ASSUMPTION UNIVERSITY (ABAC), BANGKOK

มหาวิทยาลัยอัสสัมชัญ

DEVELOPMENT OF A COMPUTER SOFTWARE FOR PROJECT SCHEDULING WITH CRITICAL PATH METHOD (CPM)

by

Mr. Waha Junsangsuk

A Final Report of the Six-Credit Course
CE 6998 - CE 6999 Project

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in Computer and Engineering Management
Assumption University

July 2002

DEVELOPMENT OF A COMPUTER SOFTWARE FOR PROJECT SCHEDULING WITH CRITICAL PATH METHOD (CPM)

by
Mr. Waha Junsangsuk

A Final Report of the Six-Credit Course
CE 6998 – CE 6999 Project

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in Computer and Engineering Management
Assumption University

July 2002

Project Title	Development of a Computer Software for Project Scheduling with Critical Path Method (CPM)
Name	Mr. Waha Junsangsuk
Project Advisor	Rear Admiral Prasart Sribhadung
Academic Year	July 2002

The Graduate School of Assumption University has approved this final report of the three-credit course, CE 6998 PROJECT, submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer and Engineering Management.

Approval Committee:

[illegible]
(Rear Admiral Prasart Sribhadung)
Advisor

[illegible]
(Prof.Dr. Srisakdi Charmonman)
Chairman

[illegible]
(Dr. Chamnong Jungthirapanich)
Dean and Co-advisor

[illegible]
(Assoc.Prof. Somchai Thayarnyong)
MUA Representative

July 2002

ABCTRACT

The purpose of this project is to study the using of the Critical Path Method (CPM) for decision making to choose the path of production process. The production process will depend on the time of each activity that will show all activities on Gantt Chart. This Gantt Chart will show every thing about production process. However, there are many factors to do in a production process.

ACKNOWLEDGEMENTS

The author wishes to take this opportunity to thank Rear Admiral Prasart Sribhadung, the advisor of this project, for his comments and recommendation for this course work.

The author would like to thank all instructors of the Graduate School of Computer & Engineering Management of Assumption University for providing their valuable knowledge during his study at Assumption University.

Finally, the author wishes to thank his family for their encouragement throughout this course.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

I. INTRODUCTION

1.1 Background of the Project

A project constructing the World Trade Center or sending a man to the moon - shares many characteristics with other types of production processes. We can picture a project with the same kind of precedence relationship with which we pictured the line flow process. Certain activities must be completed before others begin, and each activity can be expected to take a given period of time, just as in other process types. But our concern for balance in the factory, so prominent in our thinking about production processes, does not trouble us in thinking about a project since a project is, by definition, a one-time endeavor and workers can generally expect to work only so long on it and then move on to something else. It is typical for a project to employ wildly fluctuating numbers of people, many with different skills. We do not worry about idleness - construction workers move on to the next project as do space scientists and engineers.

What does trouble us in a project is getting it done on time, for projects often have important deadlines to meet. Scheduling is thus absolutely critical to the management of a project. Several techniques have been developed to highlight the project activities that must be accomplished on time (or else risk delaying the entire project) and the activities that can be delayed somewhat. One such scheduling technique is called the critical path method; what follows is a brief description of its rationale and workings.

1.2 Objectives

(a) Developing computer software to calculate Critical path and table

1.3 Scope and Limitation

(a) Developing computer software to calculate Critical path and table using CPM for choosing a critical path.

(b) Show Gantt Chart that contains all of descriptions.

1.4 Deliverables

(a) Project report

(b) Computer software for project scheduling with critical path method (CPM)

II. LITERATURE REVIEW

2.1 The Critical Path Concept

As an illustration, let us consider the simplified example of a project many companies face - the startup of a new plant. The major tasks involved in a plant startup are arrayed and described in Table 2.1. The table makes clear that some tasks cannot be started until other tasks (predecessors) have been completed.

The precedence relationships of the startup can be depicted in a graph or network, as shown in Figure 2.1. This network follows a particular convention known as activity-on-node. This convention is intuitively appealing and somewhat less confusing than the alternative convention, known as activity-on-arrow.

The key principle of the critical path concept is that a project cannot be completed any faster than the longest time path between the project's start and its finish. In Figure 1 the longest path from the start (project approval) to the finish (plant startup) involves the following activities - B, E, H, and K; they take a total of 17 months. The longest path between project approval and plant startup is termed the critical path, primarily because any delay along this path of activities sets back the entire project. It is this critical path that merits the most management attention.

All the other paths from start to finish enjoy at least a month's slack time, and so they can experience delays of varying lengths and still not harm the 17-month completion schedule. For example, the next two longest paths after the critical path involve the following activities:

(1) B-E-H-L 16 months

(2) A-D-G-K 16 months

Table 2.1. A Plant Startup Project.

Job Name	Job Description	Immediate Predecessors	Time (months)
A	Selection of plant manager	-	3
B	Site survey soil test	-	1
C	Extension of roads, water, utilities, sewer	-	6
D	Selection and purchase of equipment	A	2
E	Final engineering plans for construction	B	3
F	Employment interviews and hiring	A	3
G	Equipment delivery	D	9
H	Construction of facility	E	11
I	Precise layout of plant	D,E	1
J	Institution of management system	A	4
K	Worker training	F,G,H	2
L	Equipment and system installation	G,H,I,J	1

There can be delays of up to a month on these two paths without setting back the entire project. However, note that the first path is very much like the critical path itself; it differs only in that activity L is substituted for activity K. The slack in this path then can be taken only on activity L (equipment and system installation). If a delay occurred on any other activity, say H (construction of the facility), the critical path would also be affected, and the entire project would be delayed.

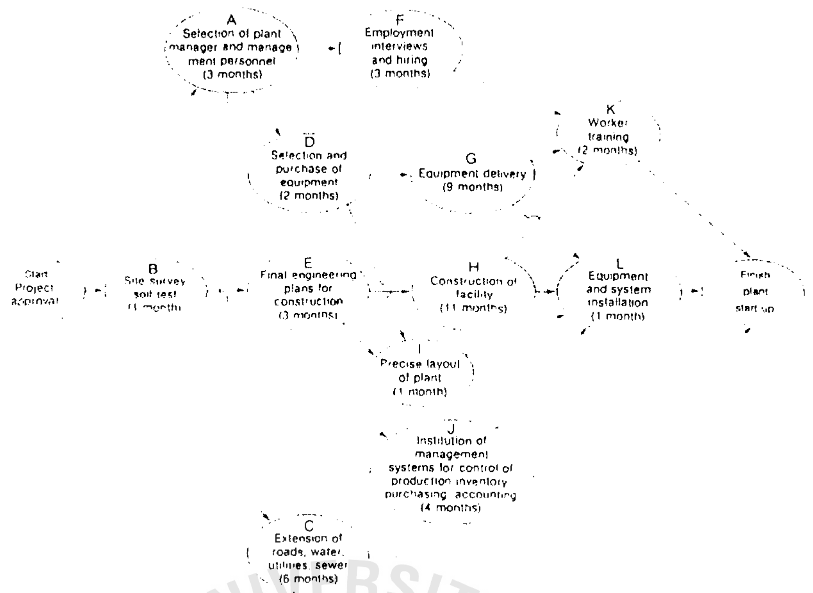

A
Selection of plant manager and management personnel (3 months)
F
Employment interviews and hiring (3 months)
K
Worker training (2 months)
D
Selection and purchase of equipment (2 months)
G
Equipment delivery (9 months)
Start Project approval
B
Site survey soil test (1 month)
E
Final engineering plans for construction (3 months)
H
Construction of facility (11 months)
L
Equipment and system installation (1 month)
Finish plant start up
I
Precise layout of plant (1 month)
J
Institution of management systems for control of production inventory purchasing accounting (4 months)
C
Extension of roads, water, utilities, sewer (6 months)

Figure 2.1. Precedence Diagram for a Plant Start-Up Project.

The second path is much more flexible about where its slack can be used up. Only one of its activities, K (worker training), is shared with the critical path. Thus its month of slack can be used up on activities A, D or G without affecting the critical path in any way.

Other paths, of course, have considerably more slack and can be accomplished at a more leisurely pace, if need be. For example, the uppermost path in the diagram (A-F-K) is expected to take 8 months, with only activity K on the critical path. Activity F could be delayed as much as 9 months without making the project late. However, activity A could not be delayed that long, since it is on one of the second longest paths and can be delayed only a month before it would affect the project's completion time.

This example raises an important point about critical paths and project scheduling. As time wears on in a project's life, delays and even some speed-ups can be expected. What may look like the critical path at a project's start (such as B-E-H-K) may not remain the critical path if many delays (or speed-ups) strike the project. This means that the project manager should periodically recalculate the critical path to check that he or she is focusing on the activities that really matter to finishing a project in the least time.

2.2 Gantt Charts and Critical Paths

Gantt charts can capture some of what the critical path gives us, but not all. A Gantt chart of the table above could be constructed. (See Tour J for both a Gantt chart and a precedence diagram of the same project steps.) Whereas the precedence characteristics can be depicted, and some feel for the slack in the system can be shown, the links between activities are missing and the critical path itself is not evident from the diagram itself; it must be imposed on it from an outside calculation.

2.3 Calculating the Critical Path

To this point we have calculated the critical path solely by inspecting all the paths and choosing the one with the longest duration. In simple cases, such as this generalized plant startup, inspection is a perfectly feasible and reasonable way of selecting the critical path. When projects get more complicated, with many more activities, inspection as a means of computing the critical path bogs down. Happily there is an alternative procedure or algorithm, which can cut down the speed of solution markedly.

In essence, the critical path algorithm is a procedure to identify which activities have some slack time and which do not. Those with no slack time, of course, make up the critical path or paths.

Before describing the algorithm, however, it is helpful to define some terms:

Early Start (ES) - The earliest time a job can begin, which is the latest time that any of the job's predecessors are finished.

Early Finish (EF) - The earliest time a job can finish, which is the early start time plus the time it takes to do the job.

For any job i, the early start and early finish can be represented symbolically.

$$ES(i) = \max\ [EF\ (\text{all of i's predecessors})]$$

or the start time of the project if we are considering the beginning jobs.

$$EF(i) = ES(i) + t(i)$$

where t(i) is the time to perform job i.

Late Start (LS) - The latest a job can start without causing the entire project to be completed any late which is the late finish time less the time for the job.

$$LS(i) = LF(i) - t)i)$$

Late Finish (LF) - The latest time a job can finish, which is the earliest time that any of the job's successors have to be started. These can be represented symbolically, as well, for any job i.

$$LE(i) = \min\ [LS(\text{all of i's successors})]$$

or the earliest finish time for the project if we are considering the ending jobs.

Total Slack (TS) - The difference between the early and late starts for a job, or the early or late finishes. Total slack is the time a particular job could be delayed without delaying the completion of the project. Symbolically,

$$TS(i) = LS(i) - ES(i) = LF(i) - EF(i)$$

The critical path algorithm involves calculating ES, EF, LS, and LF for all the jobs in the network and then comparing them to discover which nodes have zero total slack (Or a minimum total slack ii there is some discretion about the completion time possible for the project. Note the later discussion of windows of time). These nodes are the ones comprising the critical path(s). The calculation and comparison involve three steps:

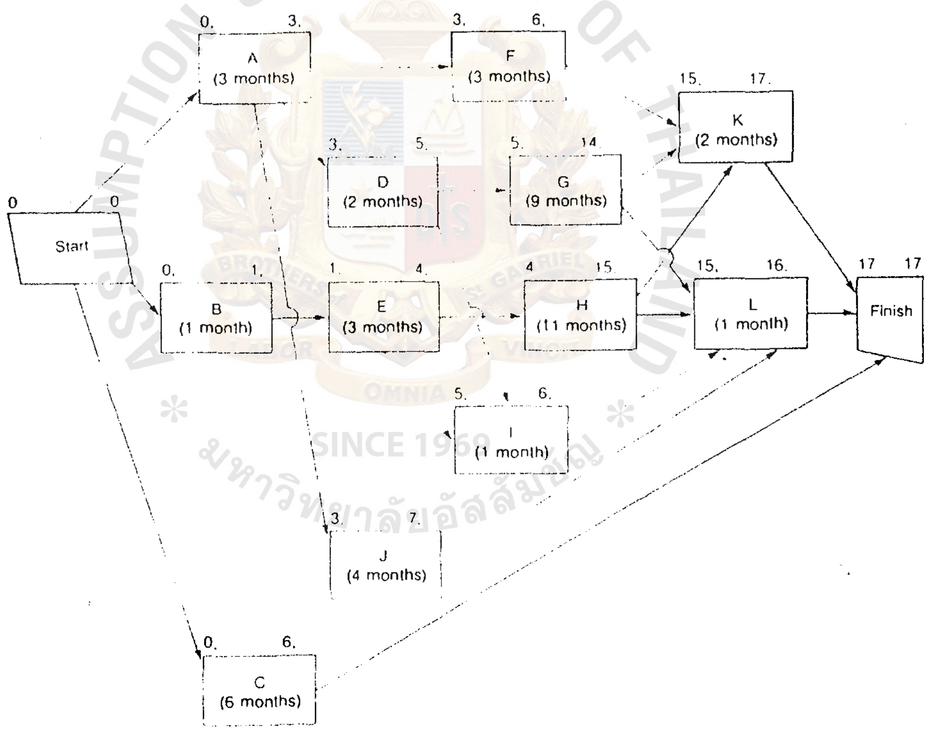


Figure 2.2. Early Start (Number at Upper Left of Box) and Early Finish (Number at Upper Right of Box) Months for the Plant Startup Project Delicate in Figure 2.1.

