MCA 201: Computer Oriented Operations Research

UNIT-I

Linear Programming: Concept of Linear Programming Model, Development of LP Model, Graphical Method, Simplex Method, Duality, Formulation of Dual Problem, Application of Duality,(Text Book 1).

UNIT-II

Transportation Problem: mathematical Model for Transportation Problem, Types of transportation problem, Finding the Initial Basic Solution, Optimal Solution by U-V method, Assignment problem, Formulation of Assignment problem-Hungerian Method, Method of Solution, Branch and Bound Technique for Assignment Problem,(Text Book 1).

UNIT-III

Network Techniques: Shortest-Path Model, Systematic Method-
Dijkstra's Algorithm, Floyid's Algorithm, Minimum Spanning Tree Problem,
Prime Algorithm, Krusakals Algorithm, Maximal Flow
Problem, Linear Programming Modeling for Maximal Flow Problem, Maximal
Flow Problem Algorithm, (Text Book 1).

UNIT-IV

Games and Strategies : Two –Person Zero- Sum Games, Maximin-Minimax Principle, Games Without Saddle Points- Mixed Strategies, Graphic Solution Of 2 x n, And m x 2 Games , Dominance Property, Arithmetic Model For n x n Games, General Solution For m x n Rectangular Games(Text Book 2).

UNIT – V

Queueing Theory: Queueing System, Elements Of Queueing System, Operating Characteristics Of Queueing System,
Probability Distributions In Queueing System, Classification Of Queueing Models, Poisson Queueing Systems, Non Poisson Queueing Systems. Network Scheduling by PERT / CPM: Rules Of Network Construction, Critical Path Analysis, Probability Considerations In PERT (Text Book 2).

Text Books:

1. R.Pannerselvam., "Operations Research" 2nd Edition, Prentice-Hall of India
2. Kanti Swarup., P.K.Gupta and Man Mohan, ., "Operations Research" 12th Edition Sultan chand & Sons

Reference Books:

1. Taha H.A., Operations Research: An Introduction, Prentice-Hall of India

Lecture Notes

UNIT-1

Linear Programming:

a method of optimising operations with some constraints. The main objective of linear programming is to maximize or minimize the numerical value. It consists of linear functions which are subjected to the constraints in the form of linear equations or in the form of inequalities. Linear programming is considered an important technique that is used to find the optimum resource utilisation. The term "linear programming" consists of two words as linear and programming. The word "linear" defines the relationship between multiple variables with degree one. The word "programming" defines the process of selecting the best solution from various alternatives.

Linear Programming is widely used in Mathematics and some other fields such as economics, business, telecommunication, and manufacturing fields. In this article, let us discuss the definition of linear programming, its components, and different methods to solve linear programming problems.

What is Linear Programming?

Linear programming (LP) or **Linear Optimisation**may be defined as the problem of maximizing or minimizing a linear function that is subjected to linear constraints. The constraints may be equalities or inequalities. The optimisation problems involve the calculation of profit and loss. Linear programming problems are an important class of optimisation problems, that helps to find the feasible region and optimise the solution in order to have the highest or lowest value of the function.

In other words, linear programming is considered as an optimization method to maximize or minimize the objective function of the given mathematical model with the set of some requirements which are represented in the linear relationship. The main aim of the linear programming problem is to find the optimal solution.

Linear programming is the method of considering different inequalities relevant to a situation and calculating the best value that is required to be obtained in those conditions. Some of the assumptions taken while working with linear programming are:

- The number of constraints should be expressed in the quantitative terms
- The relationship between the constraints and the objective function should be linear
- The linear function (i.e., objective function) is to be optimised

Graphical Method:

Graphical Method: Owing to the importance of linear programming models in various industries, many types of algorithms have been developed over the years to solve them. Some famous mentions include the Simplex method, the Hungarian approach, and others. Here we are going to concentrate on one of the most basic methods to handle a linear programming problem i.e. the graphical method.

In principle, this method works for almost all different types of problems but gets more and more difficult to solve when the number of decision variables and the constraints increases. Therefore, we'll illustrate it in a simple case i.e. for two variables only. So let's get started with the graphical method!

The Graphical Method

We will first discuss the steps of the algorithm:

Step 1: Formulate the LP (Linear programming) problem

We have already understood the mathematicalformulation of an LP problem in a previous section. Note that this is the most crucial step as all the subsequent steps depend on our analysis here.

Step 2: Construct a graph and plot the constraint lines

The graph must be constructed in 'n' dimensions, where 'n' is the number of decision variables. This should give you an idea about the complexity of this step if the number of decision variables increases.

One must know that one cannot imagine more than 3-dimensions anyway! The constraint lines can be constructed by joining the horizontal and vertical intercepts found from each constraint equation.

Step 3: Determine the valid side of each constraint line

This is used to determine the domain of the available space, which can result in a feasible solution. How to check? A simple method is to put the coordinates of the origin (0,0) in the problem and determine whether the objective function takes on a physical solution or not. If yes, then the side of the constraint lines on which the origin lies is the valid side. Otherwise it lies on the opposite one.

Step 4: Identify the feasible solution region

The feasible solution region on the graph is the one which is satisfied by all the constraints. It could be viewed as the intersection of the valid regions of each constraint line as well. Choosing any point in this area would result in a valid solution for our objective function.

Step 5: Plot the objective function on the graph

It will clearly be a straight line since we are dealing with linear equations here. One must be sure to draw it differently from the constraint lines to avoid confusion. Choose the constant value in the equation of the objective function randomly, just to make it clearly distinguishable.

Step 6: Find the optimum point

Step 7: Calculate the coordinates of the optimum point.

This is the last step of the process. Once you locate the optimum point, you'll need to find its coordinates. This can be done by drawing two perpendicular lines from the point onto the coordinate axes and noting down the coordinates.

Otherwise, you may proceed algebraically also if the optimum point is at the intersection of two constraint lines and find it by solving a set of simultaneous linear equations. The Optimum Point gives you the values of the decision variables necessary to optimize the objective function.

To find out the optimized objective function, one can simply put in the values of these parameters in the equation of the objective function. You have found your solution! Worried about the execution of this seemingly long algorithm?

Simplex Method:

simplex method, standard technique in linear programming for solving an optimization problem, typically one involving a function and several constraints expressed as inequalities. The inequalities define a polygonal region, and the solution is typically at one of the vertices. The simplex method is a systematic procedure for testing the vertices as possible solutions.

The Simplex method is an approach to solving linear programming models by hand using slack variables, tableaus, and pivot variables as a means to finding the optimal solution of an optimization problem. A linear program is a method of achieving the best outcome given a maximum or minimum equation with linear constraints. Most linear programs can be solved using an online solver such as MatLab, but the Simplex method is a technique for solving linear programs by hand. To solve a linear programming model using the Simplex method the following steps are necessary:

- Standard form

- Introducing slack variables
- Creating the tableau
- Pivot variables
- Creating a new tableau
- Checking for optimality
- Identify optimal values

Step 1: Standard Form

Standard form is the baseline format for all linear programs before solving for the optimal solution and has three requirements: (1) must be a maximization problem, (2) all linear constraints must be in a less-than-or-equal-to inequality, (3) all variables are non-negative. These requirements can always be satisfied by transforming any given linear program using basic algebra and substitution. Standard form is necessary because it creates an ideal starting point for solving the Simplex method as efficiently as possible as well as other methods of solving optimization problems.

To transform a minimization linear program model into a maximization linear program model, simply multiply both the left and the right sides of the objective function by -1.

Transforming linear constraints from a greater-than-or-equal-to inequality to a less-than-or-equal-to inequality can be done similarly as what was done to the objective function. By multiplying by -1 on both sides, the inequality can be changed to less-than-or-equal-to.

Once the model is in standard form, the slack variables can be added as shown in Step 2 of the Simplex method.

Step 2: Determine Slack Variables

Slack variables are additional variables that are introduced into the linear constraints of a linear program to transform them from inequality constraints to equality constraints. If the model is in standard form, the slack variables will always have a +1 coefficient. Slack variables are needed in the constraints to transform them into solvable equalities with one definite answer.

After the slack variables are introduced, the tableau can be set up to check for optimality as described in Step 3.

Step 3: Setting up the Tableau

A Simplex tableau is used to perform row operations on the linear programming model as well as to check a solution for optimality. The tableau consists of the coefficient corresponding to the linear constraint variables and the coefficients of the objective function. In the tableau below, the bolded top row of the tableau states what each column represents. The following two rows represent the linear constraint variable coefficients from the linear programming model, and the last row represents the objective function variable coefficients.

Once the tableau has been completed, the model can be checked for an optimal solution as shown in Step 4.

Step 4: Check Optimality

The optimal solution of a maximization linear programming model are the values assigned to the variables in the objective function to give the largest zeta value. The optimal solution would exist on the corner points of the graph of the entire model. To check optimality using the tableau, all values in the last row must contain values greater than or equal to zero. If a value is less than zero, it means that variable has not reached its optimal value. As seen in the previous tableau, three negative values exists in the bottom row indicating that this solution is not optimal. If a tableau is not optimal, the next step is to identify the pivot variable to base a new tableau on, as described in Step 5.

Step 5: Identify Pivot Variable

The pivot variable is used in row operations to identify whichvariable will become the unit value and is a key factor in the conversion of the unit value. The pivot variable can be identified by looking at the bottom row of the tableau and the indicator. Assuming that the solution is not optimal, pick the smallest negative value in the bottom row. One of the values lying in the column of this value will be the pivot variable. To find the indicator, divide the beta values of the linear constraints by their corresponding values from the column containing the possible pivot variable. The intersection of the row with the smallest non-negative indicator and the smallest negative value in the bottom row will become the pivot variable.

In the example shown below, -10 is the smallest negative in the last row. This will designate the x_2 column to contain the pivot variable. Solving for the indicator gives us a value of for the first constraint, and a value of for the second constraint. Due to being the smallest non-negative indicator, the pivot value will be in the second row and have a value of 5.

Now that the new pivot variable has been identified, the new tableau can be created in Step 6 to optimize the variable and find the new possible optimal solution.

Step 6: Create the New Tableau

The new tableau will be used to identify a new possible optimal solution. Now that the pivot variable has been identified in Step 5, row operations can be performed to optimize the pivot variable while keeping the rest of the tableau equivalent.

I. To optimize the pivot variable, it will need to be transformed into a unit value (value of 1). To transform the value, multiply the row containing the pivot variable by the reciprocal of the pivot value. In the example below, the pivot variable is originally 5, so multiply the entire row by .

the unit value has been determined, the other values in the column containing the unit value will become zero. This is because the x_2 in the second constraint is being optimized, which requires x_2 in the other equations to be zero.

n order to keep the tableau equivalent, the other variables not contained in the pivot column or pivot row must be calculated by using the new pivot values. For each new value, multiply the negative of the value in the old pivot column by the value in the new pivot row that corresponds to the value being calculated. Then add this to the old value from the old tableau to produce the new value for the new tableau. This step can be condensed into the equation on the next page:

New tableau value = (Negative value in old tableau pivot column) x (value in new tableau pivot row) + (Old tableau value)

Step 7: Check Optimality

As explained in Step 4, the optimal solution of a maximizationlinear programming model are the values assigned to the variables in the objective function to give the largest zeta value. Optimality will need to be checked after each new tableau to see if a new pivot variable needs to be identified. A solution is considered optimal if all values in the bottom row are greater than or equal to zero. If all values are greater than or equal to zero, the solution is considered optimal and Steps 8 through 11 can be ignored. If negative values exist, the solution is still not optimal and a new pivot point will need to be determined which is demonstrated in Step 8.

Step 8: Identify New Pivot Variable

If the solution has been identified as not optimal, a new pivot variable will need to be determined. The pivot variable was introduced in Step 5 and is used in row operations to identify which variable will become the unit value and is a key factor in the conversion of the unit value. The pivot variable can be identified by the intersection of the row with the smallest non-negative indicator and the smallest negative value in the bottom row.

Step 9: Create New Tableau

After the new pivot variable has been identified, a new tableau will need to be created. Introduced in Step 6, the tableau is used to optimize the pivot variable while keeping the rest of the tableau equivalent.

Step 10: Check Optimality

Using the new tableau, check for optimality. Explained in Step 4, an optimal solution appears when all values in the bottom row are greater than or equal to zero. If all values are greater than or equal to zero, skip to Step 12 because optimality has been reached. If negative values still exist, repeat steps 8 and 9 until an optimal solution is obtained.

Step 11: Identify Optimal Values

Once the tableau is proven optimal the optimal values can be identified. These can be found by distinguishing the basic and non-basic variables. A basic variable can be classified to have a single 1 value in its column and the rest be all zeros. If a variable does not meet this criteria, it is considered non-basic. If a variable is non-basic it means the optimal solution of that variable is zero. If a variable is basic, the row that contains the 1 value will correspond to the beta value. The beta value will represent the optimal solution for the given variable.

Formulation Of Dual Problem:

In mathematical optimization theory, **duality** or the **duality principle** is the principle that optimization problems may be viewed from either of two perspectives, the **primal problem**or the **dual problem**. If the primal is a minimization problem then the dual is a maximization problem (and vice versa). Any feasible solution to the primal (minimization) problem is at least as large as any feasible solution to the dual (maximization) problem. Therefore, the solution to the primal is an upper bound to the solution of the dual, and the solution of the dual is a lower bound to the solution of the primal.[1] This fact is called **weak duality**.

In general, the optimal values of the primal and dual problems need not be equal. Their difference is called the duality gap. For convex optimization problems, the duality gap is zero under a constraint qualification condition. This fact is called **strong duality**.

Usually the term "dual problem" refers to the *Lagrangian dual problem* but other dual problems are used – for example, the Wolfe dual problem and the Fenchel dual problem. The Lagrangian dual problem is obtained by forming the Lagrangian of a minimization problem by using nonnegative Lagrange multipliers to add the constraints to the objective function, and then solving for the primal variable values that minimize the original objective function. This solution gives the primal variables as functions of the Lagrange multipliers, which are called dual variables, so that the new problem is to maximize the objective function with respect to the dual variables under the derived constraints on the dual variables (including at least the nonnegativity constraints).

In general given two dual pairs of separatedlocally convex spaces and and the function , we can define the primal problem as finding such that In other words, if exists, is the minimum of the function and the infimum (greatest lower bound) of the function is attained.

If there are constraint conditions, these can be built into the function by letting where is a suitable function on that has a minimum 0 on the constraints, and for which one can prove that . The latter condition is trivially, but not always conveniently, satisfied for the characteristic function (i.e. for satisfying the constraints and otherwise). Then extend to a perturbation function such that .[2]

The duality gap is the difference of the right and left hand sides of the inequality

where is the convex conjugate in both variables and denotes the supremum(least upper bound).

The duality gap is the difference between the values of any primal solutions and any dual solutions. If is the optimal dual value and is the optimal primal value, then the duality gap is equal to . This value is always greater than or equal to 0 (for minimization problems). The duality gap is zero if and only if strong duality holds. Otherwise the gap is strictly positive and weak duality holds.

In computational optimization, another "duality gap" is often reported, which is the difference in value between any dual solution and the value of a feasible but suboptimal iterate for the primal problem. This alternative "duality gap" quantifies the discrepancy between the value of a current feasible but suboptimal iterate for the primal problem and the value of the dual problem; the value of the dual problem is, under regularity conditions, equal to the value of the *convex relaxation* of the primal problem: The convex relaxation is the problem arising replacing a non-convex feasible set with its closed convex hull and with replacing a non-convex function with its convex closure, that is the function that has the epigraph that is the closed convex hull of the original primal objective function.[

Linear programming problems are optimizationproblems in which the objective function and the constraints are all linear. In the primal problem, the objective function is a linear combination of *n* variables. There are *m*constraints, each of which places an upper bound on a linear combination of the *n*variables. The goal is to maximize the value of the objective function subject to the constraints. A *solution* is a vector (a list) of *n*values that achieves the maximum value for the objective function.

In the dual problem, the objective function is a linear combination of the *m* values that are the limits in the *m* constraints from the primal problem. There are *n* dual constraints, each of which places a lower bound on a linear combination of *m* dual variables.

Relationship between the primal problem and the dual problem:

In the linear case, in the primal problem, from each sub-optimal point that satisfies all the constraints, there is a direction or subspace of directions to move that increases the objective function. Moving in any such direction is said to remove slack between the candidate solution and one or more constraints. An *infeasible* value of the candidate solution is one that exceeds one or more of the constraints.

In the dual problem, the dual vector multiplies the constraints that determine the positions of the constraints in the primal. Varying the dual vector in the dual problem is equivalent to revising the upper bounds in the primal problem. The lowest upper bound is sought. That is, the dual vector is minimized in order to remove slack between the candidate positions of the constraints and the actual optimum. An infeasible value of the dual vector is one that is too low. It sets the candidate positions of one or more of the constraints in a position that excludes the actual optimum.

This intuition is made formal by the equations in Linear programming: Duality.

Short Answers:

1.What is Linear Programing?
2.Explain About Development Of Lp Model
3.What Is Simplex Method?
4.What is Graphical Method?

5.What Is Application Of Duality?

Long Answers:

6.Explain about Simplex and Graphical Method
7.What is Linear Programming? Explain About Development of LP
8.Explain about Formulation Of Dual Model
9.Explain About Concept Of Linear Programming
10.What Is Duality?

UNIT-2

Transportation Problem:

In operations Research Linear programming is one of the model in mathematical programming, which is very broad and vast. Mathematical programming includes many more optimization models known as Non - linear Programming, Stochastic programming, Integer Programming and Dynamic Programming - each one of them is an efficient optimization technique to solve the problem with a specific structure, which depends on the assumptions made in formulating the model. We can remember that the general linear programming model is based on the assumptions:

( ) Certainty

The resources available and the requirement of resources by competing candidates, the profit coefficients of each variable are assumed to remain unchanged and they are certain in nature.

( )Linearity

The objective function and structural constraints are assumed to be linear.

( ) Divisibility

All variables are assumed to be continuous; hence they can assume integer or fractional values.

( ) Single stage

The model is static and constrained to one decision only. And planning period is assumed to be fixed.

( ) Non-negativity

A non-negativity constraint exists in the problem, so that the values of all variables are to be 0, i.e. the lower limit is zero and the upper limit may be any positive number.

( ) Fixed technology

Production requirements are assumed to be fixed during the planning period.

) Constant profit or cost per unit

Regardless of the production schedules profit or cost remain constant.

Now let us examine the applicability of linear programming model for transportation and assignment models.

4.2. TRANSPORTATION MODEL

The transportation model deals with a special class of linear programming problem in which the objective is to transport a homogeneous commodity from various origins or factories to different destinations or markets at a total minimum cost.

To understand the problem more clearly, let us take an example and discuss the rationale of transportation problem. Three factories A, B and C manufactures sugar and are located in different regions. Factory A manufactures, b1 tons of sugar per year and B manufactures b2 tons of sugar per year and C manufactures b3 tons of sugar. The sugar is required by four markets W, X, Y and Z. The requirement of the four markets is as follows: Demand for sugar in Markets W, X, Yand Z is d1, d2, d3 and d4 tons respectively. The transportation cost of one ton of sugar from each factory to market is given in the matrix below. The objective is to transport sugar from factories to the markets at a minimu total transportation cost.

Markets

Transportation cost per ton in Rs.

Availability in tons

W

X

Y

Z

Factories

Demand in Tons.

A

c11

c12

c13

c14

b1

B

c21

c22

c23

c24

b2

C

c31

c32

c33

c34

b3

d1

d2

d3

d4

bj/ dj

For the data given above, the mathematical model will be:

MinimizeZ=c11 x11 +c12 x12 +c13 x13 +c14 x14 +c21 x21 +c22 x22 +c23 x23 +c24 x24 +

c31 x31 + c32 x32 + c33 x33 + c34 x34 subject to a condition: OBJECTIVEFUNCTION.

a11 x11 +a12 x12 + a13 x13 + a14 x14 a21 x21 +a22 x22 + a23 x23 +a24 x24

a31 x31 +a32 x32 +a33 x33 +a34 x34

b1 (because the sum must be less than or equal to the available capacity)

d1

b2 b3

MIXED STRUCTURAL CONSTRAINTS.

a11 x11 +a21 x21 +a31 x31

a12 x12 + a22 x22 + a32 x32

a13 x13 +a23 x23 +a33 x33

a14 x14 +a24 x24 +a34 x34

All xij and xji are 0 where i = 1,2,3 and j = 1,2,3,4. (This is because we cannot

supply negative elements). NON-NEGATIVITY

Linear Programming: II Transportation Model

The above problem has got the following properties:

1. It has an objective function.

2. It has structural constraints.

3. It has a non-negativity constraint.

4. The relationship between the variables and the constraints are linear.

We know very well that these are the properties of a linear programming problem. Hence the transportation model is also a linear programming problem. But a special type of linear programming

problem.

Once we say that the problem has got the characteristics of linear programming model, and then we can solve it by simplex method. Hence we can solve the transportation problem by using the simplex method. As we see in the above given transportation model, the structural constraints are of mixed type. That is some of them are of type and some of them are of type. When we start solving the transportation problem by simplex method, it takes more time and laborious. Hence we use transportation algorithm or transportation method to solve the problem. Before we discuss the transportation algorithm, let us see how a general model for transportation problem appears. The general problem will have 'm' rows and 'n' columns i.e., m × n matrix.

nm

cij xj s.t.wherei=1tomandj=1ton. aij xij bi wherei=1tomandj=1ton

Types Of Transportation Problem:

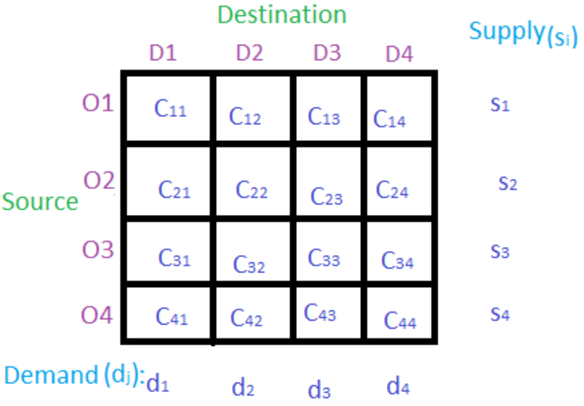
Balanced: When both supplies and demands are equal then the problem is said to be a balanced transportation problem.

Unbalanced: When the supply and demand are not equal then it is said to be an unbalanced transportation problem. In this type of problem, either a dummy row or a dummy column is added according to the requirement to make it a balanced problem. Then it can be solved similar to the balanced problem.

Methods to Solve:
To find the initial basic feasible solution there are three methods:

1. NorthWest Corner Cell Method.
2. Least Call Cell Method.
3. Vogel’s Approximation Method (VAM).

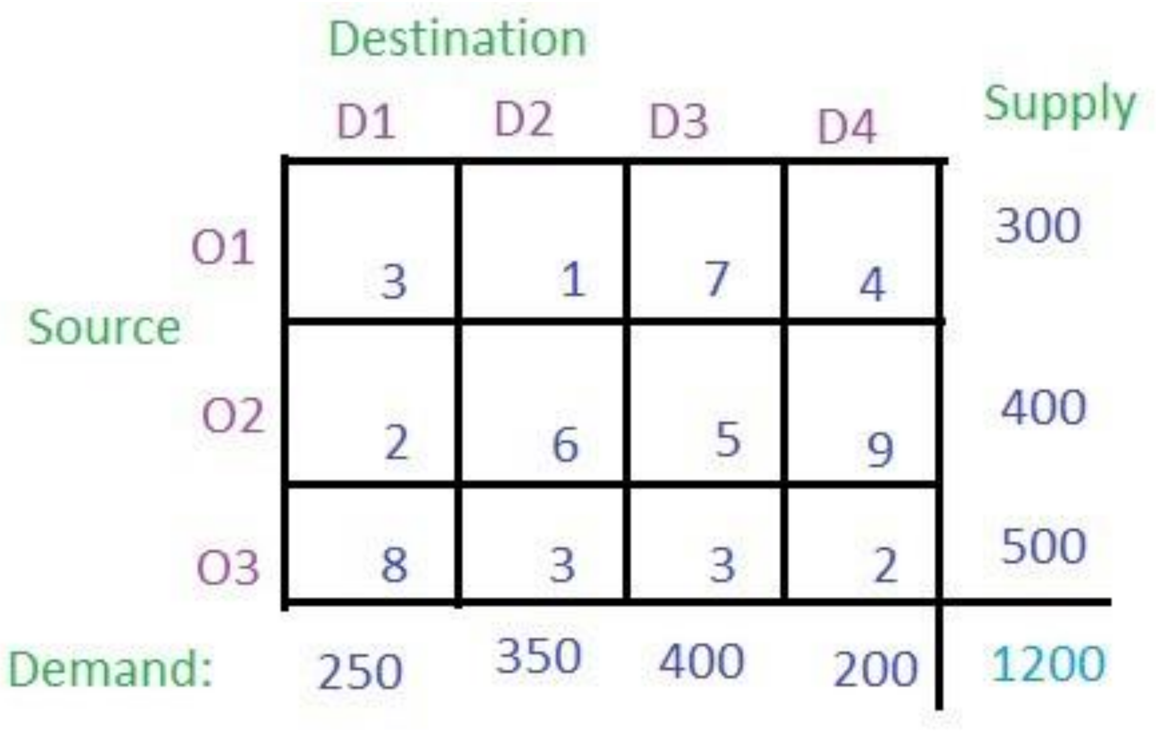
In the above table **D1**, **D2**, **D3** and **D4** are the destinations where the products/goods are to be delivered from different sources **S1**, **S2**, **S3** and **S4**. **S_i** is the supply from the source **O_i**. **d_j** is the demand of the destination **D_j**. **C_{ij}** is the cost when the product is delivered from source **S_i** to destination **D_j**.



		Destination				
		D1	D2	D3	D4	Supply(s_i)
	O1	C_{11}	C_{12}	C_{13}	C_{14}	s_1
Source	O2	C_{21}	C_{22}	C_{23}	C_{24}	s_2
	O3	C_{31}	C_{32}	C_{33}	C_{34}	s_3
	O4	C_{41}	C_{42}	C_{43}	C_{44}	s_4
Demand (d_j):		d_1	d_2	d_3	d_4	

NorthWest Corner Method

An introduction to Transportation problem has been discussed in the previous article, in this article, finding the initial basic feasible solution using the NorthWest Corner Cell Method will be discussed.

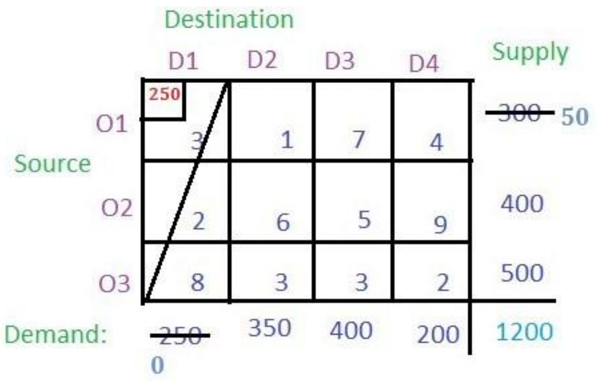


		Destination				
		D1	D2	D3	D4	Supply
	O1	3	1	7	4	300
Source	O2	2	6	5	9	400
	O3	8	3	3	2	500
Demand:		250	350	400	200	1200

Explanation: Given three sources **O1**, **O2**and **O3** and four destinations **D1**, **D2**, **D3**and **D4**. For the sources **O1**, **O2** and **O3**, the supply is **300**, **400** and **500**respectively. The destinations **D1**, **D2**, **D3**and **D4** have demands **250**, **350**, **400** and **200** respectively.

Solution: According to North West Corner method, **(O1, D1)** has to be the starting point i.e. the north-west corner of the table. Each and every value in the cell is considered as the cost per transportation. Compare the demand for column **D1** and supply from the source **O1**and allocate the minimum of two to the cell **(O1, D1)** as shown in the figure.

The demand for Column **D1** is completed so the entire column **D1** will be canceled. The supply from the source **O1** remains **300 – 250 = 50**.

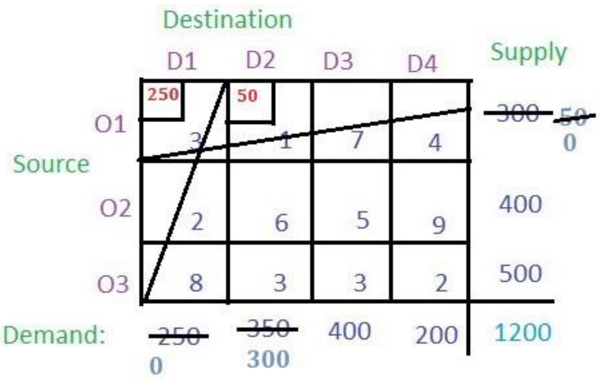


Destination

Source

	D1	D2	D3	D4	Supply
O1	250 3	1	7	4	~~300~~ 50
O2	2	6	5	9	400
O3	8	3	3	2	500
Demand:	~~250~~ 0	350	400	200	1200

Now from the remaining table i.e. excluding column **D1**, check the north-west corner i.e. **(O1, D2)** and allocate the minimum among the supply for the respective column and the rows. The supply from **O1** is **50** which is less than the demand for **D2** (i.e. 350), so allocate **50** to the cell **(O1, D2)**. Since the supply from row **O1** is completed cancel the row **O1**. The demand for column **D2** remain **350 – 50 = 300**.



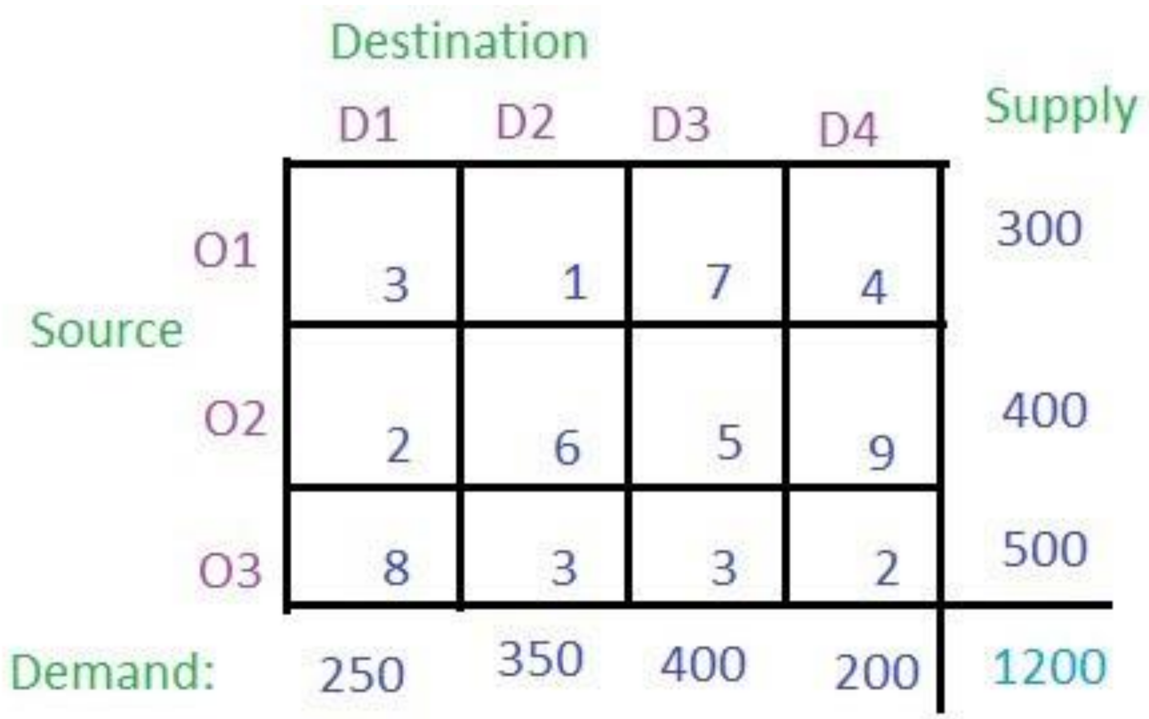
Destination

Source

	D1	D2	D3	D4	Supply
O1	250 3	50 1	7	4	~~300~~ ~~50~~ 0
O2	2	6	5	9	400
O3	8	3	3	2	500
Demand:	~~250~~ 0	~~350~~ 300	400	200	1200

Least Cost Cell Method)

The **North-West Corner** method has been discussed in the previous article. In this article, the **Least Cost Cell** method will be discussed.

