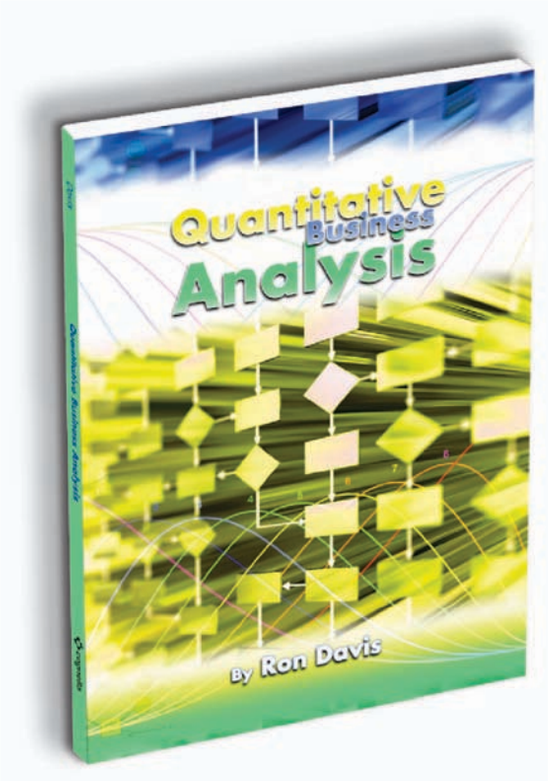
cognella academic publishing

Sneak Preview



Quantitative Business Analysis

by Ron Davis

Included in this preview:

- Copyright Page
- Table of Contents
- Excerpt of Chapter 1

For additional information on adopting this book for your class, please contact us at 800.200.3908 x501 or via e-mail at info@cognella.com

MANAGEMENT SCIENCE READER for Quantitative Business Analysis

Ron Davis

San Jose State University

cognella™
San Diego, CA

First published in the United States of America in 2010 by Cognella, a division of University Readers, Inc.

Previously published by Mathematical Programming Services
3790 El Camino Real #219
Palo Alto, California 94306
http://www.mathproservices.com

14 13 12 11 10 1 2 3 4 5

Printed in the United States of America

ISBN: 978-1-935551-32-4

cognella™

www.cognella.com 800.200.3908

Contents

0 OR/MS Methodology/ Terminology

In past decades, the term Operations Research (OR) connoted a more mathematical or algorithmic emphasis than did the term Management Science (MS) which was used in more practical or applied contexts. But in recent years, the two terms have become blurred, and overlap so much that in fact the former Operations Research Society of America and the former Institute of Management Sciences merged and became the unified INFORMS. This acronym stands for Institute For Operations Research and the Management Sciences. Two INFORMS conferences are held each year, one in the spring and one in the fall, where thousands of presentations are made by academics, practitioners, government employees, consulting firms, and the military. The best way to get a feeling for the breadth and depth of the papers presented is to go to the INFORMS web site at www.informs.org and take a look at their listings for recent and upcoming conferences on the web. There is also the Decision Sciences Institute, another professional society in this area. You can visit their web site at http://www.decisionsciences.org.

0.1 Model Building Processes

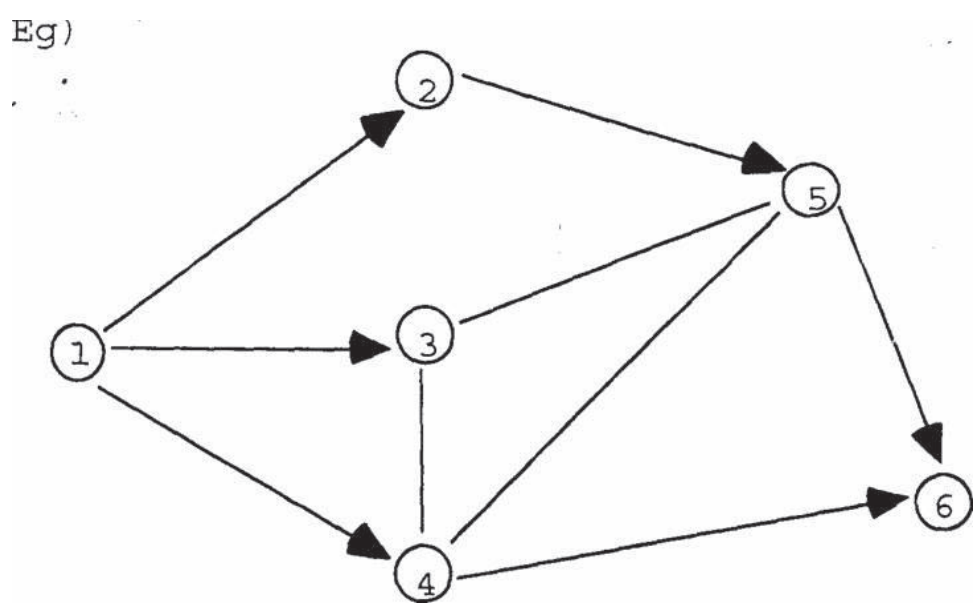
What unifies the MS/OR community is the commitment to a particular methodology for problem solving, one which is analogous to the scientific method. There is a fundamental difference, however, in that management science models incorporate one or more quantitative objective functions used as performance measures for the evaluation of system performance resulting from selected decisions or controls. There is a universal desire to improve system performance through the selection of those decision variable values or control settings that give rise to the "best" performance with respect to the performance indices which have been selected for the problem. If there is only one performance index, we seek optimal solutions that either minimize or maximize the performance index. If there are two or more performance indices, then we seek the Pareto-optimal set of non-dominated or maximal-value solutions. This concept is defined more precisely later in the chapter.

Besides the presence of one or more objective functions for evaluating system performance, another ubiquitous commitment which MS/OR practitioners share is that mathematical and computer models are central to the analysis and computation of improved solutions. MS/OR practitioners are uniformly model builders, and these models invariably have a mathematical aspect and a computational aspect. It would be a mistake to conclude that model building is the province of mathematician and computer scientists only. In order that the model built has sufficient "reality" built into it, inputs from other disciplines are required. It frequently requires the combined efforts of a team of specialists with knowledge of the engineering, production, logistics, marketing and financial aspects, all providing critical inputs to the model building process. Managing the model development process becomes a job in itself, and the process can be described in greater detail by reference to Figure 0-1 shown on the next page.

The process begins in the upper left corner, with a "real-world problem." Since model building is not free, the process must begin with the realization that there is "room for improvement" in some aspect of a business's operation, and a consequent commitment to expend the resources necessary to carry out a model-building effort. In short, the "higher-ups" in the organization must be convinced that the prospects for a positive return on investment are good. Two aspects of obtaining support and approval from the "higher-ups" are a clear demonstration that current practice is not nearly as effective or efficient as it might be. And secondly, they must be convinced that the present "state-of-the-art" tools are sufficiently powerful to handle the dimensionality and complexity of the requisite models for the decision problem at hand.

Once a decision problem has been identified for which a model-building effort is desired, the first step is to prepare a written problem statement. This gives a detailed account of the alternatives to be considered, the system structure which relates actions taken and performance indices used for defining optimal or Pareto-optimal solutions, and the data to be taken into account by the model. This problem definition is usually not written by the model builders, but is rather provided by various professionals in the organization who know what the "real world" problem is.

Figure 0-1: OR/MS MODELING METHODOLOGY



The next step is to translate the verbal statement of the problem into a mathematical formulation of the problem. The definition of the mathematical formulation involves some mathematical notation and the model builders rather than the model sponsors usually provide the mathematical thinking behind the model. This step is referred to as problem formulation since the mathematical model usually includes a number of quantitative formulas useful in stating the objectives and constraints in the model.