(1) Forward pass. The forward pass through the network calculates the early start (ES) and early finish (EF) times. It does so by moving from start to finish, figuring first the early start time and then adding the job's duration to it to figure the early finish. The early start for any job is calculated by scanning all of its immediate predecessor activities (those with arrows pointing into it in the path) and choosing as the job's early start the latest of all the predecessors' early finish times. The ES and EF times for our plant startup example are noted in Figure 2.3.

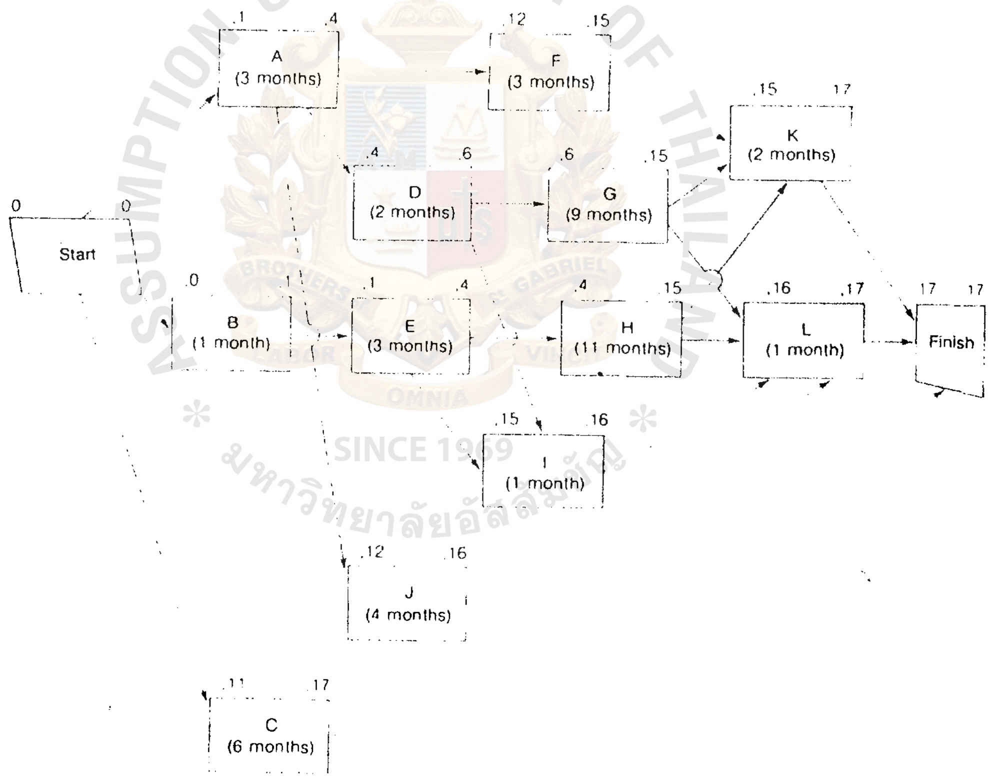


Figure 2.3. Late Start (Number at Upper Left of Box) and Late Finish (Number at Upper Right of Box) Months for the Plant Startup Project Depicted in Figure 2.1.

(2) Backward pass. The backward pass through the network calculates the late start (LS) and late finish (LF) times. It does this by moving from the finish back to the start, figuring first the late finish times and then deriving the late start times by subtracting the job's duration from the late finish time. The late finish for any job is calculated by scanning all of its immediate successor activities (those with arrows leading away from it on the path) and choosing as the job's late finish the earliest of all the successors' late start times. The LF and LS times for our plant startup example are noted in Figure 2.3.

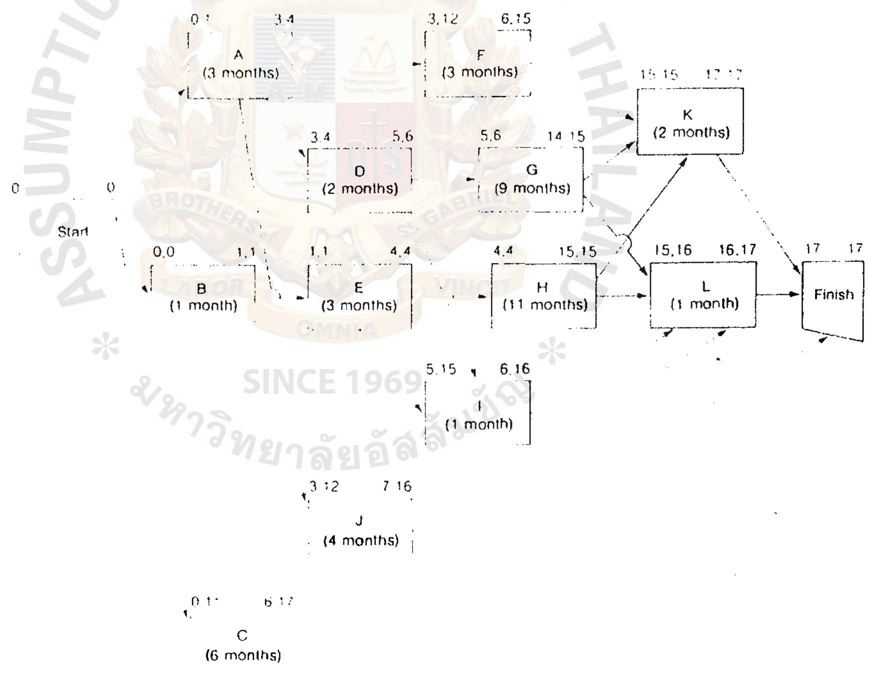


Figure 2.4. Composite of Starting and Finishing Months Given in Figure 2.2 and Figure 2.3 for the Plant Startup Project.

(3) Comparison. The comparison of either early start with early finish or late start with late finish is the way to figure whether the job has any total slack (TS) associated with it. If the LS(i) - ES(i) or LF(i} - EF(i) calculations are zero, the job has no total slack and is on the critical path. Figure 2.4 combines all the times in one diagram and displays the critical path.

For very large projects, of course, and ones that go into significant detail, even following the critical path method can become tedious. To aid in the analysis, there are computer programs that calculate the critical path. These can be used with ease to determine how differences in expected completion times for various activities (and even the introduction of variations in the network of activities itself) affect the critical path (s). Because so many things can go awry in a project, it is advantageous to compute the critical path on a regular basis so as not to be caught napping.

2.4 Project Costs and Timing and the Critical Path

Frequently, actual projects (such as the new plant startup outlined in Figure 2.1) need not be finished in the least time possible but, rather, can be finished during any portion of a "window" of time. When during the acceptable window of time the project's finish ought to be targeted depends largely on the project's managers. Often they are influenced by cost considerations, because it is not uncommon for a project's contract to stipulate rewards for an early finish and penalties for a late finish. These rewards and penalties must be weighted against the costs that hurrying up any of the project's activities could impose on the project's budget.

2.5 Windows of Time and the Critical Path

Windows of time can be easily incorporated into the calculation of the critical path; in fact, the terminology is already in place to handle them. The acceptable window of time can be denoted as the difference between the early finish and the late finish (or between the early start and the late start). To return to our example, if the plant startup could be completed between 17 and 21 months from the start, the values for Figure 2.4 would be recast as in Figure 2.5.

Building a window of time in the critical path computation makes it important to distinguish between two different kinds of slack time during the project: total slack and free slack. Total slack, as discussed previously, is the difference between the early and late finishes (or early and late starts) for any activity.

An activity's total slack time is the maximum time that activity may be delayed beyond its early start without forcing the delay of the entire project. As previously noted, this notion of total slack can be used to define the critical path; the critical path is composed of those activities that have the lowest total slack (the smallest differences - possibly zero - between early and late starts or finishes).

Free slack, on the other hand, is the time by which an activity can be delayed without delaying the early start of any other activity. Free slack is defined as the difference between an activity's early finish time and the earliest of the early start times for all its immediate successor activities. Symbolically, the free slack (FS) for any job i can be defined as

$$FS(i) = \min [ES \text{ all of i's immediate successors}] - EF(i)$$

Consider the path A-J-L in Figure 2.5. Activity L can be delayed at most a month without risking the delay of the entire project; as it is, its early finish time (16 months) is but 1 month shy of the project completion's early start month (17 months). Its free slack is 1 month. Activity J, on the other hand, has more free slack. Its early finish time of 7 months is 8 months shy of its only successor activity's (L's) early start time of 15 months; its free slack is thus 8 months. Activity A, the predecessor to activity, has no free slack because its early finish time (3 months) is equal to the early start time of activities D, F and J.

This lack of any free slack does not mean, however that if activity A were inadvertently delayed, the entire project would be delayed. Activity A, after all, does not lie on the critical path. It has a positive total slack and thus could be delayed by one month without necessarily delaying the project. It is often useful to employ a network-based Gantt chart as an aid in tracking and scheduling non-critical activities.

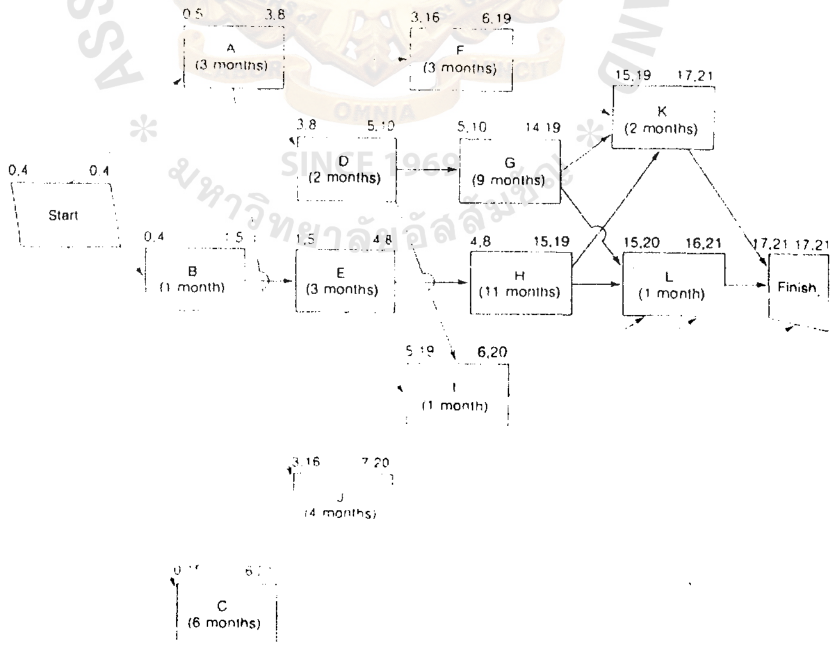


Figure 2.5. The Addition of Four Months (a Window of Time) to the Plant Startup Project.

Such charts usually display two panels, one depicting all activities at their earliest starts and the other depicting all activities at their latest starts. Network-based Gantt charts can also include data on manpower schedules per time period and cost incurred per time period. Figure 2.6 refers back to Figure 2.4's network where there is no window of time for the completion of the project. Note how the activities on the critical path coordinate with one another on the chart.

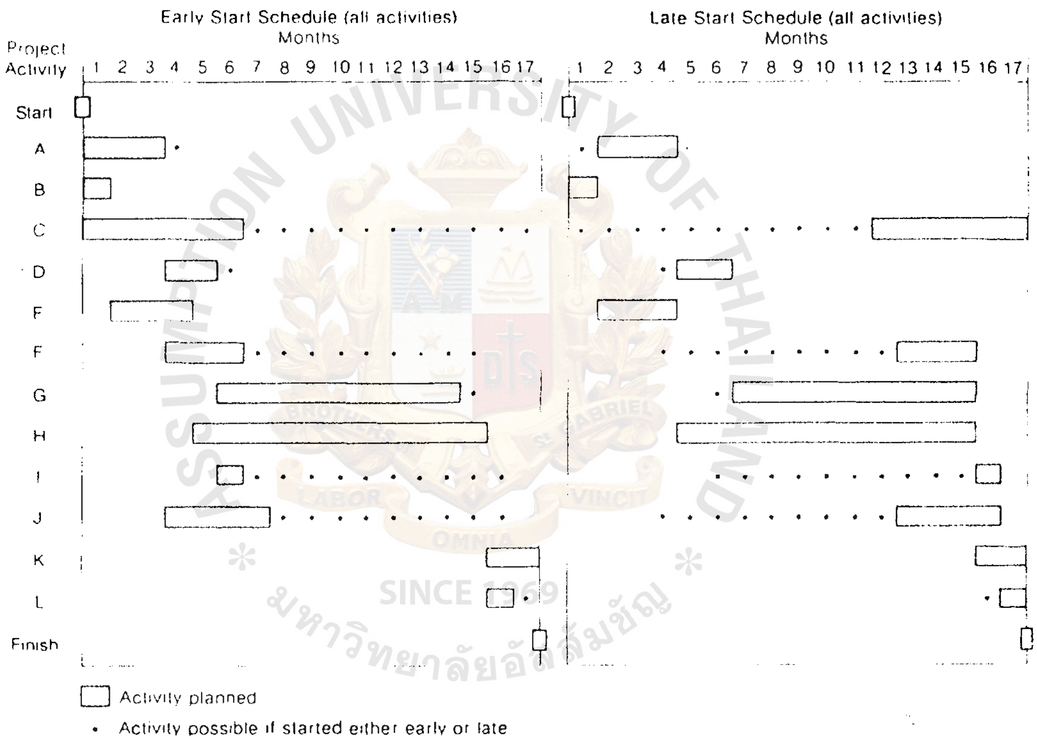


Figure 2.6. A Network-Based Gantt Chart for the Startup Project.