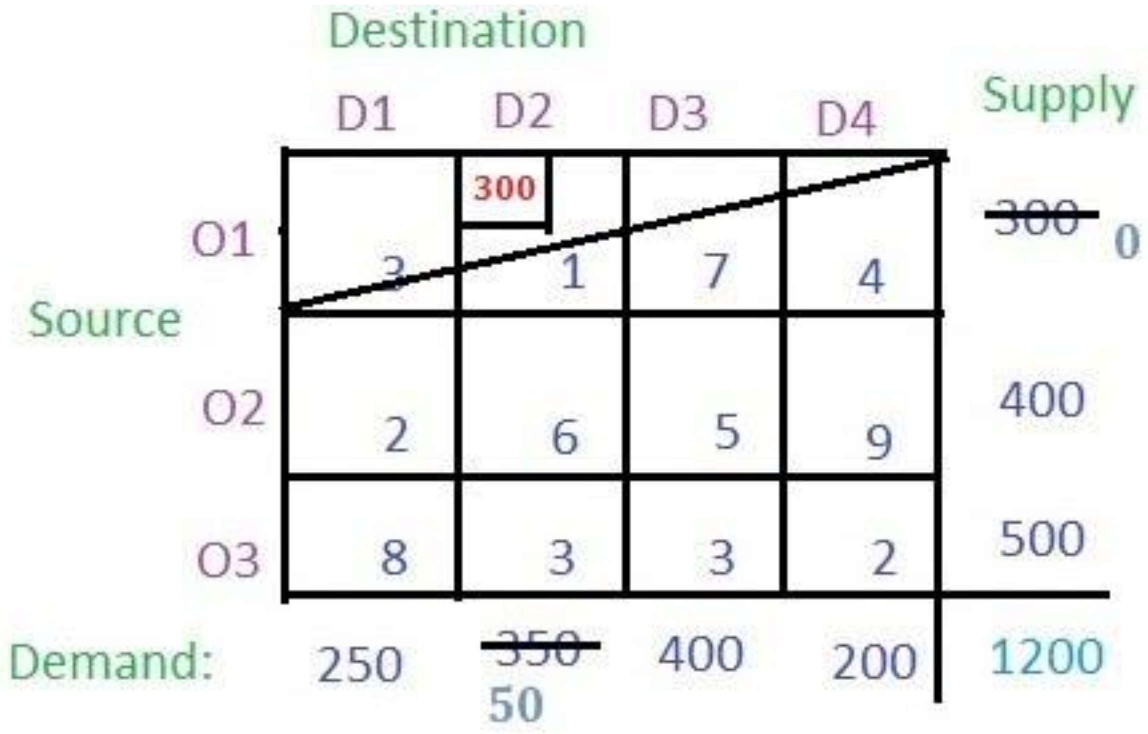


Destination

Source

	D1	D2	D3	D4	Supply
O1	3	1	7	4	300
O2	2	6	5	9	400
O3	8	3	3	2	500
Demand:	250	350	400	200	1200

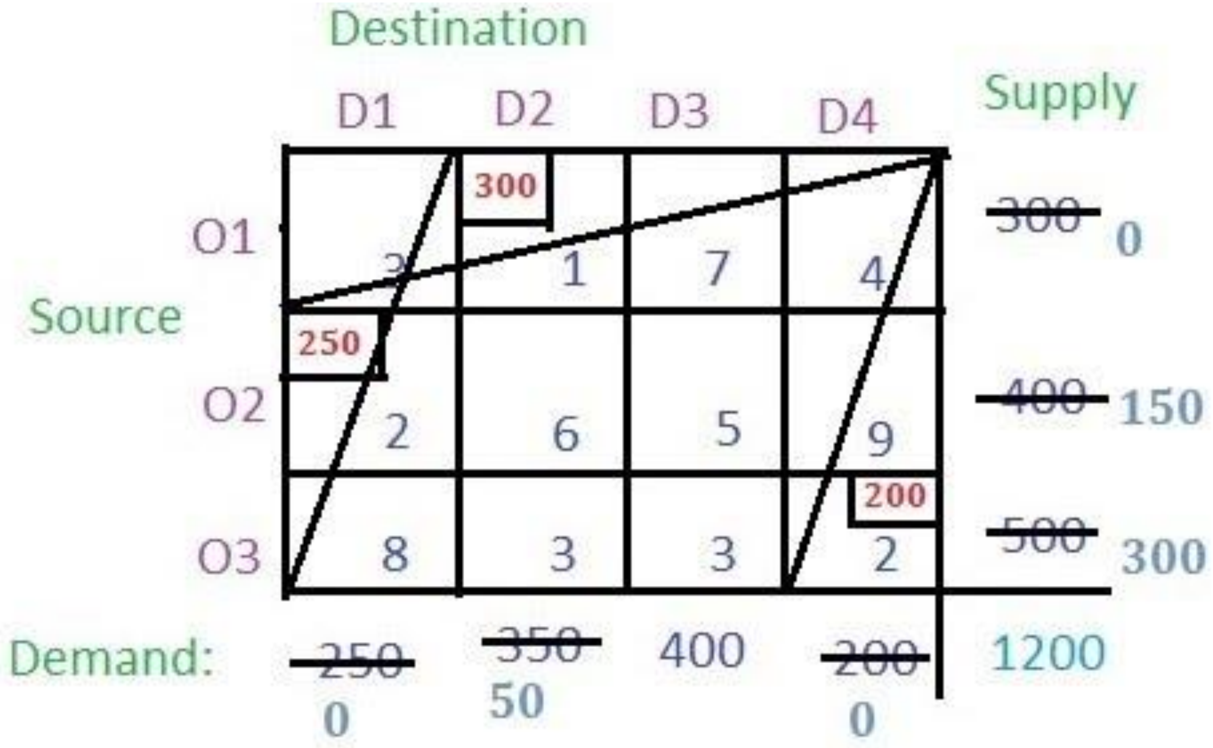
Solution: According to the Least Cost Cell method, the least cost among all the cells in the table has to be found which is **1** (i.e. cell **(O1, D2)**).
Now check the supply from the row **O1** and demand for column **D2** and allocate the smaller value to the cell. The smaller value is **300** so allocate this to the cell. The supply from **O1** is completed so cancel this row and the remaining demand for the column **D2** is **350 – 300 = 50**.



		Destination				
		D1	D2	D3	D4	Supply
Source	O1	3	300 1	7	4	~~300~~ 0
	O2	2	6	5	9	400
	O3	8	3	3	2	500
Demand:		250	~~350~~ 50	400	200	1200

Now find the cell with the least cost among the remaining cells. There are two cells with the least cost i.e. **(O2, D1)** and **(O3, D4)**with cost **2**. Lets select **(O2, D1)**. Now find the demand and supply for the respective cell and allocate the minimum among them to the cell and cancel the row or column whose supply or demand becomes **0** after allocation.

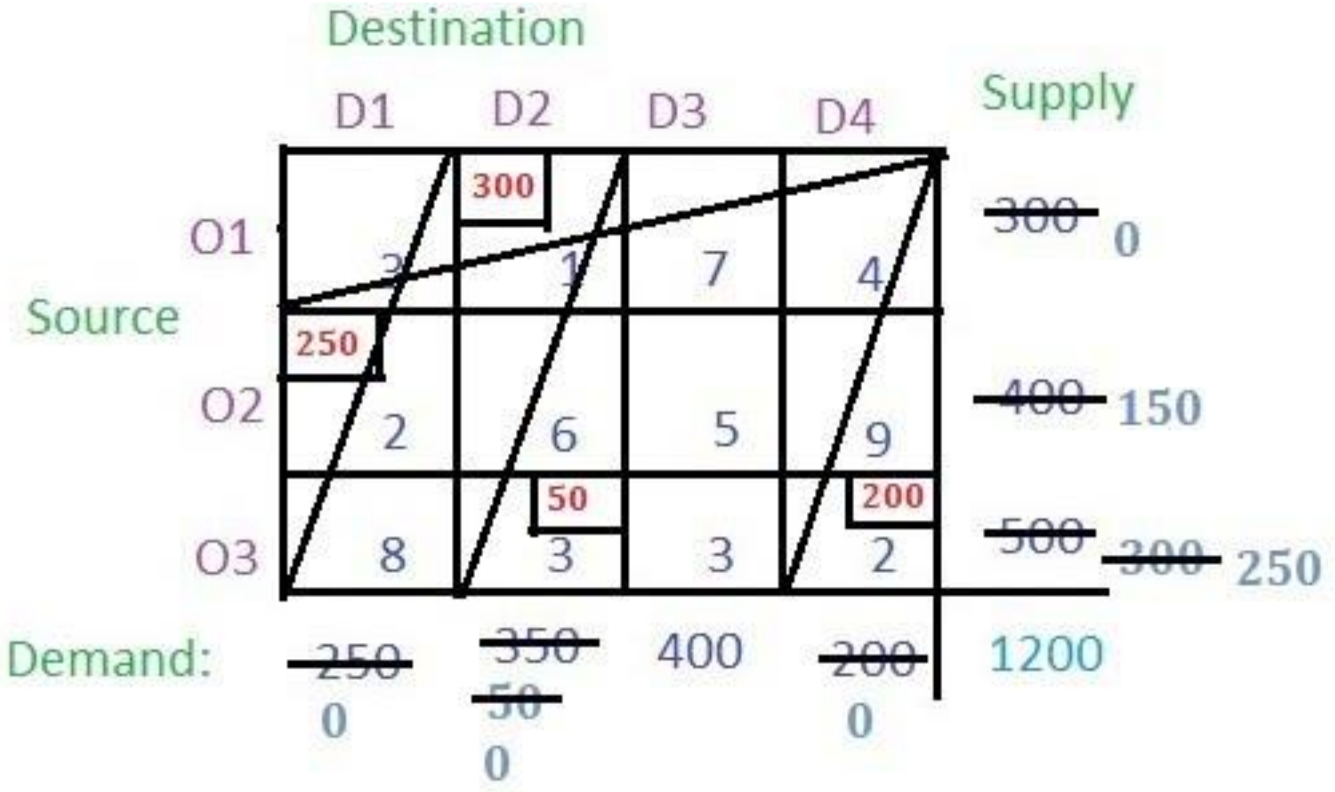
Now the cell with the least cost is **(O3, D4)**with cost **2**. Allocate this cell with **200** as the demand is smaller than the supply. So the column gets cancelled.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 150
O3	8	3	3	200 2	~~500~~ 300
Demand:	~~250~~ 0	~~350~~ 50	400	~~200~~ 0	1200

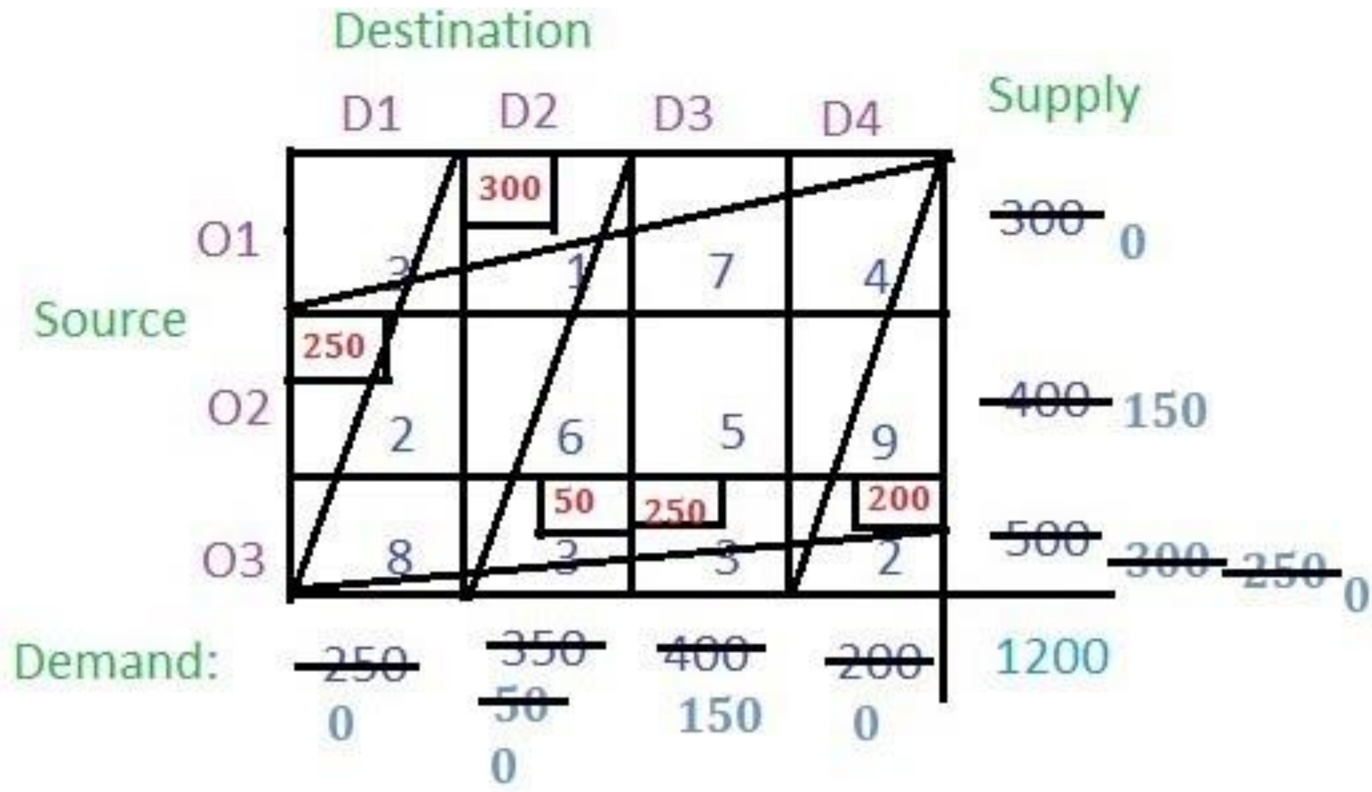
There are two cells among the unallocated cells that have the least cost. Choose any at random say **(O3, D2)**. Allocate this cell with a minimum among the supply from the respective row and the demand of the respective column. Cancel the row or column with zero value.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 150
O3	8	50 3	3	200 2	~~500~~ ~~300~~ 250
Demand:	~~250~~ 0	~~350~~ ~~50~~ 0	400	~~200~~ 0	1200

Now the cell with the least cost is **(O3, D3)**. Allocate the minimum of supply and demand and cancel the row or column with zero value.

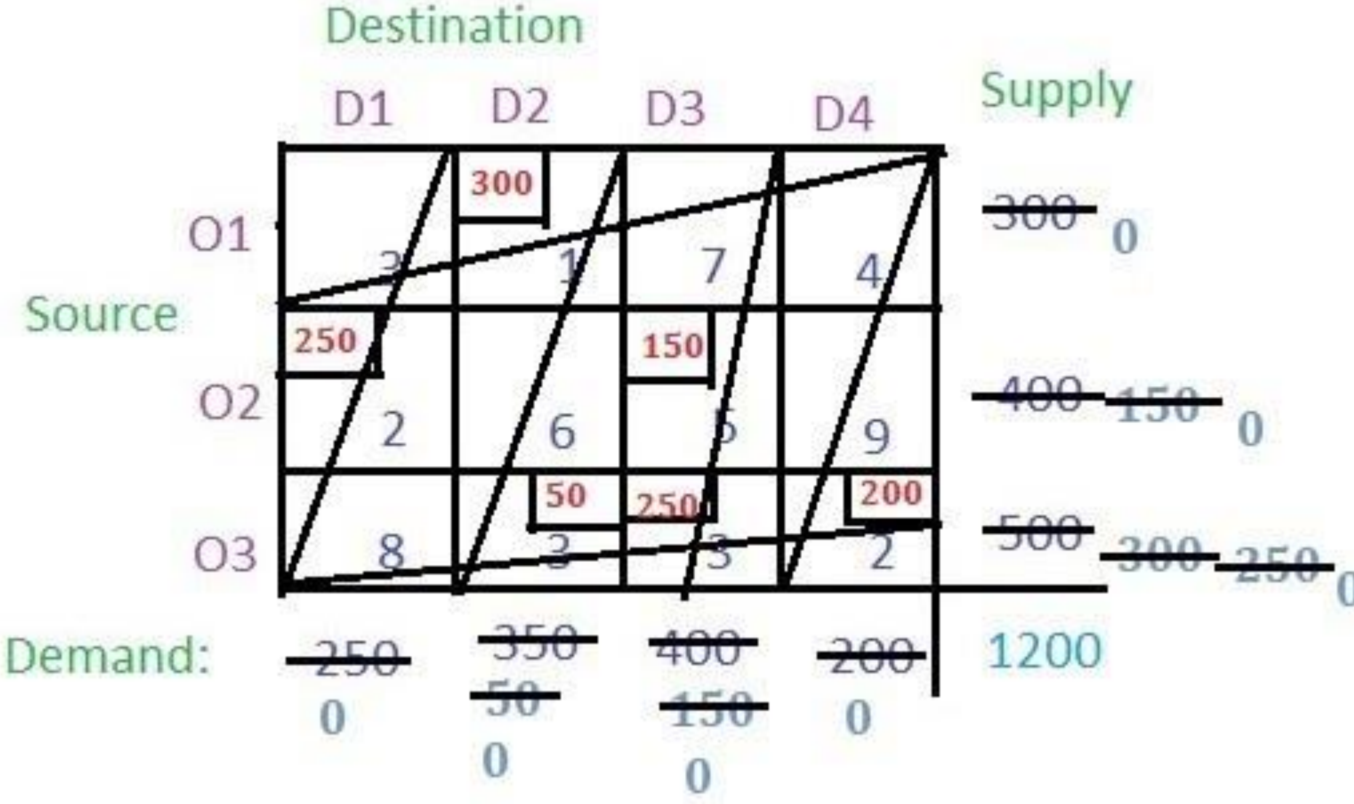


Destination

Source

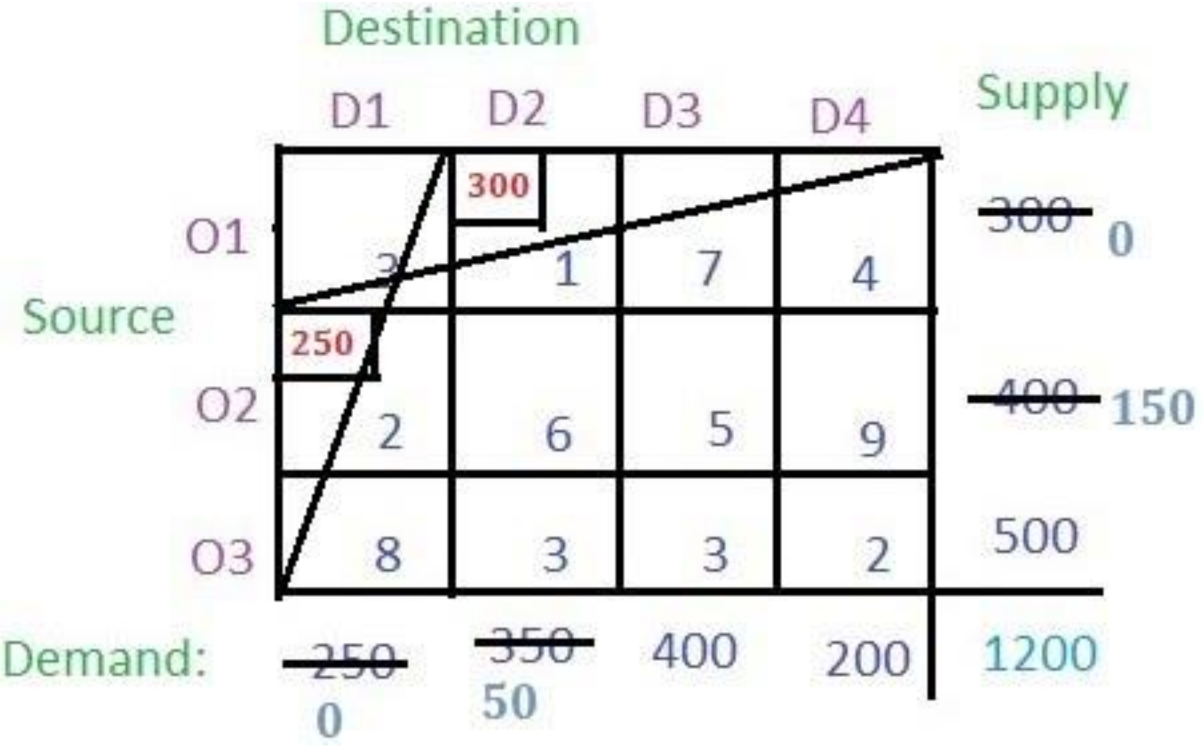
	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 150
O3	8	50 3	250 3	200 2	~~500~~ ~~300~~ ~~250~~ 0
Demand:	~~250~~ 0	~~350~~ ~~50~~ 0	~~400~~ 150	~~200~~ 0	1200

The only remaining cell is **(O2, D3)** with cost **5** and its supply is **150** and demand is **150** i.e. demand and supply both are equal. Allocate it to this cell.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	150 5	9	~~400~~ ~~150~~ 0
O3	8	50 3	250 3	200 2	~~500~~ ~~300~~ ~~250~~ 0
Demand:	~~250~~ 0	~~350~~ ~~50~~ 0	~~400~~ ~~150~~ 0	~~200~~ 0	1200

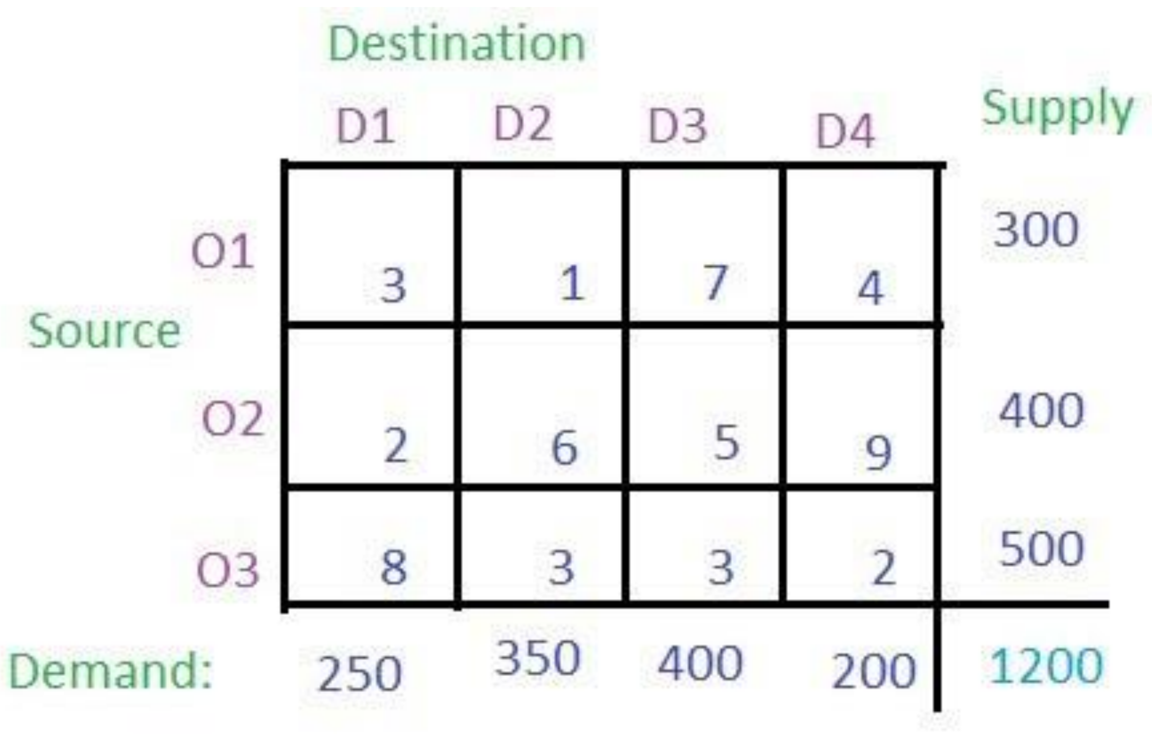


Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 150
O3	8	3	3	2	500
Demand:	~~250~~ 0	~~350~~ 50	400	200	1200

Vogel’s Approximation Method)

The **North-West Corner** method and the **Least Cost Cell** method has been discussed in the previous articles. In this article, the **Vogel’s Approximation** method will be discussed.

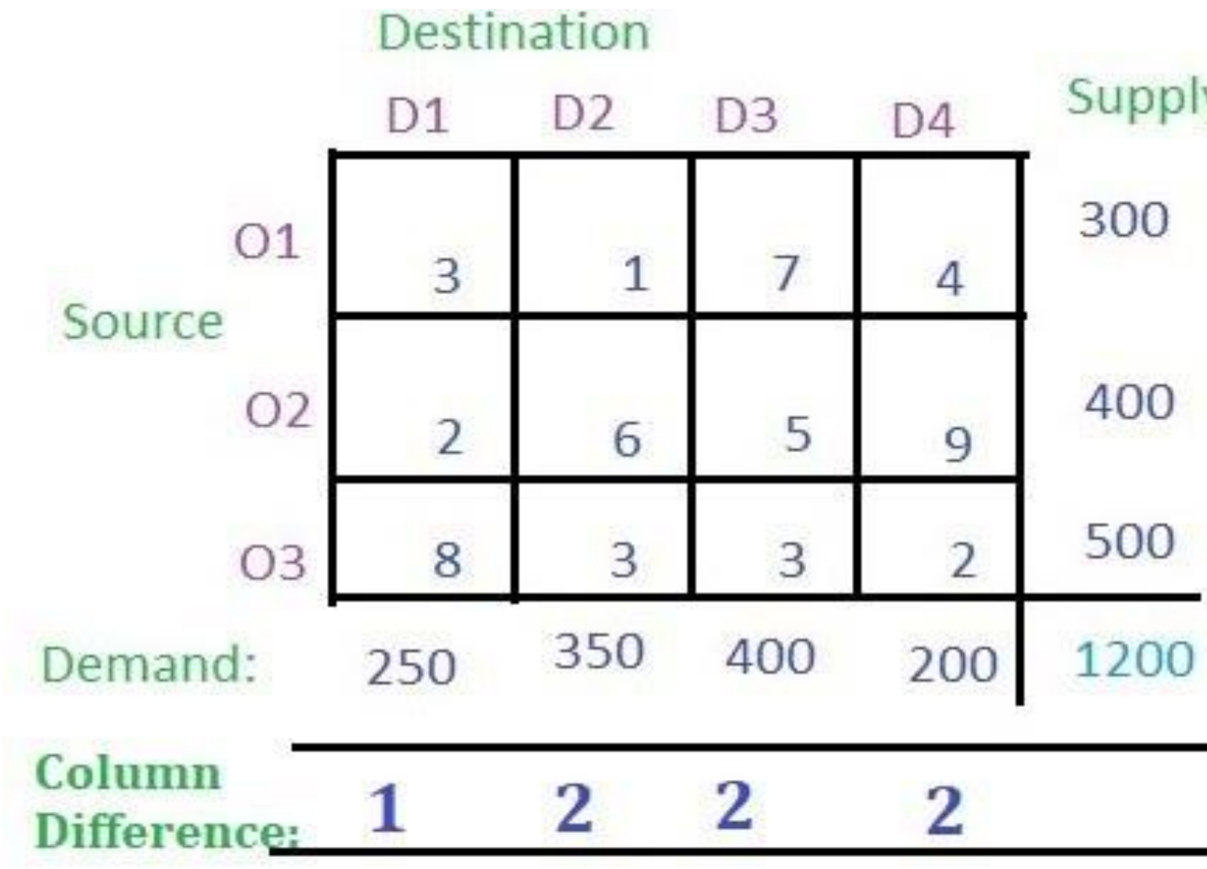


Destination

Source	D1	D2	D3	D4	Supply
O1	3	1	7	4	300
O2	2	6	5	9	400
O3	8	3	3	2	500
Demand:	250	350	400	200	1200

Solution:

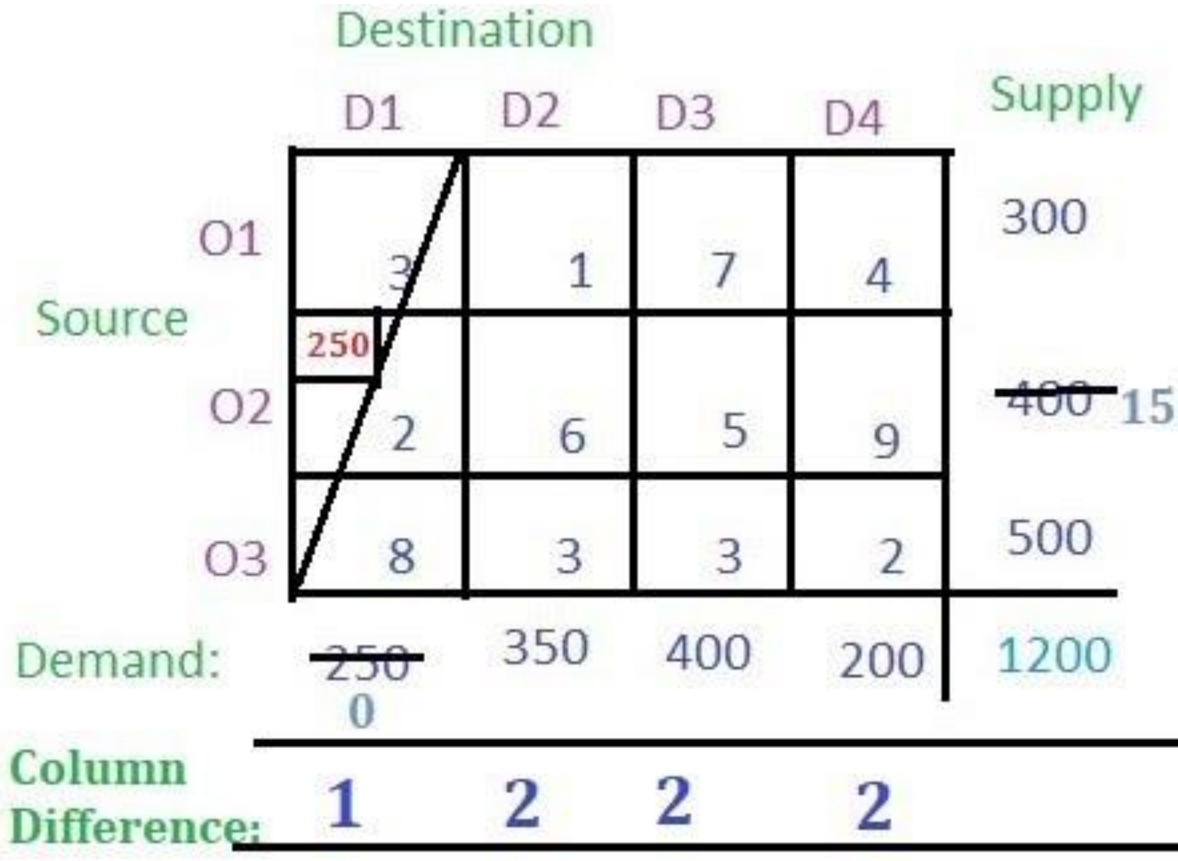
- For each row find the least value and then the second least value and take the absolute difference of these two least values and write it in the corresponding row difference as shown in the image below. In row **O1**, **1** is the least value and **3**is the second least value and their absolute difference is **2**. Similarly, for row **O2** and **O3**, the absolute differences are **3** and **1** respectively.
- For each column find the least value and then the second least value and take the absolute difference of these two least values then write it in the corresponding column difference as shown in the figure. In column **D1**, **2** is the least value and **3** is the second least value and their absolute difference is **1**. Similarly, for column **D2**, **D3** and **D3**, the absolute differences are **2**, **2** and **2** respectively.