Once formulated, the knowledgeable model builder must then classify the model and come to one of two conclusions. Either (1) this is a model type which is known and for which a solution algorithm already exists; or (2) this model is of a type which has not yet been analyzed, or for which solution procedures have not yet been developed. In the former case, it is then a matter of applying the known modeling and solution algorithm to the data associated with the problem at hand. In the latter case, the services of a mathematician and a computer scientist must be secured to develop new analysis and new solution algorithms for the problem at hand.

Once the mathematical model has been specified and a solution method either selected or developed, then a computer model must be developed which embodies both the mathematical formulation of the problem and the "real-world" data associated with the model. This enables the solution algorithm to be run on the "real" data respecting the relationships and objectives in the mathematical model. The outputs from these runs then constitute the computer solution to the problem at hand.

Since there are many points at which errors can creep into the process, it is necessary to maintain a healthy skepticism about the computer results until a thorough testing process has been complete. The model usually goes through an evolutionary process in which errors and glitches of all types are gradually eliminated from the model. Errors can occur at the formulation stage. Data entry or data alignment or scaling errors can easily creep in. There can be errors in the solution algorithms or glitches in the computer programs that implement those algorithms. There can even be errors in the report generators that create false output reports based on correct internal solution values. Hence, for a model of any meaningful size or complexity, it is extremely unlikely that the model will function correctly on the first run. It is much more common that a period of "debugging" must be endured until the errors are systematically removed from the model, until it functions correctly. Model building sponsors and managers alike must be prepared to carry out this "debugging" effort, lest the project fail before all errors are removed.

To facilitate this debugging process, two types of formal testing are generally carried out to "flush out" the glitches which need to be handled. The first round of testing is referred to as "verification" testing. This entails a comparison of model outputs with model inputs to see if the results are mathematically correct. This is a test which can be carried out by the model builders, since there will be technical specifications for how the model is supposed to work. In some cases, alternate software systems may be run in parallel on the same data to see if they produce equivalent results. During this phase of the development, the goal is to eliminate all modeling, data, algorithmic and programming errors which may be present after the first pass. Reworking of all these aspects may be necessary to pass the verification tests.

At the successful conclusion of the verification testing there follows another round of testing known as "validation" testing. The model building team, since they may have made a number of hidden assumptions that are not correct, must NOT conduct these tests. Rather, the decision support system must be "turned-over" to the model sponsors for independent testing by users not involved in the development

of the system. These "real-world" testers may provide data sets other than those used by the developers, and may turn up some malfunctions that were not seen during the verification tests. Problems identified during validation may also entail reworking of many of the same areas as verification testing required. Untrained or non-technical users may try features or exercise options that had not been tested as part of verification testing.

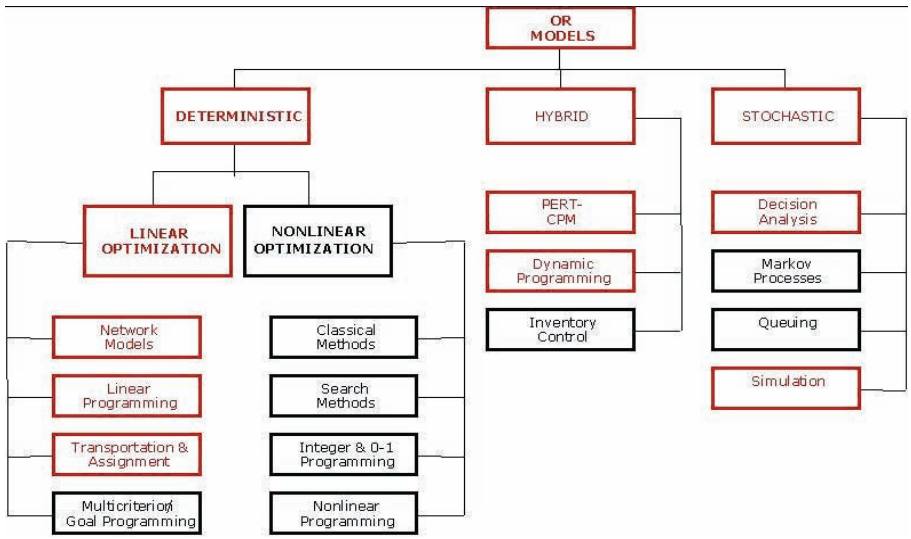
Once the decision model sponsors are satisfied with the performance of the decision support system, then it becomes a matter of accounting to see if use of the model leads to superior results compared to current practice. While success is not guaranteed, the long list of success stories is now, after 50 years, so long that the building of models is now a standard business practice. This is true also for government agencies, in the military, and even among non-profits. As they say on TV, "try it, you'll like it."

0.2 A Taxonomy of OR/MS Model Types

There have now been over 50 years since George Dantzig discovered the Simplex Algorithm for solving linear programming problems. The publication of this algorithm sparked a tremendous explosion in the development of various model types and various solution algorithms. It was found that readily apparent extensions of the Simplex Method enabled one to solve problems with non-linear objective functions and also non-linear constraints. Also a number of branch-and-bound algorithms were developed for solving problems in which some or all of the variables are either integer or binary instead of continuous. The parallel development of mathematical model types and mathematical solution algorithms has led to an ever wider class of problem types for which solution algorithms and hence also computer programs have been developed. Today, most comprehensive Management Science textbooks have over 15 chapters that discuss no less than a dozen different model types.

Figure 0-2 on this page gives a model classification that helps identify the scope of existing knowledge as well as the scope of the present Course Reader. Areas that are covered in this one-semester introductory survey are marked in red, while those areas we have to omit for lack of time are shown in black. In order to understand the structure of this diagram it is necessary to define a few terms that are commonly used in this discipline but are frequently not known or understood by the layman. The first dichotomy indicated by the chart is the distinction between a "deterministic" and a "stochastic" model. This relates to the fact that the data or parameters for a model are of two types. Parameters that are known with a fair degree of accuracy are represented as constants, i.e. by single values that may be particular specified numbers in a spreadsheet model. On the other hand, parameters about which there is considerable uncertainty, such as demand forecasts for example must instead be represented by probability distributions. These uncertain quantities are said to be random variables, and are generally described by means of standard density functions, or by means of a histogram distribution. If any of the parameters in a model are represented as random variables, then the model is said to be a STOCHASTIC MODEL. Whereas, if all parameters are represented by constants, then the model is said to be a DETERMINISTIC MODEL.

Figure 0-2: A TAXONOMY OF OR/MS MODELS



Another dichotomy indicated by the diagram is between linear and nonlinear models. A linear expression is one which can be stated as a summation of a set of "coefficient * variable" expressions. The coefficients are constants, and the variables are included at the first power - no higher order polynomials, no trig functions, no logs or exponential functions, etc. If there are no nonlinearities then the model is said to be a LINEAR MODEL.

Nonlinearities, if they exist at all, can occur in three distinct ways. There may be a need to include nonlinear terms in one or more of the objective functions. The standard Markowitz model for portfolio optimization, for example, has a quadratic form in the objective function representing the variance of the portfolio return. Nonlinearities can also occur in the constraints of the model, as can be the case, for example, if both Cartesian and polar coordinates must be included in the same formulation. And nonlinearity can arise if some of the variables in the model must be integers or binary variables. This causes the model to become an integer or mixed integer model. We will not treat any of these extensions in this course, but it is important to note their existence since many current applications utilize nonlinear formulations.

0.3 OR/MS Glossary

A link to a detailed glossary by Harvey Greenberg, adopted by Informs, is given below

http://glossary.computing.society.informs.org/

but there are a few basic concepts and definitions that need to be spelled out in detail here. Other terms and concepts will be introduced throughout the course as needed.

An ALGORITHM is a detailed step-by-step procedure for computing the solution to a quantitative problem.