2.6 Time Estimate Variations

CPM can be used to provide some insight into likely variations in the completion time of the project. Individual estimates (an optimistic one, a pessimistic one, and a most likely one) of the variance in performing each activity are made. This is accomplished by assuming a certain distribution for these variances - the beta distribution (a skewed distribution that permits the possibility of occasional very late events, but no very early events - and by insisting on the independence of each activity from others. An estimate of the time variance in accomplishing the entire project (and thus the probability for completing it by various dates) can be made by summing the variances along the critical paths. The assumptions ensure that the distribution of the summed variances will be normal.

The expected value of the time for each activity is where oe is the optimistic estimate, pe is the pessimistic estimate, and ml is the most likely estimate. The variance of the time for each activity is:

$$T = \frac{oe + 4ml + pe}{6}$$

$$O^2T = \left(\frac{(pe - 0e)}{6}\right)^2$$

It is the sum of these variances along any path in the network that is normally distributed. By using the summed variance for the path (e.g., the critical path) and the fact that the summed variance is distributed normally, a probability of completion of any event can be computed.

As may be surmised, using CPM in this way to calculate a project completion time variance and probabilities of completion by specified dates can be a bit concocted. The variance for the entire project can be only as good as the variance estimates for individual activities, and often there is little information on which to base such variance estimates. Using CPM in this way can be more trouble than it is worth. Nevertheless, CPM does make explicit a concern for time variations and the fact that the critical path itself cannot be known with certainty. These insights are useful reminders to any project manager.

2.7 Some Principles for Project Managers

Although critical paths, the critical path method, and PERT have been exceedingly useful to managers in scheduling projects, effective project management means more than the adept use of these techniques. Drawing on his own vast experience in project management, Herbert Spirer has arrived at a dozen principles that serve as a useful checklist for project managers:

(1) A project has to have a clear objective or set of objectives and these objectives ought to be stated in terms of specific items (tangible or intangible) to be delivered to the project's sponsor.

(2) The project manager ought to be able to structure the project clearly, detailing which bundles of major activities (work packages) are required and how they comprise the deliverable items that meet the project's objectives. Spirer calls this a "work breakdown structure," and he sees it as a hierarchical representation of the project.

(3) The work packages making up the work breakdown structure are composed of individual activities. These activities have to be listed and organized into groups that can be overseen by specific people. The specific activities for a major project can number in the thousands. Spirer recommends, how-ever, that no one manager have direct responsibility for more than 50 specific activities. Top-level management itself should not be responsible for more than 30 to 50 aggregations of these activities, a precept that highlights the need for an effective work breakdown structure.

(4) These activities should then be placed in a precedence diagram or network, and a critical path developed with the concomitant early and late start and finish times and slack calculations. The development of the precedence diagram or network is an important task in itself; analyzing such a network can lead to various suggestions or to resequencing activities. Calculations of the critical path need to be done periodically as the project proceeds.

(5) The calendar of activities and times that comes out of the network creation and the critical path analysis must be combined with some knowledge of resource constraints to produce a schedule that tells management when activities will be done, not simply when they must be done.

(6) Managing the resources needed to meet the schedule is eased by making use of Gantt charts and plots of cumulative resource use such as are applied in manufacturing situations.

(7) Activity times can be estimated often by analyzing similar activities and modifying their times by using common sense. Sometimes more analytic methods (regressions, cross tabulation) can also be used. The estimation of activity time is often helped by referring explicitly to the network and the critical or near-critical paths that are implied by the first-round estimates.

(8) Use pessimistic, optimistic, and most likely estimates for as many activities as possible. It is always helpful to recognize the uncertainty inherent in projects.

(9) Tracking the progress of a project is helped by establishing milestones, which are easily grouped events. Such milestones can help serve as motivation for the project as well as checkpoints for the project's progress.

(10) Assign one and only one person to be accountable for every activity.

(11) Use the plan to control the project. This is done by keeping the plan current and monitoring the actual performance in timely fashion. Determine how the actual performance measures up to the plan and what variances are implied by that contrast. Take any corrective action that is necessary, assessing the likely consequences and adjusting the plan accordingly.

(12) Assess the status of the project by using various "earned value concepts" such as the budgeted cost of work scheduled, the budgeted cost of work performed, and the actual cost of work performed. By comparing the budgeted cost of work scheduled against the budgeted cost of work performed, one can come up with a general measure of the on-schedule performance of the project. Similarly, by comparing the budgeted cost of the work performed with the actual cost of the work performed, a measure of the cost variance of the project can be ascertained.

III. THE NETWORK DIAGRAM

3.1 The Network Diagram

CPM and PERT were developed independently during the late 1950s. PERT evolved through the joint efforts of Lockheed Aircraft, the U.S. Navy Special Projects Office, and the consulting firm of Booz, Allen & Hamilton in an effort to speed up the Polaris missile project. At the time, there was considerable concern on the part of the U.S. government that the Soviet Union might be gaining nuclear superiority over the United States, and early completion of the project was given top priority by the Department of Defense. The project was a huge one, with over 3,000 contractors involved, and many thousands of activities. The use of PERT was quite successful: PERT is generally credited for shaving two years off the length of the project. Partly for that reason, PERT or some similar technique is now required on all large government projects.

CPM was developed by J. E. Kelly of the Remington Rand Corporation and M. R. Walker of Du Pont to plan and coordinate maintenance projects in chemical plants.

Although these two techniques were developed independently, they have a great deal in common. Moreover, many of the initial differences between the two techniques have disappeared as users borrowed certain features from one technique for use with the other. For example, PERT originally stressed probabilistic activity time estimates, because the environment in which it was developed was typified by high uncertainty. In contrast, the tasks for which CPM was developed were much less uncertain, so CPM originally made no provision for variable time estimates.

At present, either technique can be used with deterministic or probabilistic times. Other initial differences concerned the mechanical aspects of developing project networks. However, from a conceptual standpoint, most of these differences were relatively minor. To avoid confusion, we will not delve into the differences here. For practical purposes, the two techniques are the same; the comments and procedures described will apply to CPM analysis as well as to PERT analysis of projects.

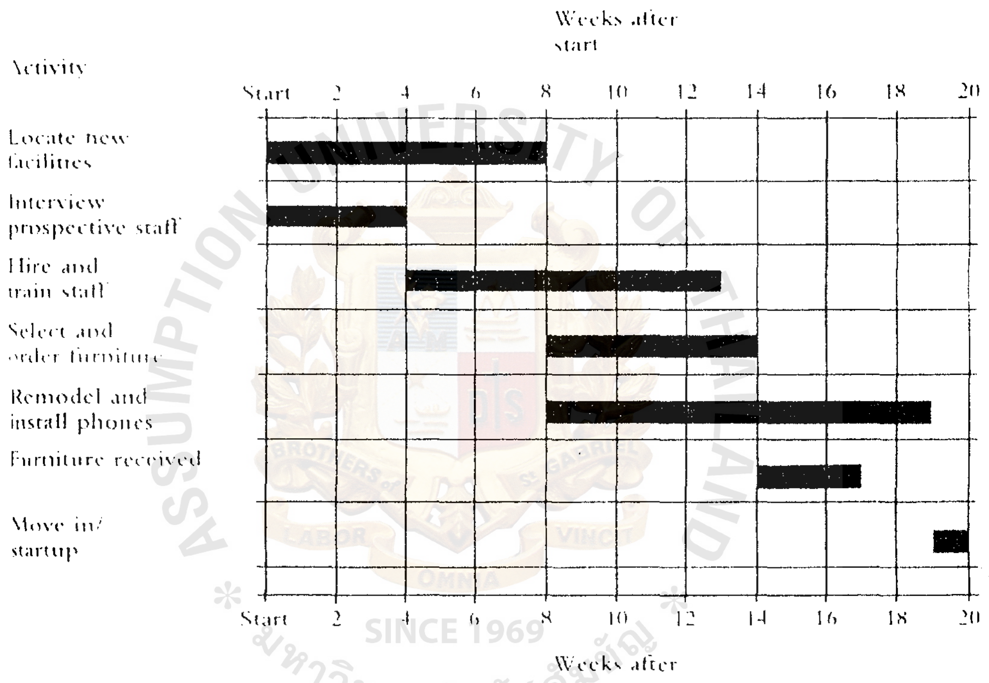


Figure 3.1. Gantt Chart for Bank Example.

One of the main features of CPM and related techniques is their use of network or precedence diagram to depict major project activities and their sequential relationships. Recall the bank example that used a Gantt chart (see Figure 3.1). A network diagram for that same problem is shown in Figure 3.2. The diagram is composed of a number of arrows and nodes. The arrows represent the project activities. Note how much clearer the sequential relationship of activities is with a network chart instead of a Gantt chart. For instance, it is apparent that ordering the furniture and remodeling both require that a location for the office have been identified.

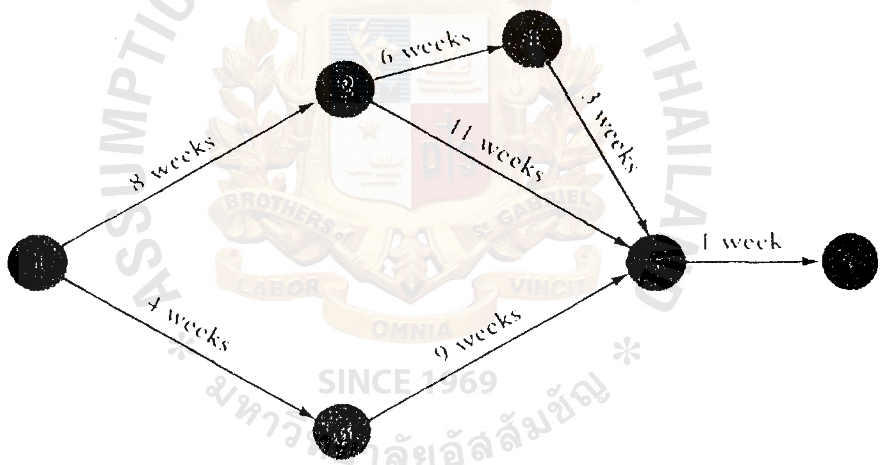


Figure 3.2. A Simple Project Network Diagram.

Likewise, interviewing must precede training. However, interviewing and training can take place independently of activities associated with locating a facility, remodeling, and so on. Hence, a network diagram is generally the preferred approach for visual portrayal of project activities.

You should know that there are two slightly different conventions for constructing these network diagrams. Under one convention, the arrows are used to designate activities; under the other convention, the nodes are used to designate activities. These conventions are referred to as **activity-on-arrow (A-O-A)** and **activity-on-node (A-o-N).** To avoid confusion, the discussion here will focus primarily on the activity-on-arrow convention. Then, later in the chapter, a comparison of the two conventions will be given. For now, we shall use the arrows for activities. Activities consume resources and/or time. The nodes in the A-O-A approach represent the starting and finishing of activities, which are called events. Events are points in time. Unlike activities, they do not consume either resources or time.

Activities can be referred to in either of two ways. One is by their endpoints (e.g., activity 2-4) and the other is by a letter assigned to an arrow (e.g., activity c). Both methods are illustrated in this chapter.

The network diagram describes sequential relationships among major activities on a project. For instance, activity 2-4 cannot be started, according to the network, until activity 1-2 has been completed. A path is a sequence of activities that leads from the starting node to the finishing node. Thus, the sequence 1-2-4-5-6 is a path. There are two other paths in this network: 1-2-5-6 and 1-3-5-6. The length (of time) of any path can be determined by summing the expected times of the activities on that path.

The path with the longest time is of particular interest because it governs project completion time. In other words, expected project duration equals the expected time of the longest path. Moreover, if there are any delays along the longest path, there will be corresponding delays in project completion time. Conversely, attempts (0 shorten project completion must focus on the longest sequence of activities. Because of its influence on project completion time, the longest path is the critical path, and its activities are referred to as critical activities.

Paths that are shorter than the critical path can experience some delays and still not affect the overall project completion time as long as the ultimate path time does not exceed the length of the critical path. The allowable slippage for any path is called the path *slack,* and it reflects the difference between the length of a given path and the length of the critical path. The critical path, then, has zero slack time.

3.2 Network Conventions

Developing and interpreting network diagrams requires some familiarity with networking conventions. Although there are many that could be mentioned, the discussion here will concentrate on some of the most basic, and most common, features of network diagrams. This will provide sufficient background for understanding the basic concepts associated with precedence diagrams and allow you to solve typical problems.

One of the main features of a precedence diagram is that it reveals which activity must be performed in sequence (i.e., there is a precedence requirement) and which can be performed independently of each other. For example, in the following diagram, activity a must be completed before activity b can begin, and activity b must be completed before activity c can begin.

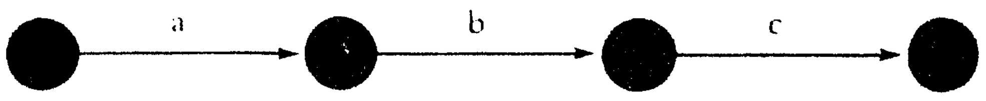


Figure 3.3. An Example Diagram, Activity a Must Be Completed before Activity b Can Begin, and Activity b Must Be Completed before Activity c Can Begin.

If the diagram had looked like this, both activity a and activity b would have to be completed before activity c could begin, but a and b could be performed at the same time; performance of a is independent of performance of b.

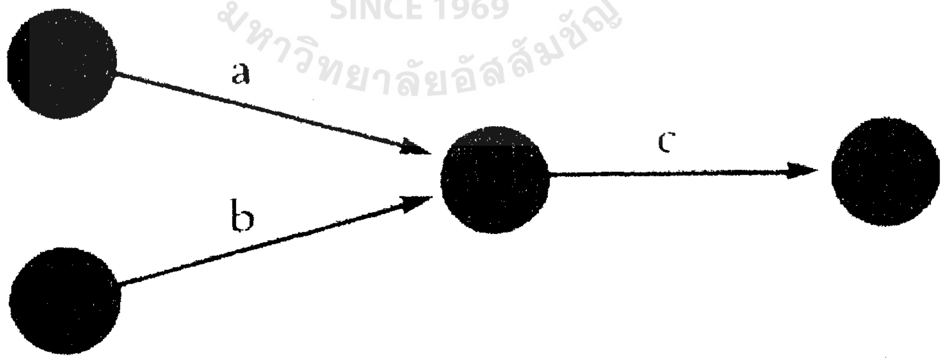


Figure 3.4 An Example That Both Activity a and Activity b Would Have to Be Completed before Activity c Could Begin.