Destination

Source	D1	D2	D3	D4	Suppl
O1	3	1	7	4	300
O2	2	6	5	9	400
O3	8	3	3	2	500
Demand:	250	350	400	200	1200
Column Difference:	**1**	**2**	**2**	**2**	

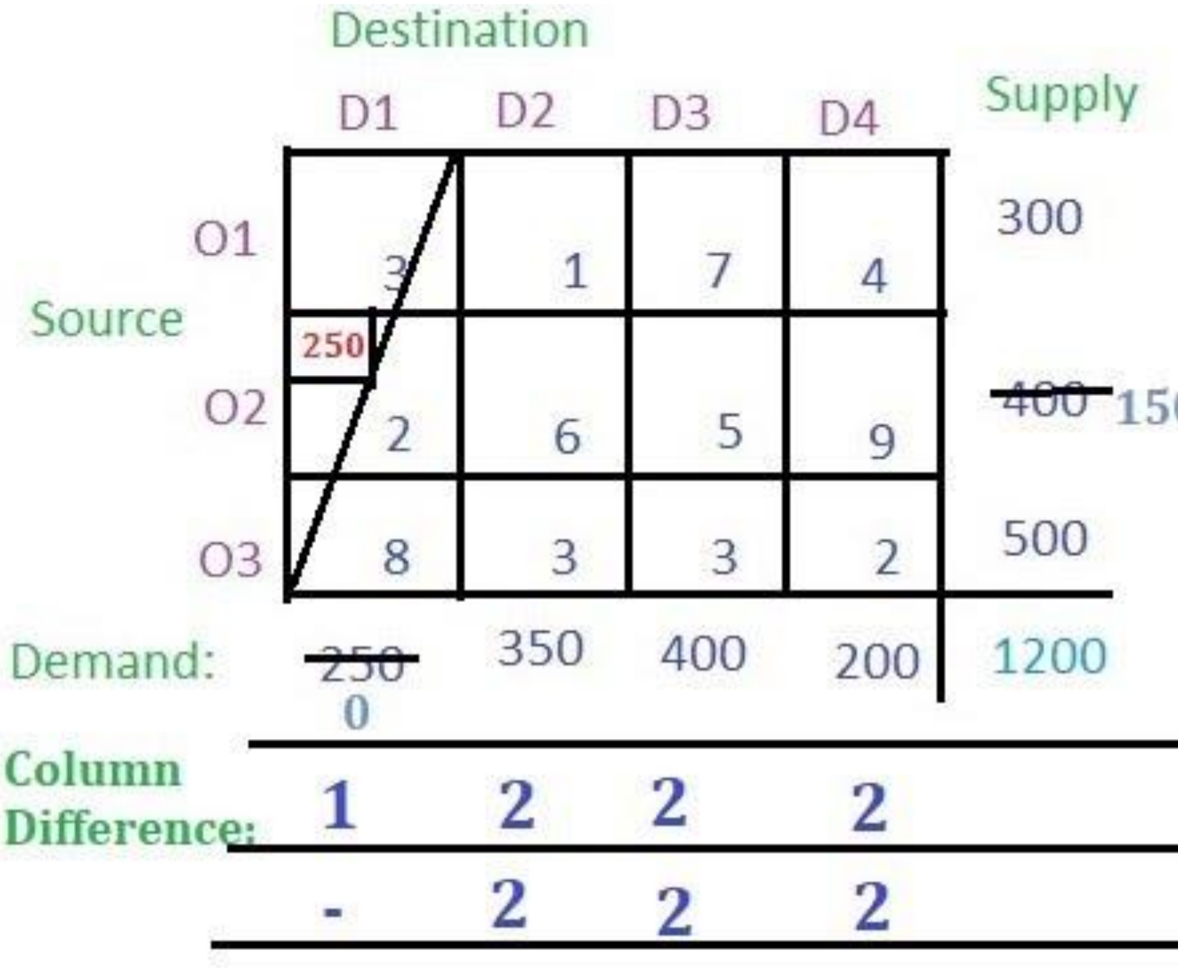
- These value of row difference and column difference are also called as penalty. Now select the maximum penalty. The maximum penalty is **3** i.e. row **O2**. Now find the cell with the least cost in row **O2** and allocate the minimum among the supply of the respective row and the demand of the respective column. Demand is smaller than the supply so allocate the column’s demand i.e. **250** to the cell. Then cancel the column **D1**.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	1	7	4	300
O2	250 2	6	5	9	~~400~~ 15
O3	8	3	3	2	500
Demand:	~~250~~ 0	350	400	200	1200
Column Difference:	**1**	**2**	**2**	**2**	

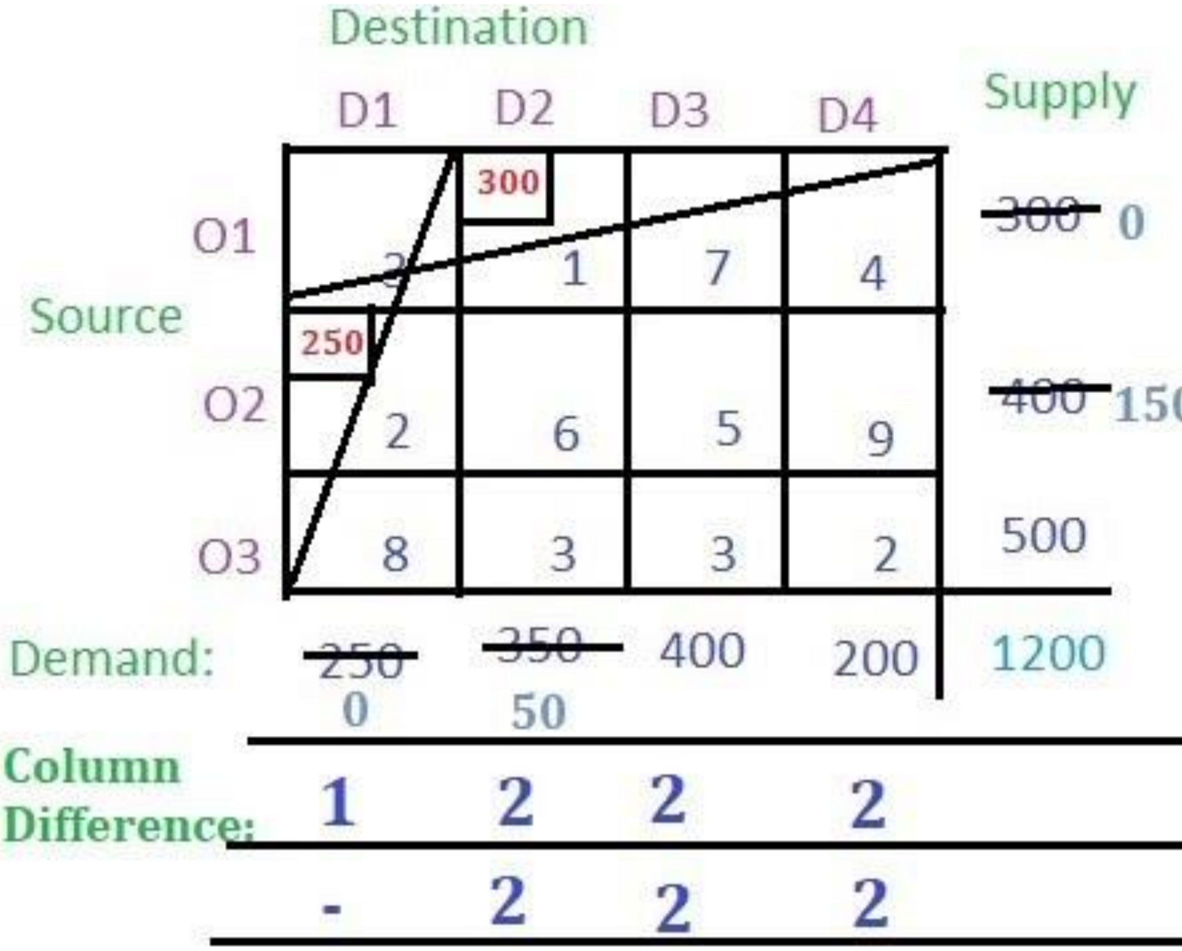
- From the remaining cells, find out the row difference and column difference.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	1	7	4	300
O2	250 2	6	5	9	~~400~~ 15
O3	8	3	3	2	500
Demand:	~~250~~ 0	350	400	200	1200
Column Difference:	**1**	**2**	**2**	**2**	
	-	**2**	**2**	**2**	

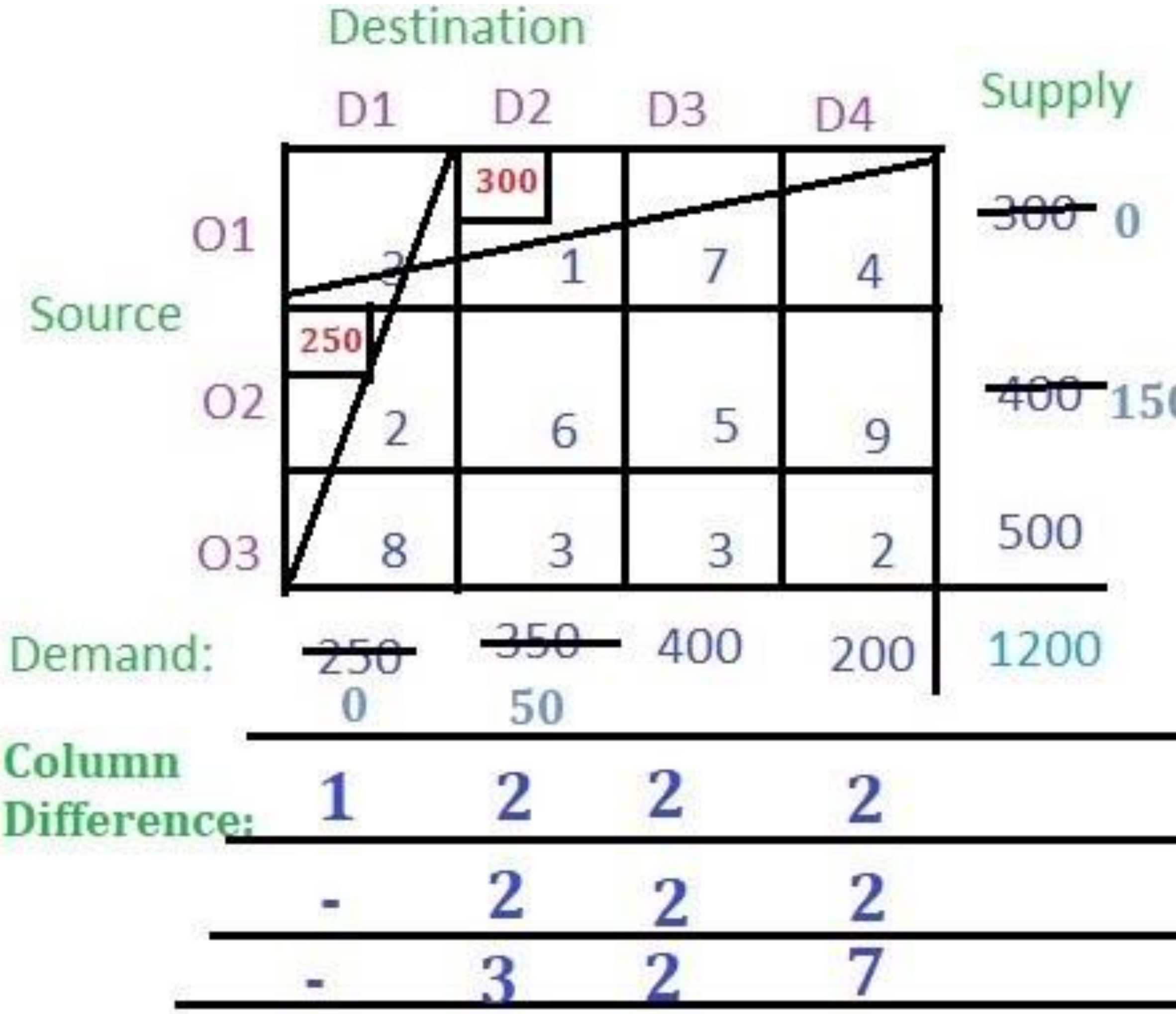
- Again select the maximum penalty which is **3** corresponding to row **O1**. The least-cost cell in row **O1** is **(O1, D2)** with cost **1**. Allocate the minimum among supply and demand from the respective row and column to the cell. Cancel the row or column with zero value.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 150
O3	8	3	3	2	500
Demand:	~~250~~ 0	~~350~~ 50	400	200	1200
Column Difference:	1	2	2	2	
	-	2	2	2	

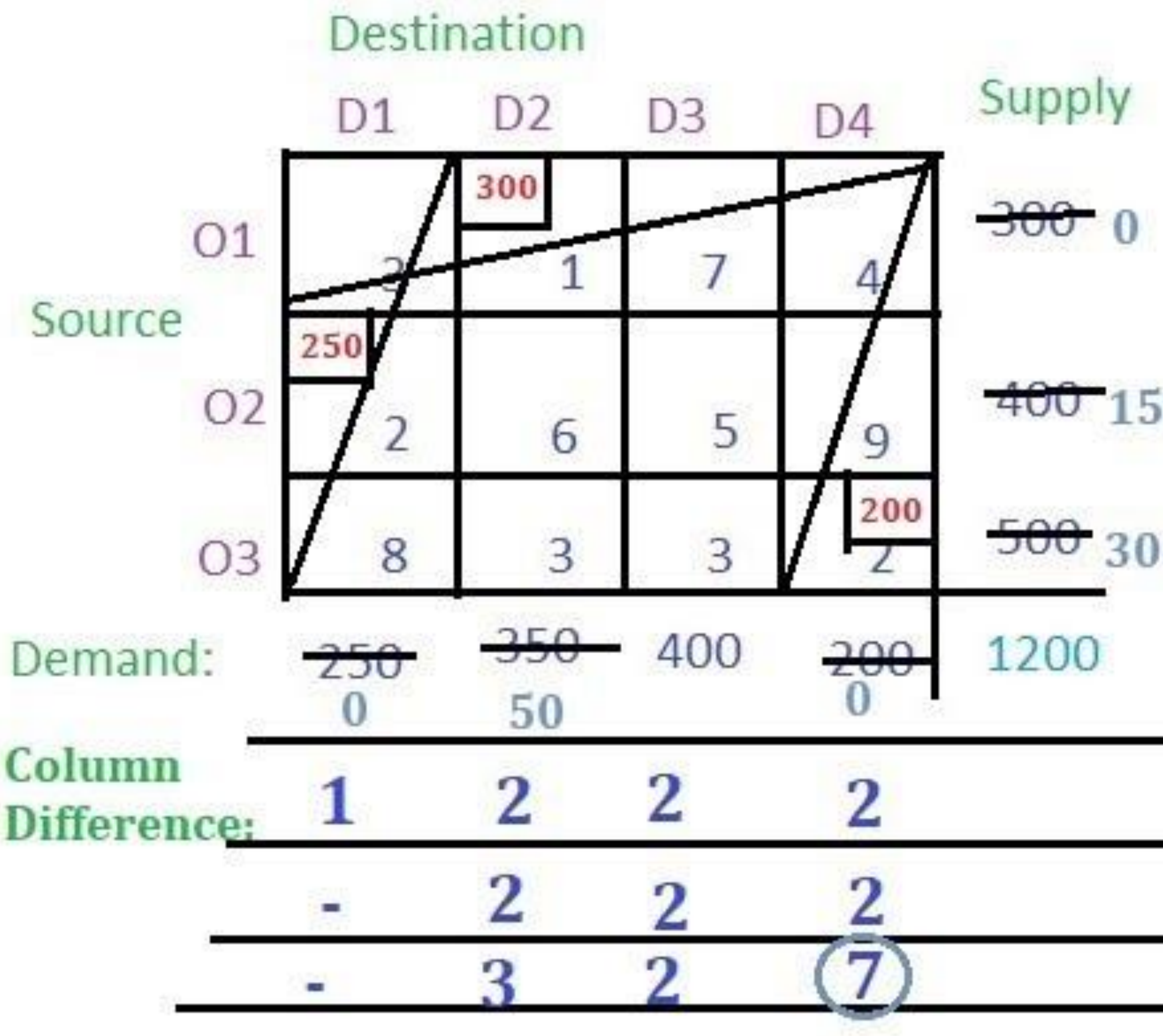
- Now find the row difference and column difference from the remaining cells.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 15
O3	8	3	3	2	500
Demand:	~~250~~ 0	~~350~~ 50	400	200	1200
Column Difference:	1	2	2	2	
	-	2	2	2	
	-	3	2	7	

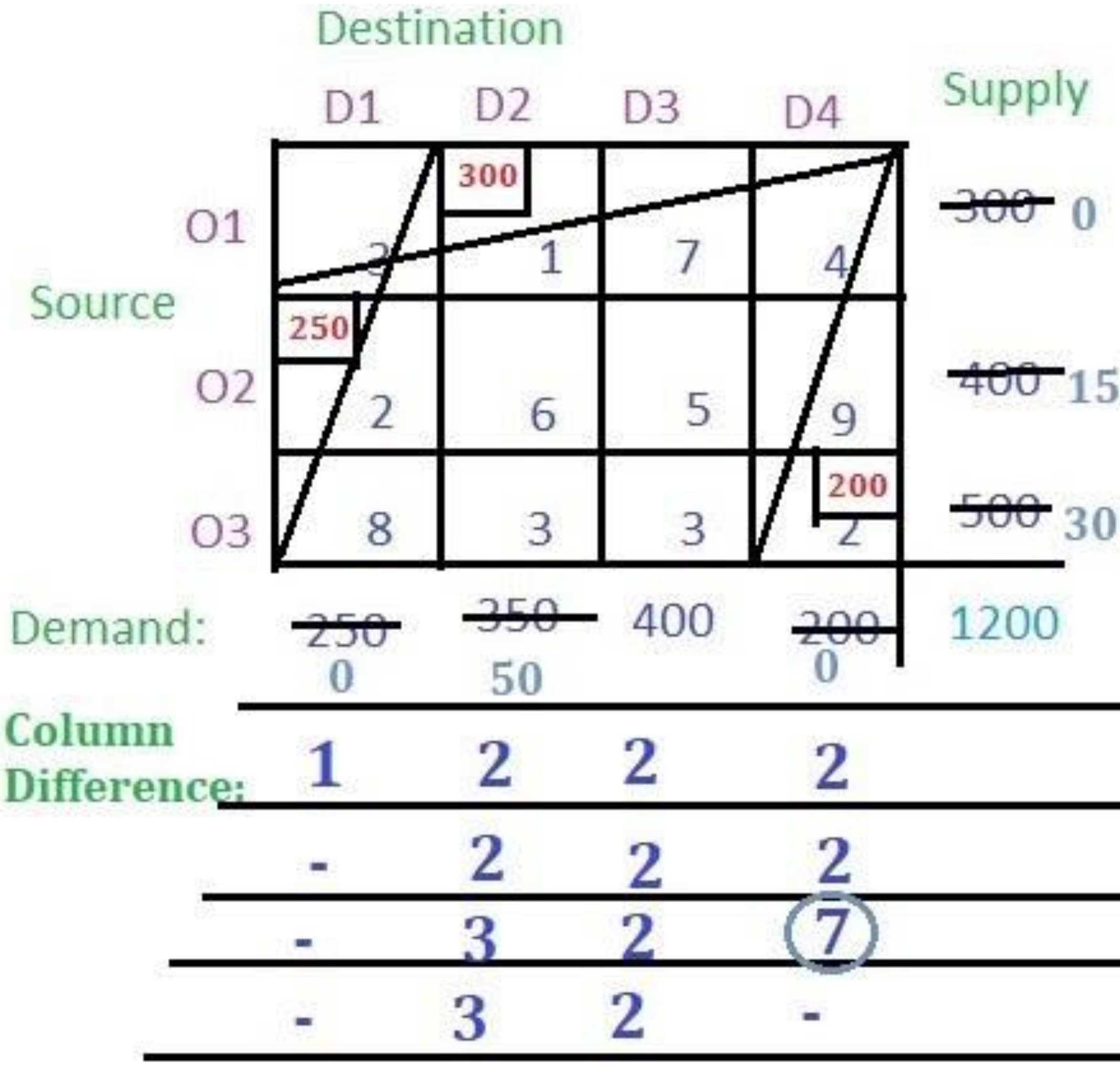
- Now select the maximum penalty which is **7** corresponding to column **D4**. The least cost cell in column **D4** is **(O3, D4)**with cost **2**. The demand is smaller than the supply for cell **(O3, D4)**. Allocate **200**to the cell and cancel the column.



Destination

	D1	D2	D3	D4	Supply
Source O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 15
O3	8	3	3	200 2	~~500~~ 30
Demand:	~~250~~ 0	~~350~~ 50	400	~~200~~ 0	1200
Column Difference:	**1**	**2**	**2**	**2**	
	-	**2**	**2**	**2**	
	-	**3**	**2**	**7**	

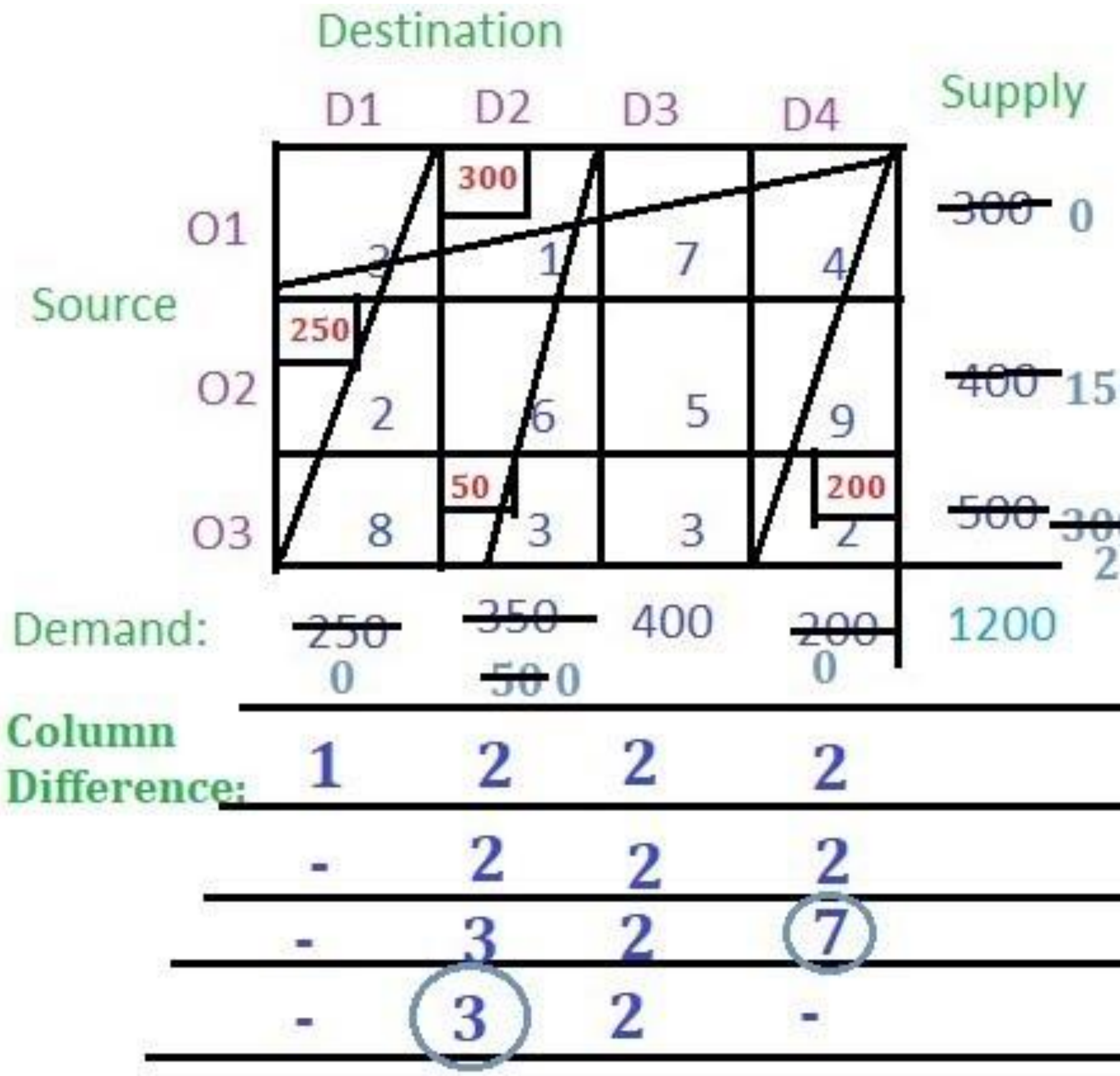
- Find the row difference and the column difference from the remaining cells.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 15
O3	8	3	3	200 2	~~500~~ 30
Demand:	~~250~~ 0	~~350~~ 50	400	~~200~~ 0	1200
Column Difference:	1	2	2	2	
	-	2	2	2	
	-	3	2	7	
	-	3	2	-	

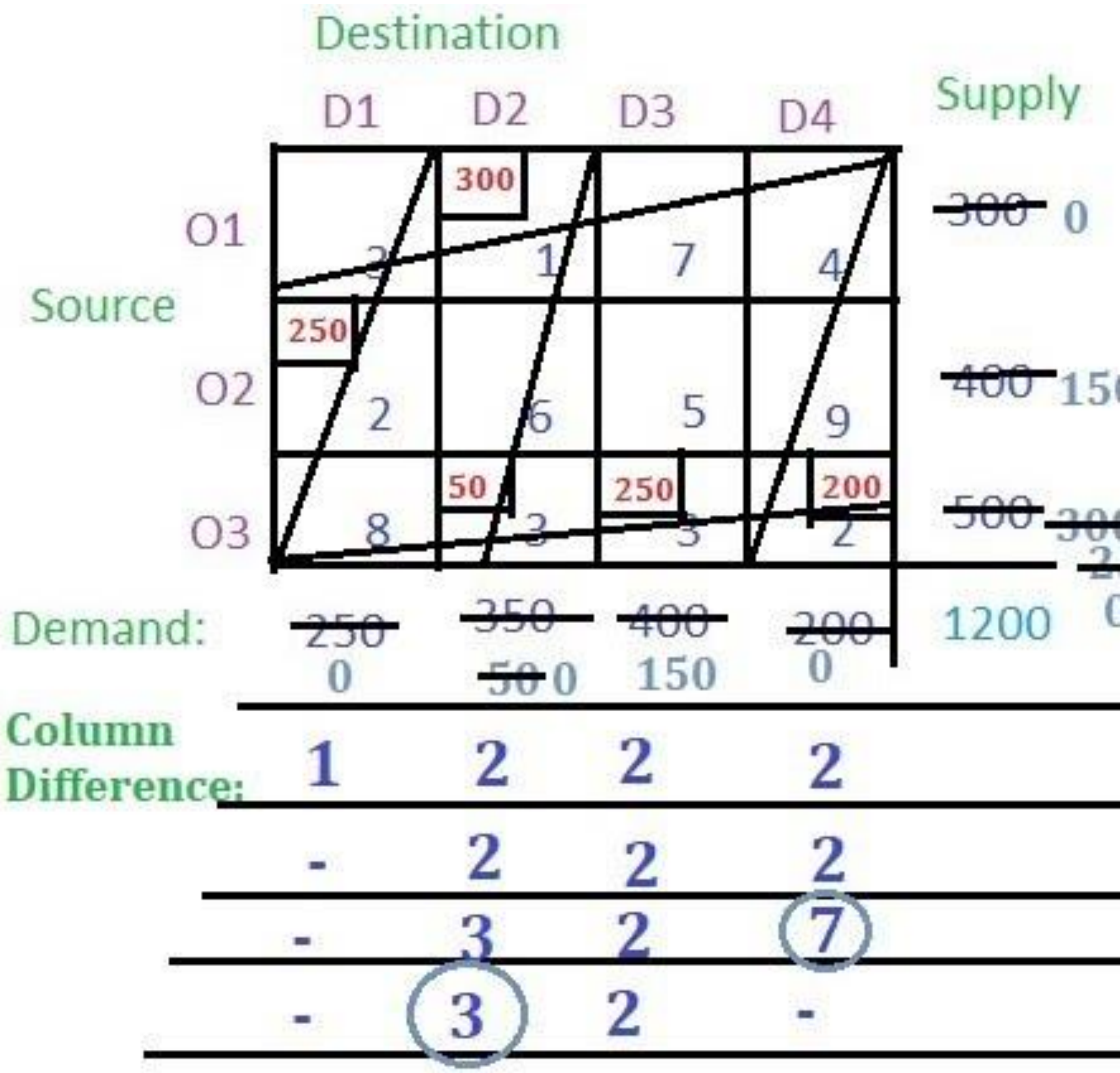
- Now the maximum penalty is **3**corresponding to the column **D2**. The cell with the least value in **D2** is **(O3, D2)**. Allocate the minimum of supply and demand and cancel the column.



Destination

Source	D1	D2	D3	D4	Supply
O1	3	300 1	7	4	~~300~~ 0
O2	250 2	6	5	9	~~400~~ 15
O3	8	50 3	3	200 2	~~500~~ ~~30~~ 2
Demand:	~~250~~ 0	~~350~~ ~~50~~ 0	400	~~200~~ 0	1200
Column Difference:	1	2	2	2	
	-	2	2	2	
	-	3	2	7	
	-	3	2	-	

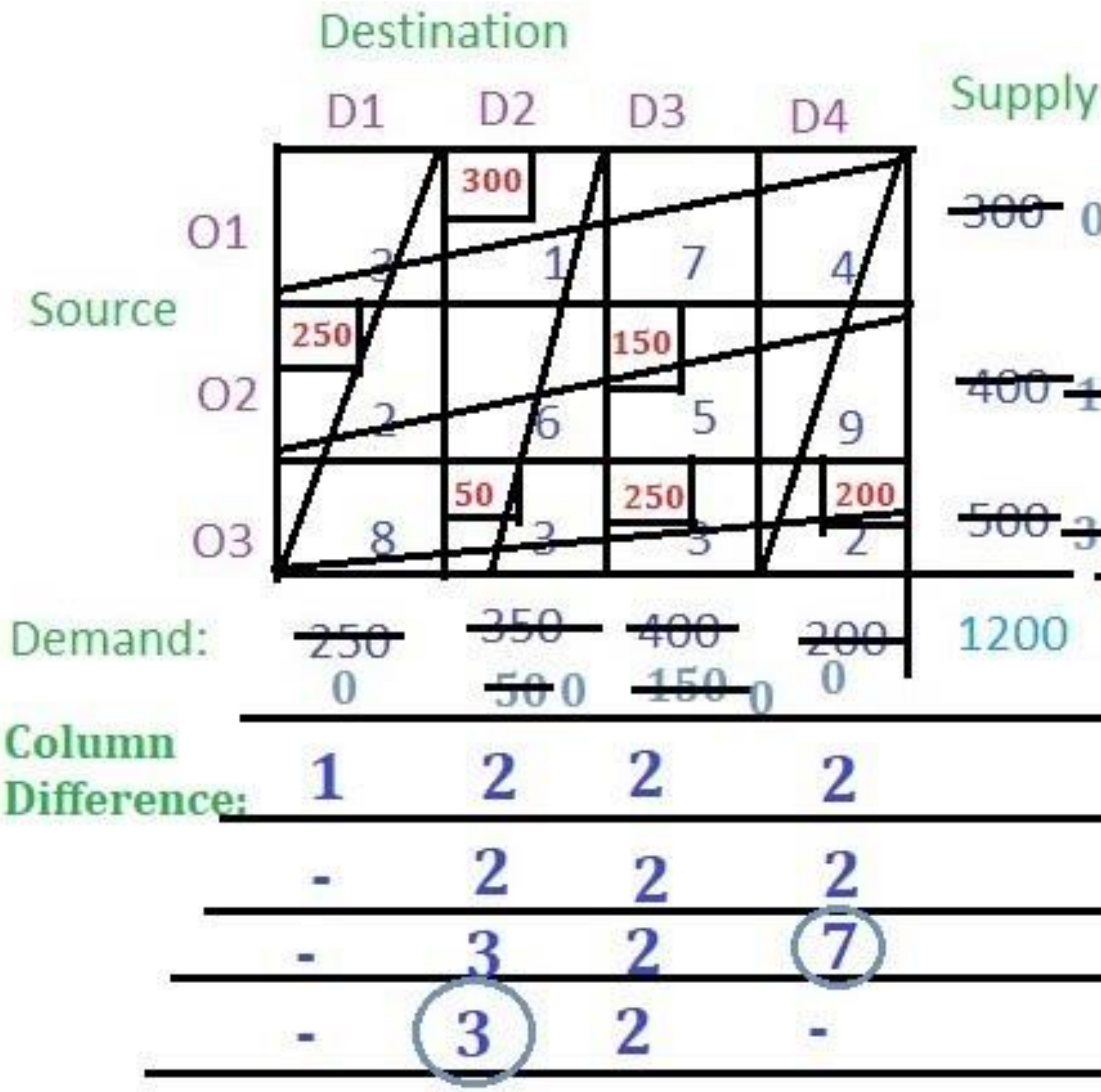
- Now there is only one column so select the cell with the least cost and allocate the value.



Destination

Source	D1	D2	D3	D4	Supply	
O1	3	300 1	7	4	~~300~~ 0	
O2	250 2	6	5	9	~~400~~ 15	
O3	8	50 3	250 3	200 2	~~500~~	~~30~~ 2
Demand:	~~250~~ 0	~~350~~ ~~50~~ 0	~~400~~ 150	~~200~~ 0	1200	0
Column Difference:	1	2	2	2		
	-	2	2	2		
	-	3	2	7		
	-	3	2	-		

- Now there is only one cell so allocate the remaining demand or supply to the cell



Destination

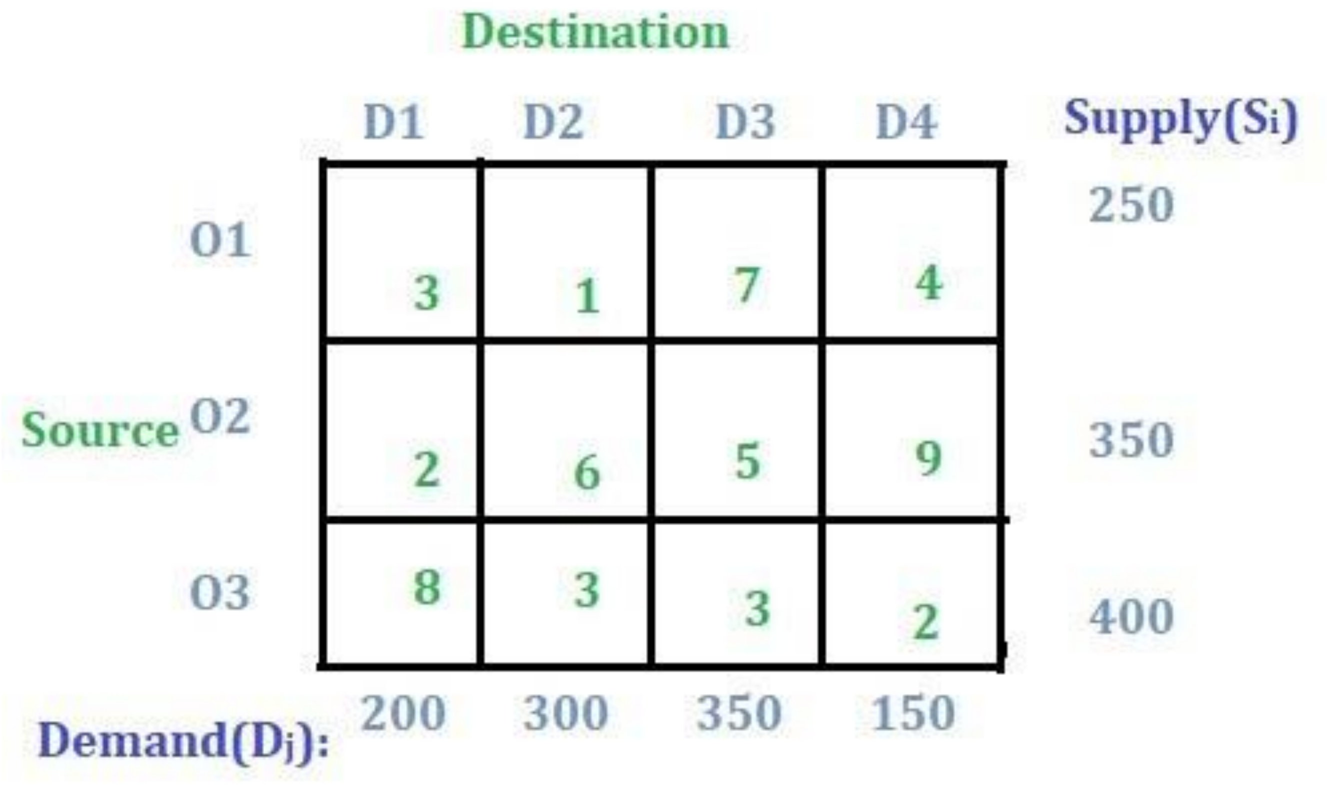
		D1	D2	D3	D4	Supply
Source	O1	3	300 1	7	4	~~300~~ 0
	O2	250 2	6	150 5	9	~~400~~ ~~1~~
	O3	8	50 3	250 3	200 2	~~500~~ ~~3~~
Demand:		~~250~~ 0	~~350~~ ~~50~~ 0	~~400~~ ~~150~~ 0	~~200~~ 0	1200
Column Difference:		**1**	**2**	**2**	**2**	
		-	**2**	**2**	**2**	
		-	**3**	**2**	**7**	
		-	**3**	**2**	**-**	

Optimal Solution By U-V Method:

There are two phases to solve the transportation problem. In the first phase, the initial basic feasible solution has to be found and the second phase involves optimization of the initial basic feasible solution that was obtained in the first phase. There are three methods for finding an initial basic feasible solution,

1. NorthWest Corner Method
2. Least Cost Cell Method
3. Vogel’s Approximation Method

This article will discuss how to optimize the initial basic feasible solution through an explained example. Consider the below transportation problem.