OPTIMIZATION is the process of minimizing or maximizing the value of an objective function, possibly subject to a set of constraints on the decision variables in the problem formulation.

An OPTIMIZATION ALGORITHM is a detailed step-by-step procedure for maximizing or minimizing the value of an objective function, possibly subject to a set of constraints on the decision variables in the problem formulation.

An OPTIMAL SOLUTION is the set of decision variable values that yield the minimum or maximum value of the objective function (possibly subject to constraints).

The OPTIMAL SOLUTION VALUE is the value of the objective function at an optimal solution point.

A HEURISTIC ALGORITHM is an approximation procedure which is designed to give a good solution with a relatively small computational effort.

Heuristics are often used to initialize an optimization algorithm. They are also used when the computational burden for obtaining a truly optimal solution is too great.

0.4 OR/MS Links

Since new sites are going up all the time, you need to do your own searches to find the latest developments. There are a few "old standbys" that you should be aware of, however, which will get you started in your search for interesting applications of the OR/MS methodology. These are presented below.

http://www.lionhrtpub.com/ORMS.shtml
http://www2.informs.org/Resources/
http://www.me.utexas.edu/~jensen/ORMM/
http://www.ece.northwestern.edu/OTC/iindex.html
http://www.worms.ms.unimelb.edu.au/toc.html
http://www.ifors.org

1 Network Models

Many practical applications of management science involve the use of network models. These arise in different ways, so the interpretation of the network elements may vary from one application to the next. For a transportation network, the nodes represent locations and the arcs represent routes between the locations. Whereas, for project analyses, the nodes represent project activities and the arcs represent precedence relationships between the activities. There are certain fundamental concepts about networks that are common to all such applications, and for our purposes, a key underlying concept is the notion of a spanning tree. This chapter is devoted primarily to explaining what a spanning tree is, and how various algorithms discover spanning trees that are associated with solutions to optimization problems posed for network models.

1.0 Definition of Terms

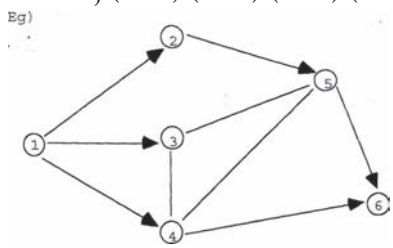
Network Diagram - set of nodes and arcs, G = {V,E} where V is the set of nodes (vertices) and E is the set of arcs (edges) in the network.

Nodes (vertices) - junction points; at which a flow originates, terminates or is relayed.

Arcs (branches or edges) - connect pairs of nodes. Represent flow or precedence.
Undirected Arcs - No arrow
Directed Arcs - Arrow

Nodes i=1,2,….., 6
Undirected arc (i,j) - between i & j (3,4) (3,5) (4,5)
Directed arc (i->j) - from i to j (1->2) (1->3) (1->4) (2->5) (4->6) (5->6)

Eg)



Chain - sequence of arcs and nodes connecting any two nodes (i&j), disregarding direction on directed arcs.

Eg) 6 (5->6) 5 (3,5) 3 (1->3) 1 is a chain from node 6 to node 1.

Path - a chain which respects specified direction along directed arcs in the chain.

Cycle - a chain connecting a node to itself, in which each arc and each node (other than the starting node) occurs only once.

Eg) (3,4) (4,5) (5,3) is a cycle.

Source Node or Origin Node - an origin of flow in a network.

Sink Node or Destination Node - a termination of flow in a network.

Other nodes in transportation networks are sometimes called Transshipment Nodes.

Two nodes are connected if there is a chain joining them. Two nodes are adjacent if there is an arc connecting them. A network is connected if there is a chain joining any pair of distinct nodes in the network.

A Sub-Network of a reference network G = {V, E} is a network G' = {V',E'} where G contains G', V contains V', and E contains E', such that the nodes connected by each arc in E' are contained in V'.

Tree - a connected sub-network containing no cycles.

Spanning Tree - A Tree which contains every node in the network.

Note: A spanning tree on N nodes contains N-1 arcs. This is proved by induction on N, noting that a tree on 2 nodes contains 1 arc, and each additional node requires the addition of one additional arc.

1.1 Minimal Spanning Tree Problem

Given: A connected network with branch lengths (distance, time or cost).

Find: A spanning tree with minimum total length, time or cost.

The solution algorithm for this problem, perhaps the simplest you will encounter, proceeds by "growing a tree" according to what is sometimes called the "Greedy Algorithm." This name refers to the fact that at each step, the cost of the next branch to be included is minimized. It is a "myopic" algorithm in that it only looks one step ahead at each iteration, but has the pleasing property that for this problem type, the optimal solution is obtained in spite of its near sightedness.

The description of the algorithm makes use of two terms which are defined for networks a little differently than they are in common parlance. Two nodes are said to be adjacent to each other if they are connected by a single arc, no matter how far away they may be from each other spatially.

And by the connected nodes we mean those which have been included in the growing tree which has been built so far, up to the present iteration. The algorithm description which follows has two parts, the initialization step, and the iterative steps, which are repeated until the stopping condition has been met.

Initialization

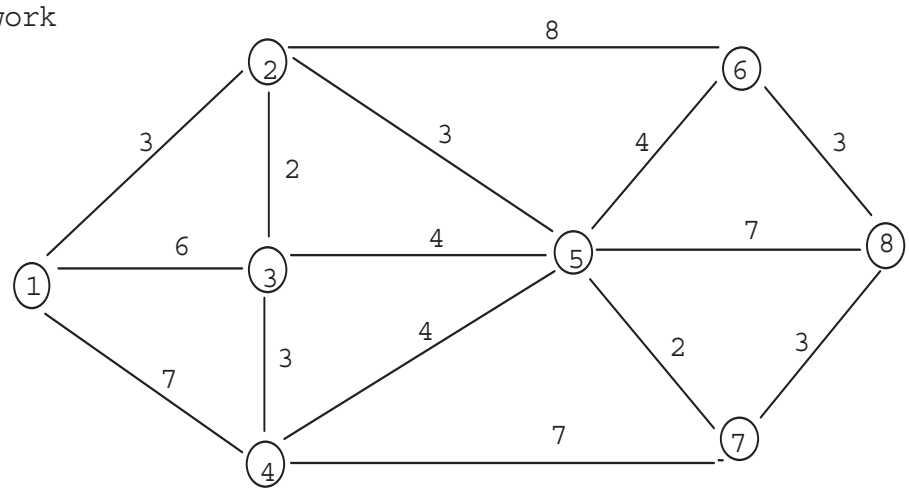
0) Arbitrarily select any node of the network and connect it to the nearest adjacent node. Alternatively, one may select as the first arc the one with the minimum arc value (distance, time or cost).

Iterations

1) Identify the set of arcs connecting one of the connected nodes to one of the adjacent unconnected nodes (no cycles are allowed in a tree). Break ties arbitrarily.
2) Select from the set of potential tree extensions identified in step 1 those with minimal value. Enter as many of these into your growing tree as you can without creating any cycles.
3) Repeat steps 1& 2 until all nodes have been connected.

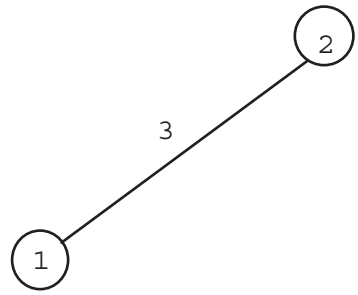
OPTIMALITV CONDITION

Computer Network

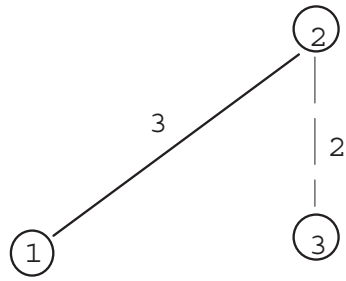


Each arc NOT included in the spanning tree, if added to the tree, forms a unique cycle. For optimality, each arc NOT included in the spanning tree must have an arc value which is greater than or equal to each of the arc values on the other arcs in the unique cycle it creates when added to the tree. For if this is not the case, then the arc NOT in the tree could be added to the tree, and the arc having highest value on the created cycle could be removed from the tree, giving rise to a new tree with a lower sum.