If activity a must precede b and c, the appropriate network would look like this:

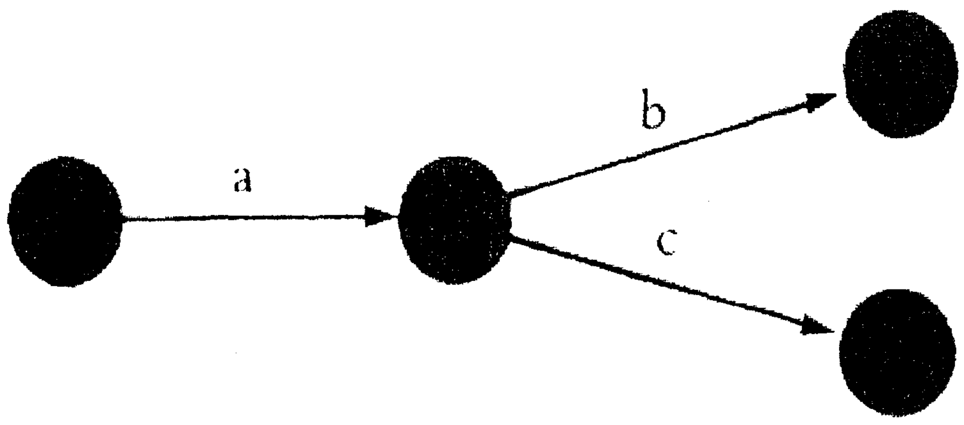


Figure 3.5. An Example That Both b and c, the Appropriate Network.

When multiple activities enter a node, this implies that all those activities must be completed before any activities that are to begin at that node can start. Hence, in this diagram, activities a and b must both be finished before either activity c or activity d can start.

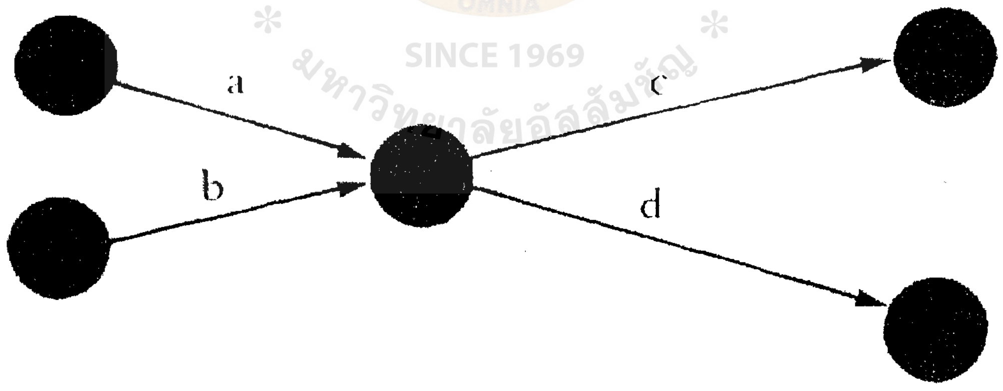


Figure 3.6. An Example That Activities a and b Must Both Be Finished Before Either Activity c or Activity d can Start.

When two activities have the same beginning and ending nodes, a dummy node and activity is used to preserve the separate identity of each activity In the diagram below, activities a and b must be completed before activity c can be started.

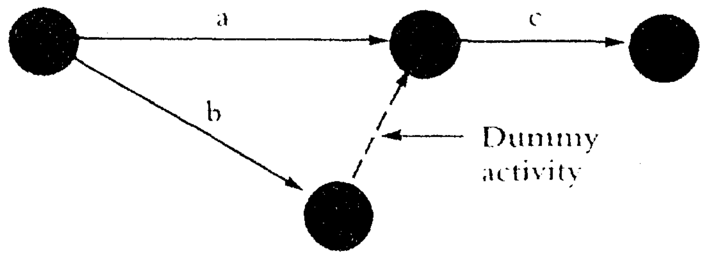


Figure 3.7. An Example that Two Activities Have the Same Beginning and Ending Nodes.

Separate identities are particularly important for computer analysis, because most computer programs identify activities by their endpoints; activities with the same endpoints could not be distinguished from each other, although they might have quite different expected times.

There are actually a number of different uses of dummy activities. Another common use is depicted below:

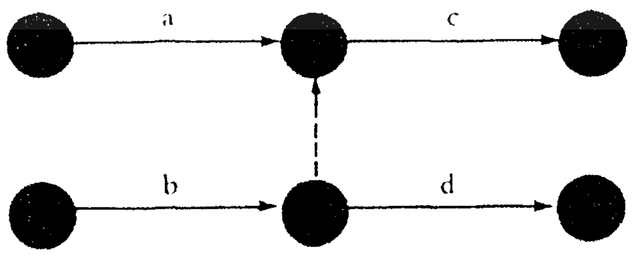


Figure 3.8. An Example Different Uses of Dummy Activities.

In this situation, activities a and b must both precede activity c. However, d's start is dependent only on completion of activity b, and not on activity completion.

The primary function of dummy activities is to clarify relationships. As far as time is concerned, a dummy activity has an activity time equal to zero.

For reference purposes, nodes are numbered typically from left to right:

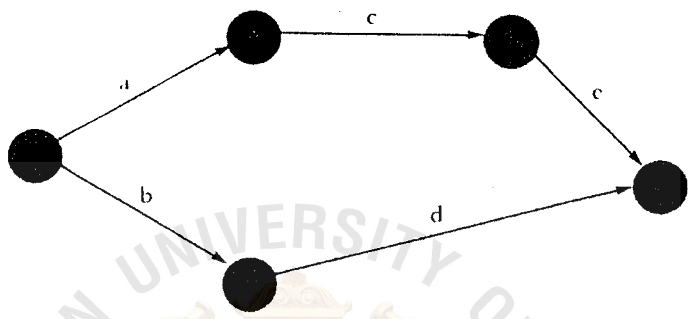


Figure 3.9. An Example, Nodes Are Numbered Typically from Left to Right.

Starting and ending arrows are sometimes used during development of a network for increased clarity.

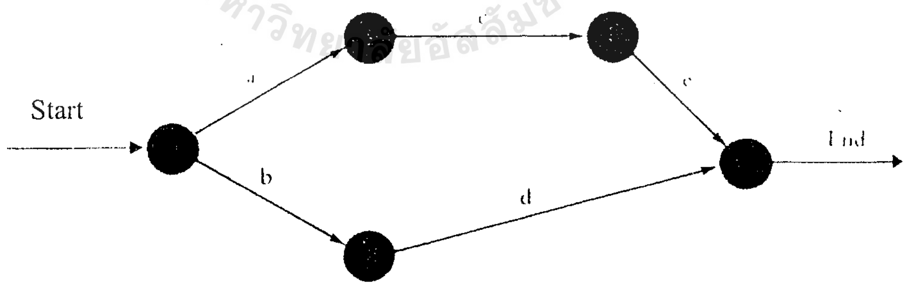


Figure 3.10. An Example Starting and Ending Arrows.

3.3 Deterministic Time: Estimatimates

The main determinant of the way CPM and PERT networks are analyzed and interpreted is whether activity time estimates are probabilistic or deterministic. If time estimates can be made with a high degree of confidence that actual times will not differ significantly, we say the estimates are deterministic. On the other hand, if estimated times are subject to variation, we say the estimates are probabilistic. Probabilistic time estimates must include an indication of the extent of probable variation.

This section describes analysis of networks with deterministic time estimates. A later section deals with probabilistic times.

One of the best ways to gain an understanding of the nature of network analysis is to consider a simple example.

EXAMPLE I

Given the information provided in the accompanying network diagram, determine each of the following.

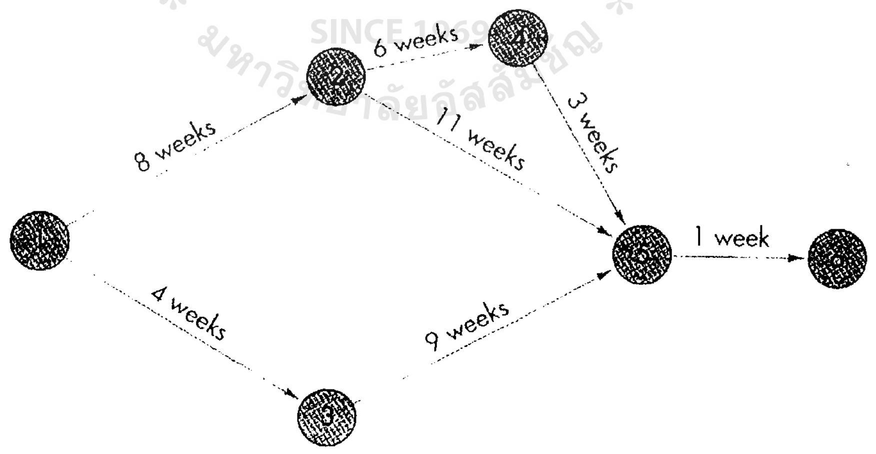


Figure 3.11. A Simple Example.

EXAMPLE I (concluded)

(a) The length of each path.

(b) The critical path.

(c) The expected length of the project.

(d) Amount of slack time for each path.

Solution

(a) As shown in the following table, the path lengths are 18 weeks, 20 weeks, and 14 weeks.

(b) The longest path (20 weeks) is 1-2-5-6, so it is the critical path.

(c) The expected length of the project is equal to the length of the critical path (i.e., 20 weeks).

(d) The slack for each path is found by subtracting its length from the length of the critical path, as shown in the last column of the table. (Note: It is sometimes desirable to know the slack time associated with activities. The next section describes a method for obtaining those slack times.)

Path	Length (weeks)	Slack
1-2-4-5-6	8 + 6 + 3 + 1 = 18	20 - 18 = 2
1-2-5-6	8 +11 + 1 = 20*	20 - 20 = 0
1-3-5-6	4 + 9 + 1 = 14	20-14 = 6

*Critical path length.

3.4 A Computing Algorithm

Many real-life project networks are much larger than the simple network illustrated in the preceding example; they often contain hundreds or even thousands of activities. Because the necessary computations can become exceedingly complex and time-consuming, large networks are generally analyzed by computer programs rather than manually. The intuitive approach just demonstrated does not lend itself to computerization because, in many instances, path sequences are not readily apparent. Instead, an algorithm is used to develop four pieces of information about the network activities:

ES, the earliest time activity can start, assuming all preceding activities start as early as possible.

EF, the earliest time the activity can finish.

LS, the latest time the activity can start and not delay the project.

LF, the latest time the activity can finish and not delay the project.

Once these values have been determined, they can be used to find:

(1) Expected project duration.

(2) Slack time.

(3) Which activities are on the critical path?

The three examples that follow illustrate how these values are computed using the precedence diagram of Example I, which is repeated here for convenient reference.

Compute the earliest starting time and earliest finishing time for each activity in the diagram shown in Figure 3.12.

Begin by placing brackets at the two ends of each starting activity:

Figure 3.12. An Example for Compute the Earliest Starting Time and Earliest Finishing Time.

We determine and place in the brackets for each activity, the earliest starting time, ES, and the earliest finishing time, EP as follows:

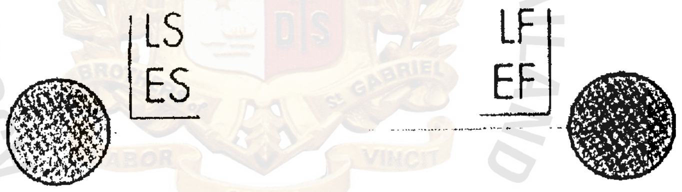


Figure 3.13. An Example to Determine and Place in the Brackets for Each Activity.

Do this for all activities, beginning at the left side of the precedence diagram and moving to the right side.

Once ES has been determined for each activity, EF can be found by adding the activity time, t, to ES: ES + t = EF

Use an ES of 0 for all starting activities. Thus, activities 1-2 and 1-3 are assigned ES values of 0. This permits computation of the EF for each of these activities:

$EF_{1-2} = 0+8 = 8$ and $EF_{1-3} = 0 + 4 = 4$

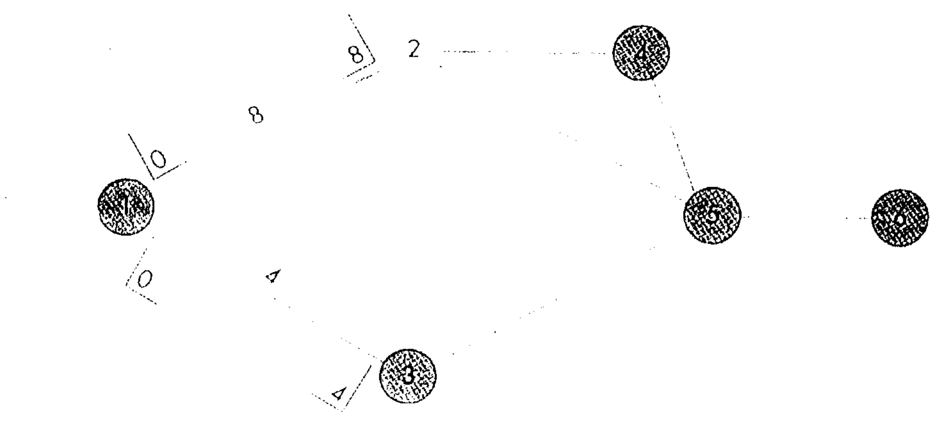


Figure 3.14. An Example for All Activities, Beginning at the Left Side of the Precedence Diagram and Moving to the Right Side.