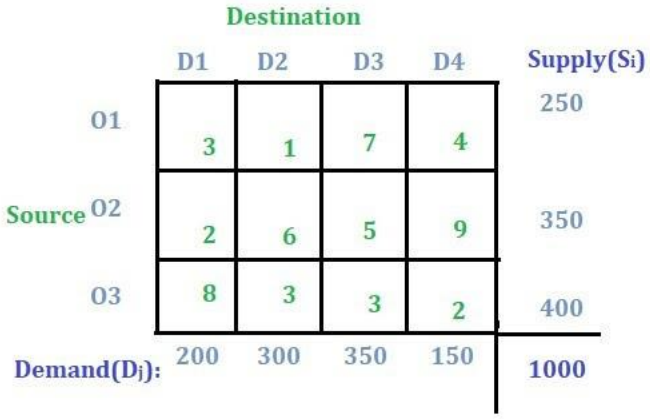


Destination

Source	D1	D2	D3	D4	Supply(Si)
O1	3	1	7	4	250
O2	2	6	5	9	350
O3	8	3	3	2	400
Demand(Dj):	200	300	350	150	

Solution:

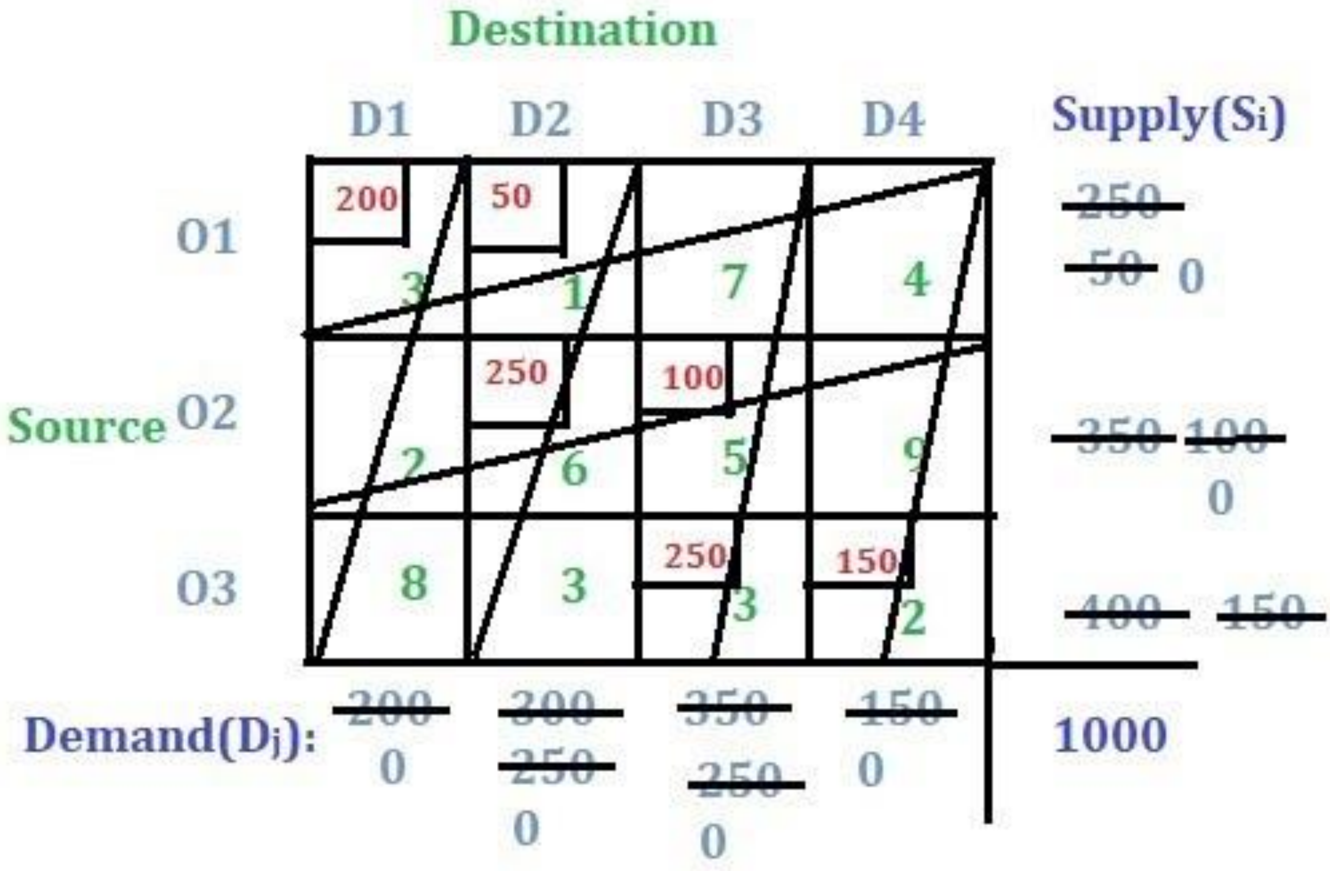
Step 1: Check whether the problem is balanced or not.
If the total sum of all the supply from sources **O1**, **O2**, and **O3** is equal to the total sum of all the demands for destinations **D1**, **D2**, **D3** and **D4** then the transportation problem is a balanced transportation problem.



		Destination				
		D1	D2	D3	D4	Supply(S_i)
	O1	3	1	7	4	250
Source	O2	2	6	5	9	350
	O3	8	3	3	2	400
Demand(D_j):		200	300	350	150	1000

Note: If the problem is not unbalanced then the concept of a dummy row or a dummy column to transform the unbalanced problem to balanced can be followed as discussed in this article.

Step 2: Finding the initial basic feasible solution.
Any of the three aforementioned methods can be used to find the initial basic feasible solution. Here, NorthWest Corner Methodwill be used. And according to the NorthWest Corner Method this is the final initial basic feasible solution:

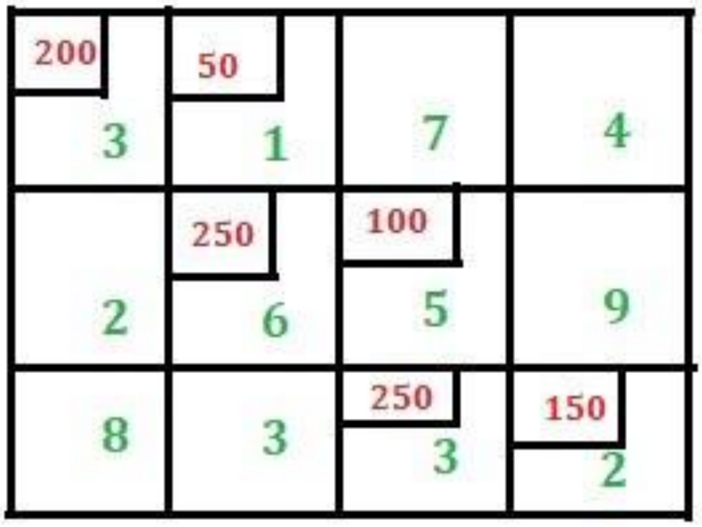


Destination

Source	D1	D2	D3	D4	Supply(Si)
O1	200 3	50 1	7	4	~~250~~ ~~50~~ 0
O2	2	250 6	100 5	9	~~350~~ ~~100~~ 0
O3	8	3	250 3	150 2	~~400~~ ~~150~~
Demand(Dj):	~~200~~ 0	~~300~~ ~~250~~ 0	~~350~~ ~~250~~ 0	~~150~~ 0	1000

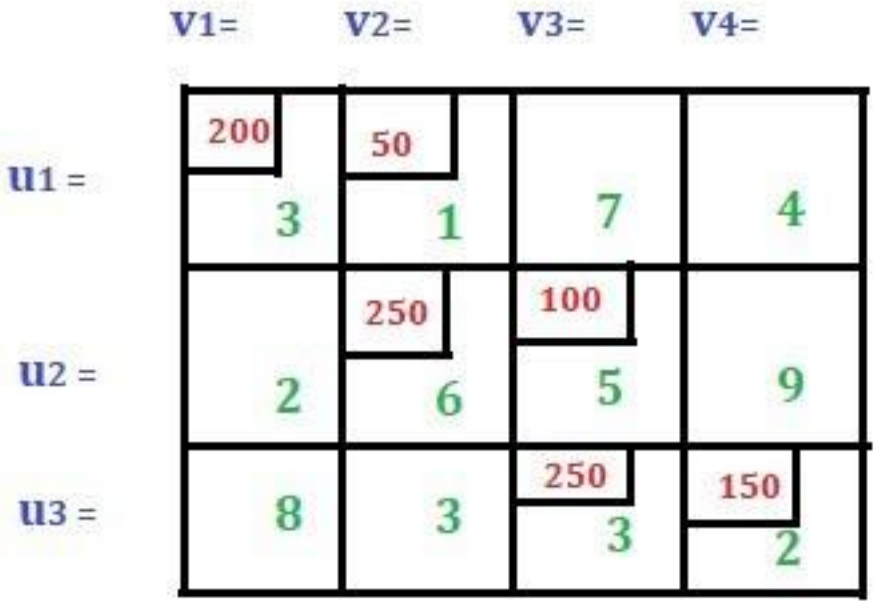
Now, the total cost of transportation will be **(200 * 3) + (50 * 1) + (250 * 6) + (100 * 5) + (250 * 3) + (150 * 2) = 3700**.

Step 3: U-V method to optimize the initial basic feasible solution.
The following is the initial basic feasible solution:



200 3	50 1	7	4
2	250 6	100 5	9
8	3	250 3	150 2

– For U-V method the values **u**$_i$ and **v**$_j$ have to be found for the rows and the columns respectively. As there are three rows so three **u**$_i$ values have to be found i.e. **u**$_1$ for the first row, **u**$_2$ for the second row and **u**$_3$for the third row.
Similarly, for four columns four **v**$_j$ values have to be found i.e. **v**$_1$, **v**$_2$, **v**$_3$ and **v**$_4$. Check the image below:



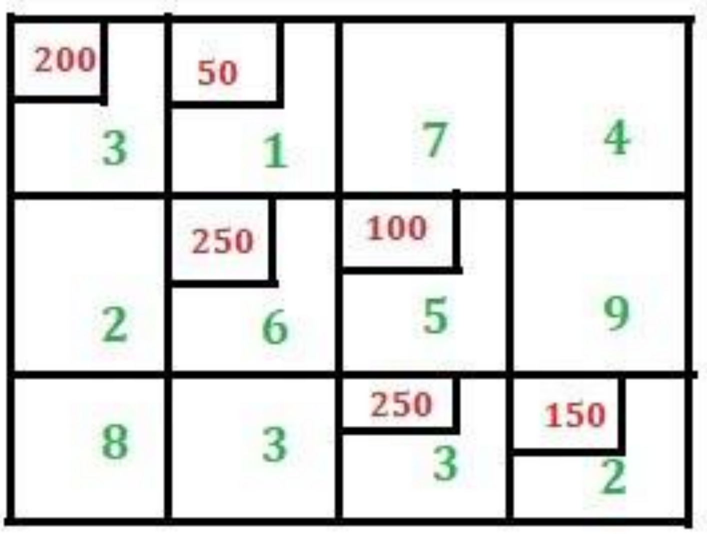
	V1=	V2=	V3=	V4=
u1 =	200 3	50 1	7	4
u2 =	2	250 6	100 5	9
u3 =	8	3	250 3	150 2

Note: If the problem is not unbalanced then the concept of a dummy row or a dummy column to transform the unbalanced problem to balanced can be followed as discussed in this article.

Step 2: Finding the initial basic feasible solution.
Any of the three aforementioned methods can be used to find the initial basic feasible solution. Here, NorthWest Corner Methodwill be used. And according to the NorthWest Corner Method this is the final initial basic feasible solution:

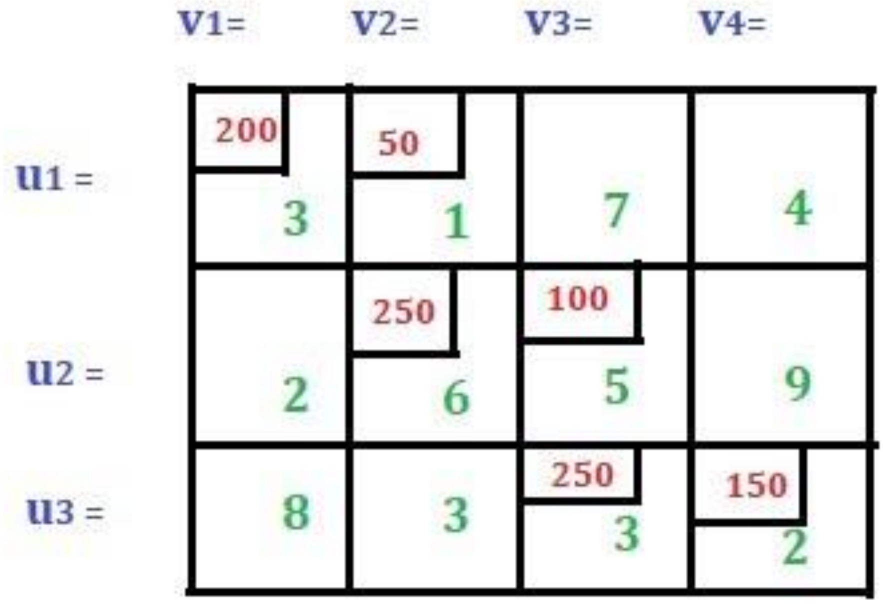
Now, the total cost of transportation will be **(200 * 3) + (50 * 1) + (250 * 6) + (100 * 5) + (250 * 3) + (150 * 2) = 3700**.

Step 3: U-V method to optimize the initial basic feasible solution.
The following is the initial basic feasible solution:



200 3	50 1	7	4
2	250 6	100 5	9
8	3	250 3	150 2

– For U-V method the values $\mathbf{u_i}$ and $\mathbf{v_j}$ have to be found for the rows and the columns respectively. As there are three rows so three $\mathbf{u_i}$ values have to be found i.e. $\mathbf{u_1}$ for the first row, $\mathbf{u_2}$ for the second row and $\mathbf{u_3}$for the third row.
Similarly, for four columns four $\mathbf{v_j}$ values have to be found i.e. $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$ and $\mathbf{v_4}$. Check the image below:



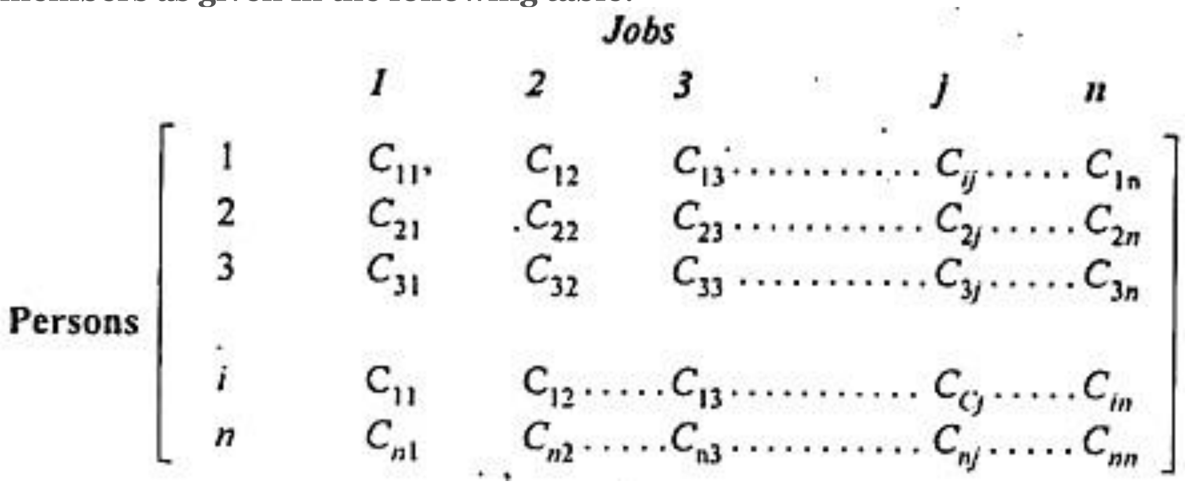
	V1=	V2=	V3=	V4=
u1 =	200 3	50 1	7	4
u2 =	2	250 6	100 5	9
u3 =	8	3	250 3	150 2

Assignment Problem:

Definition of Assignment Problem:

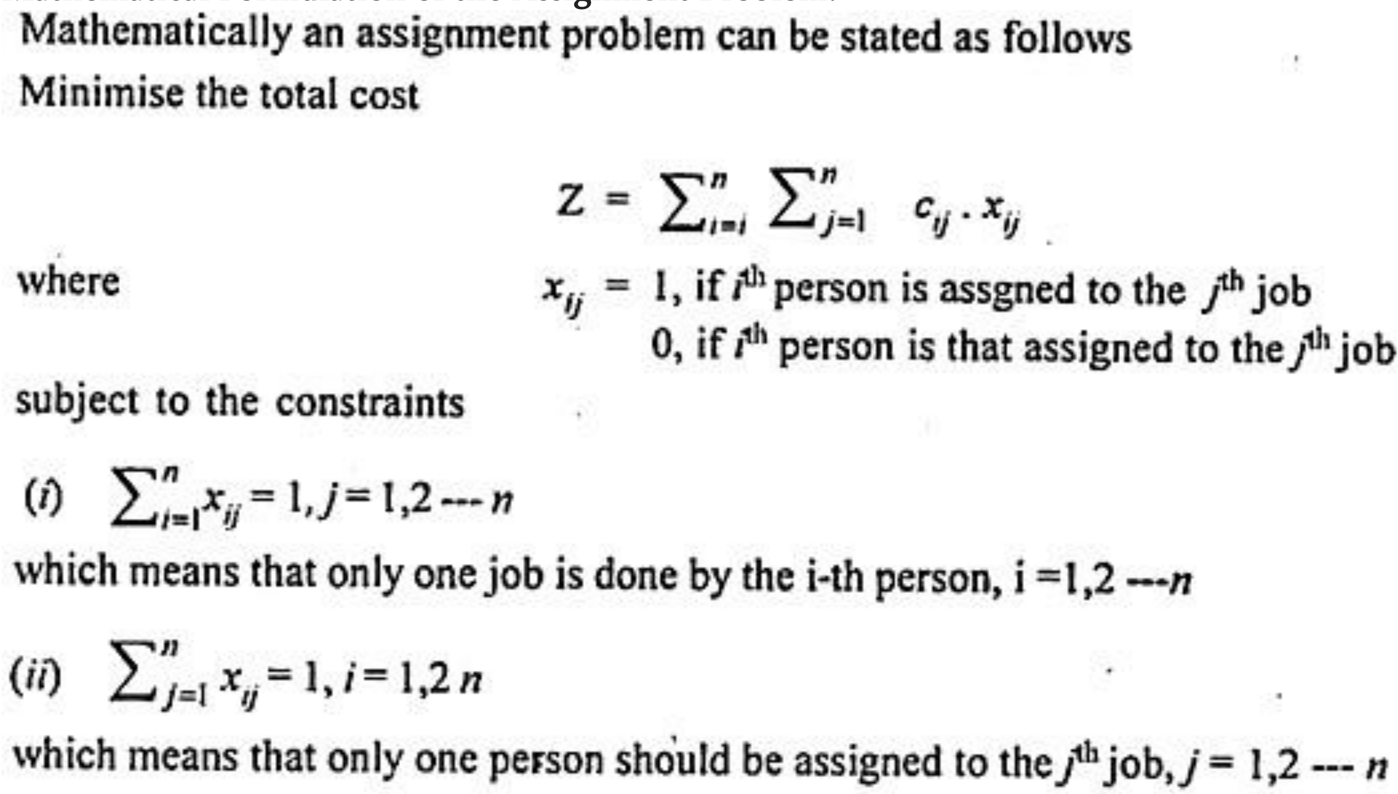
Suppose there are n jobs to be performed and n persons are available for doing these jobs. Assume that each person can do each job at a term, though with varying degree of efficiency, let c_{ij} be the cost if the i-th person is assigned to the j-th job. The problem is to find an assignment (which job should be assigned to which person one on-one basis) So that the total cost of performing all jobs is minimum, problem of this kind are known as assignment problem.

The assignment problem can be stated in the form of n x n cost matrix C real members as given in the following table:



$$\begin{array}{cc|cccccc|}
 & & & & \textit{Jobs} & & & \\
 & & I & 2 & 3 & & j & n \\
 & 1 & C_{11}, & C_{12} & C_{13} & \cdots\cdots & C_{ij} \cdots & C_{1n} \\
 & 2 & C_{21} & C_{22} & C_{23} & \cdots\cdots & C_{2j} \cdots & C_{2n} \\
\text{Persons} & 3 & C_{31} & C_{32} & C_{33} & \cdots\cdots & C_{3j} \cdots & C_{3n} \\
 & i & C_{11} & C_{12} \cdots & C_{13} & \cdots\cdots & C_{Cj} \cdots & C_{in} \\
 & n & C_{n1} & C_{n2} \cdots & C_{n3} & \cdots\cdots & C_{nj} \cdots & C_{nn}
\end{array}$$

Mathematical Formulation of the Assignment Problem:



Mathematically an assignment problem can be stated as follows

Minimise the total cost

$$Z = \sum_{i=i}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij}$$

where x_{ij} = 1, if i^{th} person is assgned to the j^{th} job

0, if i^{th} person is that assigned to the j^{th} job

subject to the constraints

(*i*) $\sum_{i=1}^{n} x_{ij} = 1, j = 1,2 \text{---} n$

which means that only one job is done by the i-th person, i =1,2 ---n

(*ii*) $\sum_{j=1}^{n} x_{ij} = 1, i = 1,2\, n$

which means that only one person should be assigned to the j^{th} job, j = 1,2 --- n

Hungarian Method for Solving Assignment Problem:

The Hungarian method of assignment provides us with an efficient method of finding the optimal solution without having to make a-direct comparison of every solution. It works on the principle of reducing the given cost matrix to a matrix of opportunity costs.

Opportunity cost show the relative penalties associated with assigning resources to an activity as opposed to making the best or least cost assignment. If we can reduce the cost matrix to the extent of having at least one zero in each row and column, it will be possible to make optimal assignment.

The Hungarian method can be summarized in the following steps:
Step 1: Develop the Cost Table from the
If the no of rows are not equal to the no of columns and vice versa, a dummy row or dummy column must be added. The assignment cost for dummy cells are always zero.

Step 2: Find the Opportunity Cost Table:
(a) Locate the smallest element in each row of the given cost table and then subtract that from each element of that row, and

(b) In the reduced matrix obtained from 2 (a) locate the smallest element in each column and then subtract that from each element. Each row and column now have at least one zero value.

Step 3: Make Assignment in the Opportunity Cost Matrix:
The procedure of making assignment is as follows:
(a) Examine rows successively until a row with exactly one unmarked zero is obtained. Make an assignment single zero by making a square around it.

(b) For each zero value that becomes assigned, eliminate (Strike off) all other zeros in the same

Repeat step 3 (a) and 3 (b) for each column also with exactly single zero value all that has not been assigned.

(d) If a row and/or column has two or more unmarked zeros and one cannot be chosen by inspection, then choose the assigned zero cell arbitrarily.

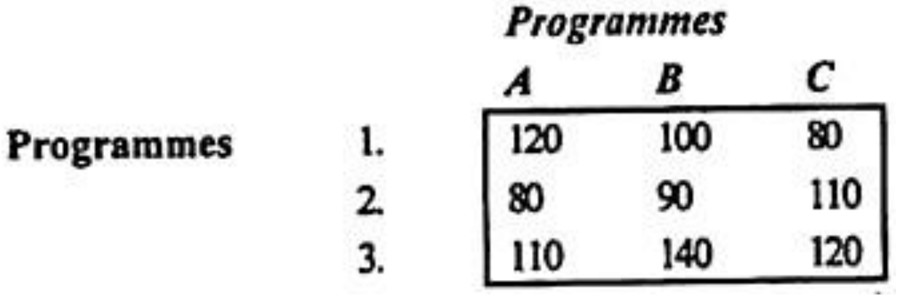
(e) Continue this process until all zeros in row column are either enclosed (Assigned) or struck off (x)

Step 4: Optimality Criterion:
If the member of assigned cells is equal to the numbers of rows column then it is optimal solution. The total cost associated with this solution is obtained by adding original cost figures in the occupied cells.

If a zero cell was chosen arbitrarily in step (3), there exists an alternative optimal solution. But if no optimal solution is found, then go to step (5).

Step 5: Revise the Opportunity Cost Table:

1. In a computer centre after studying carefully the three expert programmes, the head of computer centre, estimates the computer time in minutes required by the experts for the application programmes as follows:



		Programmes		
		A	***B***	***C***
Programmes	1.	120	100	80
	2.	80	90	110
	3.	110	140	120

Assign the programmers to the programmes in such a way that the total computer time is minimum.

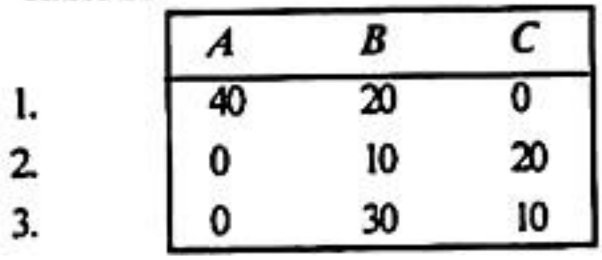
Solution:

The Hungarian method is used to obtain an optimal solution.

Step (1) & (2):
The minimum time element in row 1, 2 and 3 is 80, 80 and 110. resp. Subtract these elements from all elements in this respective row.

The reduced time matrix is shown in following table (1) Table 1:

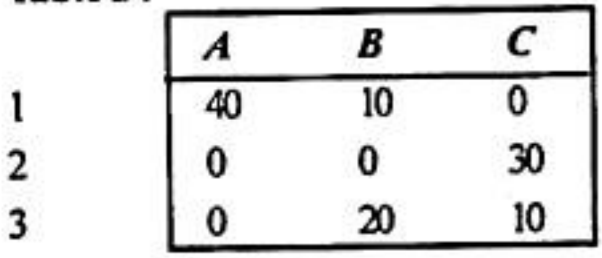
Table 1:



	A	B	C
1.	40	20	0
2.	0	10	20
3.	0	30	10

In reduced Table (1) the minimum time element in columns A, B, and C is 0,10 and 0 resp, subtract these elements from all elements in this resp. column to get the reduced time matrix as shown in Table 2.

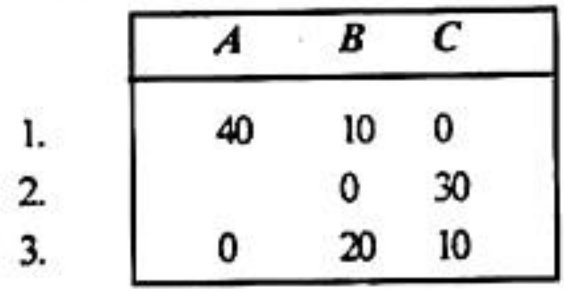
Table 2 :



	A	B	C
1	40	10	0
2	0	0	30
3	0	20	10

Step 3 (a):
Examine all the rows starting from first one- until a row containing only single zero element is located, Here, rows 1 and 3 have only one zero in the cells (1, C) and (3,A) resp, we assigned these zeros. All zeros in the assigned column are crossed off as shown in table 3.

Table 3 :



	A	B	C
1.	40	10	0
2.		0	30
3.	0	20	10

(b) We now examine each column starting from A in table 3, There is one zero in column B in the cell (2, B). Assign this cell as shown in table 4.

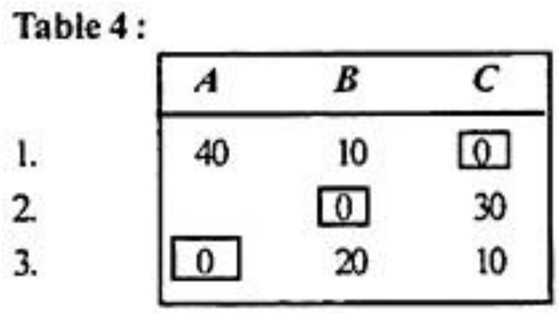
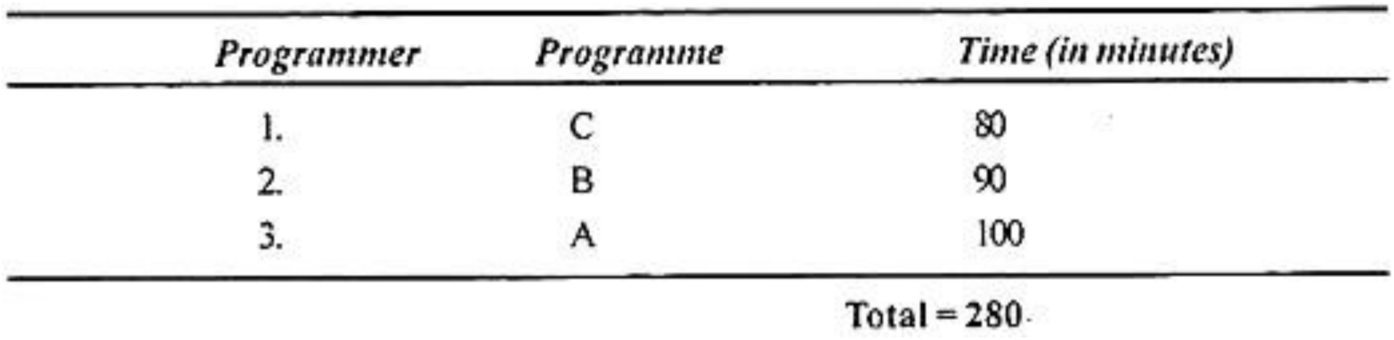


Table 4 :

	A	*B*	*C*
1.	40	10	$\boxed{0}$
2.		$\boxed{0}$	30
3.	$\boxed{0}$	20	10

(c) Since the no of Assignments (= 3) equal the no of rows (= 3), the optimal solution is obtained.

The pattern of assignment among programmers and programmes with their respective line (in minutes) is given below.



Programmer	*Programme*	*Time (in minutes)*
1.	C	80
2.	B	90
3.	A	100
		Total = 280

Branch and Bound Technique For Assignment Problem:

Branch and bound

What is Branch and bound?

Branch and bound is one of the techniques used for problem solving. It is similar to the backtracking since it also uses the state space tree. It is used for solving the optimization problems and minimization problems. If we have given a maximization problem then we can convert it using the Branch and bound technique by simply converting the problem into a maximization problem.

Let's understand through an example.

Jobs = {j1, j2, j3, j4}

P = {10, 5, 8, 3}

d = {1, 2, 1, 2}

The above are jobs, problems and problems given. We can write the solutions in two ways which are given below:

Suppose we want to perform the jobs j1 and j2 then the solution can be represented in two ways:

The first way of representing the solutions is the subsets of jobs.

S1 = {j1, j4}

The second way of representing the solution is that first job is done, second and third jobs are not done, and fourth job is done.

S2 = {1, 0, 0, 1}

The solution s1 is the variable-size solution while the solution s2 is the fixed-size solution.

As we can observe in the above figure that the breadth first search is performed but not the depth first search. Here we move breadth wise for exploring the solutions. In backtracking, we go depth-wise whereas in branch and bound, we go breadth wise.

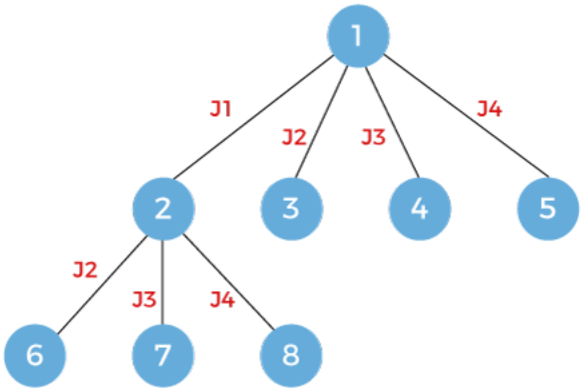
Now one level is completed. Once I take first job, then we can consider either j2, j3 or j4. If we follow the route then it says that we are doing jobs j1 and j4 so we will not consider jobs j2 and j3.

As we can observe in the above figure that the breadth first search is performed but not the depth first search. Here we move breadth wise for exploring the solutions. In backtracking, we go depth-wise whereas in branch and bound, we go breadth wise.