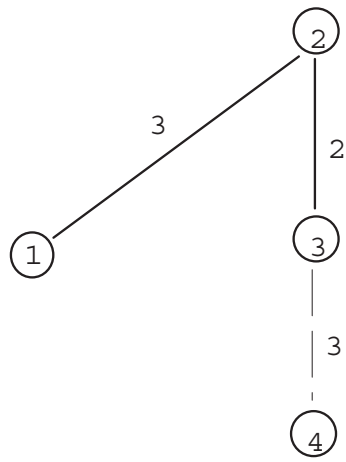
Step 1: Unconnected node nearest to 1 is node 2.



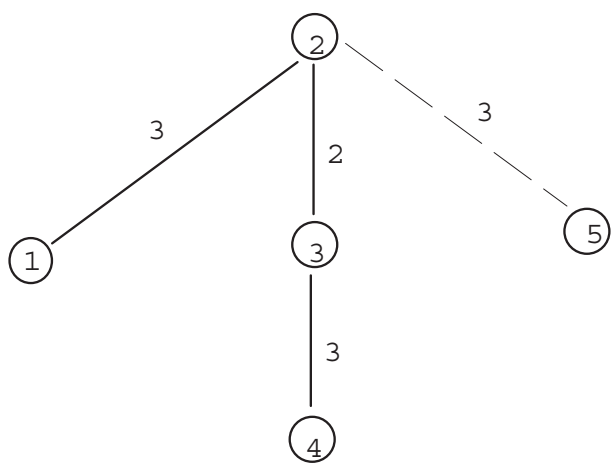
Step 2: Unconnected node nearest to 1 or 2 is node 3. (from 2)



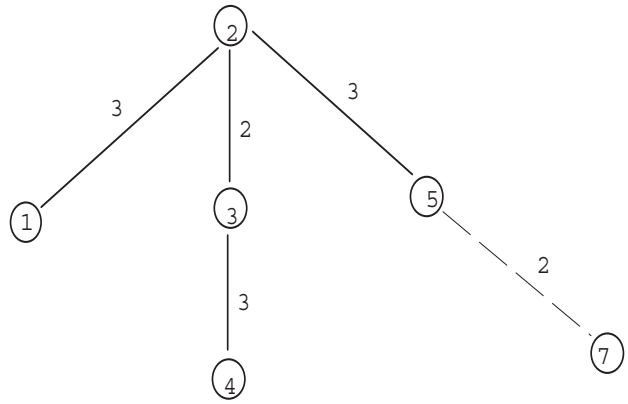
Step 3: Unconnected node nearest to (1,2,3) is node 4. (from 3)



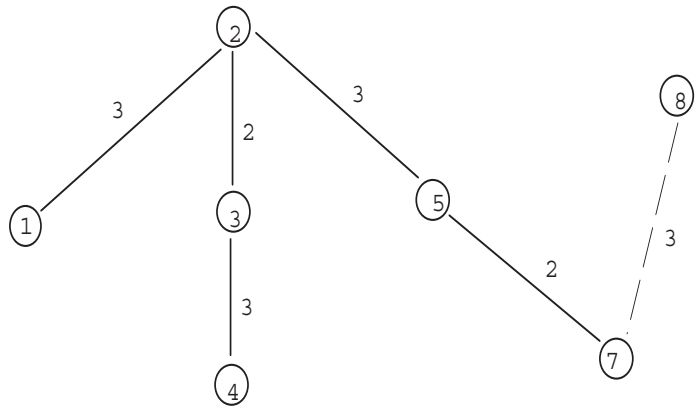
Step 4: Unconnected node nearest to (1,2,3,4) is node 5. (from 2)



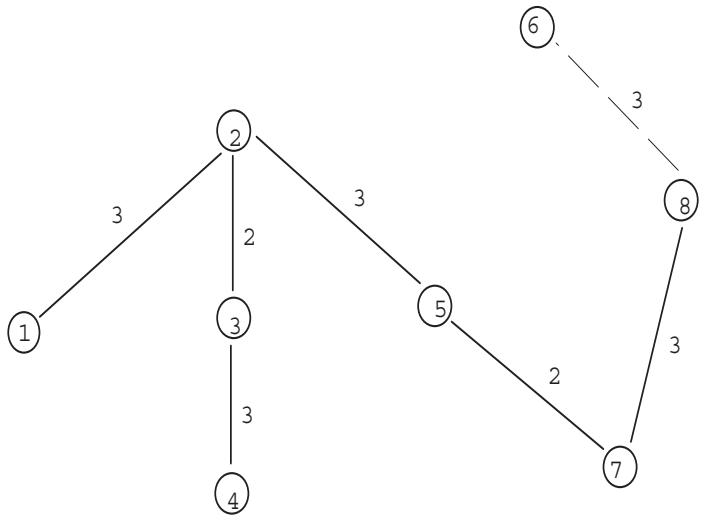
Step 5: Unconnected node nearest to (1,2,3,4,5) is node 7.(from 5)



Step 6: Unconnected node nearest to (1,2,3,4,5,7) is node 8. (from 7)



Step 7: Unconnected node nearest to (1,2,3,4,5,7,8) is node 6. (from 8)



Minimal spanning length = 3+2+3+3+2+3+3 = 19

Optimality Condition: Check each arc NOT included to see if any improvements can be made. For example, the arc from 1 to 4 creates a cycle on which the highest arc length is 7, which is the length of 1-4, so 1-4 cannot be used to reduce the sum of the spanning tree. Repeat for each of the other arcs NOT included in the solution tree to show that each of them should NOT be added to the spanning tree.

1.2 Shortest Route Problem

Given: A connected network with designated origin and destination nodes, in which each branch is given a distance dij.
Find: The shortest route from an origin to a destination through the network.

Shortest Route Solution Algorithm

In this algorithm "flag values" will be determined for each node, one at a time, that give the length of the shortest route from the origin up to that node. Hence we can Identify two sets of nodes:

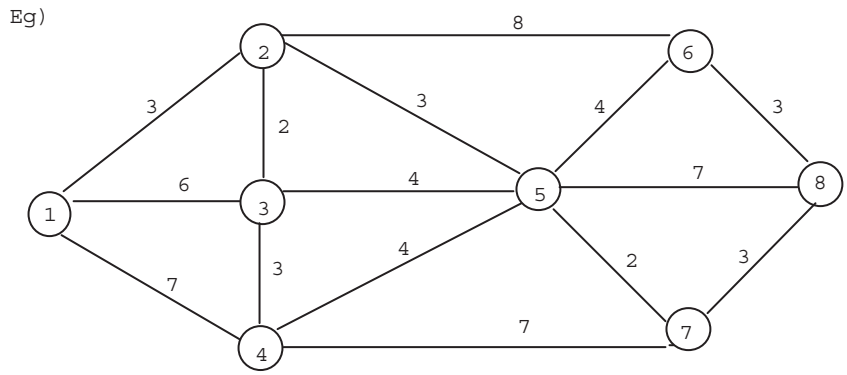
- Labeled Set - those with flags for which the minimum distance route has been found.
- Adjacent Set - nodes (without flags) connected to a node in the labeled set by a branch in the network.

The algorithm proceeds by "growing" a solution tree one branch at a time.