The EF time for an activity become the ES time for the next activity to follow it in the diagram. Hence, because activity 1-2 has an EF time of 8, both activities 2-4 and 2-5 have ES times of 8. Similarly, activity 3-5 has an ES time of 4.

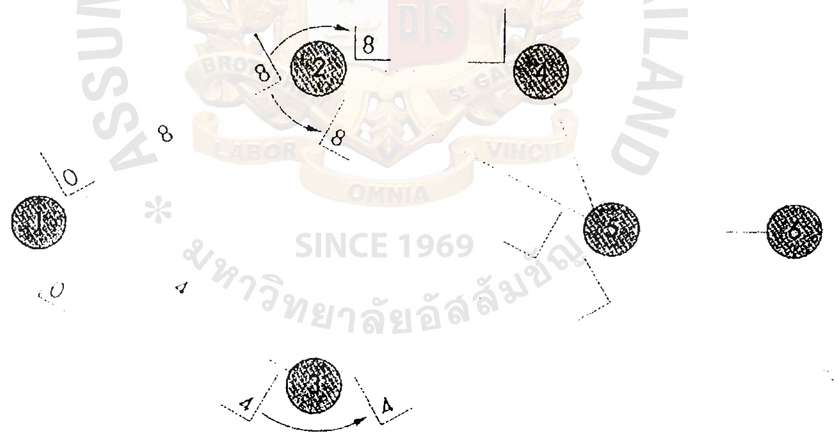


Figure 3.15. An Example for The EF Time for an Activity Become the ES Time for the Next Activity.

This permits calculation of the EF times for these activities: $EF_{2-4} = 8 + 6 = 14$; $EF_{2-5} = 8 + 11 = 19$; and $EF_{3-5} = 4 + 9 = 13$.

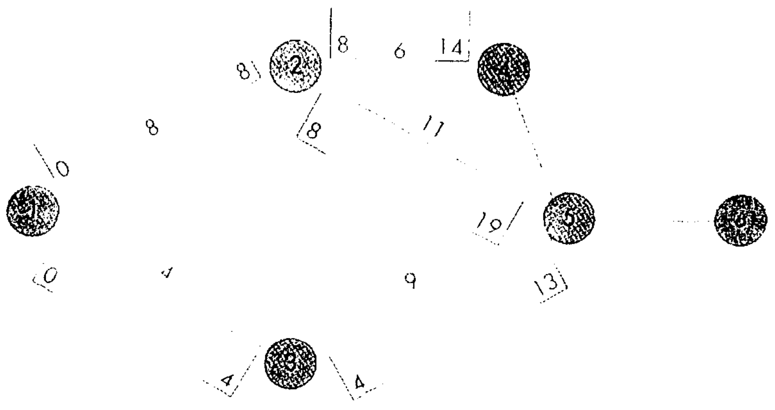


Figure 3.16. An Example to Permits Calculation of the EF Times for These Activities.

The ES for activity 4-5 is the EF time of activity 2-4, which is 14. Urine this value, we fine the EF for activity 4-5 is 17; $EF_{4-5} = 14 + 3 = 17$.

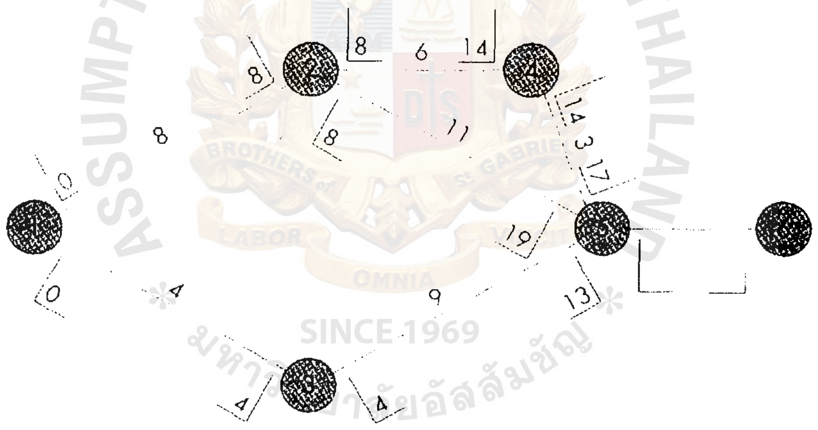


Figure 3.17. An Example That ES for Activity 4-5 is the EF Time of Activity 2-4, Which Is 14. Urine This Value, We Fine the EF for Activity 4-5 is 17; $EF_{4-5} = 14 + 3 = 17$.

In order to determine the ES for activity 5-6, we must realize that activity 5-6 cannot start until every activity that precedes it is finished. Therefore the largest of the EF time for the three activities that precede activity 5-6 determines ES_{5-6}. Hence, the ES for activity 5-6 is 19.

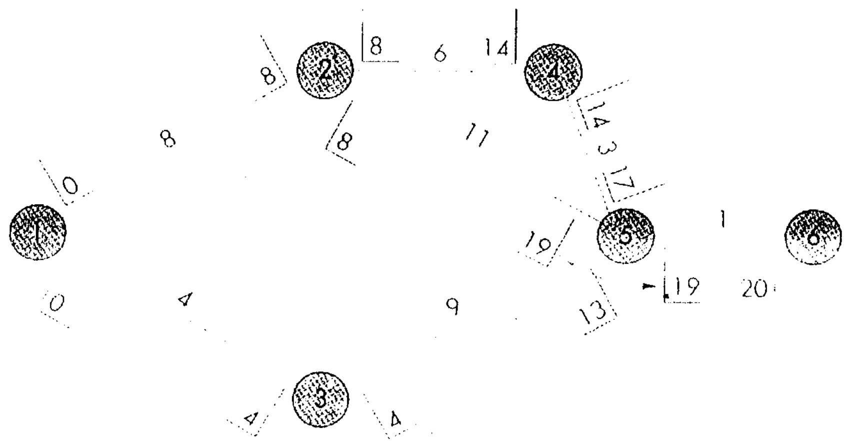


Figure 3.18. An Example That EF for the Last Activity, 5-6, is 20; $EF_{5-6}=19 + 1 = 20$.

Then the EF for the last activity, 5-6, is 20; $EF_{5-6} = 19 + 1 = 20$. Note that the latest EF is the project duration. Thus, the expected length of the project is 20 weeks.

3.5 Computing ES and EF Times

Computation of earliest starting and finishing times is aided by two simple rules:

(1) The earliest finish time for any activity is equal to its earliest start time plus its expected duration, r.

$$EF = ES + t$$

(2) For nodes with one entering arrow, ES for activities at such nodes is equal to EF of the entering arrow. For nodes with multiple entering arrows, ES for activities leaving such nodes equals the largest EF of the entering arrow.

EXAMPLE II

Compute the earliest starting time and earliest finishing time for each activity in the diagram shown in Figure 3.19.

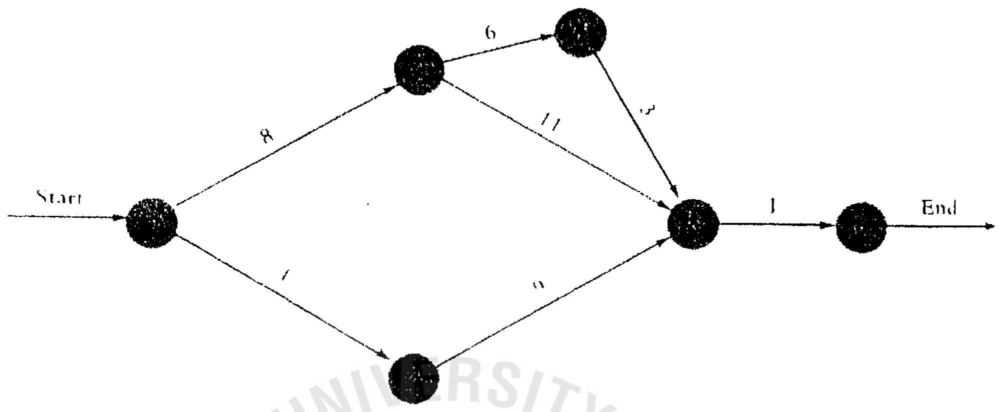


Figure 3.19. Example II.

Solution

Assume an ES of 0 for activities without predecessors. Thus, activities 1-2 and 1-3, as initial activities, are assigned early starting times equal to zero. The earliest finishing times for these activities are:

EXAMPLE II (concluded)

$EF_{1-2} = 0 + 8 = 8$ and $EF_{1-3} = 0 + 4 = 4$

The EF of activity 1-2 becomes the ES for the two activities that follow it: 2-4 and 2-5. Likewise, the EF of activity 1-3 becomes the ES for activity 3-5. Thus:

$ES_{2-4} = 8$, $ES_{2-5} = 8$, and $ES_{3-5} = 4$

The corresponding EF times for these activities are:

$EF_{2-4} = 8 + 6 = 14$

$EF_{2-5} = 8 + 11 = 19$

$EF_{3-5} = 4 + 9 = 13$

Activity 4-5 has an early starting time equal to EF_{2-4}, or 14, and an early finish time of 14 + 3 = 17. Finally, activity 5-6, with three predecessors, has an early starting time equal to the largest EF of the three activities that precede it. Hence, it has an ES of 19. Its EF time is 19 + 1 = 20.

These results are summarized in the following.

Activity	Duration	ES	EF
1-2	8	0	8
1-3	4	0	4
2-4	6	8	14
2-5	11	8	19
3-5	9	4	13
4-5	3	14	17
5-6	1	19	20

Note that the latest EF is the project duration. Thus, the expected length of the project is 20 weeks.

3.6 Computing LS and LF Times

Computation of the latest starting and finishing times is aided by the use of two rules:

(1) The latest starting time for each activity is equal to its latest finishing time minus its expected duration:

$$LS = LF - t$$

(2) For nodes with one leaving arrow, LF for arrows entering that node equals the LS of the leaving arrow. For nodes with multiple leaving arrows: LF for arrows entering that node equals the smallest LS of leaving arrows.

Finding ES and EF times involves a "forward pass" through the network; finding LS and LF times involves a "backward pass" through the network. Hence, we must begin with the EF of the last activity and use that time as the LF for the last activity. Then we obtain the LS for the last activity by subtracting its expected duration from its LF.

EXAMPLE III

Compute the latest finishing and starting times for each activity shown in Figure 3.19.

Solution

Set LF of the last activity equal to the EF of that activity. Thus,

$$LF_{5\text{-}6} - EF_{5\text{-}6} = 20 \text{ weeks}$$

Next, compute the latest starting time:

$$\begin{aligned} LS_{5\text{-}6} &= LF_{5\text{-}6} - t \\ &= 20 - 1 = 19 \end{aligned}$$

In order for activity 5-6 to start no later than week 19, all immediate predecessors must finish no later than that time. Thus,

$$LF_{4\text{-}5} - LF_{2\text{-}5} = LF_{3\text{-}5} = 19$$

The respective LS times for each activity are:

$$LS_{4\text{-}5} = 19 - 3 = 16$$

$$LS_{2\text{-}5} = 19 - 11 = 8$$

$$LS_{3\text{-}5} = 19 - 9 = 10$$

Similarly, $LF_{2-4} = LS_{4-5} = 16$, and $LS_{2-4} = 16 - 6 = 10$. Hence, there are two arrows leaving node 2: 24, with LS = 10, and 2-5, with LS = 8. The latest finish for activity 1-2 thus becomes 8, which is the smallest LS for a leaving arrow. The LF for 1-3 is equal to the LS for 3-5:

$$LF_{1-3} = LS_{3-5} = 10$$

The LS for activity 1-3 is:

$$LS_{1-3} = 10 - 4 = 6$$

The LS for activity 1-2 is:

$$LS_{1-2} = LF_{12} - t$$

$$= 8-8 = 0$$

The LS and LF computations are summarized in the following.

Activity	Duration	LF	LS
5-6	1	20	19
4-5	3	19	16
2-5	11	19	8
3-5	9	19	10
2-4	6	16	10
1-2	8	8	0
1-3	4	10	6

3.7 Computing Slack Times

The slack time can be computed in either of two ways:

Slack = LS - ES or LF - EF

EXAMPLE 4

Compute slack times for the precedence diagram of Figure 3.19.

Solution

We have the option of using either the starting times or the finishing times. Suppose we choose the starting times. Using ES times computed in Example II and LS times computed in Example III, slack times are:

Activity	LS	ES	(LS - ES) Slack
1-2	0	0	0
1-3	6	0	6
2-4	10	8	2
2-5	8	8	0
3-5	10	4	6
4-5	16	14	2
5-6	19	19	0

The critical path using this computing algorithm is denoted by activities with zero slack time. Thus, the table in the proceeding example indicates that activities 1-2, 2-5, and 5-6 are all critical activities, which agrees with the results of the intuitive approach demonstrated in Example I.

Knowledge of slack times provides managers with greater detail for planning allocation of scarce resources and for directing control efforts toward those activities that might be most susceptible to delaying the project than the more simplistic intuitive approach does. In this regard, it is important to recognize that the activity slack times are based on the assumption that all of the activities on the same path will be started as early as possible and not exceed their expected times. Furthermore, if two activities are both on the same path (e.g., activities 2-4, and 4-5 in the preceding example) and have the same slack (e.g., two weeks), this will be the total slack available to both.

In essence, the activities have shared slack. Hence, if the first activity uses all this slack, there will be zero slack for the other activity and that much less slack for all following activities on that same path.