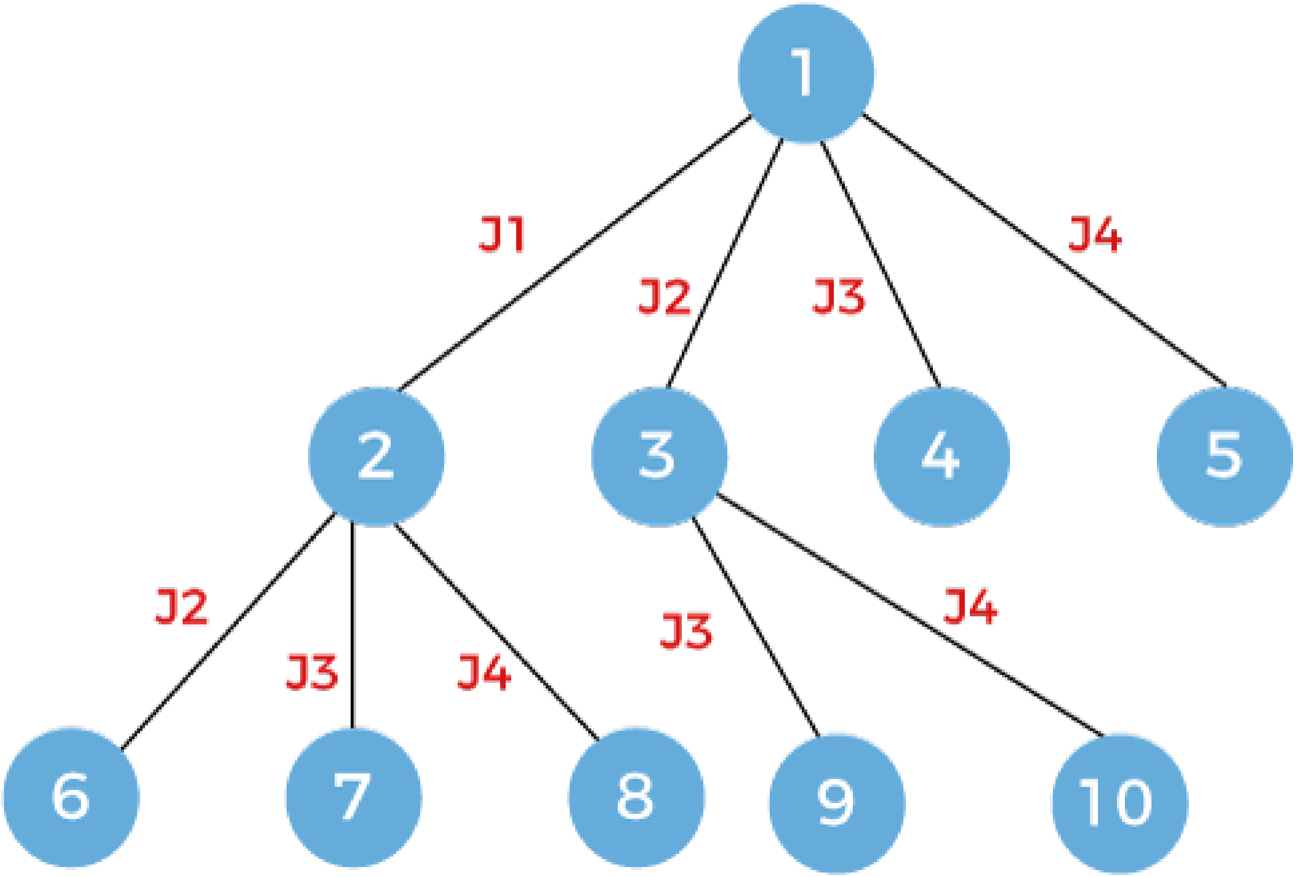
Now one level is completed. Once I take first job, then we can consider either j2, j3 or j4. If we follow the route then it says that we are doing jobs j1 and j4 so we will not consider jobs j2 and j3.

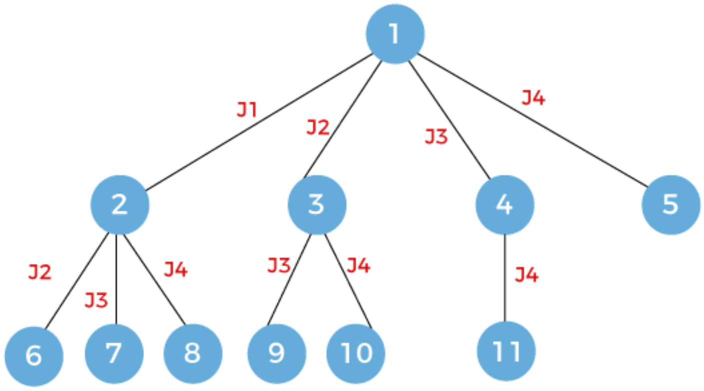
As we can observe in the above figure that the breadth first search is performed but not the depth first search. Here we move breadth wise for exploring the solutions. In backtracking, we go depth-wise whereas in branch and bound, we go breadth wise.

Now one level is completed. Once I take first job, then we can consider either j2, j3 or j4. If we follow the route then it says that we are doing jobs j1 and j4 so we will not consider jobs j2 and j3.



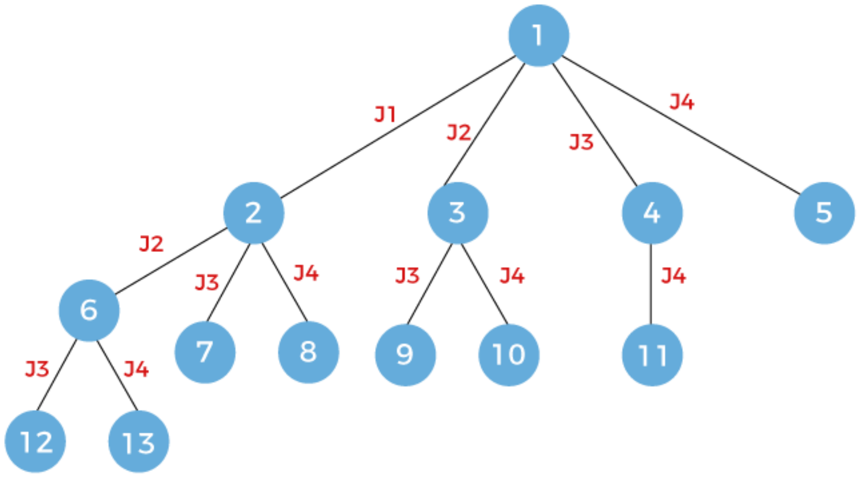
Now we will consider the node 3. In this case, we are doing job j2 so we can consider either job j3 or j4. Here, we have discarded the job j1.





Now we will expand the node 4. Since here we are doing job j3 so we will consider only job j4.

Now we will expand node 6, and here we will consider the jobs j3 and j4.



Now we will expand node 7 and here we will consider job j4.

Now we will expand node 9, and here we will consider job j4.

The above is the state space tree for the solution s1 = {j1, j4}

Short Answers:

1.Explain about Transportation Problem

2.What are Types of Transportation Problem?

3.Explain about U-V Method
4.Explain about Branch and Bound Technique
5.Explain about Initial Basic Solution

Long Answers:

6.Explain about Hungerian Method
7.Explain about Assignment Problem
8.Explain about Mathematical Model of Transportation Problem
9.Explain about Formulation of Assignment Problem
10.Explain about Branch Bound Techniques

UNIT-3

Network Techniques:

SCOPE AND DEFINITION OF NETWORK MODELS
A multitude of operations research situations can be modeled and solved as networks (nodes connected by branches):
1. Designofanoffshorenatural-gaspipelinenetworkconnectingwellheadsin the

Gulf of Mexico to an inshore delivery point.The objective of the model is to mini- mize the cost of constructing the pipeline.
2. Determination of the shortest route between two cities in an existing network of roads.
3. Determination of the maximum capacity (in tons per year) of a coal slurry pipeline network joining coal mines in Wyomingwith power plants in Houston. (Slurry pipelines transport coal by pumping water through specially designed pipes.)
4. Determination of the time schedule (start and completiondates) for the activ- itiesof a construction project.
5. Determination of the minimum-cost flow schedule from oil fields to refineries through a pipeline network.

The solution of these situations, and others like it, is accomplished through a variety of network optimization algorithms. This chapter presents four of these algorithms.

.1 Minimal spanning tree (situation 1)
2. Shortest-route algorithm (situation 2)
3. Maximal-flow algorithm (situation 3)
4. Critical path (CPM) algorithm (situation 4)

For the fifth situation, the minimum-cost capacitated network algorithm is presented in Section 20.1 on the CD.

Network Definitions. A network consists ofa set ofnodes linked by arcs (or branches). Thenotation for describing a network is (N, A), where N is the set of nodes and A is the set of arcs. As an illustration, the network in Figure 6.1 is described as

N= {1, 2, 3, 4,5}

A = {(1,2), (1, 3), (2, 3), (2, 5), (3, 4), (3, 5), (4, 2), (4,5))

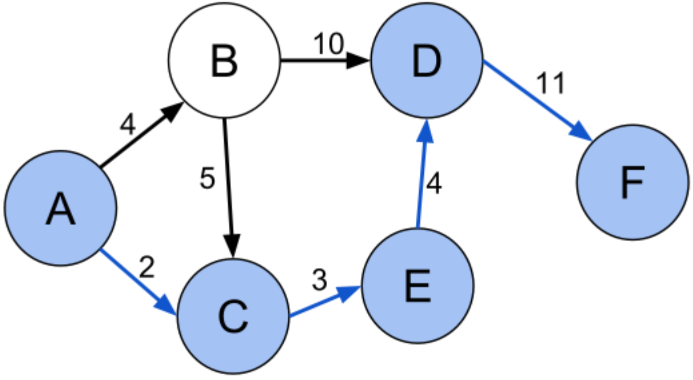
Associated with each network is a flow (e.g., oil products flow in a pipeline and automobile traffic flows in highways). In general, the flow in a network is limited by a network is limited by the capacity of its arcs.

Examples of a tree and a spanning tree

Spanning tree

An arc is said to be directed or oriented if it allows positive flow in one direction and zero flow in the opposite direction. A directed network has alldirected arcs.

A path is a sequence of distinct arcs that join two nodes through other nodes regardless of the



direction of flow in each arc. A path forms a cycle or a loop if it con- nectsa node to itself through other nodes. For example, in Figure 6.1, the arcs (2,3), (3, 4), and (4, 2) form a cycle.

A connected network is such that every two distinct nodes are linked by at least one path. The network in Figure 6.1 demonstrates this type of network. A tree is a cycle-free connected network comprised of a subset of all the nodes, and a spanning tree is a tree that links all the nodes of the network. Figure 6.2 provides examples

Shortest-Path Model:

In graph theory, the **shortest path problem** is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.

The problem of finding the shortest path between two intersections on a road map may be modeled as a special case of the shortest path problem in graphs, where the vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of the segment.

The shortest path problem can be defined for graphs whether undirected, directed, or mixed. It is defined here for undirected graphs; for directed graphs the definition of path requires that consecutive vertices be connected by an appropriate directed edge.

Two vertices are adjacent when they are both incident to a common edge. A path in an undirected graph is a sequence of vertices such that is adjacent to for . Such a path is called a path of length from to . (The are variables; their numbering here relates to their position in the sequence and needs not to relate to any canonical labeling of the vertices.)

Let be the edge incident to both and . Given a real-valued weight function , and an undirected (simple) graph , the shortest path from to is the path (where and) that over all possible minimizes the sum When each edge in the graph has unit weight or , this is equivalent to finding the path with fewest edges.

The problem is also sometimes called the **single-pair shortest path problem**, to distinguish it from the following variations:

- The **single-source shortest path problem**, in which we have to find shortest paths from a source vertex *v* to all other vertices in the graph.
- The **single-destination shortest path problem**, in which we have to find shortest paths from all vertices in the directed graph to a single destination vertex *v*. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.
- The **all-pairs shortest path problem**, in which we have to find shortest paths between every pair of vertices *v*, *v'* in the graph.

These generalizations have significantly more efficient algorithms than the simplistic approach of running a single-pair shortest path algorithm on all relevant pairs of vertices.

The most important algorithms for solving this problem are:

- Dijkstra's algorithm solves the single-source shortest path problem with non-negative edge weight.
- Bellman–Ford algorithm solves the single-source problem if edge weights may be negative.
- A* search algorithm solves for single-pair shortest path using heuristics to try to speed up the search.
- Floyd–Warshall algorithm solves all pairs shortest paths.
- Johnson's algorithm solves all pairs shortest paths, and may be faster than Floyd–Warshall on sparse graphs.
- Viterbi algorithm solves the shortest stochastic path problem with an additional probabilistic weight on each node.

Shortest-Route Algorithms:

This section presents two algorithms for solving both cyclic (i.e., containing loops) and acyclic networks:
.1 Dijkstra's algorithm
2. Floyd's algorithm
Dijkstra's algorithm is designed to determine the shortest routes between the source node and every other node in the network. Floyd's algorithm is more general because it allows the determination of the shortest route between any two nodes in the
network.
Dijkstra's Algorithm:

Let u; be the shortest distance from source node 1to node i, and define dy (20) as the length of arc (i, j).
Then thealgorithm defines thelabel for an immediately succeeding node j as
[u; i] = [u; +dis il, di =0
The label for the starting node is [0,--], indicatingthat the node has no predecessor. Node

labelsi n Diikstra's algorithm are of twotypes: temporary and permanent.
A temporary label ismodified if a shorterroute to a node can be found. If no better route can be found, the status of the temporary label is changed to permanent.
Step 0. Label the source node(node 1) with the permanent label [0, .J- Set i= .1 Step i.
(a) Computethe temporary labels [u; + di, i] for each node j that can be reached from node i, provided i is not permanently labeled. If node i is already labeled with [u; k] through another node k and fi u; + di < u;, replace (u;, k] with [u; + dijni].
(b) If all the nodes have permanent labels, stop. Otherwise, select the label [ur, s] having the shortest distance (= ur) among all

Dijikstras Algorithm:

Dijkstra's algorithm is one of the most popular algorithms for solving many single-source shortest path problems having non-negative edge weight in the graphs i.e., it is to find the shortest distance between two vertices on a graph. It was conceived by computer scientist **Edsger W. Dijkstra** in 1956 and published three years later.

Dijkstra's Algorithm has several real-world use cases, some of which are as follows:

1. **Digital Mapping Services in Google Maps:** Many times we have tried to find the distance in G-Maps, from one city to another, or from your location to the nearest desired location. There encounters the Shortest Path Algorithm, as there are various routes/paths connecting them but it has to show the minimum distance, so Dijkstra's Algorithm is used to find the minimum distance between two locations along the path. Consider India as a graph and represent a city/place with a vertex and the route between two cities/places as an edge, then by using this

algorithm, the shortest routes between any two cities/places or from one city/place to another city/place can be calculated.

2. **Social Networking Applications:** In many applications you might have seen the app suggests the list of friends that a particular user may know. How do you think many social media companies implement this feature efficiently, especially when the system has over a billion users. The standard Dijkstra algorithm can be applied using the shortest path between users measured through handshakes or connections among them. When the social networking graph is very small, it uses standard Dijkstra's algorithm along with some other features to find the shortest paths, and however, when the graph is becoming bigger and bigger, the standard algorithm takes a few several seconds to count and alternate advanced algorithms are used.
3. **Telephone Network:** As we know, in a telephone network, each line has a bandwidth, 'b'. The bandwidth of the transmission line is the highest frequency that line can support. Generally, if the frequency of the signal is higher in a certain line, the signal is reduced by that line. Bandwidth represents the amount of information that can be transmitted by the line. If we imagine a city to be a graph, the vertices represent the switching stations, and the edges represent the transmission lines and the weight of the edges represents 'b'. So as you can see it can fall into the category of shortest distance problem, for which the Dijkstra is can be used.
4. **IP routing to find Open shortest Path First:** Open Shortest Path First (OSPF) is a link-state routing protocol that is used to find the best path between the source and the destination router using its own Shortest Path First. Dijkstra's algorithm is widely used in the routing protocols required by the routers to update their forwarding table. The algorithm provides the shortest cost path from the source router to other routers in the network.
5. **Flighting Agenda:** For example, If a person needs software for making an agenda of flights for customers. The agent has access to a database with all airports and flights. Besides the flight number, origin airport, and destination, the flights have departure and arrival time. Specifically, the agent wants to determine the earliest arrival time for the destination given an origin airport and start time. There this algorithm comes into use.
6. **Designate file server:** To designate a file server in a LAN(local area network), Dijkstra's algorithm can be used. Consider that an infinite amount of time is required for transmitting files from one computer to another computer. Therefore to minimize the number of "hops" from the file server to every other computer on the network the idea is to use Dijkstra's algorithm to minimize the shortest path between the networks resulting in the minimum number of hops.
7. **Robotic Path:** Nowadays, drones and robots have come into existence, some of which are manual, some automated. The drones/robots which are automated and are used to deliver the packages to a specific location or used for a task are loaded with this algorithm module so that when the source and destination is known, the robot/drone moves in the ordered direction by following the shortest path to keep delivering the package in a minimum amount of tim

8. **Examples:**

Let u; be the shortest distance from source node 1to node i, and define dy (20) as the length of arc (i, j).

Then thealgorithm defines thelabel for an immediately succeeding node j as

[u; i] = [u; +dis il, di =0

The label for the starting node is [0,--], indicatingthat the node has no predecessor. Node labelsi n Diikstra's algorithm are of twotypes: temporary and permanent.

A temporary label ismodified if a shorterroute to a node can be found. If no better route can be found, the status of the temporary label is changed to permanent.

Step 0. Label the source node(node 1) with the permanent label [0, .J- Set i= .1 Step i.

(a) Computethe temporary labels [u; + di, i] for each node j that can be reached from node i, provided i is not permanently labeled. If node i is already labeled with [u; k] through another node k
and fi u; + di < u;, replace (u;, k] with [u; + dijni].

Floyds Algorithms:

Floyd Warshall Algorithm

The Floyd Warshall Algorithm is for solving all pairs of shortest-path problems. The problem is to find the shortest distances between every pair of vertices in a given edge-weighted directed Graph.

the **Floyd–Warshall algorithm** (also known as **Floyd's algorithm**, the **Roy–Warshall algorithm**, the **Roy–Floyd algorithm**, or the **WFI algorithm**) is an algorithm for finding shortest paths in a directed weighted graph with positive or negative edge weights (but with no negative cycles).[1][2] A single execution of the algorithm will find the lengths (summed weights) of shortest paths between all pairs of vertices. Although it does not return details of the paths themselves, it is possible to reconstruct the paths with simple modifications to the algorithm. Versions of the algorithm can also be used for

finding the transitive closure of a relation , or (in connection with the Schulze voting system) widest paths between all pairs of vertices in a weighted graph.

.

Floyd-Warshall Algorithm

Let the vertices of G be V = {1, 2........n} and consider a subset {1, 2........k} of vertices for some k. For any pair of vertices i, j ∈ V, considered all paths from i to j whose intermediate vertices are all drawn from {1, 2.......k}, and let p be a minimum weight path from amongst them. The Floyd-Warshall algorithm exploits a link between path p and shortest paths from i to j with all intermediate vertices in the set {1, 2.......k-1}. The link depends on whether or not k is an intermediate vertex of path p.

If k is not an intermediate vertex of path p, then all intermediate vertices of path p are in the set {1, 2........k-1}. Thus, the shortest path from vertex i to vertex j with all intermediate vertices in the set {1, 2.......k-1} is also the shortest path i to j with all intermediate vertices in the set {1, 2.......k}.

If k is an intermediate vertex of path p, then we break p down into i → k → j.

Let $d_{ij}^{(k)}$ be the weight of the shortest path from vertex i to vertex j with all intermediate vertices in the set {1, 2.......k}.

The strategy adopted by the Floyd-Warshall algorithm is **Dynamic Programming**. The running time of the Floyd-Warshall algorithm is determined by the triply nested for loops of lines 3-6. Each execution of line 6 takes O (1) time. The algorithm thus runs in time $\theta(n^3)$.

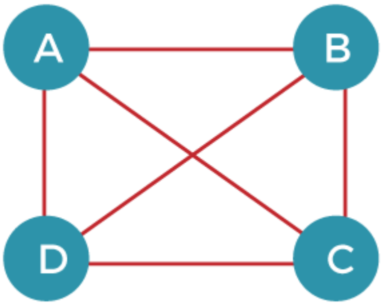
Minimum Spanning Tree:

Before knowing about the minimum spanning tree, we should know about the spanning tree.

To understand the concept of spanning tree, consider the below graph:

Before knowing about the minimum spanning tree, we should know about the spanning tree.

To understand the concept of spanning tree, consider the below graph:



The above graph can be represented as G(V, E), where 'V' is the number of vertices, and 'E' is the number of edges. The spanning tree of the above graph would be represented as G`(V`, E`). In this case, V` = V means that the number of vertices in the spanning tree would be the same as the number of vertices in the graph, but the number of edges would be different. The number of edges in the spanning tree is the subset of the number of edges in the original graph. Therefore, the number of edges can be written as:

E` € E

It can also be written as:

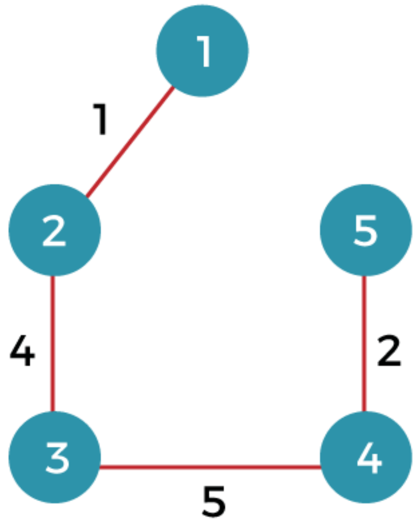
E` = |V| - 1

Two conditions exist in the spanning tree, which is as follows:

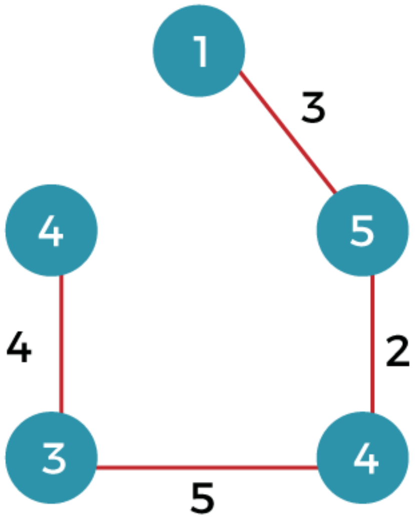
- The number of vertices in the spanning tree would be the same as the number of vertices in the original graph.
  V` = V
- The number of edges in the spanning tree would be equal to the number of edges minus 1.
  E` = |V| - 1
- The spanning tree should not contain any cycle.
- The spanning tree should not be disconnected.

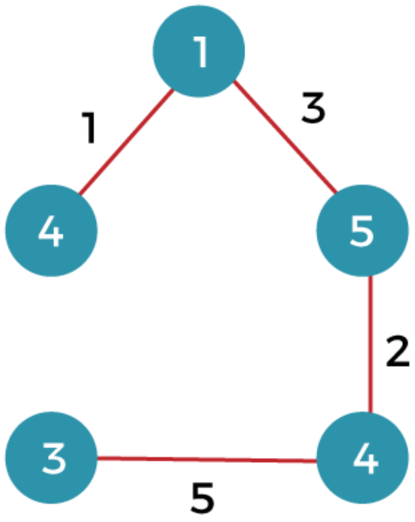
  Consider the below graph:

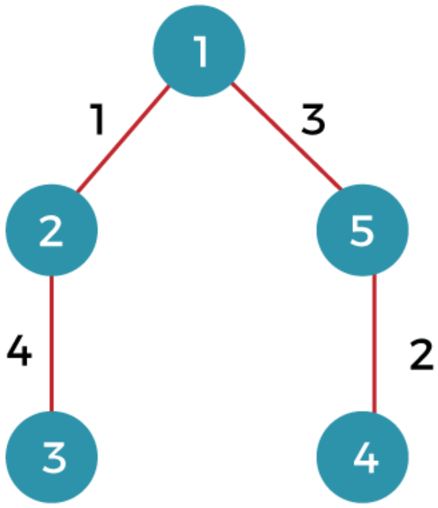
- The above graph contains 5 vertices. As we know, the vertices in the spanning tree would be the same as the graph; therefore, V` is equal 5. The number of edges in the spanning tree would be equal to (5 - 1), i.e., 4. The following are the possible spanning trees:



-


1
3
4
5
4
2
3
4
5


1
1
3
4
5
2
3
4
5


1
1
3
2
5
4
2
3
4

The following are the spanning trees that we can make from the above graph.

- The first spanning tree is a tree in which we have removed the edge between the vertices 1 and 5 shown as below: The sum of the edges of the above tree is (1 + 4 + 5 + 2): 12
- The second spanning tree is a tree in which we have removed the edge between the vertices 1 and 2 shown as below: The sum of the edges of the above tree is (3 + 2 + 5 + 4) : 14
- The third spanning tree is a tree in which we have removed the edge between the vertices 2 and 3 shown as below: The sum of the edges of the above tree is (1 + 3 + 2 + 5) : 11
- The fourth spanning tree is a tree in which we have removed the edge between the vertices 3 and 4 shown as below: The sum of the edges of the above tree is (1 + 3 + 2 + 4) : 10. The edge cost 10 is minimum so it is a minimum spanning tree.

Prime Algorithm:

Prim's Algorithm

In this article, we will discuss the prim's algorithm. Along with the algorithm, we will also see the complexity, working, example, and implementation of prim's algorithm.

Before starting the main topic, we should discuss the basic and important terms such as spanning tree and minimum spanning tree.

Spanning tree - A spanning tree is the subgraph of an undirected connected graph.

Minimum Spanning tree - Minimum spanning tree can be defined as the spanning tree in which the sum of the weights of the edge is minimum. The weight of the spanning tree is the sum of the weights given to the edges of the spanning tree.

Now, let's start the main topic.

Prim's Algorithm is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

How does the prim's algorithm work?

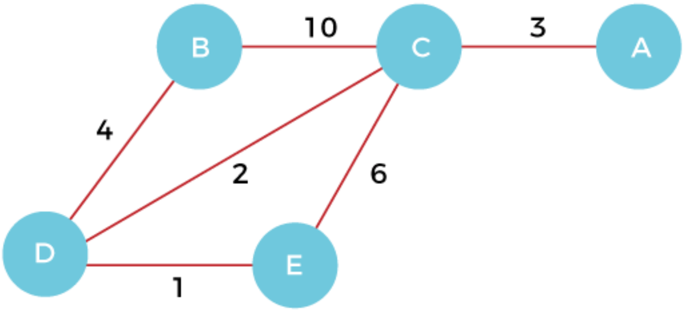
Prim's algorithm is a greedy algorithm that starts from one vertex and continue to add the edges with the smallest weight until the goal is reached. The steps to implement the prim's algorithm are given as follows -

- First, we have to initialize an MST with the randomly chosen vertex.
- Now, we have to find all the edges that connect the tree in the above step with the new vertices. From the edges found, select the minimum edge and add it to the tree.
- Repeat step 2 until the minimum spanning tree is formed.

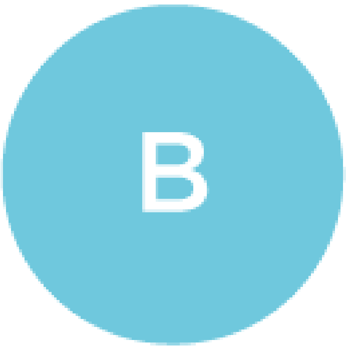
Example of prim's algorithm

Now, let's see the working of prim's algorithm using an example. It will be easier to understand the prim's algorithm using an example.

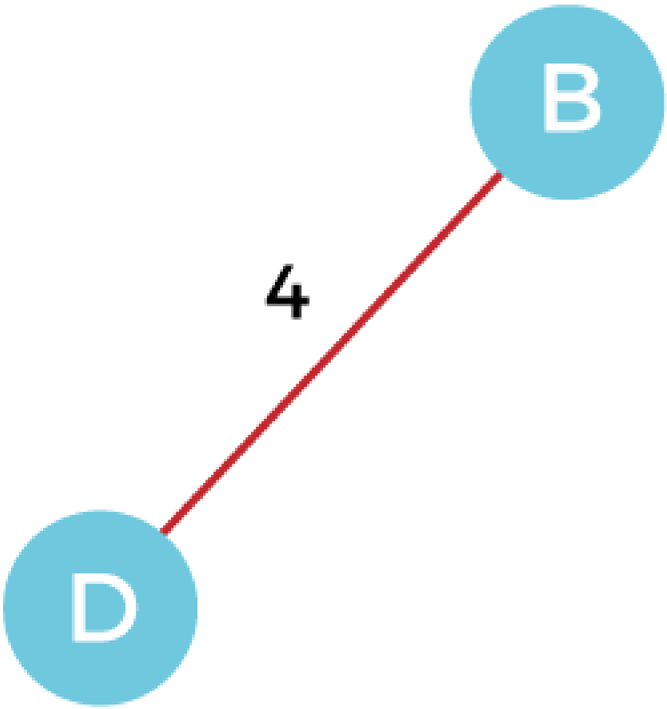
Suppose, a weighted graph is -



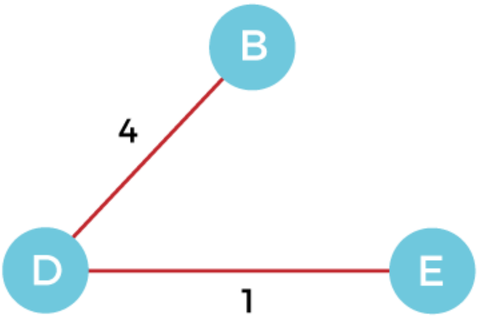
Step 1 - First, we have to choose a vertex from the above graph. Let's choose B.



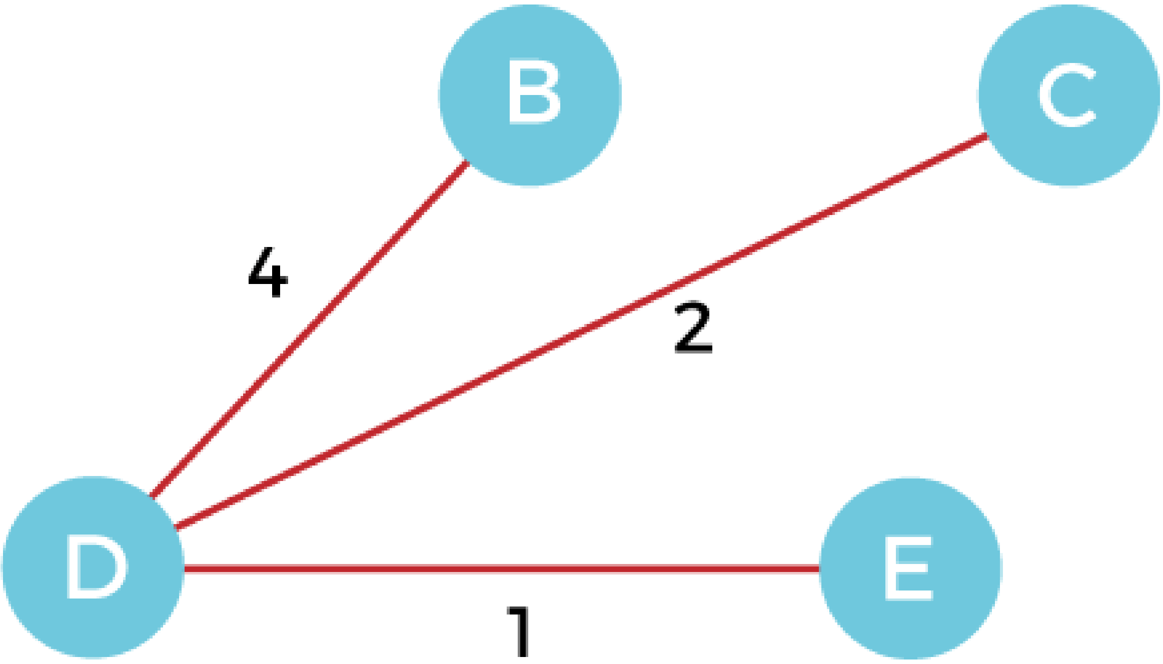
Step 2 - Now, we have to choose and add the shortest edge from vertex B. There are two edges from vertex B that are B to C with weight 10 and edge B to D with weight 4. Among the edges, the edge BD has the minimum weight. So, add it to the MST.



Step 3 - Now, again, choose the edge with the minimum weight among all the other edges. In this case, the edges DE and CD are such edges. Add them to MST and explore the adjacent of C, i.e., E and A. So, select the edge DE and add it to the MST.

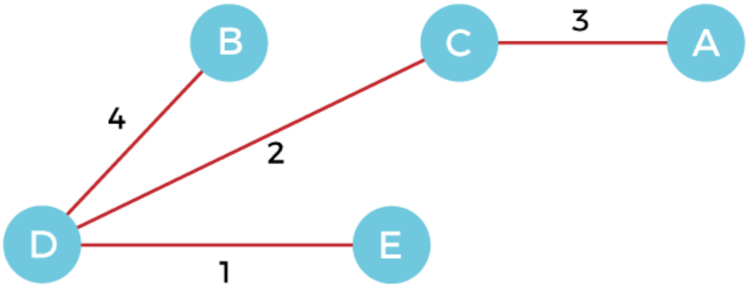


Step 4 - Now, select the edge CD, and add it to the MST.



Step 5 - Now, choose the edge CA. Here, we cannot select the edge CE as it would create a cycle to the graph. So, choose the edge CA and add it to the MST.

So, the graph produced in step 5 is the minimum spanning tree of the given



graph. The cost of the MST is given below –

Krushkals Algorithm:

we will discuss Kruskal's algorithm. Here, we will also see the complexity, working, example, and implementation of the Kruskal's algorithm.

But before moving directly towards the algorithm, we should first understand the basic terms such as spanning tree and minimum spanning tree.

Spanning tree - A spanning tree is the subgraph of an undirected connected graph.

Minimum Spanning tree - Minimum spanning tree can be defined as the spanning tree in which the sum of the weights of the edge is minimum. The weight of the spanning tree is the sum of the weights given to the edges of the spanning tree.

Now, let's start with the main topic.

Kruskal's Algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using which we can traverse every vertex of the graph. It follows the greedy approach that finds an optimum solution at every stage instead of focusing on a global optimum.