1) Initially, put the origin in the labeled set, with a flag value of 0;
2) Identify the set of arcs leading to nodes adjacent to the current labeled set;
3) Identify the arc leading to a node in the adjacent set with the minimal new flag value; the new flag value is given by the sum ("old flag value" + intervening arc value dij).
4) Save the connecting branch (including its direction) which leads to the selected adjacent node as a permanent arc in the solution tree. Delete from further consideration any other branches leading from the labeled set to the selected adjacent node.
5) Add the selected adjacent node to the labeled set, labeled with a flag value giving its minimum distance to the origin.

NOTE: This algorithm finds the minimum distance from the origin to every other node.

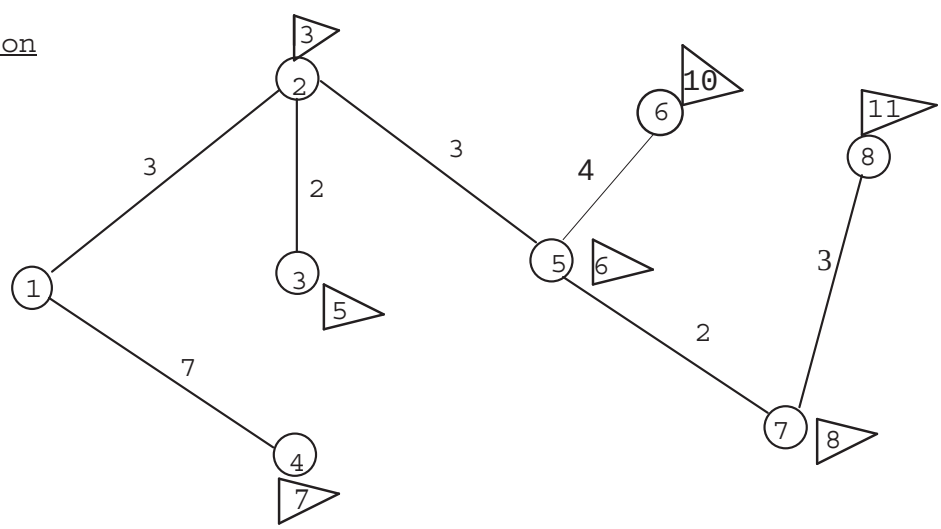
Eg)



Iter	Perm	Adj	Connecting Branch	Perm		d_{ij} Label		Total Dist to Source
1.	{1}	{2,3,4}	(1,2)	0	+	3	=	3 *
			(1,3)	0	+	6	=	6
			(1,4)	0	+	7	=	7
		Action:						
		Save (1->2).						
			Add node 2 to Labeled set with label = 3					
2.	{1,2}	{3,4,5,6}	(1,3)	0	+	6	=	6
			(1,4)	0	+	7	=	7
			(2,6)	3	+	8	=	11
			(2,5)	3	+	3	=	6
			(2,3)	3	+	2	=	5 *
		Action:						
		Save (2-->3), delete (1,3).						
			Add node 3 to Perm set with label = 5					
3.	{1,2,3}	{4,5,6}	(1,4)	0	+	7	=	7
			(2,6)	3	+	8	=	11
			(2,5)	3	+	3	=	6 *
			(3,5)	5	+	4	=	9
			(3,4)	5	+	3	=	8
		Action:						
		Save (2-->5), delete (3,5).						
			Add node 5 to Perm set with label = 6					
4.	{1,2,3,5}	{4,6,7,8}	(1,4)	0	+	7	=	7 *
			(2,6)	3	+	8	=	11
			(3,4)	5	+	3	=	8
			(5,6)	6	+	4	=	10
			(5,8)	6	+	7	=	13
			(5,7)	6	+	2	=	8
			(5,4)	6	+	4	=	10
		Action:						
		Save (1-->4), delete (3,4) & (5,4).						
			Add node 4 to Perm set with label = 7					
5.	{1,2,3,4,5}	{6,7,8}	(2,6)	3	+	8	=	11
			(5,6)	6	+	4	=	10
			(5,8)	6	+	7	=	13
			(5,7)	6	+	2	=	8 *
			(4,7)	7	+	7	=	14

		Action:						
		Save (5-->7), delete (4,7).						
			Add node 7 to Perm set with label = 8					
6.	{1,2,3,4,5,7}	{6,8}	(2,6)	3	+	8	=	11
			(5,6)	6	+	4	=	10*
			(5,8)	6	+	7	=	13
			(7,8)	8	+	3	=	11
		Action:						
		Save (5->6), delete (2,6).						
			Add node 6 to Labeled set with label =10					
7.	{1,2,3,4,5,6,7}	{8}	(5,8)	6	+	7	=	13
			(7,8)	8	+	3	=	11 *
			(6,8)	10	+	3	=	13
		Action:						
		Save (7-->8), delete (5,8) & (6,8).						
			Add node 8 to Labeled set and					
			stop: Solution tree obtained.					

Final Solution



Shortest route from node 1 to node 8 is 1-2-5-7-8 with length 11.

1.3 The Critical Path Method (CPM)

One of the most common applications of network models is project scheduling and project management. It is common practice to break a large project down into a large number of individual work steps or tasks that are well defined and have a logical sequencing relationship to the other tasks in the project.

There are two different conventions for representing projects as project networks, AOA (Activity on Arc) and AON (Activity on Node). Most modern computer programs use the AON convention since it leads to simpler computational algorithms and is easier to update and maintain. Hence it is the convention that will be used here.

In the AON model, each project activity is associated with a node, and the precedence relations between one activity and its immediate predecessors are represented as directed arcs.

1.3.1 AON NETWORK REPRESENTATION

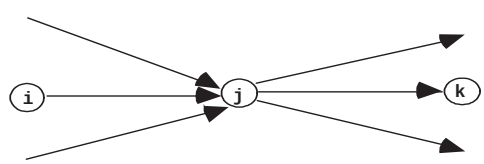
Activities <--> Nodes
Require time and utilize resources. Nodes represent individual tasks in the project.

Precedence Relations <--> Arcs
Arcs show activities which must be complete in order for another activity to begin.

Project Network
Represents a complete set of interrelated activities and precedence relationships.

Originating Event of Project = Start (BOP may be added to network if not already indicated)
Terminal Event of Project = End (EOP may be added to network if not already indicated)

Precedence Relationships



Activity J cannot begin until activity I is complete; similarly activity K cannot begin until activity J is complete.

Project Network - a network diagram with event nodes and precedence arcs arranged so as to represent the desired precedence relationships.

Rules for Diagramming a Project Network

1) Each activity is represented by a single node.
2) The project network has ONE beginning node (node 1) and ONE ending node (node N). If these are not already present, they may be added as "milestone" nodes with zero time duration at the beginning and end of the project. The beginning of project node, when added, will be referred to as the BOP, and the end of project node, when added, will be referred to as the EOP.
3) Numbering of nodes is such that the head of each precedence arrow has the higher number.

4) Directed arcs, or arrows, are inserted only to represent immediate predecessors; implied precedence need not be shown explicitly.
5) Nodes should be arranged to minimize the number of line crossings. Embedded "milestone" nodes may be positioned in the project when by doing so crossing lines are eliminated. Keep it Simple.

1.3.2 TIME CONCEPTS

Let di = estimated duration of activity (node) i

Critical Path - the path with the longest total time through the network from start to finish. A project may have several critical paths.

From the activity durations di, we can derive a set of times to associate with each node and a set of times to associate with each activity. There are the early start and early finish times (denoted ESi and EFi respectively) and the late start and late finish times (denoted LSi and LFi respectively). By an early time, we mean the earliest time that event or activity start or finish may occur without violating any of the precedence relationships that have been specified. In other word, all preceding activities must have been completed before a given activity can occur or begin. By a late time, we mean the latest time an event or activity start or finish can occur without causing a delay in the completion time of the project. With these definitions in mind, we can then introduce the following variables which will be used to determine the critical path(s) through the network, as well as the slack time available for each activity, if any.

