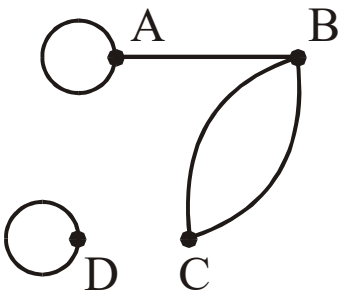
The Hungarian algorithm and the optimal assignment problem

Graphs

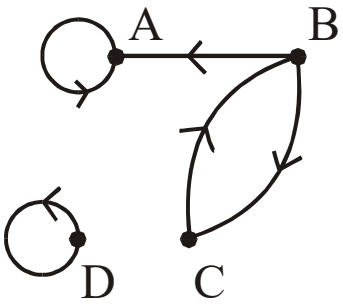
You are reminded that a *graph* is a line drawing connecting vertices to one another by means of edges. An edge is a line joining two vertices.

An *incidence table* shows the number of connections between vertices.



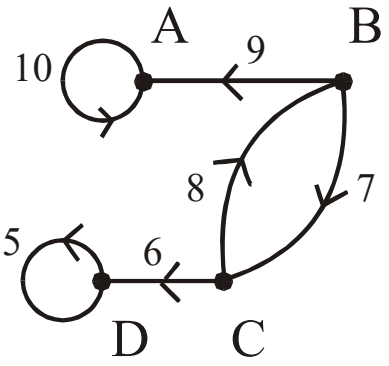
	A	**B**	**C**	**D**
A	1	1	0	0
B	1	0	2	0
C	0	2	0	0
D	0	0	0	1

A *diagraph* is a graph where the edges are directed.



	A	**B**	**C**	**D**
A	1	0	0	0
B	1	0	1	0
C	0	1	0	0
D	0	0	0	1

A *weighted digraph* is one in which a number is associated with every edge.



	A	B	C	D
A	10	-	-	-
B	9	-	7	-
C	-	8	-	6
D	-	-	-	5

Problems on weighted digraphs include

1. Finding the minimum spanning tree by means of Prim or Kruskal's algorithm.
2. Finding the shortest path from one vertex to another by means of dynamic programming or Dijkstra's algorithm.

Optimal Assignment Problem

A *simple graph* is a graph with no parallel edges and loops.

A *bipartite graph* is a simple graph in which the sets of vertices can be partitioned into two sets, *X* and *Y*, such that every edge is between a vertex in *X* and a vertex in *Y*.

A *weighted bipartite graph* is one in which each edge of the graph is assigned a real number which is its weight.

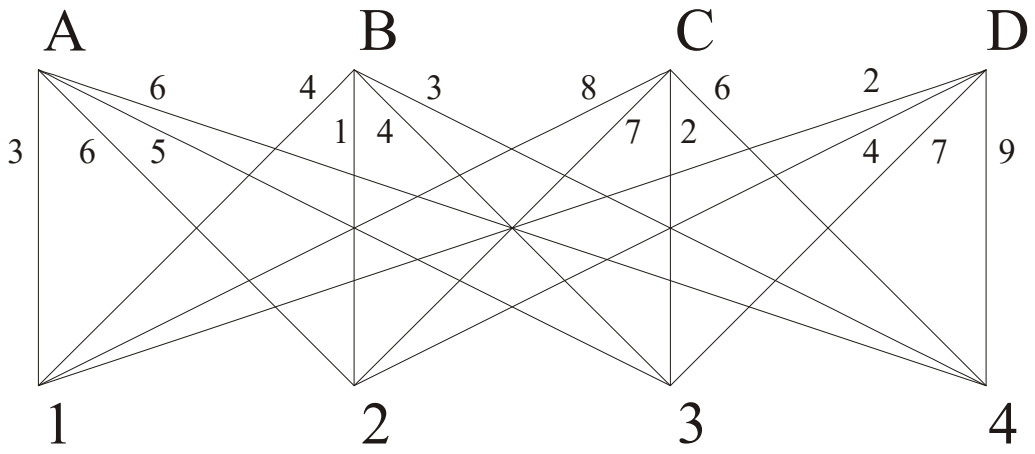
In the *optimal allocation graph* the sets *X* and *Y* must have the same order. That is, there is the same number of vertices in *X* and there is in *Y*.

The *weight matrix* is the matrix which is the incidence table of the graph corresponding to the optimal allocation problem.

The optimal allocation problem is also a problem in *matching*. The elements two sets, *X* and *Y*, have to be matched in some way that optimises a certain value.
Example

The following graph represents the possible assignments of bomber squadrons (A, B, C and D) to towns (1, 2, 3 or 4) on a day during a campaign of aerial bombardment.

The weightings represent the time (in hours) each squadron will spend in the air to deliver their payload to the relevant towns.



The problem is to find the assignment of bombers to towns that minimises time in air.

The weight matrix is as follows.