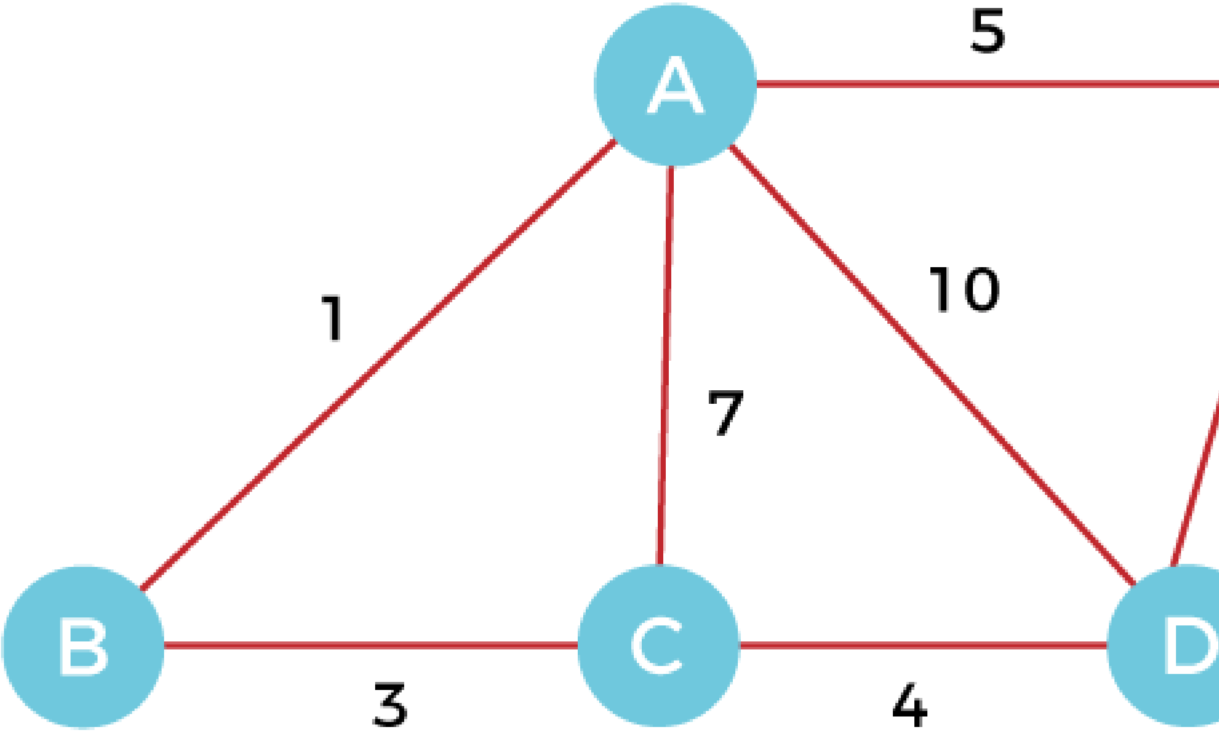
How does Kruskal's algorithm work?

In Kruskal's algorithm, we start from edges with the lowest weight and keep adding the edges until the goal is reached. The steps to implement Kruskal's algorithm are listed as follows -

- First, sort all the edges from low weight to high.
- Now, take the edge with the lowest weight and add it to the spanning tree. If the edge to be added creates a cycle, then reject the edge.
- Continue to add the edges until we reach all vertices, and a minimum spanning tree is created.

Example of Kruskal's algorithm

- Now, let's see the working of Kruskal's algorithm using an example. It will be easier to understand Kruskal's algorithm using an example
- is -



- The weight of the edges of the above graph is given in the below table

The weight of the edges of the above graph is given in the below table -

Edge	AB	AC	AD	AE	BC	CD	DE
Weight	1	7	10	5	3	4	2

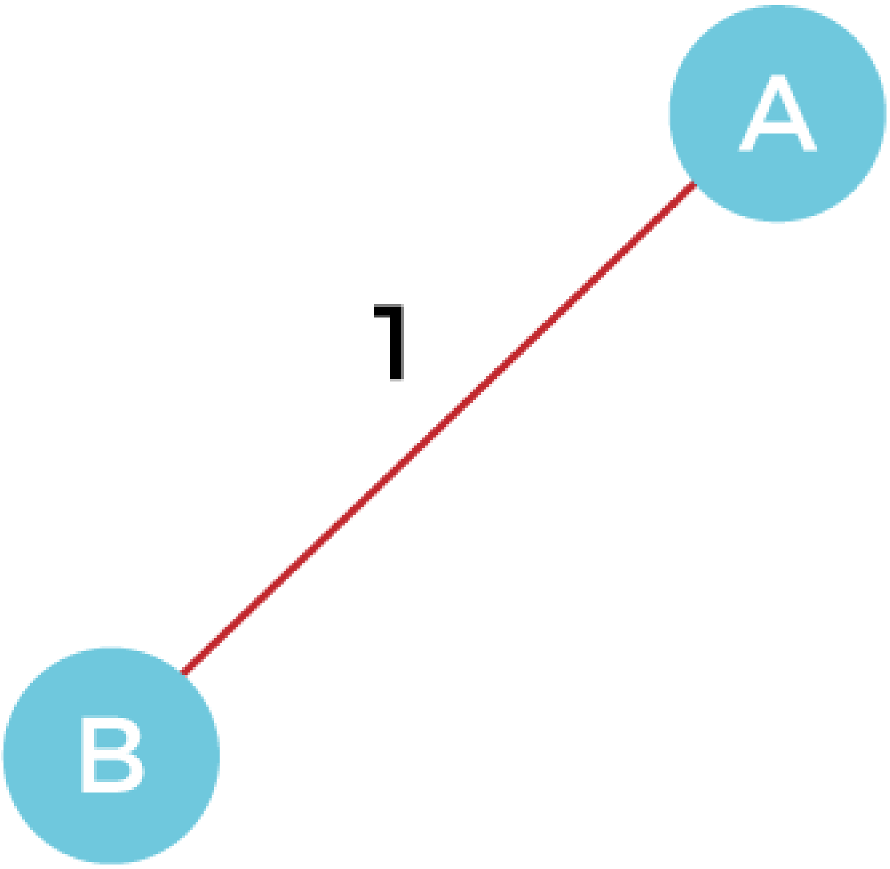
Now, sort the edges given above in the ascending order of their weights.

Edge	AB	DE	BC	CD	AE	AC	AD
Weight	1	2	3	4	5	7	10

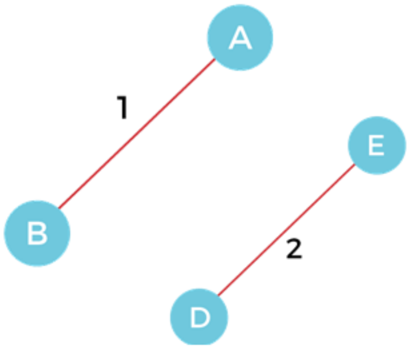
Step 1 - First, add the edge **AB** with weight **1** to the MST.

Step 2 - Add the edge **DE** with weight **2** to the MST as it is not creating the cycle.

Step 1 - First, add the edge **AB** with weight **1** to the MST.

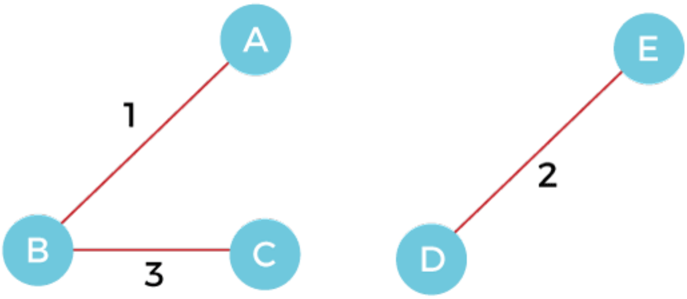


Step 2 - Add the edge **DE** with weight **2** to the MST as it is not creating the cycle.



Step 3 - Add the edge **BC** with weight **3** to the MST, as it is not creating any cycle or loop.

Step 4 - Now, pick the edge **CD** with weight **4** to the MST, as it is not forming the cycle.

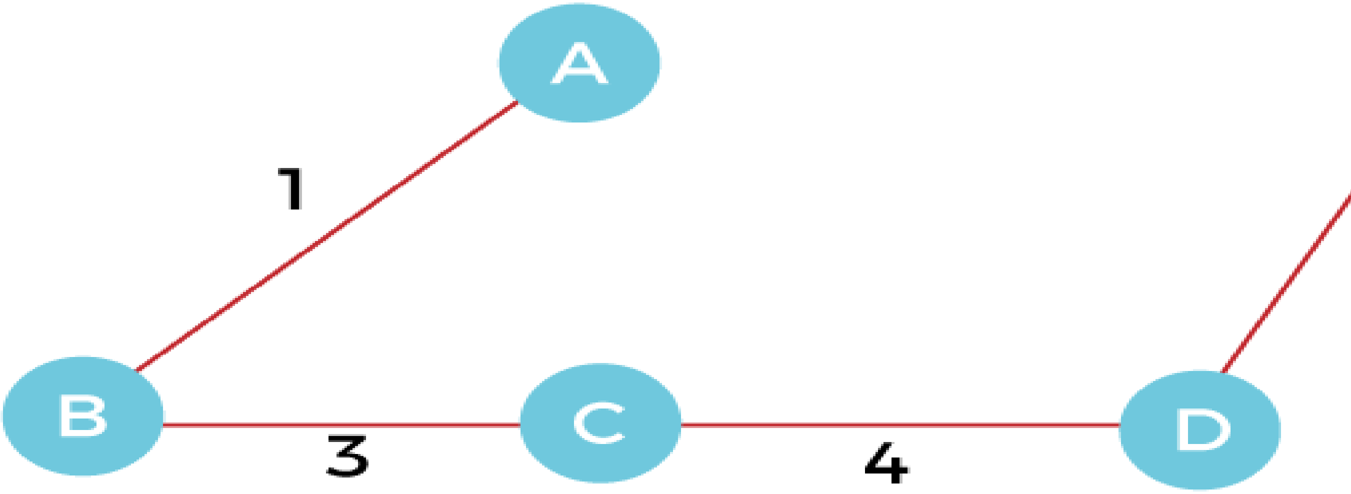


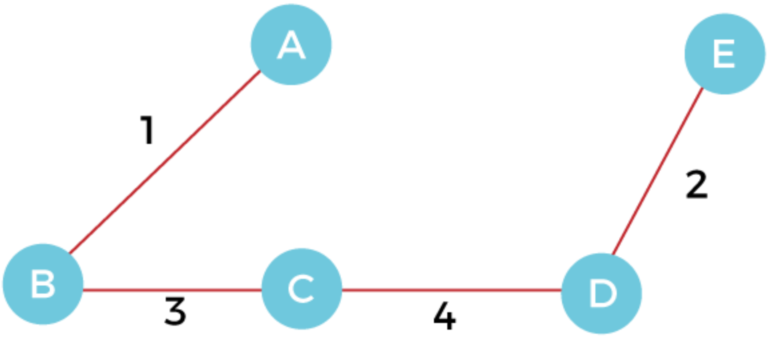
Step 5 - After that, pick the edge **AE** with weight **5.** Including this edge will create the cycle, so discard it.

Step 6 - Pick the edge **AC** with weight **7.**Including this edge will create the cycle, so discard it.

Step 7 - Pick the edge **AD** with weight **10.**Including this edge will also create the cycle, so discard it.

So, the final minimum spanning tree obtained from the given weighted graph by using Kruskal's algorithm is -





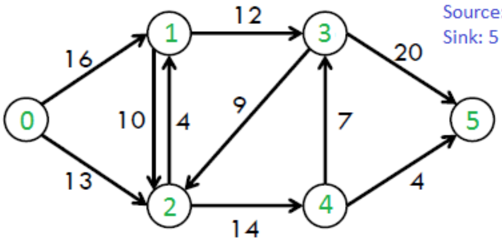
The cost of the MST is = AB + DE + BC + CD = 1 + 2 + 3 + 4 = 10.

Now, the number of edges in the above tree equals the number of vertices minus

Maximal Flow Problem:

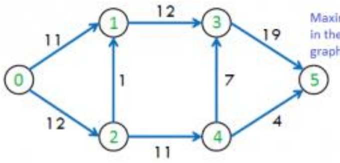
Max Flow Problem Introduction

Maximum flow problems involve finding a feasible flow through a single-source, single-sink flow network that is maximum. Let’s take an image to explain how the above definition wants to say



Source: 0
Sink: 5

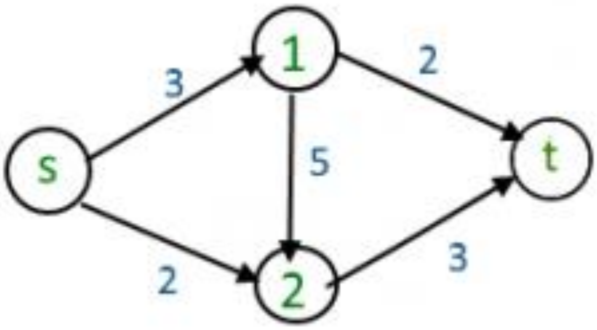
Each edge is labeled with capacity, the maximum amount of stuff that it can carry. The goal is to figure out how much stuff can be pushed from the vertex s(source) to the vertex t(sink). .



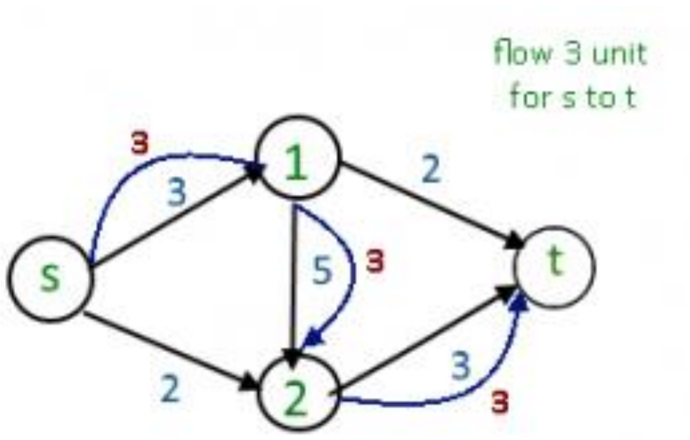
Maximum Flow
in the above
graph is 23

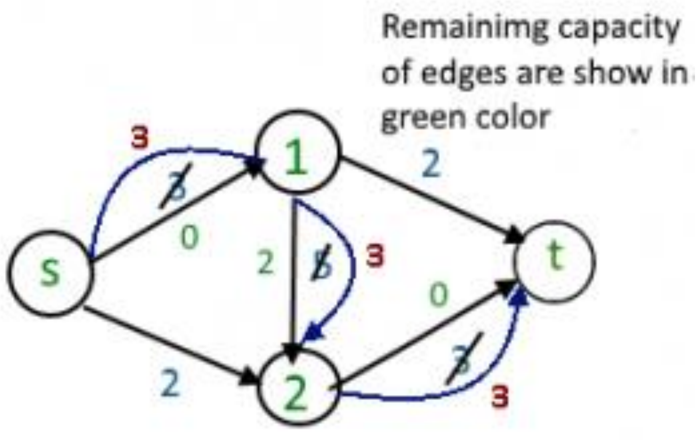
maximum flow possible is : **23** Following are different approaches to solve the problem : **1. Naive Greedy Algorithm Approach (May not produce an optimal or correct result)** Greedy approach to the maximum flow problem is to start with the all-zero flow and greedily produce flows with ever-higher value. The natural way to proceed from one to the next is to send more flow on some path from s to t How Greedy approach work to find the maximum flow :

Note that the path search just needs to determine whether or not there is an s-t path in the subgraph of edges e with f(e) < C(e). This is easily done in linear time using BFS or DFS.

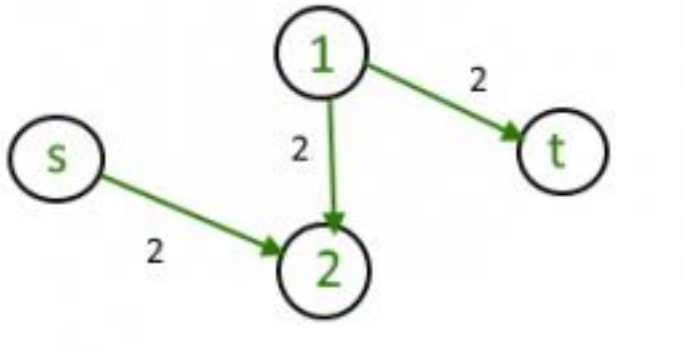


There is a path from source (s) to sink(t) [s -> 1 -> 2 -> t] with maximum flow 3 unit (path show in blue color)





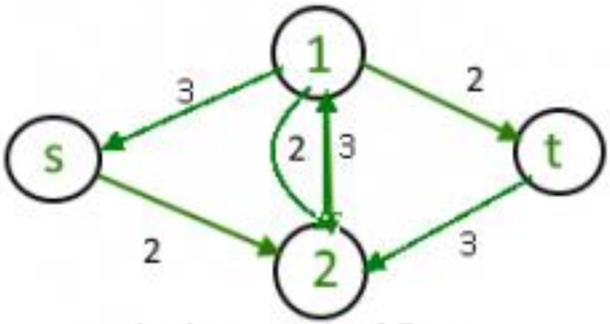
After removing all useless edge from graph it's look like



For above graph there is no path from source to sink so maximum flow : 3 unit
But maximum flow is 5 unit. to over come from this issue we use residual Graph.

2. Residual Graphs

The idea is to extend the naive greedy algorithm by allowing "undo" operations. For example, from the point where this algorithm gets stuck in above image, we'd like to route two more units of flow along the edge (s, 2), then backward along the edge (1, 2), undoing 2 of the 3 units we routed the previous iteration, and finally along the edge (1,t)



residual network of flow

backward edge : (f(e)) and forward edge : (C(e) – f(e)) We need a way of formally specifying the allowable "undo" operations. This motivates the following simple but important definition, of a residual network. The idea is that, given a graph G and a flow f in it, we form a new flow network G_f that has the same vertex set of G and that has two edges for each edge of G. An edge e = (1,2) of G that carries flow f(e) and has capacity C(e) (for above image) spawns a "forward edge" of $G_f$with capacity C(e)-f(e) (the room remaining) and a "backward edge" (2,1) of G_f with capacity f(e) (the amount of previously routed flow that can be undone). source(s)- sink(t) paths with f(e) < C(e) for all edges, as searched for by the naive greedy algorithm, corresponding to the special case of s-t paths of G_f that comprise only forward edges. The idea of residual graph is used The Ford-Fulkerson

Short Answers:

1.What are Network Techniques?
2.What are Steps Involved In Networking Tecnique?
3.Explain about Shortest Path Model
4.Explain about Prims Algorithm
5.Explain about Krushkals

Long Answers:

6.Explain about Dijikstras Algorithm
7.Explain about Maximal Flow Problem

8.Explain about Minimum Spanning Tree
9.Explain about Floyds Algorithm
10.Explain About Systematic Method

UNIT-4

Game Theory:

Game Theory Definition

The game theory is said to be the science of strategies which comes under the probability distribution. It determines logical as well as mathematical actions that should be taken by the players in order to obtain the best possible outcomes for themselves in the games. The games studied in game theory may range from chess to tennis and from child-rearing to takeovers. But there is one thing common that such an array of games is interdependent, i.e. outcome for each player depends upon the strategies of all.

In other words, game theory deals with mathematical models of cooperation and conflicts between rational decision-makers. Game theory can be defined as the study of decision-making in which the players must make strategies affecting the interests of other players.

Zero-Sum Game Theory

There is a special kind of game studied in game theory, called zero-sum games. They are constant-sum games. In such games, the available resources can neither be increased nor decreased. Also, the total benefit in zero-sum games for all combination of strategies, always adds to zero. We can say that in zero-sum games, one wins and exactly one opponent loses. The sum of benefits of all the players for any outcome is equal to zero is called a zero-sum game. Thus, the interest of the two players is opposed.

Several games, game theory are non-zero-sum games, since net result of outcome is less than or greater than zero. So, when one player's gain does not correspond to other's loss, it is called a non-zero sum game.

Game Theory Applications

The game theory is widely applied to study human as well as animal behaviours. It is utilized in economics to understand the economic behaviours, such as behaviours of consumers, markets and firms. Game theory has been commonly used in social sciences as well. It is applied in the study of sociological, political and psychological behaviours. The use of analysis based on game theory is seen in biology too. In addition to behavioural prediction, game theory utilized in the development of theories of normative or ethical behaviour.

Game Theory Example

The best example of game theory is a classical hypothesis called “Prisoners Dilemma”. According to this situation, two people are supposed to be arrested for stealing a car. They have to serve 2-year imprisonment for this. But, the police also suspects that these two people have also committed a bank robbery. The police placed each prisoner in a separate cell. Both of them are told that they are suspects of being bank robbers. They are inquired separately and are not able to communicate with each other.

The prisoners are given two situations:

- If they both confess to being bank robbers, then each will serve 3-year imprisonment for both car theft and robbery.
- If only one of them confesses to being a bank robber and the other does not, then the person who confesses will serve 1-year and others will serve 10-year imprisonment.

According to game theory, the prisoners will either confess or deny the bank robbery. So, there are four possible outcomes :

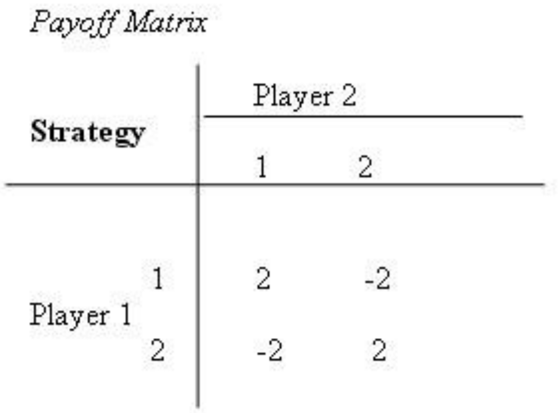
Two-Person Zero-Sum Games: Basic Concepts

Game theory provides a mathematical framework for analyzing the decision-making processes and strategies of adversaries (or *players*) in different types of competitive situations. The simplest type of competitive situations are **two-person, zero-sum games**. These games involve only two players; they are called *zero-sum* games because one player wins whatever the other player loses.

Example: Odds and Evens

Consider the simple game called **odds and evens**. Suppose that player 1 takes evens and player 2 takes odds. Then, each player simultaneously shows either one finger or two fingers. If the number of fingers matches, then the result is *even*, and player 1 wins the bet ($2). If the number of fingers does not match, then the result is *odd*, and player 2 wins the bet ($2). Each player has two possible strategies: show one finger or show two fingers. The *payoff matrix* shown below represents the payoff to player 1.

Consider the simple game called **odds and evens**. Suppose that player 1 takes evens and player 2 takes odds. Then, each player simultaneously shows either one finger or two fingers. If the number of fingers matches, then the result is *even*, and player 1 wins the bet ($2). If the number of fingers does not match, then the result is *odd*, and player 2 wins the bet ($2). Each player has two possible strategies: show one finger or show two fingers. The *payoff matrix* shown below represents the payoff to player 1.



Payoff Matrix

Strategy		Player 2	
		1	2
Player 1	1	2	-2
	2	-2	2

Basic Concepts of Two-Person Zero-Sum Games

This game of odds and evens illustrates important concepts of simple games.

- A two-person game is characterized by the strategies of each player and the payoff matrix.
- The payoff matrix shows the gain (positive or negative) for player 1 that would result from each combination of strategies for the two players. *Note that the matrix for player 2 is the negative of the matrix for player 1 in a zero-sum game.*
- The entries in the payoff matrix can be in any units as long as they represent the *utility (or value)* to the player.
- There are two key assumptions about the behavior of the players. The first is that both players are *rational*. The second is that both players are *greedy* meaning that they choose their strategies in their own interest (to promote their own wealth).

Maximin-Minimax Principle:

The maximin-minimax principle is used for the selection of optimal strategies by two players. This method is also known as a **calculus method**.

The Minimax Theorem

For every finite two-person zero-sum game,

- there is a number Vv, called the value of the game

- there is a mixed strategy for Player AA such that AA's **average gain is at least Vv no matter what Player BB does, and**
- **there is a mixed strategy for Player BB such that BB's average loss is at most Vv no matter what Player AA does.**

Player AA wishes to maximize his gain while player BB wishes to minimize his loss. Since player AA would like to maximize his minimum gain, we obtain
the maximin value for player AA and the corresponding strategy is called the maximin strategy. Player AA's corresponding gain is called the *maximin value* or *lower value* (V—v_) of the game.
the same time, player BB wishes to minimize his maximum loss, we obtain
the **minimax** value for player BB and the corresponding strategy is called the **minimax strategy**. Player BB's corresponding loss is called the *minimax value* or *upper value* (V¯¯¯v¯) of the game.
Usually the minimax value is greater than or equal to the maximin value. If the equality holds, i.e., $\max_i \min_j a_{ij} = \min_j \max_i a_{ij} = v$maximinjaij=minjmaxiaij=vthen the corresponding pure strategies are called **optimal strategies** and the game is said to have a saddle point.
When these two values (miximin and minimax) are equal, the corresponding strategies are called **optimal strategies**.

When these two values (miximin and minimax) are equal, the corresponding strategies are called **optimal strategies**.
The player A's selection is called the *maximin strategy*. Player B's selection is called the *minimax value* or *upper value* ($\bar{v}$) of the game.
Payoff Matrix for Player A is as follows:

Step by step procedure of Maximin-minimax principle

Step 1

Select the minimum element of each row of the payoff matrix,

i.e., $\min_j a_{ij}, i=1,2,\cdots,m$.i.e., minjaij,i=1,2,⋯,m.

Step 2

Select the maximum element of each column of the payoff matrix,

i.e., $\max_i a_{ij}, j=1,2,\cdots,n$.i.e., maxiaij,j=1,2,⋯,n.

Step 3

Obtain the maximum value of each row minimum,

i.e., $\max_i \min_j a_{ij} = \underline{v}$.i.e., maximinjaij=v_.

Step 4

Obtain the minimum value of each column maximum,

i.e., $\min_j \max_i a_{ij} = \overline{v}$.i.e., minjmaxiaij=v¯.

Step 5

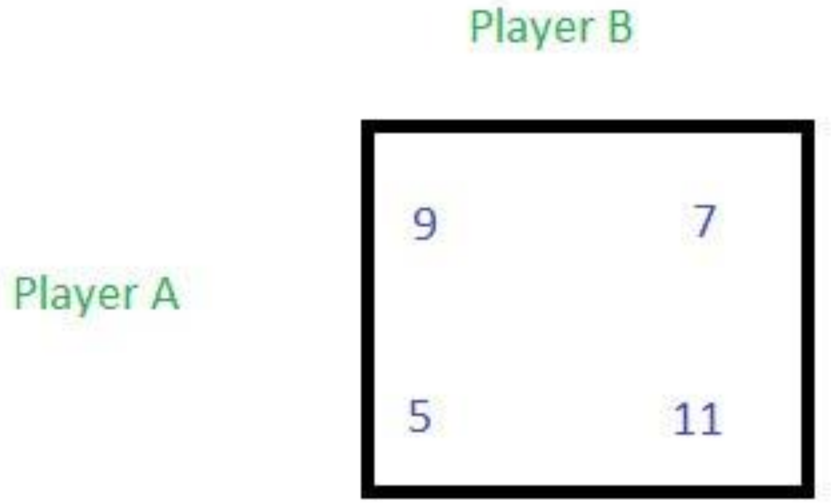
If $\min_j \max_i a_{ij} = \max_i \min_j a_{ij} = v$minjmaxiaij=maximinjaij=v,
i.e., $\max(\min)=\min(\max)$max(min)=min(max) then the position of that element is a **saddle point** of the payoff matrix. And the corresponding strategy is called **optimal strategy**

Games Without Saddle Points:

If a game has no saddle point then the game is said to have mixed strategy.

Game with Mixed Strategy)

Consider the following payoff matrix with respect to player A and solve it optimally.

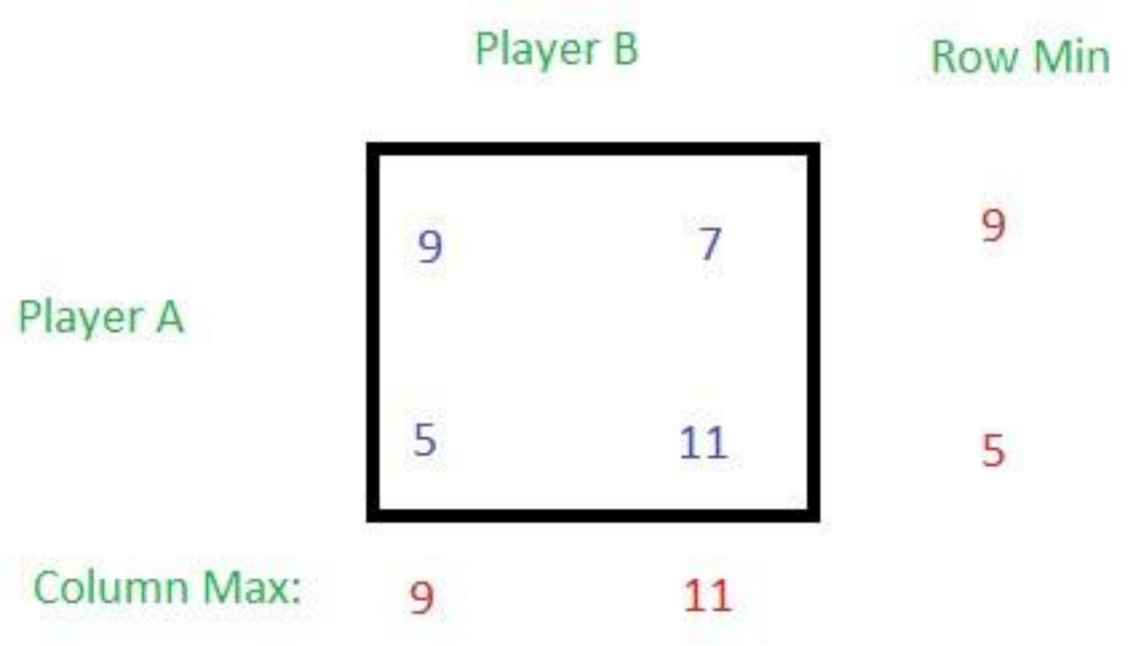


	Player B	
Player A	9	7
	5	11

Solution:

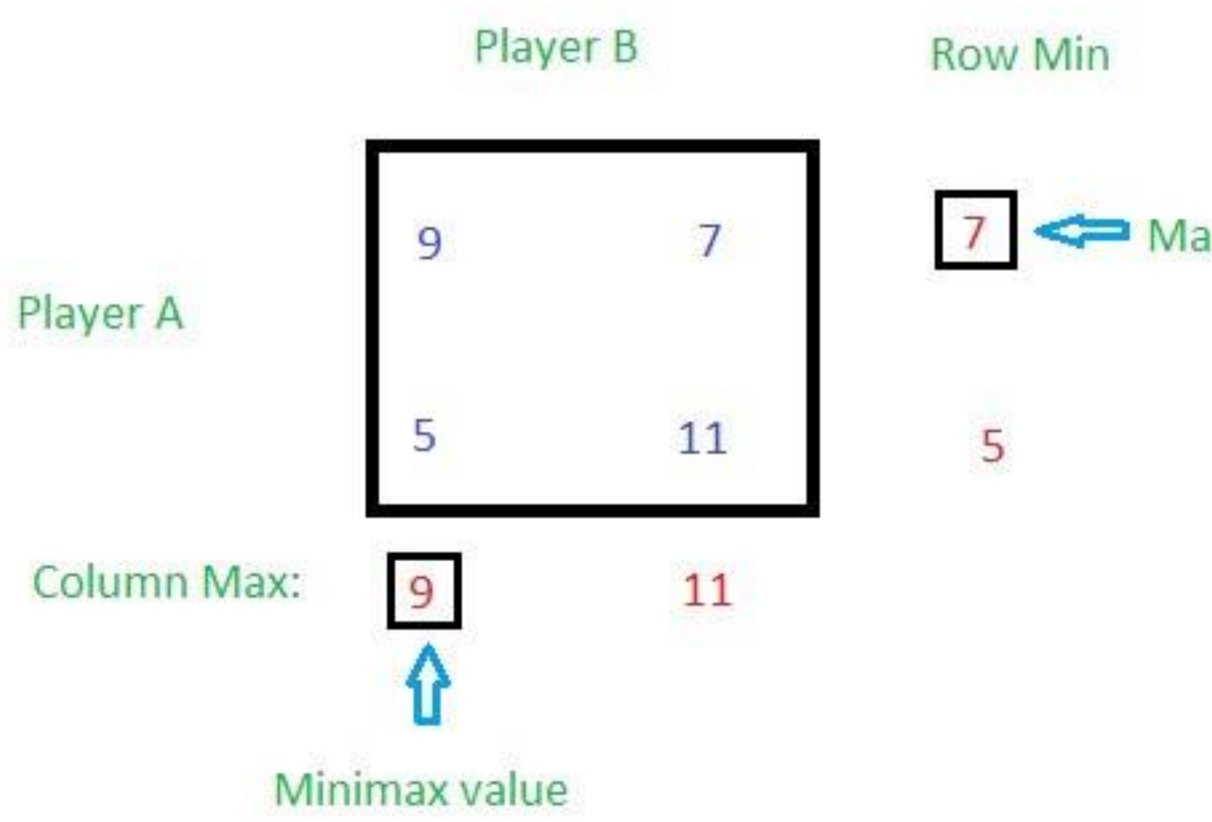
If a game has no saddle point then the game is said to have mixed strategy.

- **Step 1:** Find out the row minimum and column maximum.