→ ESj = early start time for activity (node) j
EFj = early finish time for activity (node) j

Note that EFi = ESi + di for all i. Also

ESj =MAXi { EFi: activity i precedes activity j}

Turning now to the late times, we define

→ LFj = late finish time for activity (node) i
→ LSi = late start time for activity (node) i

Note that LSi = LFi – di all i. Also

LFi = MINj { LSj: Activity j follows activity i}.

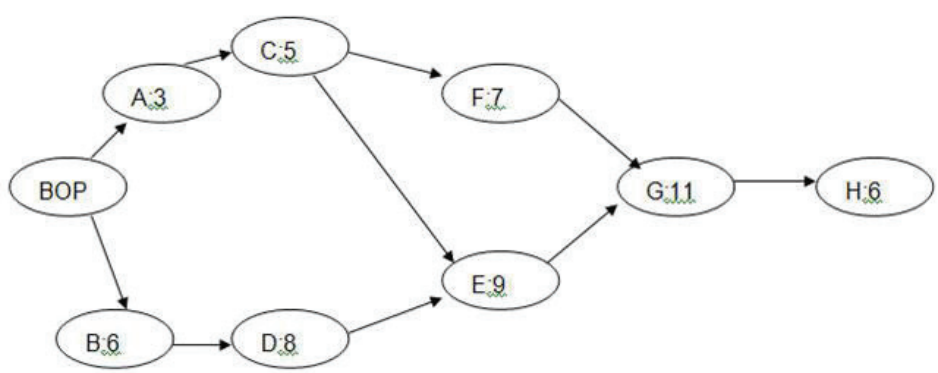
The algorithm for computing these times and hence the critical paths and slacks for non-critical activities proceeds in two passes: (1) a forward pass through the network to determine all of the early times, followed by (2) a backward pass through the network to determine all of the late times. With these in hand, computation of the slacks and determination of the critical path(s) is straightforward.

1.3.3 A NUMERICAL EXAMPLE

Suppose a project is characterized by the activity precedences and durations given in the following table.

Activity	Immediate Predecessors	Expected Duration
A	-	3
B	-	6
C	A	5
D	B	8
E	C, D	9
F	C	7
G	E, F	11
H	G	6

To draw the network corresponding to this table, we note that since neither A nor B have any predecessors, they can be done concurrently and hence insertion of the BOP milestone is called for. Similarly C follows A and D follows B and these may be done concurrently as well. Then F follows C and E follows C&D. Finally G follows E&F and H follows G. H is clearly the last node in the project, so no EOP is needed in this case.



FORWARD PASS

One sets the clock to 0 at the beginning of the project, thus ESi =0 for the initial node. Then one computes the early start and early finish times for the remainder of the nodes in the network by means of a recursion formula which involves the maximum of the finish times for activities preceding each node, as follows:

$$ES_j = MAX_i \{ EF_i : \text{activity i precedes activity j} \} = MAX_i \{ ES_i + d_i \}$$

Here it is understood that i ranges over all immediate predecessors of j. Thus the early start for activity at node j equals the maximum of the early finishes for all activities immediately preceding j. Iterating these formulas forwards gives the following results.

Node	$ES_j = Max_i \{ES_i + d_i\}=Max \{EF_i\}$	$EF_j = ES_j + d_j$
0	0	0+0=0
A	Max {0} =0	0+3=3
B	Max {0} =0	0+6=6
C	Max {3} =3	3+5=8
D	Max {6} =6	6+8 = 14
E	Max {8,14} =14	14+9 = 23
F	Max{8}=8	8+7=15
G	Max{23,15}=23	23+11=34
H	Max {34}=34	34+6=40

The early finish time at node H (the final node in the project), is the minimal project duration. Thus the project in this case will take at least 40 days to complete. These computations may also be done directly on the network. In this case, the ES times are usually shown above and to the left of the node, and the EF times are shown above and to the right of the node.

BACKWARD PASS

Since we do not want to allow any delay in the project at the final node, one sets the late time at N equal to the early time at H, i.e. LFH = EFH. Then late times are computed recursively backwards through the network by means of the following formula:

$$LF_j = MIN_k \{LS_k : \text{Activity k is immediately preceded by activity j }\}= MN_k \{LF_k-d_k\}$$

where it is understood that k ranges over all nodes such that j is an immediate predecessor of k, in other words, over all immediate successors of node j. Thus the late finish for any activity equals the minimum of the late starts for all activities following j. Repeated application of this formula in the backward direction through the network constitutes the "backward pass" and gives rise to a set of late times which can be used to compute slack times for each activity. For the present example the results are as shown below:

Node	$LF_j = Min_k \{LF_k - d_k\}=Min_k \{LS_k\}$	$LS_j=LF_j -d_j$
H	40	34=40-6
G	Min{34}=34	23=34-11
F	Min{23}=23	16=23-7
E	Min{23}=23	14=23-9
D	Min {14} = 14	6=14-8
C	Min{14-16}=14	9=14-5
B	Min {6}=6	0=6-6
A	Min {9}	6=9-3
0	Min {6,0} = 0	0=0-0

Again, these computations may be done directly on the project network. In this case, the late finish times are usually shown below and to the right of the node name, and the late start times are shown

below and to the left of the node name. Note that since we set the late finish equal to the early finish at the final project activity H, the late start time at the beginning node came out to be zero, as it must if there is to be no delay at the end.

SLACK TIMES

Activity (Node) Slack for node I is the amount of time activity I can be delayed without delaying the completion time for the project, assuming all other activities have started and will start at the earliest possible time, i.e.

Si = LSi –ESi =LFi -EFi

For the present example one obtains:

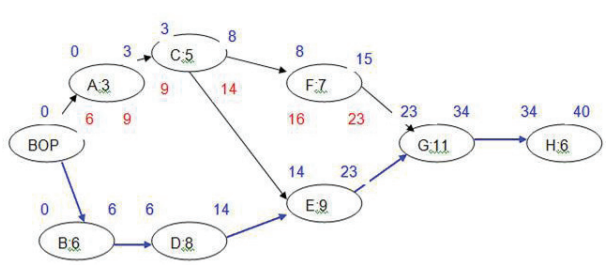
Node	$S_j = LS_j - ES_j$
A	6-0=6
B	0-0=0
C	9-3=6
D	6-6=6
E	14-14=0
F	16-8=8
G	23-23=0
H	34-34=0

Note: the node slacks for activities on a critical path will always be 0.

Critical Activities are those with zero slack (B, D, E, G, and H). Critical activities are never isolated, they always lay on a critical path.

A Critical Path is a path from project beginning to project end formed entirely by critical activities. Consequently, for the example at hand the critical path is B-D-E-G-H and project duration is 40

The tabular format for these computations is favored when working the problem in a spreadsheet. When working the problem by hand, it is much easier to depict and develop these results directly on the network diagram. In this case the ES and EF times are noted above the nodes and the LS and LF times are noted below the nodes, as in the following diagram. For brevity, late times are shown only where different from early times. Early times and Late times are always the same on the critical path, so slack is always zero there and late times are sometimes omitted. In this case, the non-critical activities are A, C and F; slack is 6 for A and C, but slack is 8 for F.



1.4 CPM in the Spreadsheet

In this section we present a way to implement the CPM in the spreadsheet environment that yields not only the critical path but also the length or duration of the critical path. The spreadsheet views presented below relate to the example of the previous section for which the solution has already been computed. The critical path is BDEGH and the CP duration is 40 days.

The first three columns in the CPM spreadsheet contain the data for the problem, namely, the letter name of the activities, the immediate predecessors for each activity, and the stated duration of the activities. The next two columns carry out the forward pass computations, and the next two after that carry out the backward pass computations. The formulas in the columns for the ES and LF quantities are based on the precedence relationships that have been given for the project activities. Finally, there are three more columns that compute slack, identify the critical activities, and compute the name of the critical path. The critical path results are shown at the bottom of the table.