IV. FINANCIAL ANALYSIS

Benefits/Cost Analysis

4.1 Cost Analysis

The costs of proposed system are as follows:

Software cost

(1) Microsoft Visual Basic	12,000 Baht
(2) Microsoft Window Xp Thai Editions	7,000 Baht
(3) Microsoft Office Xp	8,650 Baht
Total cost of software	27,650 Baht

Installation cost estimated:

Pentium IV 2.0 GHz

(1) Hard Disk 40.0 GB (7200rpm)

(2) DVD-ROM 16X

(3) DDR-RAM 256 MB(PC2700)

(4) Monitor 17" Flat Screen

(5) Case ATX 350 Walt(for Pentium IV)

(6) Back Up unit 800 VA

(7) Main Board INTEL socket 478

(8) CPU INTEL 2.0 GHz(Pentium IV)

Total cost of installation	60,000 Baht
Total cost	87,650 Baht

4.2 Intangible Benefits

(1) It easy for end user.

(2) It can change number of activity and activity time.

(3) Can display Gantt chart that easy for planing.

(4) User will know when they should to start their activity.

(5) Can calculate when should be order a material.

(6) For manager can use for long term planning.

(7) Can reduce some position that knows only CPM.

4.3 Break Even Analysis.

Suppose that can reduce worker that knows only CPM, production manager that get salaries equal 20,000 Baht and user such as Forman to use this program.

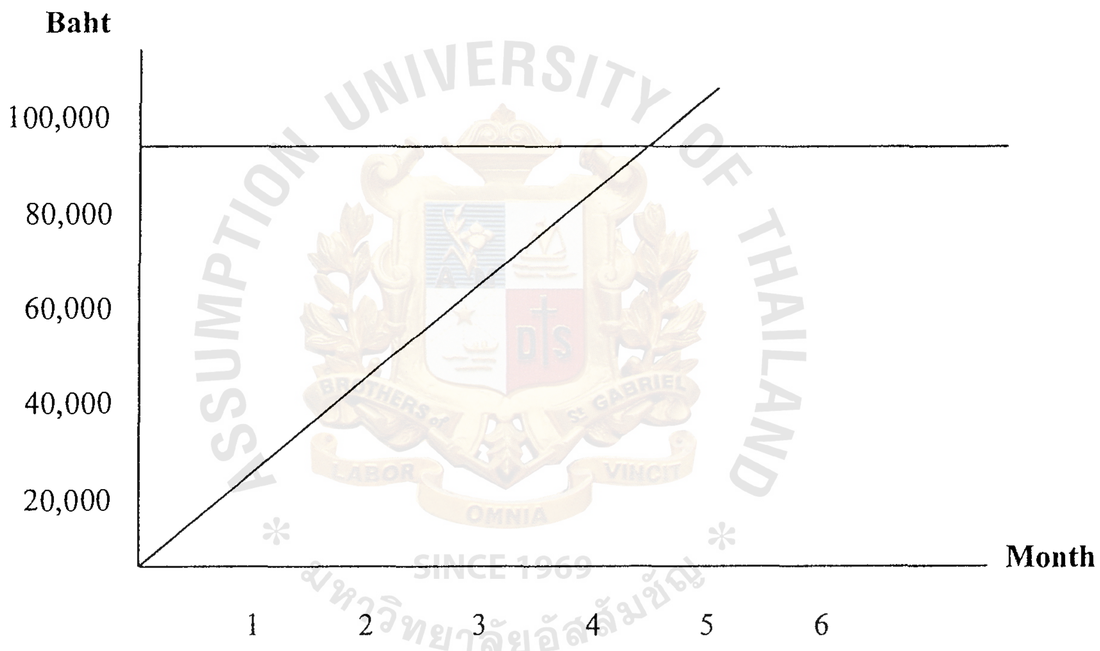


Figure 4.1. Break Even Analysis.

From Figure 4.1 will see, this project use just 5 months for return.

V. DEVELOPING VISUAL BASIC FOR CALCULATE CRITICAL PATH

5.1 Overview of Visual Basic

The emphasis with object-oriented/event-driven (OOED) languages, such as Visual Basic, is on the objects included in the user interface and on the events that occur through those objects. Consequently, the procedure-oriented approach to the programming solution of the step-by-step, top-to-bottom development is inappropriate.

The goal of the Visual Basic programmer is to develop an interface that gives the user as much control as possible, while guarding against application errors.

The programming process in Visual Basic consists of a five-step process: plan the application, build the user interface, code the application, test and debug the application, and deliver the application (which includes project documentation). The application plan culminates in an identification of tasks the application needs to perform, the objects needed to accomplish these tasks, and the events required to trigger an object to perform its task.

The interactive nature of the development environment allows the programmer to continue refinements and development of the user interface while involved in the coding process. You can test event code immediately as each event process is completed. Delivering the application involves gathering all documentation for delivery with the executable file. This text will emphasize the build, code, and test processes of application development.

5.2 Introduction for Uses CPM Program.

A Figure 5.1. show the flow chart for end user to use CPM software and the other Figures show an CPM computer software step by step.

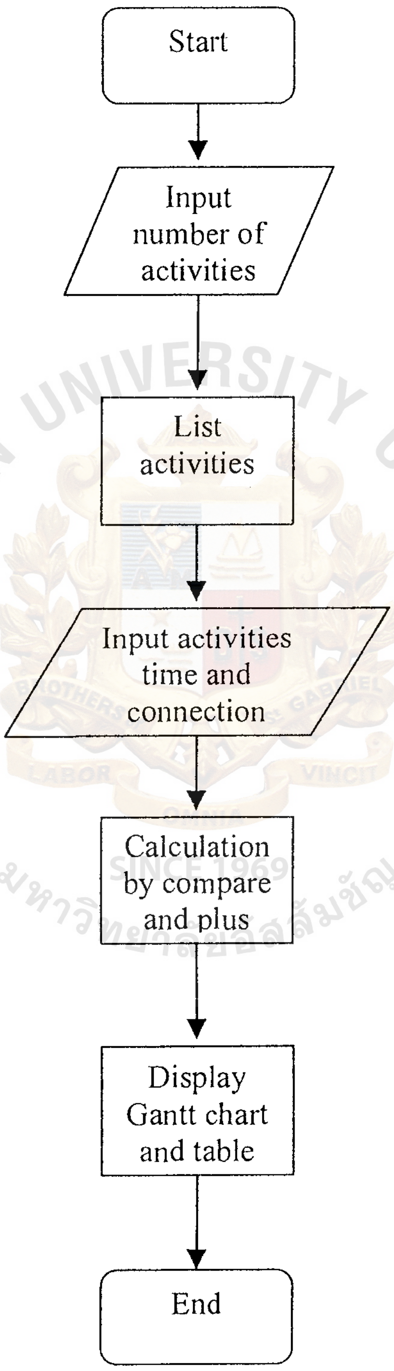

Start
Input number of activities
List activities
Input activities time and connection
Calculation by compare and plus
Display Gantt chart and table
End

Figure 5.1. Flow Chart for End User.

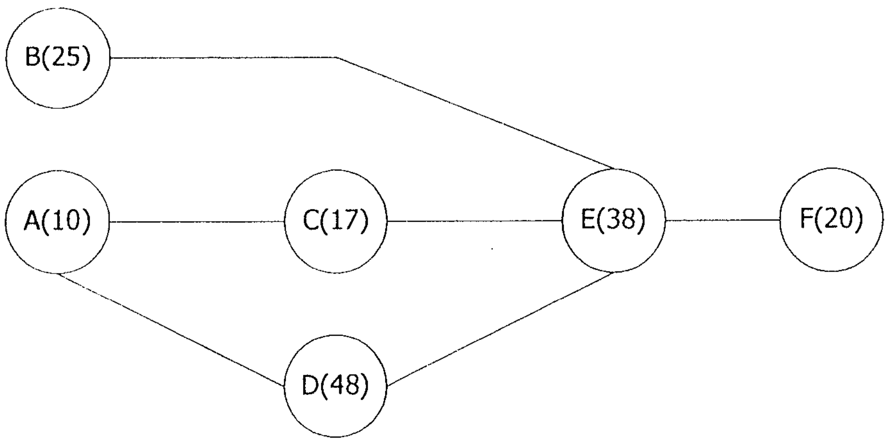


Figure 5.2. Example Network for Test Program.

Before use this program should have network like Figure 2.5.

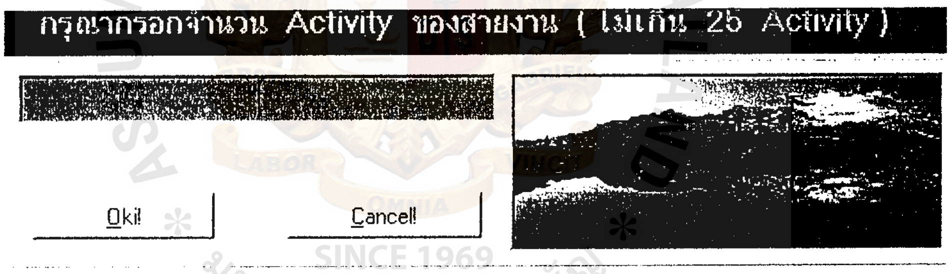
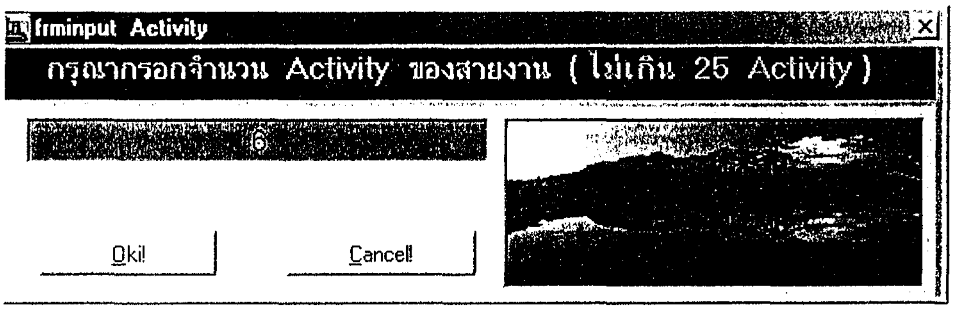

กรุณากรอกจำนวน Activity ของสายงาน (ไม่เกิน 25 Activity)

Figure 5.3. The First Page When Run a CPM Program.

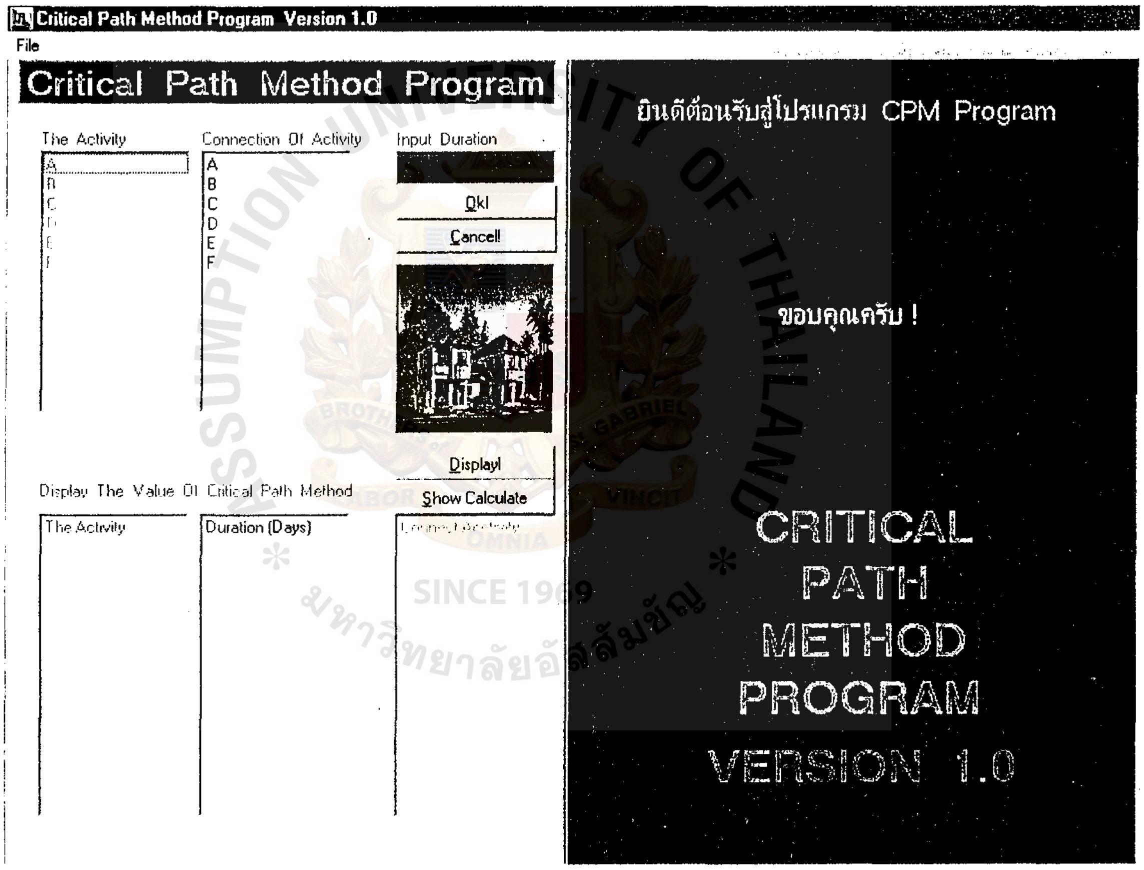
In this page you will see a n opening for input number of activity that no over 25 activities because the program use a capital in English for list a table.


frminput Activity

กรุณากรอกจำนวน Activity ของสายงาน (ไม่เกิน 25 Activity)

6

Figure 5.4. Input Number of Activity by Use Figure 5.2.