	A	*B*	*C*	*D*
1	3	4	8	2
2	6	1	7	4
3	5	4	2	7
4	6	3	6	9

Optimal assignment problem

A solution of the optimal assignment problem is a choice of n elements from the weighted graph such that (1) no two selected elements lie in the same row or same column; (2) the sum of the n selected entries is as small as possible.

The Hungarian Algorithm

The Hungarian algorithm is a procedure for solving the optimal assignment problem.

Suppose a choice of n elements from the weight matrix is a solution to the optimal assignment problem. Then that choice can also be found as the corresponding choice of n elements in another matrix, called the *modified matrix* by a process called *deleting zeros*.

Finding the modified matrix

This is done by reducing either rows (or columns) in the weight matrix, followed by reducing columns (or rows). In the process of reducing a row (column), you subtract the smallest number in the row from each element of the row (column). Repeat the process for each column (or row, if you started with columns). This is best explained by illustration.

Reducing rows

In this example we will start by reducing the rows. The smallest number in each row is circled in the diagram below.

	A	*B*	*C*	*D*
1	3	4	8	(2)
2	6	(1)	7	4
3	5	4	(2)	7
4	6	(3)	6	9

To reduce the rows we subtract these numbers from each element in the relevant row, as follows.

	A	*B*	*C*	*D*	
1	3	4	8	2	–2
2	6	1	7	4	–1
3	5	4	2	7	–2
4	6	3	6	9	–3

The result is

	A	*B*	*C*	*D*
1	(1)	2	6	0
2	5	0	6	3
3	3	2	0	5
4	3	0	3	6

In this matrix there is only one column that does not have a zero. This element has been circled. Now we reduce this matrix by subtracting this number from each entry in the column.

Reducing columns

	A	*B*	*C*	*D*
1	1	2	6	0
2	5	0	6	3
3	3	2	0	5
4	3	0	3	6
	-1			

The result is the modified matrix we are looking for.

	A	*B*	*C*	*D*
1	0	2	6	0
2	4	0	6	3
3	2	2	0	5
4	2	0	3	6

You can proceed either by reducing the rows then the columns, or by reducing the columns then the rows. However, the two alternative orders (rows → columns or columns → rows) do not necessarily produce the same modified matrix and you are usually told in the question whether to begin by reducing rows or columns first.

Whichever order is adopted, a solution to the optimisation problem will eventually be produced, but the solutions could be different, because there can be more than one solution to the optimisation problem. Now we proceed to the second stage of the algorithm.

Deleting zeros

Locate a row or column with the smallest number of zeros. Select one of those entries. Mark the selection (for example, by putting it in a box).

If the selection is in a row draw a vertical line through the column in which it appears. Ignore that column when making further selections.

You may not select two elements from the same row or column.

If the selection is in a column draw a horizontal line through the row in which it appears. Ignore that row when making further selections.

Continue this process until either:

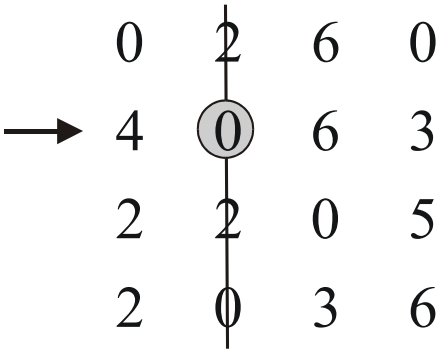
(1) You have n lines and n identified elements;
(2) You cannot identify any more zeros and delete any more lines; you have less than n identified elements.

In case (1) the assignment problem is solved; the n identified elements correspond to the n elements in the original matrix. The sum of the n elements in the original matrix is equal to the total minimum allocation.

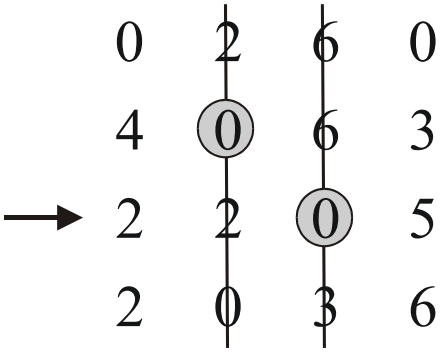
In case (2) you have to employ an additional process, possibly repeatedly, until you reach a situation described by case (1). This additional process is finding an *augmented matrix*.

Example continued

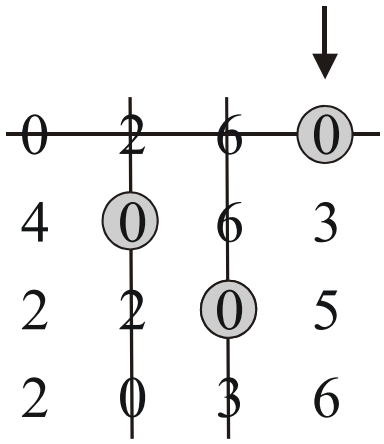
Select row 2, delete column 2



Select row 3, delete column 3



Select column 4, delete row 1



We have 3 selected items in a 4 x 4 matrix: i.e. we are one solution short. We must, therefore, find the augmented matrix.