Activity	Preds	Duration	ES	EF	LS	LF	SLACK	Crit(0/1)	CritAct
A		3	0	3	6	9	6	0	
B		6	0	6	0	6	0	1	B
C	A	5	3	8	9	14	6	0	
D	B	8	6	14	6	14	0	1	D
E	C, D	9	14	23	14	23	0	1	E
F	C	7	8	15	16	23	8	0	
G	E, F	11	23	34	23	34	0	1	G
H	G	6	34	40	34	40	0	1	H
CP		40							BDEGH

To see how the formulas for the forward and backward passes depend on the precedence structure, it is helpful to study the formula view of the spreadsheet.

	A	B	C	D	E	F	G
1	Activity	Preds	Duration	ES	EF	LS	LF
2	A		3	0	=SUM(C2:D2)	=G2-C2	=F4
3	B		6	0	=SUM(C3:D3)	=G3-C3	=F5
4	C	A	5	=E2	=SUM(C4:D4)	=G4-C4	=MIN(F6:F7)
5	D	B	8	=E3	=SUM(C5:D5)	=G5-C5	=F6
6	E	C, D	9	=MAX(E4:E5)	=SUM(C6:D6)	=G6-C6	=F8
7	F	C	7	=E4	=SUM(C7:D7)	=G7-C7	=F8
8	G	E, F	11	=MAX(E6:E7)	=SUM(C8:D8)	=G8-C8	=F9
9	H	G	6	=E8	=SUM(C9:D9)	=G9-C9	=E9
10	CP		=E9				

Note that there are three different cases that occur in the ES column, i.e. column D. If an activity has no predecessors (as for A and B) then the Early Start time is just 0. If an activity has only one predecessor (as for C, D, F and H) then the Early Start time is just the EF time of the preceding activity, which is

obtained by a reference to the EF column (which is column E). In those cases where an activity has more than one predecessor (as for E and G) the MAX() function is evaluated where the arguments are the EF times for the various predecessors in question. It is important to use formulas that refer to column E because if the activity durations change (as they will when we simulate the project later in this book) we want the forward pass to "recompute" based on the new durations, and not remain the same as for the previous durations.. The EF times in column E are uniformly computed as a simple summation of the ES time and the duration for each activity.

Note that in cell C10 there is a simple formula =E9 that is an example of what we call a "replication formula." It simply picks up a value that has already been computed in the spreadsheet and shows it again ("as is") in another cell. In the present instance, this is because the EF time of the last activity in the network (or the EOP in general) is itself the duration of the project. That is to say, cell C10 shows the duration of the critical path for the project, which is computed initially as the EF time for the last node in the network.

	H	I	J
1	SLACK	Crit(0/1)	CritAct
2	=F2-D2	=IF(H2=0,1,0)	=IF(I2=1,A2,"")
3	=F3-D3	=IF(H3=0,1,0)	=IF(I3=1,A3,"")
4	=F4-D4	=IF(H4=0,1,0)	=IF(I4=1,A4,"")
5	=F5-D5	=IF(H5=0,1,0)	=IF(I5=1,A5,"")
6	=F6-D6	=IF(H6=0,1,0)	=IF(I6=1,A6,"")
7	=F7-D7	=IF(H7=0,1,0)	=IF(I7=1,A7,"")
8	=F8-D8	=IF(H8=0,1,0)	=IF(I8=1,A8,"")
9	=F9-D9	=IF(H9=0,1,0)	=IF(I9=1,A9,"")
10			=CONCATENATE(J2,J3,J4,J5,J6,J7,J8,J9)

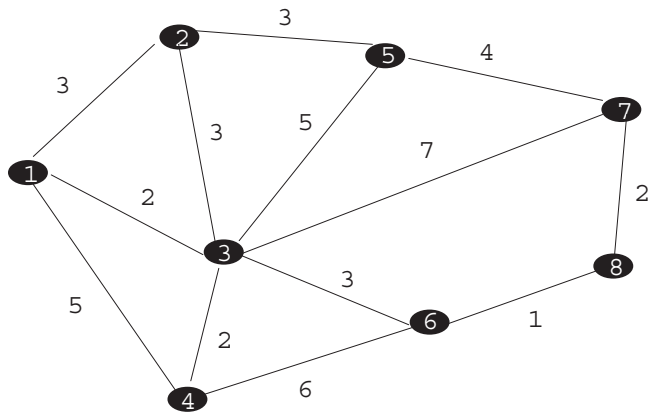
Moving over to the LF column, we again find three cases. By definition, the LF time is equal to the EF time for the last node in the network, so there is a reference to column E in the row for activity H (this is another "replication formula" based on the result in E9). In those cases where an activity has only one succeeding activity there is a reference to column F to obtain the LS time for the single following activity. In those cases where there are two or more successors, the MIN() function is invoked to compute the minimum of the following LS times. Once again it is important that all of these cells contain formulas since the values obtained may change when the activitiy durations change. The LS times are uniformly computed as a simple difference of the LF time minus the duration of each activity.

Panning to the right three columns, we see the formulas that are used to identify the slacks and the critical path for the project. In column H the early start is subtracted from the late start to obtain the slack for each activity. Then in column I a criticality flag is set to "1" if the activity is critical (slack is zero) or to "0" if the activity is non-critical (slack is positive). Finally, in column J the IF statements show the name of the activity from column A if the criticality flag is a "1" and leave the field blank if the activity is non-critical. Then at the bottom of column J, the names of the critical activities are put next to each other in a single string by use of the CONCATENATE function. For some reason, it does not work to put the argument in as J2:J9, but if you list the cells to be concatenated individually with commas between the arguments, it works fine.

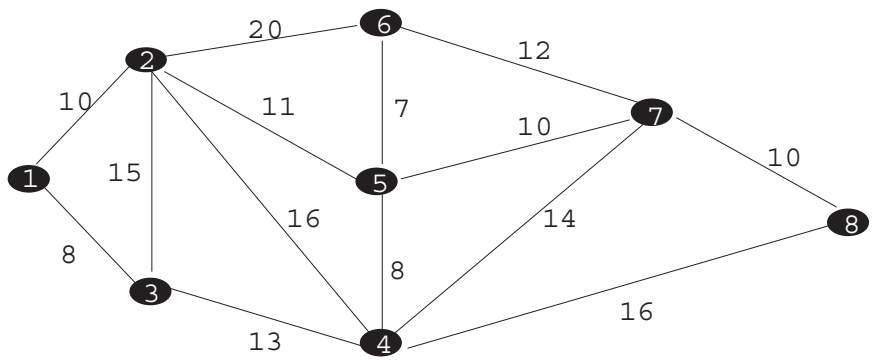
1.5 Practice Problems

Minimal Spanning Trees Find the minimum spanning trees for the networks below. Then rework to find the **shortest route spanning trees** starting from node 1 in each case.

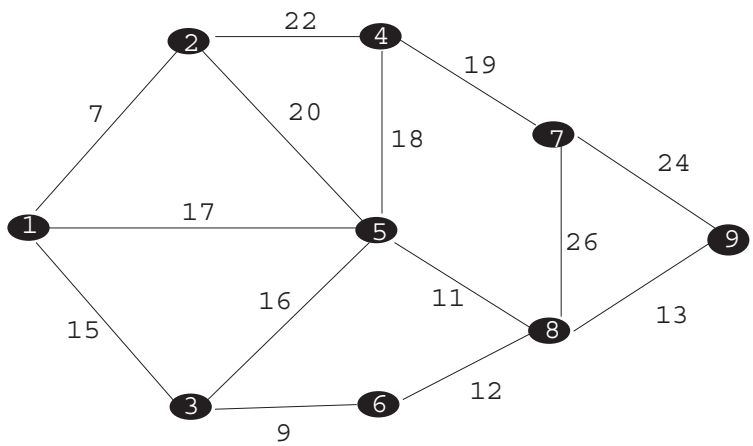
1.