	Player B		Row Min
Player A	9	7	9
	5	11	5
Column Max:	9	11	

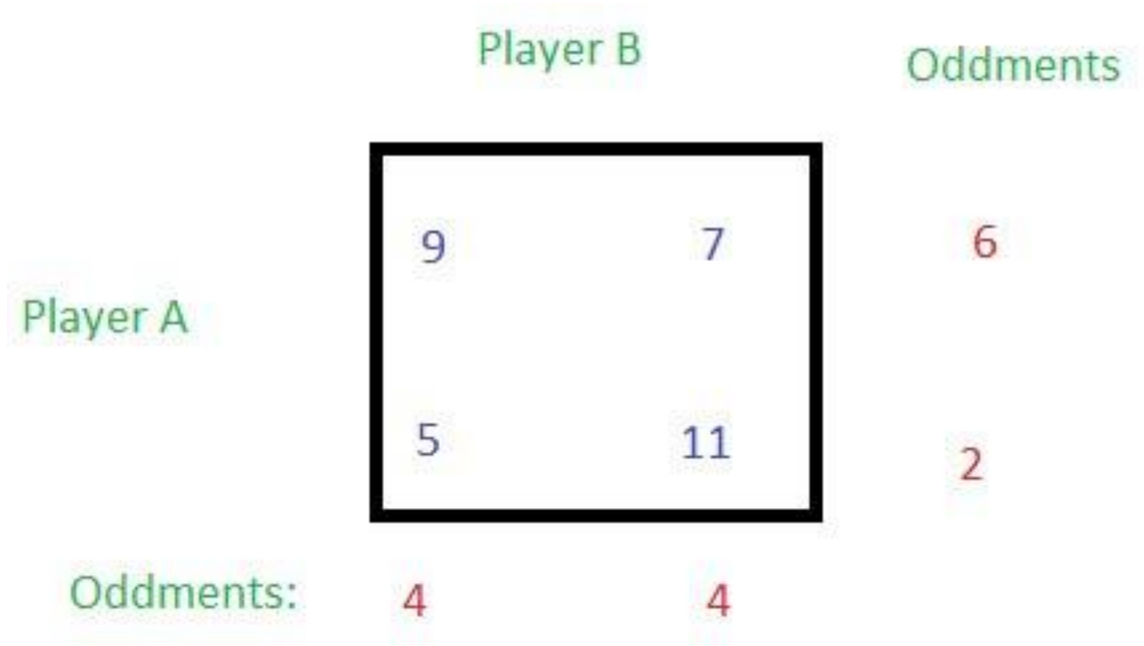
- **Step 2:** Find out the minimax and maximin values.



	Player B		Row Min
Player A	9	7	$\boxed{7}$ ⇦ Ma
	5	11	5
Column Max:	$\boxed{9}$	11	

Since minimax and maximin value of this game are not equal, this game has no saddle point.

- **Step 3:** Now take the 2×2 matrix and find out the oddments for both row and column.

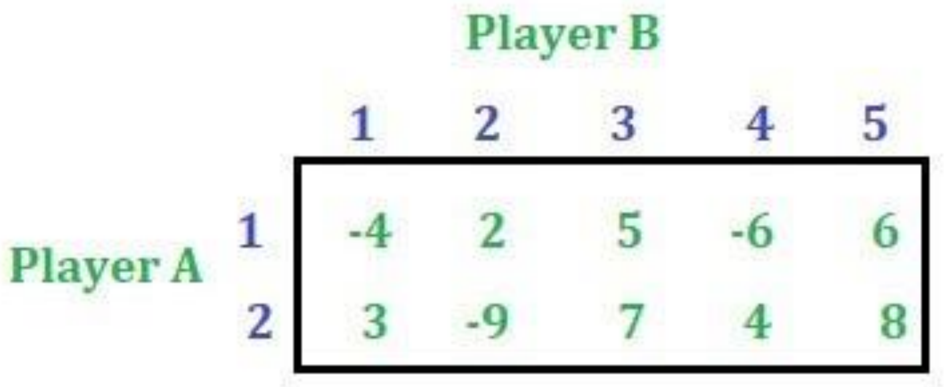


	Player B		Oddments
Player A	9	7	6
	5	11	2
Oddments:	4	4	

Graphic Solution Of 2xn and mx2:

Game Theory (Normal-form Game) | Set 6 (Graphical Method [2 X N] Game)

The payoff matrix of a **2 * N** game consists of **2 rows** and **N columns**. This article will discuss how to solve a **2 * N** game by graphical method.
Consider the below 2 * 5 game:

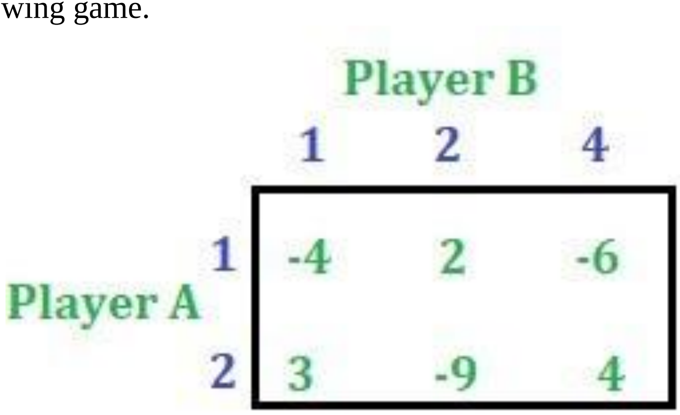


		Player B				
		1	2	3	4	5
Player A	1	-4	2	5	-6	6
	2	3	-9	7	4	8

Solution: First check the saddle point of the game. This game has no saddle point.

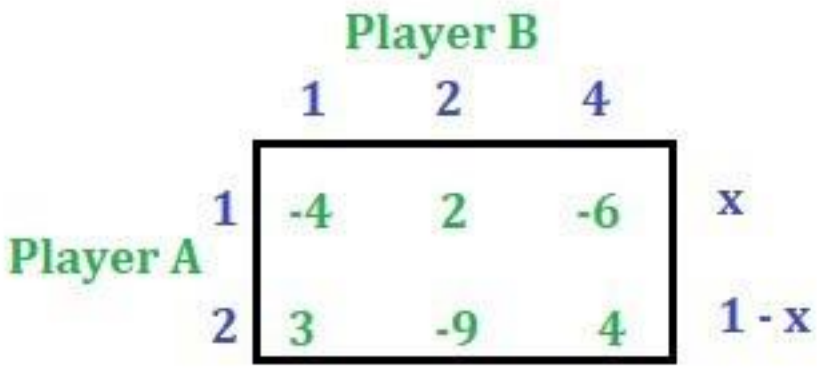
Step 1: Reduce the size of the payoff matrix by applying dominance property, if it exists. This step is not compulsory. The size is being reduced to just simplify the problem. The game can be solved without reducing the size also.
After reducing the above game with the help of dominance property we get the

following game.



		Player B		
		1	2	4
Player A	1	-4	2	-6
	2	3	-9	4

Step 2: Let **x** be the probability of selection of alternative 1 by player A and **(1 – x)** be the probability of selection of alternative 2 by player A.



		Player B			
		1	2	4	
Player A	1	-4	2	-6	x
	2	3	-9	4	1 - x

Derive the expected gain function of player A with respect to each of the alternatives of player B. To do this just multiply the column values of B's alternative with its corresponding probability of selection of alternatives by player A. For example, the first alternative of player B is column number 1, so multiply **-4** with **x** and **3** with **(1 – x)** and add them then the expression obtained is the A's expected payoff function. Similarly, the second alternative of player B is column number 2, so multiply **2** with **x** and **-9** with **(1 – x)** and add them. Similarly, the third alternative of player B is the column number 4, so multiply **-6** with **x** and **4** with **(1 – x)** and add them. Please refer the shown table.

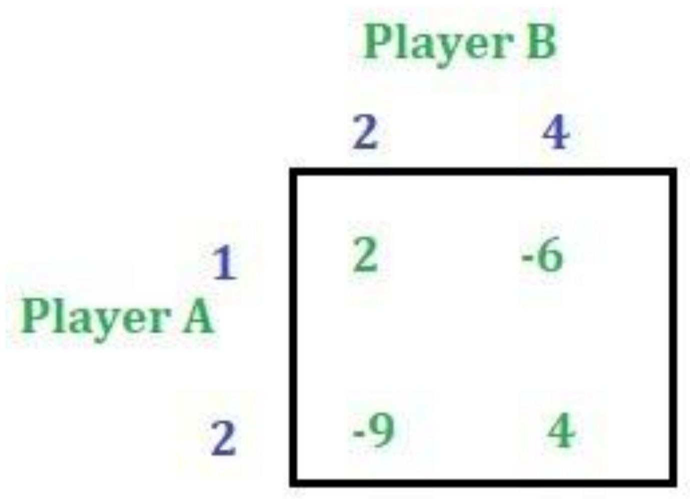
Step 3: Find the value of the gain when **x = 0** and **x = 1**. See the table below:

Step 4: Now plot the gain function on a graph by assuming suitable scale. [Keep x on the x-axis and the gain on y-axis]
If **B** selects the first alternative i.e. first strategy, when **x = 0** A's expected gain is **3**and when **x = 1** A's expected gain is **-4**.
If **B** selects the second alternative i.e. second strategy, when **x = 0** A's expected gain is **-9** and when **x = 1** A's expected gain is **2**.
If **B** selects the third alternative i.e. fourth strategy, when **x = 0** A's expected gain is **4**and when **x = 1** A's expected gain is **-6**.

Using the above information plot the graph.

Step 5: Find the highest intersection point in the lower boundary of the graph –> **Maximum point** as A is the **Maximin player**.
The lower boundary is ABC. And the highest point among A, B and C is B. This intersection point B is called the **Maximin point**.

Step 6: If the number of lines passing through the maximin point is only two, form 2 * 2 payoff matrix then solve the game as per this article.
If not, identify any two lines with opposite slopes passing through that point. Form a 2 * 2 payoff matrix and then solve. This point will be discussed in the next article.
Since we have two lines passing through this point, the payoff matrix using B1 and B2 alternatives are:

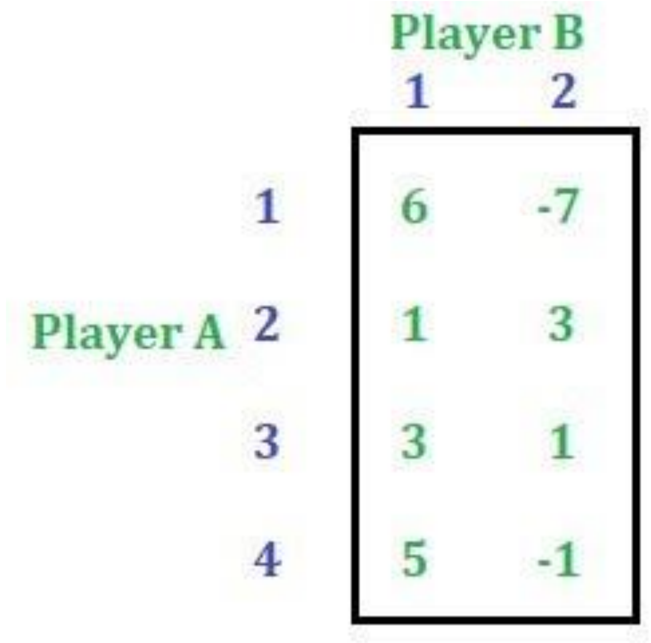


		Player B	
		2	4
Player A	1	2	-6
	2	-9	4

Game Theory (Normal-form Game) | Set 7 (Graphical Method [M X 2] Game)

The payoff matrix of an **M * 2** game consists of **M** rows and two columns. This article will discuss how to solve an **M * 2**game by graphical method. Also, this article will discuss if more than two lines intersect the same point in the graph then how can a **2 * 2** payoff matrix be formed.

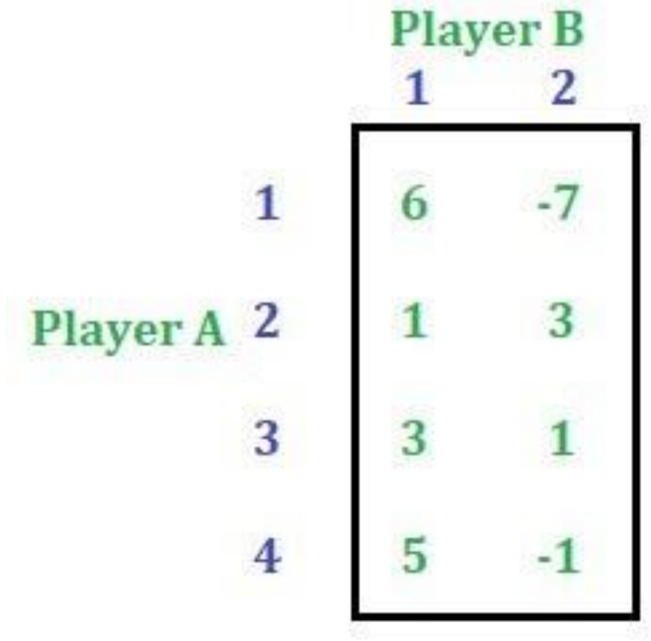
Consider the below problem:



$$\begin{array}{cc|cc|} & & \text{Player B} & \\ & & 1 & 2 \\ \hline & 1 & 6 & -7 \\ \text{Player A} & 2 & 1 & 3 \\ & 3 & 3 & 1 \\ & 4 & 5 & -1 \\ \hline \end{array}$$

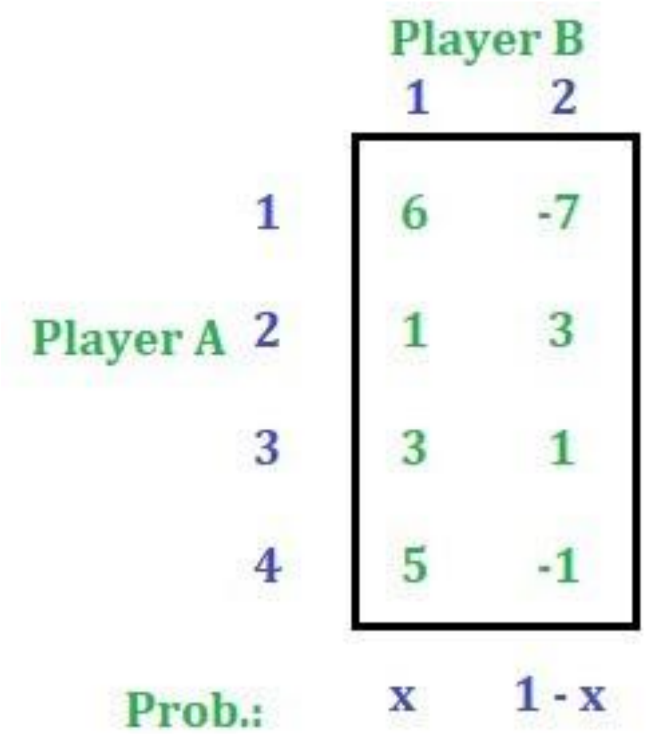
Solution: First check whether the problem has got saddle point or not. This game has no saddle point.

Step 1: Reduce the size of the payoff matrix of player A by applying the dominance property, if it exists. The size is being reduced just to simplify the problem. The game can be solved without reducing the size also.
In this problem the dominance property is not applicable. We cannot simplify this matrix more than this. So, we are remain with the following game.



$$\begin{array}{cc|cc|} & & \text{Player B} & \\ & & 1 & 2 \\ \hline & 1 & 6 & -7 \\ \text{Player A} & 2 & 1 & 3 \\ & 3 & 3 & 1 \\ & 4 & 5 & -1 \\ \hline \end{array}$$

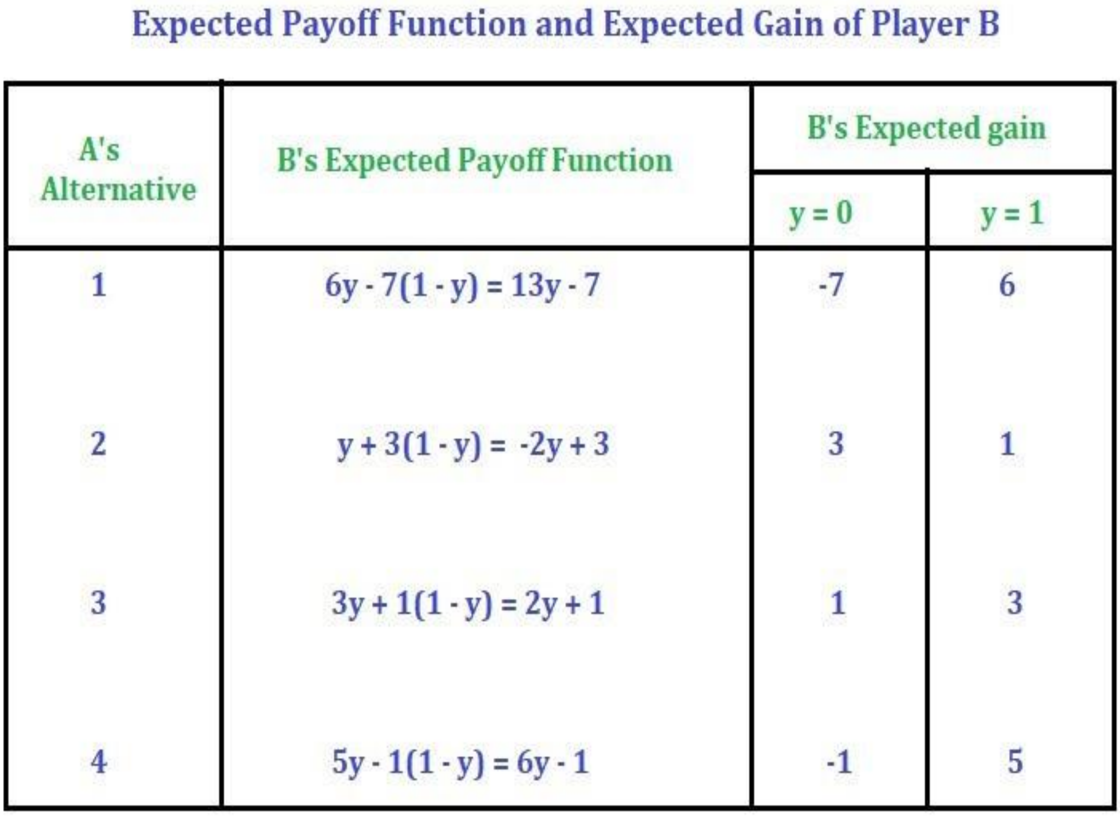
Step 2: Let **y** be the probability of selection of alternative 1 by player B and **(1 – y)** be the probability of selection of alternative 2 by player B.



		Player B	
		1	2
	1	6	-7
Player A	2	1	3
	3	3	1
	4	5	-1
Prob.:		x	1 - x

Derive the expected gain function of player B with respect to each of the alternatives of player A. See the table below.

Along with find the value of the gain when **y = 0** and **y = 1**. See the table below

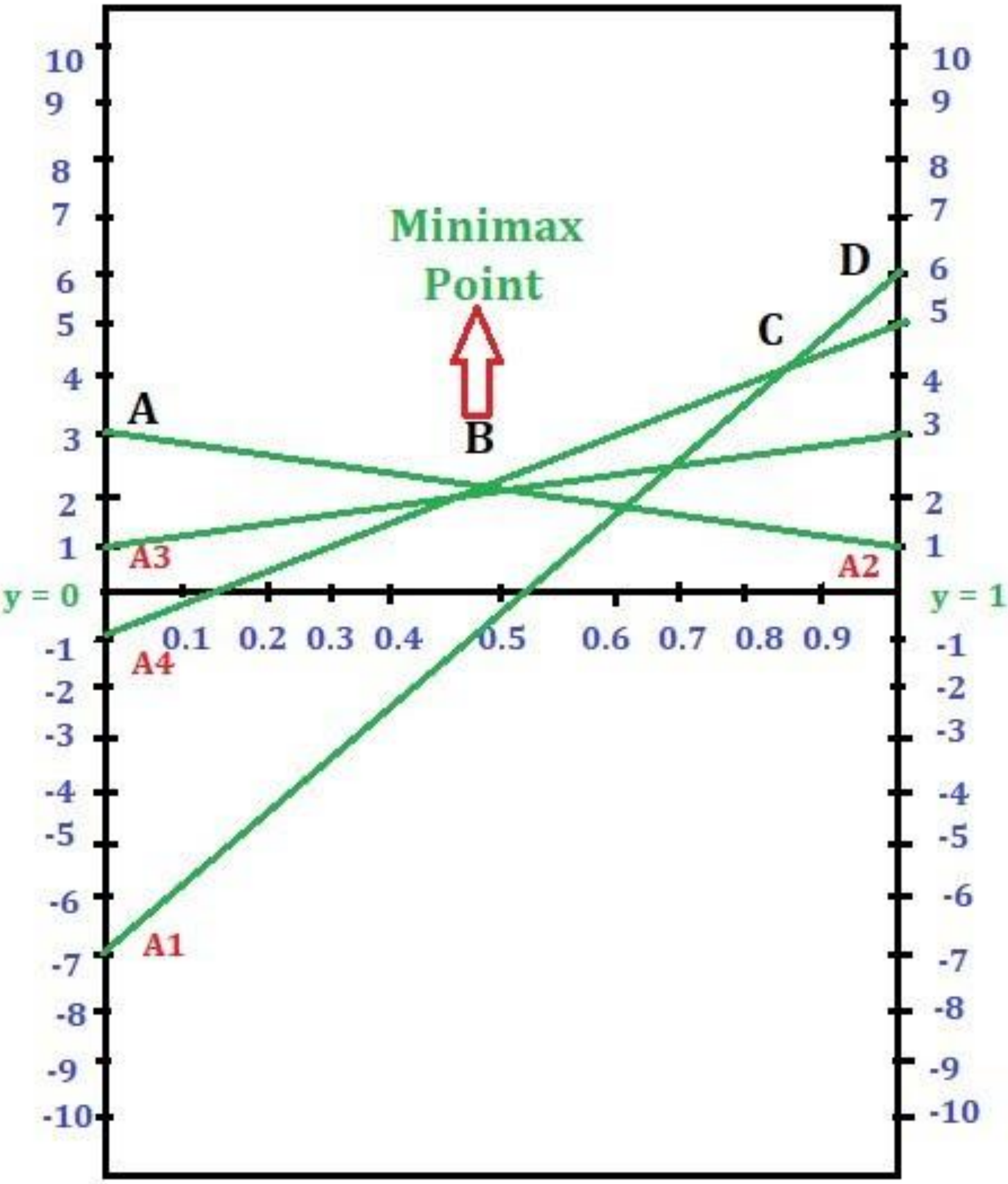


Expected Payoff Function and Expected Gain of Player B

A's Alternative	B's Expected Payoff Function	B's Expected gain	
		y = 0	y = 1
1	6y - 7(1 - y) = 13y - 7	-7	6
2	y + 3(1 - y) = -2y + 3	3	1
3	3y + 1(1 - y) = 2y + 1	1	3
4	5y - 1(1 - y) = 6y - 1	-1	5

Step 3: Plot the gain function on a graph by assuming a suitable scale. Keep **y** on X-axis and the gain on Y-axis.

Step 3: Find the lowest intersection point in the upper boundary of the graph –> Minimax point.
ABCD is the upper boundary in the given graph. There are four intersection points where B is the lowest intersection. So this point is called Minimax point.



Dominance Property:

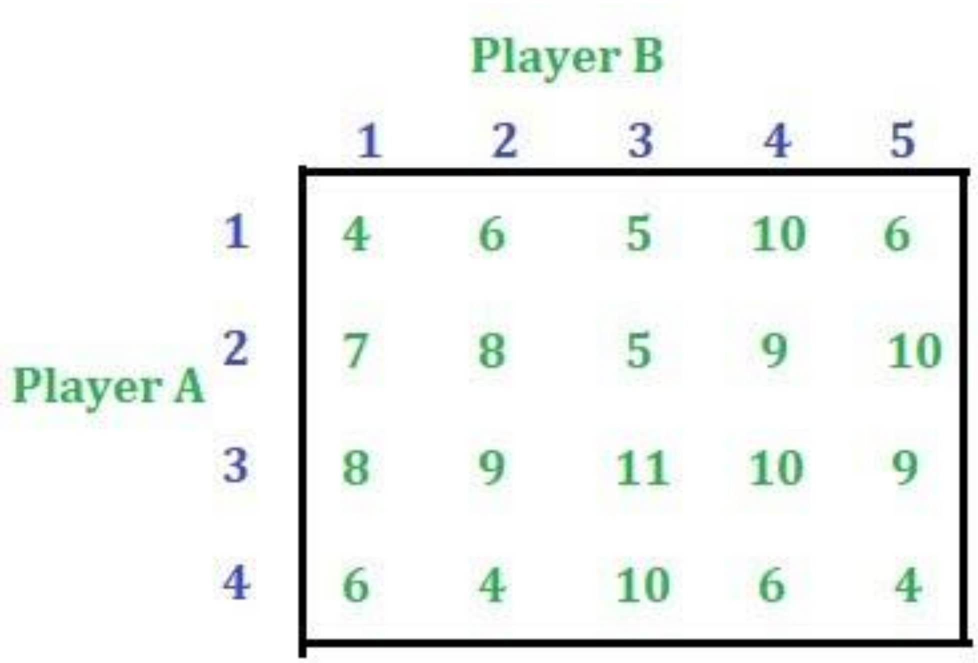
Game Theory (Normal-form Game) | Set 4 (Dominance Property-Pure Strategy)

In some of the games, it is possible to reduce the size of the payoff matrix by eliminating rows (or columns) which are dominated by other rows (or columns) respectively.

Dominance property for rows: X ≤ Y i.e. if all the elements of a particular row **X** are less than or equal to the corresponding elements of another row **Y** then delete row **X** (row **X** is dominated by row **Y**). The elements of a particular row **X** can also be compared with an average of two or more other rows and if the elements of the row **X**are less than or equal to the corresponding elements after taking the average then delete row **X**.

Dominance property for columns: X ≥ Y i.e. if all the elements of a particular column **X**are greater than or equal to the corresponding elements of another column **Y** then delete column **X** (column **X** is dominated by column **Y**). The elements of the column **X** can also be compared with an average of two or more columns and if the elements of column **X** are greater than the corresponding elements after taking the average then delete the column **X**.

Consider the below game:

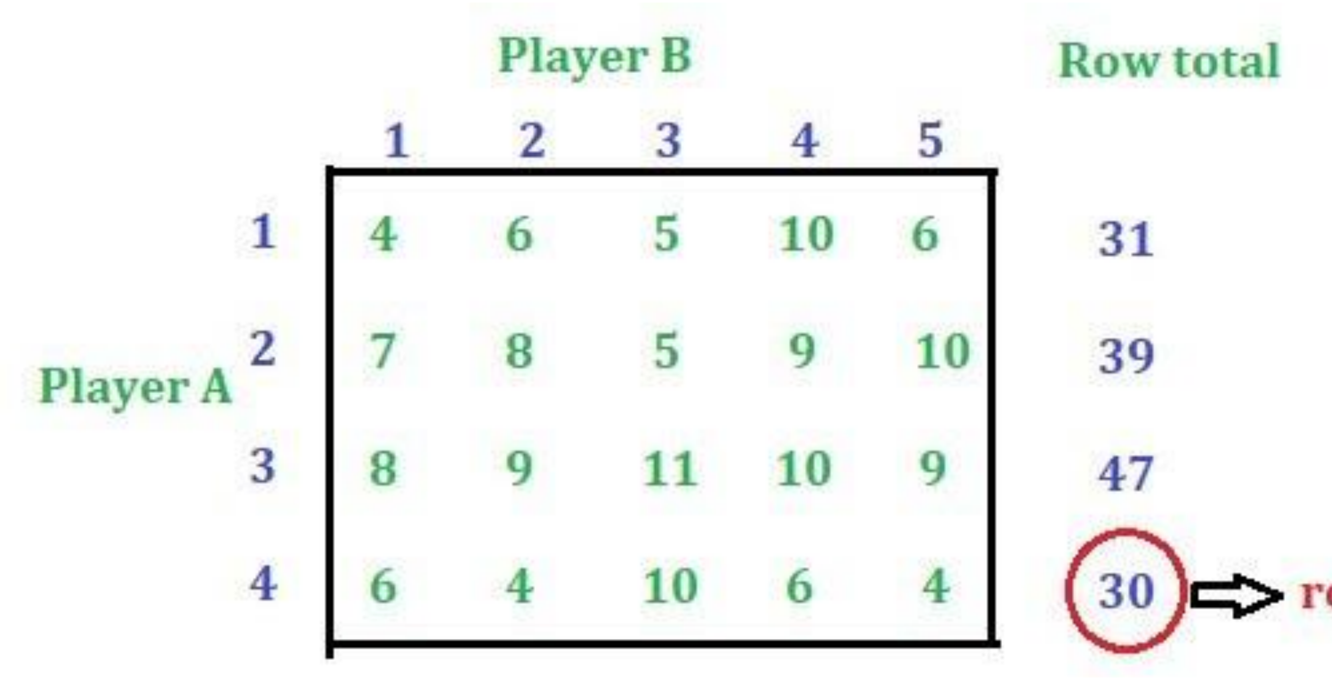


		Player B				
		1	2	3	4	5
	1	4	6	5	10	6
Player A	2	7	8	5	9	10
	3	8	9	11	10	9
	4	6	4	10	6	4

Solution:
Pure Strategy: The solution of the above game by Pure Strategy will be,
– the value of the game (V) = 8
– A[P1, P2, P3, P4] = A[0, 0, 1, 0]
– B[Q1, Q2, Q3, Q4, Q5] = B[1, 0, 0, 0, 0]
Where,

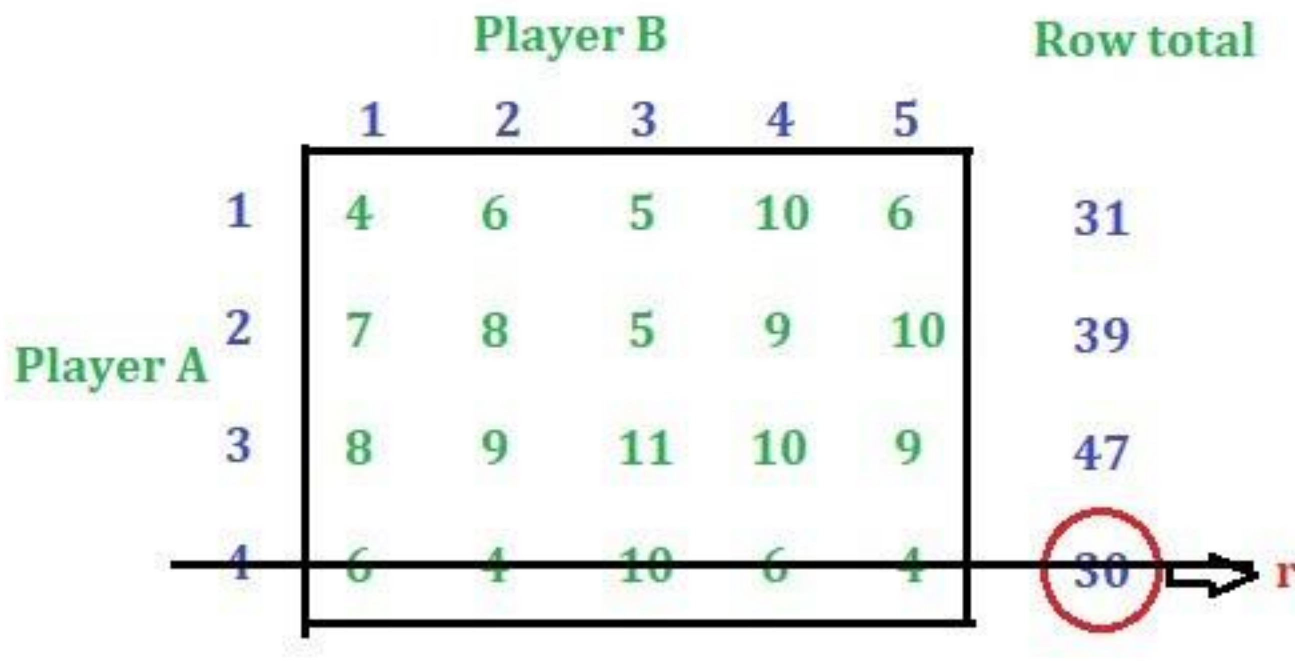
P1, P2, P3 and P4 are probabilities of strategy 1, 2, 3 and 4 respectively for player A.
Q1, Q2, Q3, Q4 and Q5 are probabilities of strategy 1, 2, 3, 4 and 5 respectively for player B
For both the players, the total probability is 1.

Dominance Property: The same results will be obtained by using dominance property as well.
– Take the total of each row and select the least among them.



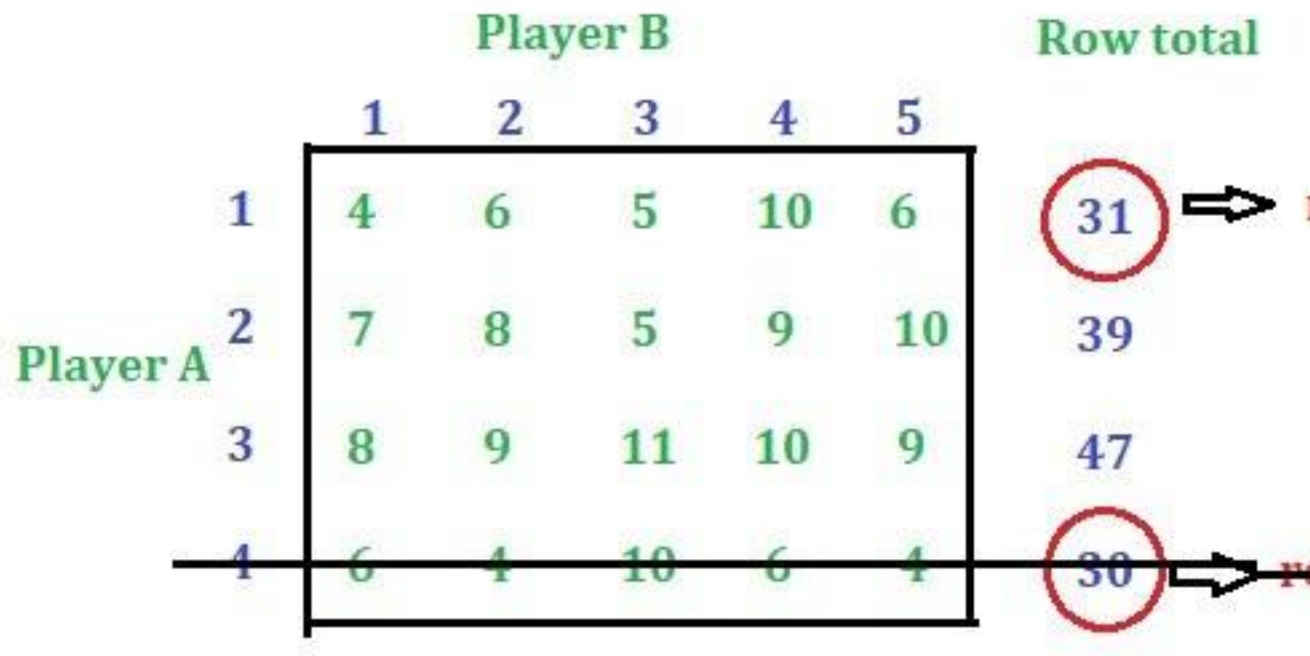
		Player B					Row total
		1	2	3	4	5	
Player A	1	4	6	5	10	6	31
	2	7	8	5	9	10	39
	3	8	9	11	10	9	47
	4	6	4	10	6	4	30

– Using dominance property for row, elements of row 4 are smaller than the elements of row 3 i.e. row 4 is dominated by row 3. Delete row 4.