Critical Path Method Program Version 1.0

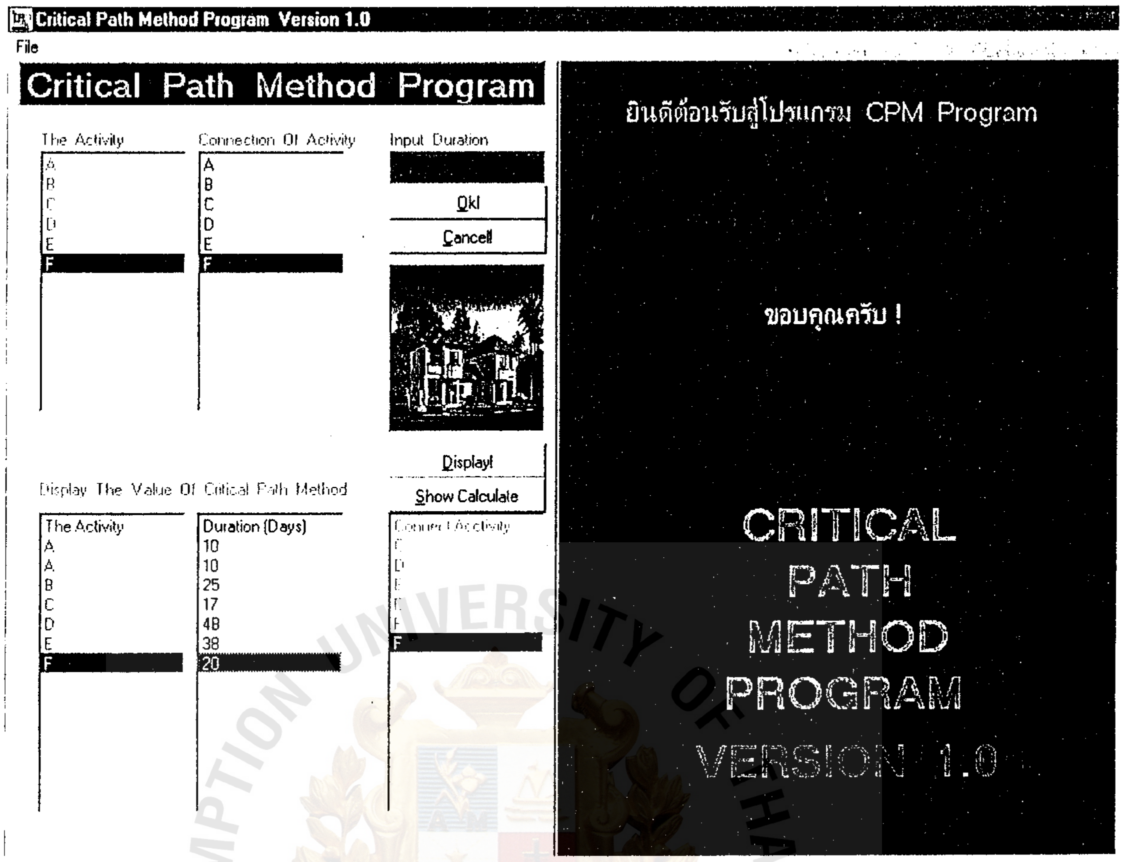
A
B
C
D
E
F

ยินดีต้อนรับสู่โปรแกรม CPM Program

ขอบคุณครับ !

CRITICAL PATH METHOD PROGRAM VERSION 1.0

Figure 5.5. List of Activity.


Critical Path Method Program Version 1.0
File
Critical Path Method Program
The Activity
Connection Of Activity
Input Duration
Ok!
Cancel!
Display!
Show Calculate
ยินดีต้อนรับสู่โปรแกรม CPM Program
ขอบคุณครับ !
CRITICAL PATH METHOD PROGRAM VERSION 1.0

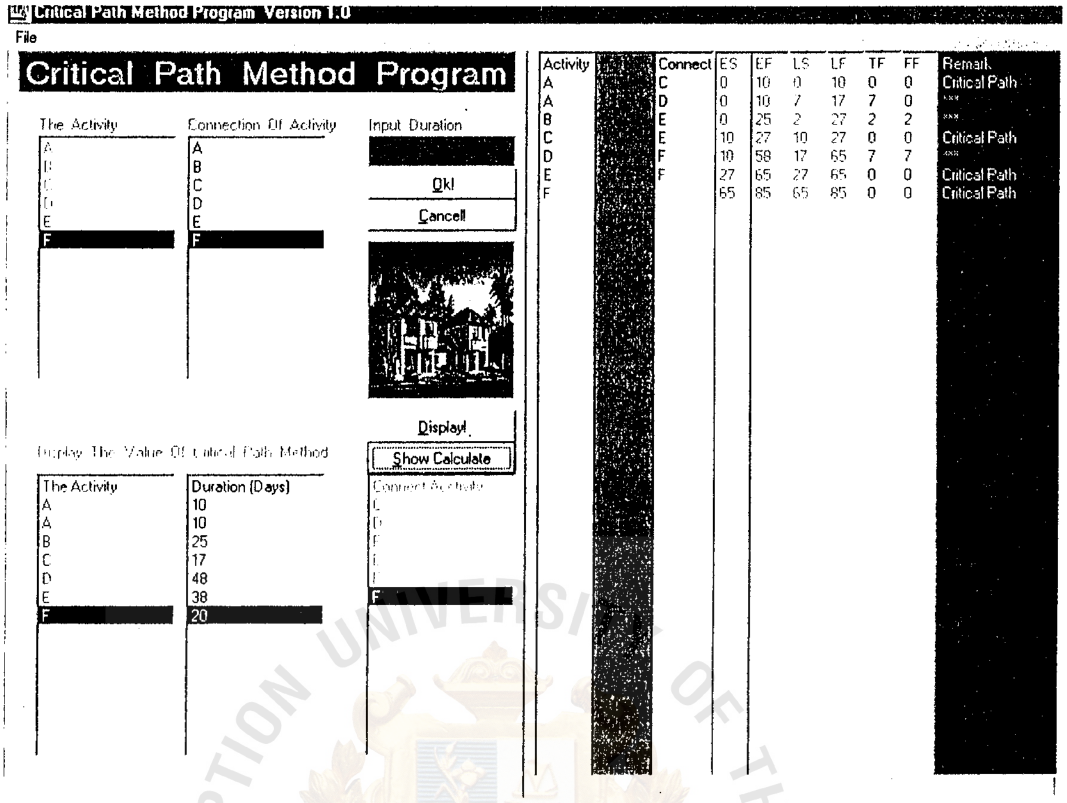
The Activity	Connection Of Activity
A	A
B	B
C	C
D	D
E	E
F	F

Display The Value Of Critical Path Method

The Activity	Duration (Days)	Connect Activity
A	10	C
A	10	D
B	25	E
C	17	E
D	48	F
E	38	F
F	20	

Figure 5.6. Input Times Duration and Set Activity Connection.

From Figure 5.6 will see that has activity A 2 times, that come from Figure 5.2 will see activity A have connect to activity C and D. When end user use this program they must input activity like a network, it mean when a network have more 1 connection (like activity A in Figure 5.2) they should input an activity equal the connection. If you input correctly the number of an opening "The Activity" will more than an opening "Connect Activity" by 1.



Critical Path Method Program Version 1.0

File

Critical Path Method Program

The Activity	Connection Of Activity
A	A
B	B
C	C
D	D
E	E
F	F

Input Duration

Ok!

Cancel!

Display!

Show Calculate

Display The Value Of Critical Path Method

The Activity	Duration (Days)	Connect Activity
A	10	C
A	10	D
B	25	E
C	17	E
D	48	F
E	38	F
F	20	

Activity	[illegible]	Connect	ES	EF	LS	LF	TF	FF	Remark
A		C	0	10	0	10	0	0	Critical Path
A		D	0	10	7	17	7	0	***
B		E	0	25	2	27	2	2	***
C		E	10	27	10	27	0	0	Critical Path
D		F	10	58	17	65	7	7	***
E		F	27	65	27	65	0	0	Critical Path
F			65	85	65	85	0	0	Critical Path

Figure 5.7. Table from Calculates.

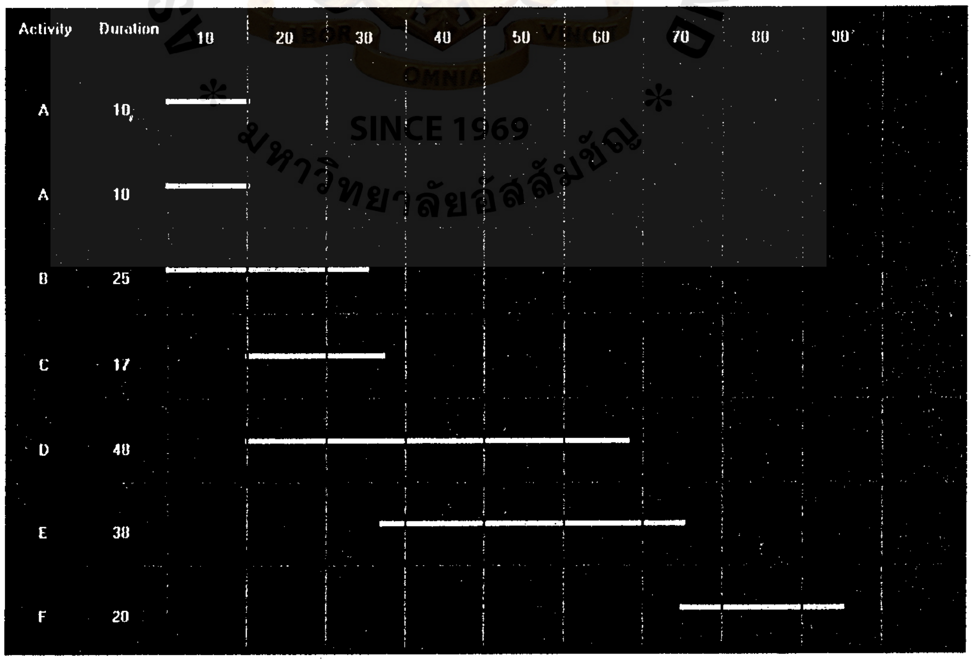


Figure 5.8. Gantt Chart.

5.3 Problem about Visual Basic

In this program that use Visual Basic, that include about "Data Base" for generate table and Gantt chart. In this case the problem has happen when Compile, first of all when we have compiled that can have 2 type EXE file and SETUP file. The EXE file can run every where and every operation (Win9x,WinME and Windows2000) but SETUP file can setup into windows and run from Program File, that can setup only Windows 98 SE may be cause from "Data Base". Maybe this effect will relationship about "Data Base" that use Windows98 SE to develop.

VI. CONCLUSIONS

The CPM program provides a convenient approach for solving complex network problems in production. However, as this paper demonstrated with some illustrative examples, its use to network diagram problems should be a cautious one. There is way to input data (times duration and connection) if input it correctly the program will show that you want to know about CPM.

This is true with Critical Path method. The examples in this paper, will lookalike not complicated that was intentionally for have easy to understand especially to end user that sometime they have a network diagram and introduction of this program.

In addition it can help another people such as manager to use for long planning for example to find when should order the material for each activity.

The computer software the CPM can calculate the best way and it shows result step by step for studies how to use the CPM to choose the best way.