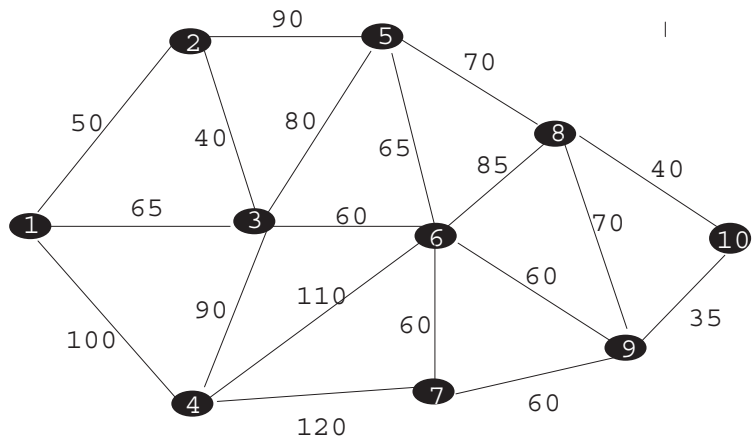
2.



3.



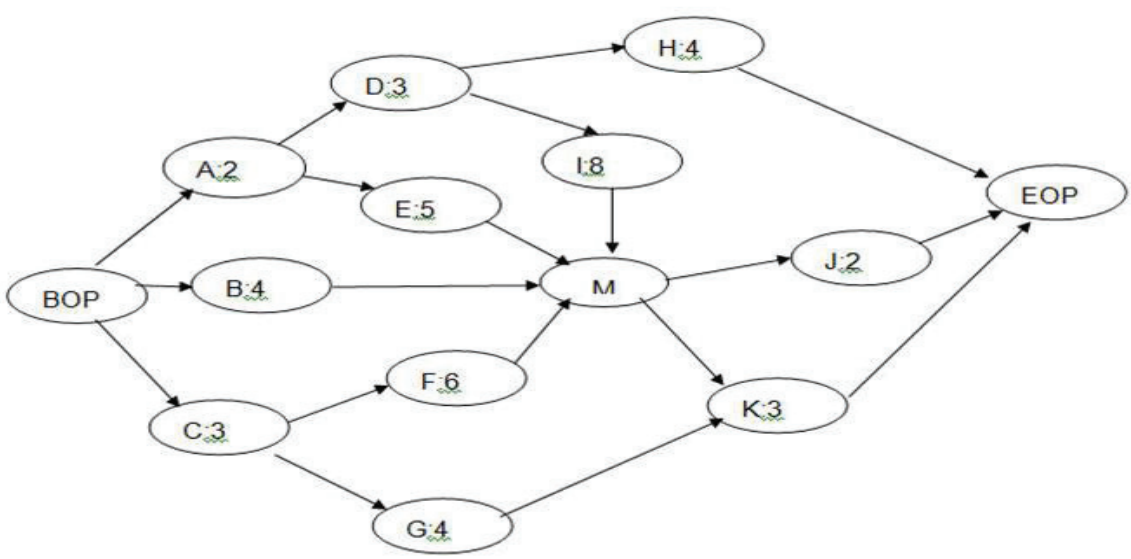
4.



CPM PROBLEMS

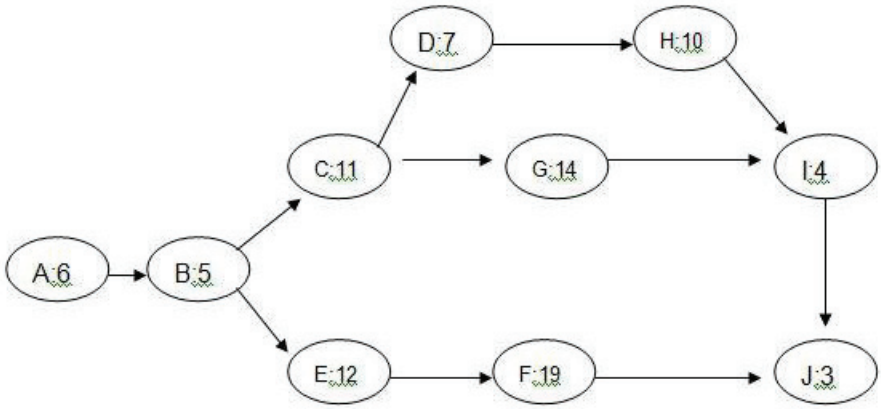
In the project networks shown below, for problems 1-3, find (a) the ESj, EFj, LFj and LSj for each activity node; (b) the critical path and its duration; and (c) the slack on all milestones and activities.

CPM1

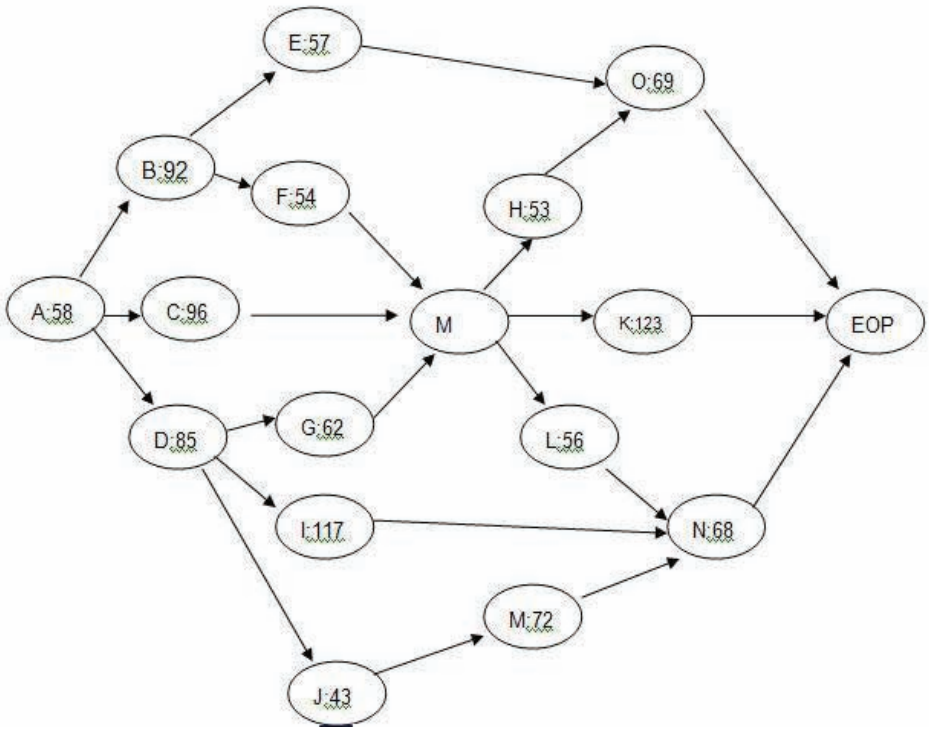


Note: There are tree milestone nodes in this project network, all of which have zero activity durations. BOP and EOP are the "beginning-of-project "node and "end –of-project" nodes of the project respectively; M is an intermediate milestone introduced solely to simplify the network diagram. Note that activity j depends on M being complete, whereas K depends on both M1 and G.

CPM2


D:7
H:10
C:11
G:14
I:4
A:6
B:5
E:12
F:19
J:3

CPM3


E:57
O:69
B:92
F:54
H:53
A:58
C:96
M
K:123
EOP
D:85
G:62
L:56
I:117
N:68
M:72
J:43