		Player B					Row total
		1	2	3	4	5	
Player A	1	4	6	5	10	6	31
	2	7	8	5	9	10	39
	3	8	9	11	10	9	47
	~~4~~	~~6~~	~~4~~	~~10~~	~~6~~	~~4~~	~~30~~

– Select the least value among the remaining ‘Row total’ and apply the dominance property.



		Player B					Row total
		1	2	3	4	5	
Player A	1	4	6	5	10	6	31
	2	7	8	5	9	10	39
	3	8	9	11	10	9	47
	~~4~~	~~6~~	~~4~~	~~10~~	~~6~~	~~4~~	~~30~~

Short Answers:

1.What is Game Theory?

2.What is Minimax Principle?

3.What is Maxmini Principle?

4.What is Graphic Solution?

5.What is NxN Rectangular Games?

Long Answers:

6.Explain about Maxmini and Minimax Principle

7.Explain about Dominance Property

8.Explain About Arithmetic Model

9.Explain about Mixed Strategies

10.Explain about Saddle Points

Unit-5

Queuing Theory:

Queuing theory

Queuing theory deals with problems which involve queuing (or waiting). Typical examples might be:

- banks/supermarkets - waiting for service
- computers - waiting for a response
- failure situations - waiting for a failure to occur e.g. in a piece of machinery
- public transport - waiting for a train or a bus

As we know queues are a common every-day experience. Queues form because resources are limited. In fact it makes *economic sense* to have queues. For example how many supermarket tills you would need to avoid queuing? How many buses or trains would be needed if queues were to be avoided/eliminated?

In designing queueing systems we need to aim for a balance between service to customers (short queues implying many servers) and economic considerations (not too many servers).

In essence all queuing systems can be broken down into individual sub-systems consisting of *entities* queuing for some *activity* (as shown below).

Typically we can talk of this individual sub-system as dealing with ***customers*** queuing for ***service***. To analyse this sub-system we need information relating to:

- **arrival process:**
  - how customers arrive e.g. singly or in groups (batch or bulk arrivals)

  - how the arrivals are distributed in time (e.g. what is the probability distribution of time between successive arrivals (the ***interarrival time distribution***))
    - whether there is a finite population of customers or (effectively) an infinite number
- **service mechanism:**
  - a description of the resources needed for service to begin
  - how long the service will take (the ***service time distribution***)
  - the number of servers available
  - whether the servers are in series (each server has a separate queue) or in parallel (one queue for all servers)
  - whether preemption is allowed (a server can stop processing a customer to deal with another "emergency" customer)

    Assuming that the service times for customers are independent and do not depend upon the arrival process is common. Another common assumption about service times is that they are exponentially distributed.

- **queue characteristics:**
  - how, from the set of customers waiting for service, do we choose the one to be served next (e.g. FIFO (first-in first-out) - also known as FCFS (first-come first served); LIFO (last-in first-out); randomly) (this is often called the *queue discipline*)
  - do we have:
    - balking (customers deciding not to join the queue if it is too long)
    - reneging (customers leave the queue if they have waited too long for service)
    - jockeying (customers switch between queues if they think they will get served faster by so doing)
    - a queue of finite capacity or (effectively) of infinite capacity

    Changing the queue discipline (the rule by which we select the next customer to be served) can often reduce congestion. Often the queue discipline "choose the customer with the lowest service time" results in the smallest value for the time (on average) a customer spends queuing.

Elements of Queuing System:

Characteristics of Queuing Systems

The key elements of queuing systems are customers and servers.

- The term *customer* can refer to people, machines, trucks, airplanes etc etc. Anything that arrive at a facility and requires service.

- The term *server* can refer to receptionist, repair personnel, runways in airport, washing machines etc etc. Any resource that provides the requested service.

Below we describe the elements of queuing systems in more details.

The Calling Population

The population of potential customers, referred to as the *calling population*, will be assumed to be infinite, even though the number of potential customers is actually finite. When the population of potential customers is large this assumption is innocuous and actually can simplify the model. This is especially true when we believe that at any given time the number of customers being served or waiting for service is a small proportion of the whole population.

The assumption of an infinite population is such that the rate of arrival of customers is not affected by the number of customers that have already joined the queuing system. In addition, this will usually entail that the rate of arrival is constant throughout time.

System Capacity

In many queuing systems there is a limit to the number of customers that may be in the waiting line or system. An arriving customer who finds the system full does not enter but returns immediately to the calling population. However, there are other systems that may simply have an infinite capacity. As we will see later, when a system has a limited capacity, a distinction is made between the arrival rate (i.e. the number of arrivals per time unit) and the effective arrival rate (the number who arrive and enter the system per time unit).

The Arrival Process

The arrival process for infinite population models is usually characterized in terms of inter-arrival times of successive customers. Arrivals may occur at scheduled times or random times. When at random times, the inter-arrival times are usually characterized by a probability distribution. In addition, customers may arrive one at a time or in batches. The batch may be of constant size or of random size.

The most important model, and the only one we will consider, for random arrivals is the Poisson arrival process. If A_n represents the inter-arrival time between customer $n-1$ and customer n, then for a Poisson arrival process A_n is exponentially distributed with mean $1/\lambda$per time unite. The arrival rate is λ customers per time unit. The number of arrivals in a time interval of length t has the Poisson distribution with mean λt customers.

Queue Behavior and Queue Discipline

Queue behavior refers to the actions of customers while in a queue waiting for service to begin. In some situation, there is a possibility that incoming customers will balk, renege, or jockey (move from one line to another if they think they have chosen a slow line).

Queue discipline refers to the logical ordering of customers in a queue and determines which customer will be chosen for service when a server becomes free. Common queue disciplines include first-in-first-out (FIFO), last-in-first-out (LIFO), service in random order (SIRO) etc. Notice that a FIFO queue discipline implies that services begin in the same order as arrivals, but that customers could leave the system in a different order because of different length service times.

Service Times and Service Mechanism

The service times of successive arrivals are denoted by $S_1, S_2, \ldots$. They may be constant or of random duration. In the latter case $\{S_1, S_2, S_3, \ldots\}$ is usually characterized as a sequence of independent and identically distributed random variables. The Exponential, Normal etc. are often used to model service times. Sometimes services are identically distributed for all customers of a given type or class or priority, whereas customers of different types might have completely different service-time distributions. In addition in some systems service times depend upon the time of the day or upon the length of the waiting line.
A queuing system consists of a number of service counters and interconnecting queues. Each service center consists of some number of server, c, working in parallel; that is, upon getting to the head of the line, a customer takes the first available server. Parallel service mechanisms are either single server ($c=1$), multiple server ($1<c<\infty$), or unlimited servers ($c=\infty$).

Probability Distribution:

t is common to use to use the symbols:

- *lamda* to be the mean (or average) number of arrivals per time period, i.e. the mean arrival rate
- µ to be the mean (or average) number of customers served per time period, i.e. the mean service rate

There is a standard notation system to classify queueing systems as A/B/C/D/E, where:

- A represents the probability distribution for the arrival process
- B represents the probability distribution for the service process
- C represents the number of channels (servers)
- D represents the maximum number of customers allowed in the queueing system (either being served or waiting for service)
- E represents the maximum number of customers in total

Common options for A and B are:

- M for a Poisson arrival distribution (exponential interarrival distribution) or a exponential service time distribution
- D for a deterministic or constant value
- G for a general distribution (but with a known mean and variance)

If D and E are not specified then it is assumed that they are infinite.

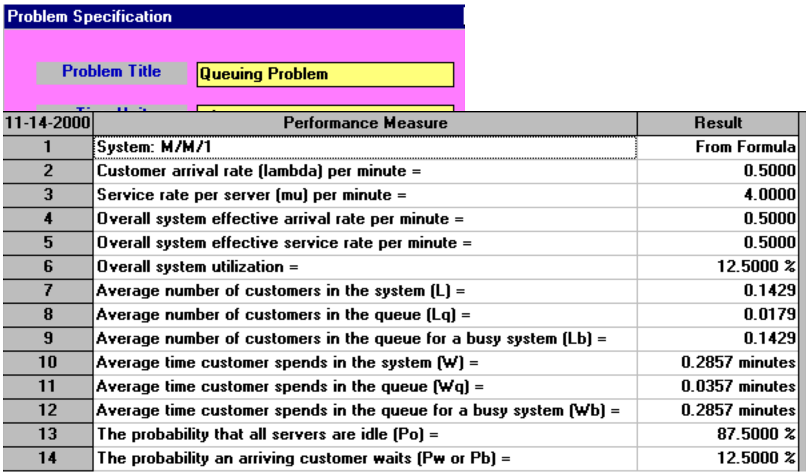
For example the M/M/1 queueing system, the simplest queueing system, has a Poisson arrival distribution, an exponential service time distribution and a single channel (one server).

Note here that in using this notation it is always assumed that there is just a **single queue**(waiting line) and customers move from this single queue to the servers.

Simple M/M/1 example

Suppose we have a single server in a shop and customers arrive in the shop with a Poisson arrival distribution at a mean rate of *lamda*=0.5 customers per minute, i.e. on average one customer appears every 1/*lamda* = 1/0.5 = 2 minutes. This implies that the interarrival times have an exponential distribution with an average interarrival time of 2 minutes. The server has an exponential service time distribution with a mean service rate of 4 customers per minute, i.e. the service rate µ=4 customers per minute. As we have a Poisson arrival rate/exponential service time/single server we have a M/M/1 queue in terms of the standard notation.

We can analyse this queueing situation using the package. The input is shown below:



Problem Specification

Problem Title: Queuing Problem

11-14-2000	Performance Measure	Result
1	System: M/M/1	From Formula
2	Customer arrival rate (lambda) per minute =	0.5000
3	Service rate per server (mu) per minute =	4.0000
4	Overall system effective arrival rate per minute =	0.5000
5	Overall system effective service rate per minute =	0.5000
6	Overall system utilization =	12.5000 %
7	Average number of customers in the system (L) =	0.1429
8	Average number of customers in the queue (Lq) =	0.0179
9	Average number of customers in the queue for a busy system (Lb) =	0.1429
10	Average time customer spends in the system (W) =	0.2857 minutes
11	Average time customer spends in the queue (Wq) =	0.0357 minutes
12	Average time customer spends in the queue for a busy system (Wb) =	0.2857 minutes
13	The probability that all servers are idle (Po) =	87.5000 %
14	The probability an arriving customer waits (Pw or Pb) =	12.5000 %

he first line of the output says that the results are from a formula. For this very simple queueing system there are exact formulae that give the statistics above under the assumption that the system has reached a **steady state - that is that the system has been running long enough so as to settle down into some kind of equilibrium position.**

Naturally real-life systems hardly ever reach a steady state. Simply put life is not like that. However despite this simple queueing formulae can give us some insight into how a system might behave very quickly.
The package took a fraction of a second to produce the output seen above.

One factor that is of note is *traffic intensity* = (arrival rate)/(departure rate) where arrival rate = number of arrivals per unit time and departure rate = number of departures per unit time. Traffic intensity is a measure of the congestion of the system. If it is near to zero there is very little queuing and in general as the traffic intensity increases (to near 1 or even greater than 1) the amount of queuing increases. For the system we have considered above the arrival rate is 0.5 and the departure rate is 4 so the traffic

Classification of Queuing Models:

Different models in queuing theory are classified by using special (or standard) notations described initially by D.G.Kendall in 1953 in the form (a/b/c). Later A.M.Lee in 1966 added the symbols d and c to the Kendall notation. Now in the literature of queuing theory the standard format used to describe the main characteristics of parallel queues is as follows:

$$\{(a/b/c) : (d/c)\}$$

Where

a = arrivals distribution

b = service time (or departures) distribution

c = number of service channels (servers) d = max. number of customers allowed in the system (in queue plus in service)

e = queue (or service) discipline.

Certain descriptive notations are used for the arrival and service time distribution (i.e. to replace notation a and b) as following:

M = exponential (or markovian) inter-arrival times or service-time distribution (or equivalently poisson or markovian arrivel or departure distribution)

D = constant or deterministic inter-arrival-time or service-time.

G = service time (departures) distribution of general type, i.e. no assumption is made about the type of distribution.

GI = Inter-arrival time (arrivals) having a general probability distribution such as as normal, uniform or any empirical distribution.

E_k = Erlang-k distribution of inter-arrival or service time distribution with parameter k (i.e. if k= 1, Erlang is equivalent to exponential and if k = , Erlang is equivalent to deterministic).

For example, a queuing system in which the number of arrivals is described by a Poisson probability distribution, the service time is described by an exponential distribution, and there is a single server, would be designed by M/M/I.

The Kendall notation now will be used to define the class to which a queuing model belongs. The usefulness of a model for a particular situation is limited by its assumptions.

Model 1 :{(M/M1): (/FCFS)} single server, unlimited queue model

The derivation of this model is based on certain assumptions about the queuing system:

1. Exponential distribution of inter-arrival times or poisson distribution of arrival rate.
2. Single waiting line with no restriction on length of queue (i.e. infinite capacity) and no banking or reneging.
3. Queue discipline is ‘first-come, first-serve
4. Single serve with exponential distribution of service time

Performance characteristics

P_w = probability of server being busy (i.e. customer has to wait) = 1-$P_{o= \lambda / \mu}$

1.Expected (or average) number of customer in the system (customers in the line plus the customer being served)

2.Expected (or average) queue length or expected number of customers waiting in the queue

3.Expected (or average) waiting time of a customer in the queue

4.Expected (or average) waiting time of a customer in the system

5.Expected (or average) waiting time in the queue for busy system

6.Probability of k or more customers in the system

7.The variance (fluctuation) of queue length Var (n)

=

8.Expected non-empty queue length

9.Probability that waiting time is more than

Poisson Queuing Systems:

Poisson Queues

A Poisson queue is a queuing model in which the number of arrivals per unit of time and the number of completions of service per unit of time, when there are customers waiting, both have the Poisson distribution.

The Poisson distribution is good to use if the arrivals are all random and independent of each other.

For the Poisson distribution, the probability that there are exactly x arrivals during t amount of time is: The Poisson

Distribution

t is a duration of time. Its units are, e.g., hours or days.

8 (Greek letter lambda) is the expected (average) number of arrivals per hour, or day, or whatever units t is measured in.

8t is therefore the expected number of arrivals during t amount of time.

x is a possible number of arriving customers.

x! ("x factorial") means 1×2×...×(x-1)×x. For example, 5! = 1×2×3×4×5. We define 0! = 1.

If arrivals are distributed according to the above formula, then we say "the arrivals are Poisson" or "have the Poisson distribution."

Spreadsheet Implementation

Here is a spreadsheet layout for calculating values for the Poisson distribution. Detailed explanations of the formulas will follow.

For illustration, I've picked a 8 of 4 arrivals per hour expected, and a t of 2 hours. This makes P of x the probability that exactly x people will arrive in the next 2 hours, given that the expected number of arrivals is 4.

P of <= x, in column C, is the probability that x or fewer people will arrive in the next two hours.

Column A has the values of x. I use formulas like =A6+1, rather than typing in 1,2,3,..., to make it easier to extend the table down with one copying operation.

Here’s an explanation of the formula =EXP(-B3)*B3^A6/FACT(A6) in B6:

EXP is the Excel function for raising e to a power. =EXP(-$B$3) means e-$B$3, which is e-8t, since

QUEUING THEORY I 4

-8t is in B3. The dollar signs keep the cell reference on B3 when the formula is copied and pasted to another cell.

The caret symbol ^ is for raising a number to a power. $B$3^A6 means ($B$3)A6, which means (8t)x.

The FACT function (@FACT in Quattro Pro) calculates factorials. Here, it is calculating x!, the factorial of the x value in column A.

The formula in the C column, starting with C7, gives a running total of the P of x's. Each cell is the cell above, which is the previous total, plus the cell to the left, the current P of x.

When you are constructing the table of probabilities in the spreadsheet:

1. Fill in row 6 as shown.

2. In A7, type =A6+1 as shown.

3. Copy cell B6 and paste to B7.

4. In C7, type =B7+C6 as shown. That should give you row 7 complete as in the diagram above.

5. Copy A7:C7. Paste to a range starting in A8 and extending down the A column as far down as

you want. (For the homework, you’ll want to go down about twenty rows.)

Here are the numbers I see after I copy A7:C7 to A8:A24.

Network Scheduling by PERT/CPM:

Project scheduling is a difficult job for project managers. Managers estimate the time and resources required to complete activities and organize them into a coherent sequence. **Program Evaluation and Review Technique (PERT)** and the **Critical Path Method (CPM)** are two project scheduling methods that can be used in software development. Both are driven by information already developed in earlier project planning activities. Here I will discuss the use of PERT only. PERT is used to schedule, organize, and coordinate tasks within a project. It is basically a method to analyze the tasks involved in completing a given project, especially the time needed to complete each task, and to identify the minimum time needed to complete the total project. The main objective of PERT is to facilitate decision making and to reduce both the time and cost required to complete a project. PERT planning involves the following steps:

- Identify the specific activities and milestones.

- Determine the proper sequence of the activities.
- Construct a network diagram.
- Estimate the time required for each activity.
- Determine the critical path.
- Update the PERT chart as the project progresses.

PERT planning involves the following steps that are described below.

1. **Identify the specific activities and milestones**. The activities are the tasks required to complete a project. The milestones are the events marking the beginning and the end of one or more activities. It is helpful to list the tasks in a table that in later steps can be expanded to include information on sequence and duration.
2. **Determine the proper sequence of the activities**. This step may be combined with the activity identification step since the activity sequence is evident for some tasks. Other tasks may require more analysis to determine the exact order in which they must be performed.
3. **Construct a network diagram**. Using the activity sequence information, a network diagram can be drawn showing the sequence of the serial and parallel activities. Each activity represents a node in the network, and the arrows represent the relation between activities. Software packages simplify this step by automatically converting tabular activity information into a network diagram.
4. **Estimate the time required for each activity**. Weeks are a commonly used unit of time for activity completion, but any consistent unit of time can be used. A distinguishing feature of PERT is its ability to deal with uncertainty in activity completion time. For each activity, the model usually includes three time estimates:

- Optimistic time – generally the shortest time in which the activity can be completed. It is common practice to specify optimistic time to be three standards deviations from the mean so that there is a approximately a 1% chance that the activity will be completed within the optimistic time.
- Most likely time – the completion time having the highest probability. Note that this time is different from the expected time.

- Pessimistic time – the longest time that an activity might require. Three standard deviations from the mean is commonly used for the pessimistic time.

PERT assumes a beta probability distribution for the time estimates. For a beta distribution, the expected time for each activity can be approximated using the following weighted average:

- Expected time = (Optimistic + 4 x Most likely + Pessimistic) / 6
- This expected time may be displayed on the network diagram.
- To calculate the variance for each activity completion time, if three standard deviation times were selected for the optimistic and pessimistic times, then there are six standard deviations between them, so the variance is given by: [(Pessimistic - Optimistic) / 6]^2

1. **Determine the critical path**. The critical path is determined by adding the times for the activities in each sequence and determining the longest path in the project. The critical path determines the total calendar time required for the project. If activities outside the critical path speed up or slow down (within limits), the total project time does not change. The amount of time that a non – critical path activity can be delayed without the project is referred to as a *slack time*.

If the critical path is not immediately obvious, it may be helpful to determine the following four quantities for each activity:

ES – Earliest Start time
EF - Earliest Finish time
LS – Latest Start time
LF - Latest Finish time

These times are calculated using the expected time for the relevant activities. The earliest start and finish times of each activity are determined by working forward through the network and determining the earliest time at which an activity can start and finish considering its predecessors activities. The latest start and finish times are the latest times that an activity can start and finish without delaying the project. LS and LF are found by working backward through the network. The difference in the latest and earliest finish of each activity is that activity’s slack. The critical path then is the path through the network in which none of the activities have slack.

The variance in the project completion time can be calculated by summing the variances in the completion times of the activities in the critical path. Given this variance, one can calculate the probability that the project will be completed by the certain date assuming a normal probability distribution for the critical path. The normal distribution assumption holds if the number of activities in the path is large enough for the central limit theorem to be applied.

Since the critical path determines the completion date of the project, the project can be accelerated by adding the resources required to decrease the time for the activities in the critical path. Such a shortening of the project sometimes is referred to as *project crashing*.

Update the PERT chart as the project progresses. Make adjustments in the PERT chart as the project progresses. As the project unfolds, the estimated times can be replaced with actual times. In cases where there are delays, additional resources may be needed to stay on schedule and the PERT chart may be modified to reflect the new situation.

Some of the **benefits of PERT** are:

- Provides expected completion time.
- The probability of completion before a specific date.
- The critical path activities that directly impact the completion time.
- The activities that have slack time and that can be lend resources to critical path activities.
- An activity start and end date.

As a prerequisite, personnel should already have a common understanding of formal project management terminology, tools, and techniques. They should be able to create a project plan, choose the most appropriate scheduling method, and select and organize a team to perform project tasks.

Project schedules are usually represented in a set of charts showing the work breakdown, activities dependencies and staff allocations. Here is an example case study where the project manager knows the succession of the project activities and the optimistic, pessimistic, and most likely time (in weeks) for the following activities.

Rules of Network Constructon:

Construction of network:

Rules for constructing network

For the construction of a network, generally, the following rules are followed:

(i) Each activity is represented by one and only one arrow.(i.e) only one activity can connect any two nodes.

(ii) No two activities can be identified by the same head and tail events.

(iii) Nodes are numbered to identify an activity uniquely. Tail node (starting point) should be lower than the head node (end point) of an activity.

(iv) Arrows should not cross each other.

(v) Arrows should be kept straight and not curved or bent.

(vi) Every node must have atleast one activity preceding it and atleast one activity following it except for the node at the beginning and at the end of the network.

Numbering the Events

After the network is drawn in a logical sequence, every event is assigned a number. The number sequence must be such as to reflect the flow of the network. In event numbering, the following rules should be observed:

(i) Event numbers should be unique.

(ii) Event numbering should be carried out on a sequential basis from left to right.

(iii) The initial event is numbered 0 or 1.

(iv) The head of an arrow should always bear a number higher than the one assigned at the tail of the arrow.

(v) Gap should be left in the sequence of event numbering to accommodate subsequent inclusion of activities, if necessary.

Example

Draw the logic network for the following:

Activities C and D both follow A , activity E follows C , activity F follows D , activity E and F precedes B.

Solution:

The required network for the above information.

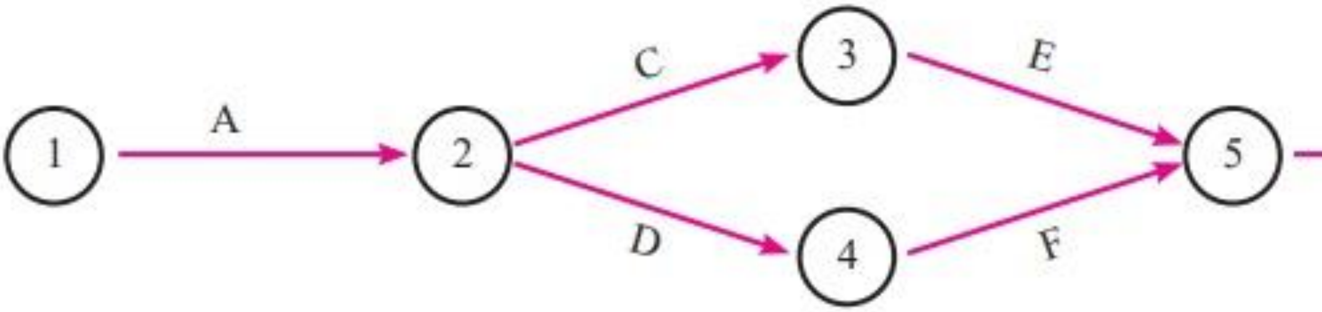


Fig 10.5

Example

Develop a network based on the following information:

Activity :	A	B	C	D	E	F	G
Immediate predecessor:	-	-	A	B	C,D	C,D	E

Solution:

Using the immediate precedence relationships and following the rules of network construction, the required network is shown in following figure

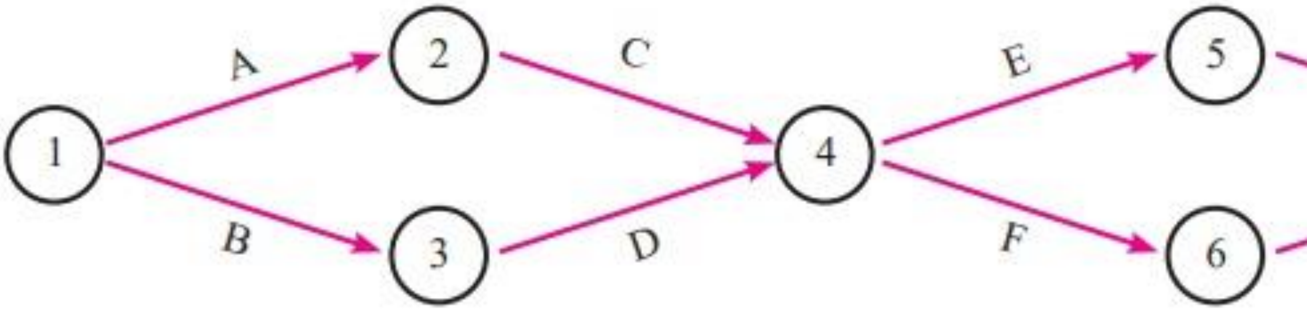


Fig 10.6

Critical Path Analysis:

Critical path analysis is a project-management technique that lays out all of the activities needed to complete a task, the time it will take to complete each activity and the relationships between the activities. Also called the "critical path method", critical path analysis can help predict whether a project can be completed on time and can be used to reorganize the project both before starting it, and as it progresses, to keep the project's completion on track and ensure that deliverables are ready on time. Either manually or using computer software, the project manager first lists each activity, the order it must be completed in and how long it is expected to take, and then diagrams the process.

Advantages: The benefit of using Critical Path Analysis (CPA) within the planning process is to help develop and test a plan to ensure that it is robust. Critical Path Analysis formally identifies the tasks which must be completed on time for the whole project to be completed on time.

Limitations: The disadvantage of CPA is that it does not show the relation of tasks to time as clearly as other planning tools, such as Gantt Charts. This can make them more difficult to understand when used for management and communication of plans within organizations.

What Is Critical Path Analysis?

Critical path analysis (CPA) is a project management technique that requires mapping out every key task that is necessary to complete a project. It includes identifying the amount of time necessary to finish each activity and the dependencies of each activity on any others.

Also known as the critical path method, CPA is used to set a realistic deadline for a project and to track its progress along the way.

KEY TAKEAWAYS

- Critical path analysis is a project planning method that focuses on identifying tasks that are dependant on other tasks for their timely completion.
- Understanding the dependencies between tasks is key to setting a realistic deadline for a complex project.

- Critical path analysis is used in most industries that undertake highly complex projects.

What Is the Purpose of Critical Path Analysis?

In the late 1950s, James Kelley of Remington Rand and Morgan Walker of DuPontdeveloped a project management technique called the critical path method (CPM). Earlier versions of their technique were being practiced before then and are said to have contributed to the completion of the Manhattan Project, the secret American defense program to build an atomic bomb in order to end World War II.

Critical path analysis identifies the sequence of crucial and interdependent steps that comprise a work plan from start to finish. It also identifies non-critical tasks. These may also be important, but if they hit an unexpected snag they will not hold up any other tasks and thus jeopardize the execution of the entire project.

The concept of a critical path recognizes that completion of some tasks in a project is dependent on the completion of other tasks. Some activities cannot start until others are finished. Inevitably, that presents the risk of bottlenecks.

CPA is used widely in industries devoted to extremely complex projects, from aerospace and defense to construction and product development.

How to Use CPA

CPA detects and defines all of the critical and noncritical tasks involved in a work plan and identifies both the minimum and the maximum amount of time associated with each. It also notes those dependencies among activities, and that tells them the amount of float or slack time that can be associated with each in order to arrive at a reasonable overall deadline date.

Probability Considerations in PERT:

How to Calculate Probability in PERT

A Project Evaluation and Review Technique (PERT) is a tool that project management uses to manage a project. The PERT chart contains information about subprojects and estimated completion times. Each subproject is assigned an estimated completion time, based upon probability distributions for the expected completion time for the subproject from start to finish. The probability distribution for expected time is calculated by the following equation: Expected time = (Optimistic Time + 4 x Most likely Time + Pessimistic Time) / 6.

step 1.

Add the Optimistic Time for a subproject to the Pessimistic time for the same subproject. For example, if your subproject’s optimistic time is one day, and Pessimistic Time is seven days, the total for this step is eight days.

step 2.

Multiply the Most Likely Time by four. If the most likely time is two days, then two days times four is eight days.

step 3.

Add your answer from Step one to your answer from Step two. For the example, eight days + eight days is 16 days.

step 4.

Divide your answer in Step 3 by 6. For the example, 16 days / 6 is 2.67 days (rounded). 2.67 days is the expected time for this subproject.

step 5.

Repeat Steps 1 through 4 for each subproject in your PERT Chart. When you finish, you'll have calculated expected values for each step of your entire project schedule.

Short Answers:

1.Explain about Queuing Theory

2.Explain about Characteristics of Queuing System

3.Explain about Elements of Queuing System

4.Explain about Non-Poission Queuing System

5.Explain about Rules Of Rules of Network Construction

Long Answers:

6.Explain about Network Scheduling by PERT/CPM

7.Explain about queuing Theory and Classification of Queuing Models

8.Explain about Poission Queuing Systems And Non-Poission Queuing Systems

9.Explain about Critical Path Analysis

10. Explain about Probabilistic Distributions In Queuing Theory.

Question papers for Sample:

S.V.U. COLLEGE of COMMERCE MANAGEMENT AND

COMPUTER SCIENCE

TIRUPATHI

DEPARTMENT OF COMPUTER SCIENCE

Time:**2hours** **INTERNAL EXAMINATIONS -I** MAX MARKS:30

Section-A

Answer any five from the following 5*2=10m

1.What is Linear Programming?

2.What is Transportation Problem?

3.What are Networking Techniques?

4.Explain about Simplex Method

5.Explain about U-V Method

6.What is Duality?

7.What is Mathematical Model of Transportation?

8.What is Assignment Problem?

Section-B

Answer any one Question from each unit 2*10=20marks

Unit-1

9.(a).Explain about LP Model

(b).Explain about Graphical Method

(Or)

10.(a) Explain about Formulation of Dual Problem

(b) Explain about the Concept of Duality

Unit-2

11.Explain about Formation of Assignment Problem

12.(a)Explain about Dijikstras Algorithm

(b) Explain about Floyds Algorithm

S.V.U. COLLEGE of COMMERCE MANAGEMENT AND
COMPUTER SCIENCE
TIRUPATHI
DEPARTMENT OF COMPUTER SCIENCE

Time:**2hours** **INTERNAL EXAMINATIONS -ll** MAX MARKS:30

Section-A

1.What is Systematic Method?

2.What is Maximal Flow Problem?

3.What is Game Theory?

4.What is Queuing Theory?

5.What are Rules for Network Construction?

6.What is Systematic Method?

7.What is Dominance Property?

8.What is Probablistic Distributions In Queuing System?

Section-B

Answer any one Question from each unit 2*10=20marks

Unit-1

9.(a)Explain about Minimum Spanning Tree

(b)Explain about Krushkals Algorithm

(or)

10.(a)Explain about Shortest -Path Model

(b)Explain about Maximal Flow Problem

Unit-2

11.(a)Explain about Game Theory and Strategies

(b) Explain about Two -Person -Zero -Sum Games

(or)

12.(a)Explain about Queuing System and it's Elements

(b)Explain about Classification of Queuing Models

MASTER OF COMPUTER APPLICATIONS DEGREE

EXAMINATION

SECOND SEMESTER

Paper MCA 201:: Computer Oriented Operation Research
Under C.B.S.C Revised Regulations w.e.f.2021-2023

(Common paper to University and all Affliated Colleges)

Time:3 hours Max.Marks:70

PART-A

(Compulsory)

Answer any five of the following questions each question carries 2 marks. (5*2=10)

1.a.What is Linear Programming?

b. What is Application of Duality?

C. What is Transportation Problem?

d. What is Hungerian Method?

e. What is Networking Technique?

f. What is Prims Algorithm?

g. What is Game Theory?

h. What is Rectangular Games?

i. What is Queuing Theory?

j. What are Probabilistic Considerations?

PART-B

Answer any ONE full question from each unit

Each question carries 12 Marks (5*12=60)

UNIT-1

2.Explain about Graphical Method

(Or)

3.Explain about Simplex Method

Unit-2

4.Explain about Formulation of Assignment Problem

(or)

5.Explain about Optimal Solution by U-V Method

Unit-3

6.Explain about Linear Programming Modelling For Maximal Flow Problem

(or)

7.Explain about Minimum Spanning Tree

Unit-4

8.Explain about Graphic Solution of 2xn and Mx2 Games

(or)

9.Explain about Arithmetic Model For nxn Games

Unit-5

10.Explain about Classification of queuing Models

(or)

11.Explain about Poission Queuing Systems.