0, 2 then column minima 0, 3, 0 give **0,** 2, 2; 4, 4, 0; 0, 0, 4, whose zeros need 3 lines.
The allocation is **P** to 1, Q to 3, R to 2, with profit 14 + 15 + 10 = **39**.
M1 A1 for turning the profits into costs, M1 for the row and column reduction, A1 for the allocation, A1 for the profit.
As a check, the opportunity cost lost is 1 + 0 + 5 = 6 and 3(15) - 39 = 6.

B3 Add a **dummy** job, a fourth column of zeros, so the matrix is square. Whoever is allocated the dummy is the worker left idle, at no cost. [3]
For the refusal, give that one cell a cost **large** enough that the algorithm will never choose it, and say in your working that you have done so. The rest of the method runs unchanged. B1 for the dummy column of zeros, M1 A1 for the prohibitive cost in the refused cell.

SECTION C · Stretch

C1 Row minima 6, 9, 5 give 0, 5, 5; 0, 8, 4; 7, 2, 0. Column minima 0, 2, 0 then give **0,** 3, 5; 0, 6, 4; 7, 0, 0. [7]
Every zero is covered by column 1 together with row 3, so only **2** lines are needed against a size of 3, and the test fails. State the number of lines explicitly; that sentence is a mark on its own.
The uncovered entries are 3, 5, 6 and 4, so the smallest is **3**. Subtract 3 from every uncovered entry and add 3 to the entry covered twice, which is the 7 at the foot of column 1, giving **0,** 0, 2; 0, 3, 1; 10, 0, 0. Forgetting the addition at the crossing point is the usual slip, and it leaves a matrix with no valid allocation.
Three lines are now needed, so an allocation exists: **P** to 2, Q to 1, R to 3, costing 11 + 9 + 5 = **25**.
M1 A1 for the row and column reduction, B1 for stating that only two lines are needed, M1 for subtracting the smallest uncovered entry and adding it at the crossing, A1 for the augmented matrix, A1 for the allocation, A1 for the cost.
Checking all six allocations confirms 25 is the least.