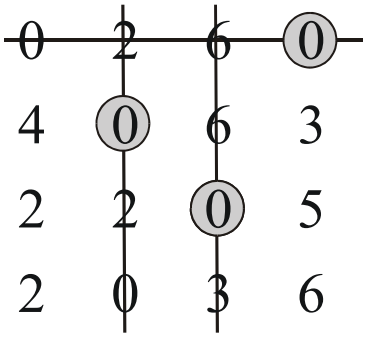
Finding the augmented matrix

Find the smallest uncovered entry. Suppose its value is *t*. Subtract *t* from each entry in any uncovered row and add *t* to each entry in a covered column.

Repeat the deleting zeros process on the augmented matrix.

If the matrix provides *n* identified elements you are done; if not, then iterate the process of finding the augmented matrix until you do.

Example continued



The uncovered elements are in columns 1 and 4. The smallest element is 2. We subtract 2 from each uncovered row and add 2 to each covered column.

0	2	6	0	
4	0	6	3	–2
2	2	0	5	–2
2	0	3	6	–2
	+2	+2		

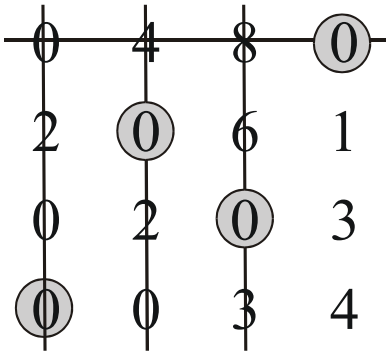
The result is

0	4	8	0
2	0	6	1
0	2	0	3
0	0	3	4

We now repeat the process of deleting zeroes

One order in which this can be done (the order is not unique) is as follows:

Select row 2, delete column 2
Select row 3, delete column 3
Select column 4, delete row 1
Select row 4, delete column 1



We select those elements in the original weight matrix.

	A	*B*	*C*	*D*
1	3	4	8	2
2	6	1	7	4
3	5	4	2	7
4	6	3	6	9

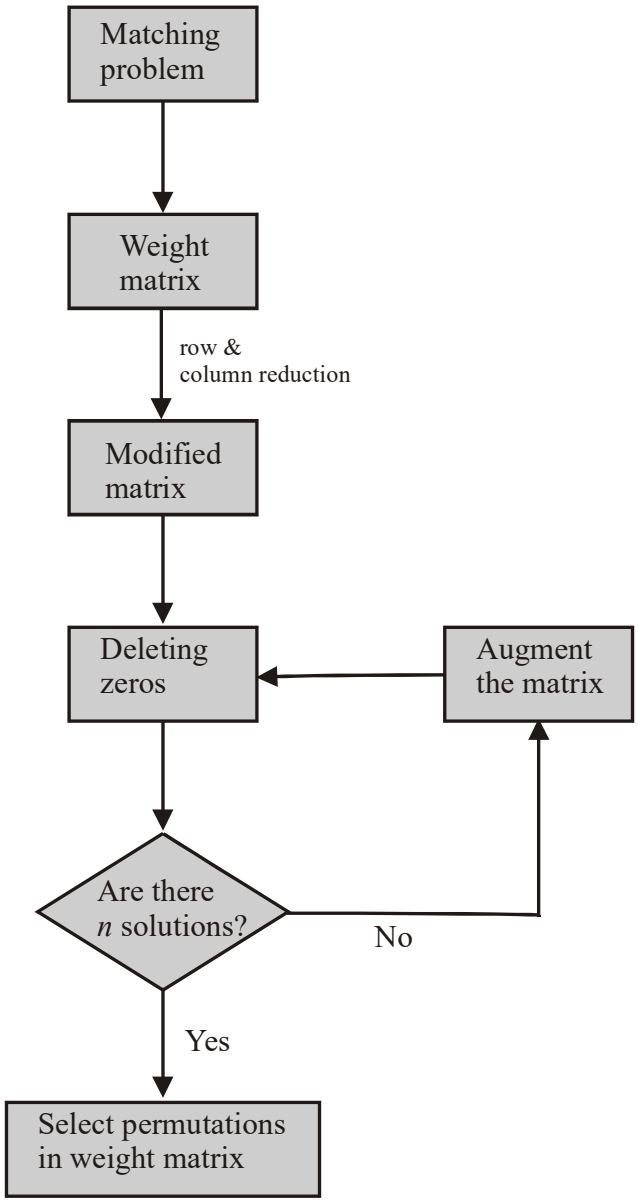
This gives the matching

4 ⟷ A, 2 ⟷ B, 3 ⟷ C, 1 ⟷ D

With minimum air time

6 + 1 + 2 + 2 = 11 hours

A flow diagram for the entire algorithm is:



Black's Academy

Visit Black's Academy

We hope that you have found this article helpful. For more information about our database visit Black's Academy on-line at

http://www.blacksacademy.com

Contact us also at support@blacksacademy.com and tells us more about what you are studying and how you think we might be able to help you.