APPENDIX A

EXAMPLE SOURCE CODE

```
Option Explicit

Const NumFrames = 72

Dim a, b, c, d, e, i, J, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, es, et, f, g, h, joe, jub, nha,

phu, nhing, pop, jaib, Mm, Fly As Integer

Dim Mun, Moon As String

Dim picArray(1 To NumFrames) As Picture

Private Sub Command1_Click()

List5.AddItem Text1.Text

Text1.Text = ""

End Sub

Private Sub Command2_Click()

Dim i As Integer

i = List2.ListIndex

If i >= 0 Then

   List2.RemoveItem i

   Else

      Beep

End If

End Sub

Private Sub Command3_Click()

If List3.ListCount < Vlac Then

  MsgBox " ?", vbOKOnly, "CPM Confirm !"

  List1.SetFocus

ElseIf List3.ListCount >= Vlac + 1 And List5.ListCount >= Vlac + 1 Then

Dim er, wr, i, k, l, m, n, o, p, q As Integer
```

```
nub = 1
er = Val(List3.ListCount - 1)
wr = Val(List4.ListCount - 1)
ReDim AC(1 To er, 1 To wr) As Integer
For k = 1 To er
   For l = 1 To wr
      If l > wr Then
        GoTo Jrr
      ElseIf l <= wr Then
      list3 <>  list4        If List3.List(k) <> List4.List(l) Then
        List6.List(k) = "0": GoTo Prr
        list3 = list4            ElseIf List3.List(k) = List4.List(l) Then
             List6.List(k) = "-": GoTo Jrr
      End If
      End If
Prr:
   Next l
Jrr:
Next k
For m = 1 To er
      '       l ist6 = 0 list7 = list5 + list6
      If List6.List(m) = "0" Then
        List7.List(m) = Val(List5.List(m)) + Val(List6.List(m))
      'list6 = - list7 = -
      ElseIf List6.List(m) = "-" Then
```

```
            List7.List(m) = "-"
        End If
Next m
For n = 1 To er
        For p = 1 To wr
            ' list3  list4              If List3.List(n) <> List4.List(p) Then
              GoTo Zxx
            'list3 = list4 list6 = list7
            ElseIf List3.List(n) = List4.List(p) Then
                    List6.List(n) = List7.List(p)
                    List7.List(n) = Val(List5.List(n)) + Val(List6.List(n))
                    GoTo Yxx
            End If
Zxx:
        Next p
Yxx:
Next n

ReDim Dl4(1 To wr) As String
For o = 1 To wr
        Dl4(o) = List4.List(o)
        Mm = Val(List7.List(o))
        For p = 1 To wr
            If Dl4(o) = List4.List(p) Then
              If Mm <= List7.List(p) Then
```

```
                Mm = Val(List7.List(p))
                    For q = 1 To er
                          If List3.List(q) = Dl4(o) Then
                            List6.List(q) = Mm
                            List7.List(q) = Val(List6.List(q)) + Val(List5.List(q))
                          End If
                    Next q
              End If
            End If
      Next p
Next o
For r = 1 To er
      List8.AddItem "-"
      List9.AddItem "-"
      List10.AddItem "-"
      List11.AddItem "-"
      List15.AddItem "-"
Next r
List9.List(er) = List7.List(er)
List8.List(er) = List6.List(er)
es = er - 1
et = wr - 1
For s = 0 To es
      For t = 0 To et
            ' list3 list4
            If List3.List(er - s) <> List4.List(wr - t) Then
              GoTo Zxy
            'list3 = list4 list6 = list7
            ElseIf List3.List(er - s) = List4.List(wr - t) Then
                    List9.List(wr - t) = List8.List(er - s)
                    List8.List(wr - t) = Val(List9.List(wr - t)) - Val(List5.List(wr - t))
                    GoTo Zxy
           End If
Zxy:
     Next t
Next s
For joe = 0 To es
     Moon = List3.List(er - joe)
     Fly = Val(List8.List(er - joe))
     For jub = 1 To er
           If Moon = List3.List(jub) Then
             If Fly < List8.List(jub) Then
             For phu = 1 To wr
                   If List4.List(phu) = Moon Then
                     List9.List(phu) = List8.List(er - joe)
                     List8.List(phu) = Val(List9.List(phu)) - Val(List5.List(phu))
                   End If
             Next phu


           End If
```

```
            End If
          Next jub
  Next joe
  List11.List(er) = Val(List7.List(er)) - Val(List7.List(er))
  For u = 1 To er
        List10.List(u) = Val(List9.List(u)) - Val(List7.List(u))
  Next u
  For v = 1 To er
        For w = 1 To wr
              If List3.List(v) = List4.List(w) Then
                List11.List(w) = Val(List6.List(v)) - Val(List7.List(w))
                GoTo ker
              End If
  ker:
       Next w
  Next v
  For x = 1 To er
       List12.List(x) = List3.List(x)
       List13.List(x) = List5.List(x)
  Next x
  For y = 1 To wr
       List14.List(y) = List4.List(y)
  Next y
  For z = 1 To er
        If List10.List(z) = "0" And List11.List(z) = "0" Then
        List15.List(z) = "Critical Path"
      Else
        List15.List(z) = "***"
      End If
Next z
ReDim Limit(1 To er) As Single
Limit(1) = 7200 / er
Dim Art As Integer
ReDim DtaN(1 To er) As String
'input duration
For Art = 1 To er
   If Art > er Then
      GoTo Sat:
   ElseIf Art <= er Then
   DtaN(Art) = List5.List(Art)
   frmDisplay.Label4(Art - 1).Caption = DtaN(Art)
   frmDisplay.Label4(Art - 1).BackStyle = 0
   If List5.List(Art) = List5.List(Art + 1) Then
      frmDisplay.Label4(Art - 1).Caption = List5.List(Art + 1)
      frmDisplay.Label4(Art - 1).BackStyle = 0
   End If
  End If
Next Art
Sat:
'not main
```

```
Dim Pol2 As Integer
Dim Jat As Integer
Dim Rtg, pol As Integer
    pol = Val(List7.List(er))
    Pol2 = (pol / 10) + 1
Dim aa As Integer
ReDim DtDu(1 To Pol2 + 1) As Integer
ReDim Par(1 To Pol2 + 1) As Integer
'if pol < 200
If pol < 200 Then
   For l = 1 To Pol2 + 1
      DtDu(l) = 10 * l
      frmDisplay.Label5(l - 1).Caption = DtDu(l)
      frmDisplay.Label5(l - 1).BackStyle = 0
      On Error GoTo hjk:
   Next l
hjk:
   Par(1) = 9600 / Pol2
   'plot vertical line
   For aa = 1 To Pol2 - 1
      If aa > Pol2 Then
      GoTo Cft:
      ElseIf aa <= Pol2 Then
      frmDisplay.Line10(aa).X1 = 2160 + Par(aa)
      frmDisplay.Line10(aa).X2 = 2160 + Par(aa)
```

```
        frmDisplay.Line10(aa).Y1 = 360
        frmDisplay.Line10(aa).Y2 = 8160
        frmDisplay.Line10(aa).BorderWidth = 2
        frmDisplay.Line10(aa).BorderColor = vbBlue
        frmDisplay.Label5(aa - 1).FontBold = True
        frmDisplay.Label5(aa - 1).AutoSize = True
        frmDisplay.Label5(aa - 1).ForeColor = RGB(255, 0, 0)
        frmDisplay.Label5(aa - 1).Top = 600
        frmDisplay.Label5(aa - 1).Left = (2160 + Par(aa)) - ((Par(1) / 2) + 150)
        Par(aa + 1) = Par(aa) + Par(1)
        On Error GoTo Cft:
        End If
    Next aa
Cft:
'if pol >  200
ElseIf pol >= 200 Then
    Pol2 = (pol / 20) + 1
    For l = 1 To Pol2 + 1
        DtDu(l) = 20 * l
        frmDisplay.Label5(l - 1).Caption = DtDu(l)
        frmDisplay.Label5(l - 1).BackStyle = 0
        On Error GoTo jjk:
    Next l
jjk:
    Par(1) = 9600 / Pol2
```

```
   For aa = 1 To Pol2 - 1
      If aa > Pol2 Then
      GoTo jft:
      ElseIf aa <= Pol2 Then
      frmDisplay.Line10(aa).X1 = 2160 + Par(aa)
      frmDisplay.Line10(aa).X2 = 2160 + Par(aa)
      frmDisplay.Line10(aa).Y1 = 360
      frmDisplay.Line10(aa).Y2 = 8160
      frmDisplay.Line10(aa).BorderWidth = 2
      frmDisplay.Line10(aa).BorderColor = vbBlue
      frmDisplay.Label5(aa - 1).BackStyle = 0
      frmDisplay.Label5(aa - 1).FontBold = True
      frmDisplay.Label5(aa - 1).AutoSize = True
      frmDisplay.Label5(aa - 1).ForeColor = RGB(255, 0, 0)
      frmDisplay.Label5(aa - 1).Top = 600
      frmDisplay.Label5(aa - 1).Left = (2160 + Par(aa)) - ((Par(1) / 2) + 150)
      Par(aa + 1) = Par(aa) + Par(1)
      On Error GoTo jft:
      End If
   Next aa
jft:
End If
'plot time - schedul diagram
For i = 1 To er + 1
   If i > er Then
```

```
    GoTo ggg:
    ElseIf i < er Then
    frmDisplay.Line8(i).X1 = (Val(List6.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).X2 = (Val(List7.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).Y1 = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Line8(i).Y2 = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Line8(i).BorderWidth = 4
    frmDisplay.Line8(i).BorderColor = vbGreen
    Limit(i + 1) = Limit(i) + Limit(1)
    ElseIf i = er Then
    frmDisplay.Line8(i).X1 = (Val(List6.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).X2 = (Val(List7.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).Y1 = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Line8(i).Y2 = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Line8(i).BorderWidth = 4
    frmDisplay.Line8(i).BorderColor = vbGreen
    End If
Next i
ggg:
'plot horizontal line
For i = 1 To er + 1
    If i > er Then
    GoTo pgg:
    ElseIf i < er Then
```

```
frmDisplay.Line9(i).X1 = 240
frmDisplay.Line9(i).X2 = 11760
frmDisplay.Line9(i).Y1 = 960 + Limit(i)
frmDisplay.Line9(i).Y2 = 960 + Limit(i)
frmDisplay.Line9(i).BorderWidth = 2
frmDisplay.Line9(i).BorderColor = vbBlue
frmDisplay.Label3(i - 1).BackStyle = 0
frmDisplay.Label3(i - 1).Caption = List3.List(i)
frmDisplay.Label3(i - 1).AutoSize = True
frmDisplay.Label3(i - 1).ForeColor = RGB(200, 100, 50)
frmDisplay.Label3(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)
frmDisplay.Label3(i - 1).Left = 600
frmDisplay.Label3(i - 1).FontBold = True
frmDisplay.Label4(i - 1).BackStyle = 0
frmDisplay.Label4(i - 1).FontBold = True
frmDisplay.Label4(i - 1).AutoSize = True
frmDisplay.Label4(i - 1).ForeColor = RGB(255, 0, 100)
frmDisplay.Label4(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)
frmDisplay.Label4(i - 1).Left = 1480
Limit(i + 1) = Limit(i) + Limit(1)
ElseIf i = er Then
frmDisplay.Line9(i).X1 = 240
frmDisplay.Line9(i).X2 = 11760
frmDisplay.Line9(i).Y1 = 960 + Limit(i)
frmDisplay.Line9(i).Y2 = 960 + Limit(i)
```

```
    frmDisplay.Line9(i).BorderWidth = 2
    frmDisplay.Line9(i).BorderColor = vbBlue
    frmDisplay.Label3(i - 1).BackStyle = 0
    frmDisplay.Label3(i - 1).Caption = List3.List(i)
    frmDisplay.Label3(i - 1).AutoSize = True
    frmDisplay.Label3(i - 1).ForeColor = RGB(200, 100, 50)
    frmDisplay.Label3(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Label3(i - 1).Left = 600
    frmDisplay.Label3(i - 1).FontBold = True
    frmDisplay.Label4(i - 1).BackStyle = 0
    frmDisplay.Label4(i - 1).FontBold = True
    frmDisplay.Label4(i - 1).AutoSize = True
    frmDisplay.Label4(i - 1).ForeColor = RGB(255, 0, 100)
    frmDisplay.Label4(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Label4(i - 1).Left = 1480
    End If
Next i
pgg:
Load frmDisplay
frmDisplay.Show
Me.Hide
End If
End Sub

Private Sub Command4_Click()
```

```
Frame2.Visible = False


End Sub
Private Sub Form_Activate()
Me.Frame2.Visible = True
If nub = 1 Then
  GoTo Jasss
ElseIf nub = 0 Then
Dim m, n, o, p, q, r, s, t, u As Integer
      Me.Refresh


      For m = 1 To 8600 Step 0.04
            Label6.Visible = True
            Label6.Top = 9200 - m
      Next m
      For n = 1 To 7400 Step 0.04
            Label7.Visible = True
            Label7.Top = 9200 - n
      Next n
      For o = 1 To 6440 Step 0.04
            Label8.Visible = True
            Label8.Top = 9200 - o
      Next o
      For p = 1 To 4280 Step 0.065
            Label9.Visible = True
            Label9.Top = 9200 - p
        Next p
        For q = 1 To 3680 Step 0.065
            Label10.Visible = True
            Label10.Top = 9200 - q
        Next q
        For r = 1 To 3080 Step 0.065
            Label11.Visible = True
            Label11.Top = 9200 - r
        Next r
        For s = 1 To 2480 Step 0.065
            Label12.Visible = True
            Label12.Top = 9200 - s
        Next s
        For t = 1 To 1760 Step 0.065
            Label13.Visible = True
            Label13.Top = 9200 - t
        Next t
List1.SetFocus
End If
Jasss:
End Sub

Private Sub Form_Load()
nub = 0
```

```
Me.Top = 0
Me.Left = 0
Text1.Visible = True
Label4.Visible = True
Frame2.Visible = True
Label6.Visible = False
Label7.Visible = False
Label8.Visible = False
Label9.Visible = False
Label10.Visible = False
Label11.Visible = False
Label12.Visible = False
Label13.Visible = False
Dim i As Integer
Dim strFile As String
Dim intI As Integer
For i = 1 To Vlac
   List1.AddItem Dataactivity(i)
   List2.AddItem Dataactivity(i)
Next i
List3.AddItem "The Activity"
List4.AddItem "Connect Acctivity"
List5.AddItem "Duration (Days)"
List6.AddItem "ES"
List7.AddItem "EF"
```

```
List8.AddItem "LS"
List9.AddItem "LF"
List10.AddItem "TF"
List11.AddItem "FF"
List12.AddItem "Activity"
List13.AddItem "Duration"
List14.AddItem "Connect"
List15.AddItem "Remark"
End Sub

Private Sub Form_Unload(Cancel As Integer)
End
End Sub

Private Sub List1_DblClick()
Dim n, i As Integer
If List1.ListCount <> List2.ListCount Then
For n = 0 To (List1.ListCount - 1)
   If List1.Selected(n) = True Then
   List2.AddItem List1.List(n)
   End If
Next n
ElseIf List1.ListCount = List2.ListCount Then
     Text1.Visible = True
     Label4.Visible = True
     For i = 0 To (List1.ListCount - 1)
        If List1.Selected(i) = True Then
          List3.AddItem List1.List(i)
        End If
     Text1.SetFocus
     Next i
End If
End Sub
Private Sub List2_DblClick()
Dim r As Integer
For r = 0 To (List2.ListCount - 1)
   If List2.Selected(r) = True Then
   List4.AddItem List2.List(r)
   End If
Next r
End Sub
Private Sub List3_DblClick()
Dim i As Integer
i = List3.ListIndex
If i >= 0 Then
   List3.RemoveItem i
   Else
      Beep
End If
End Sub
```

```
Private Sub List4_DblClick()
Dim i As Integer
i = List4.ListIndex
If i >= 0 Then
   List4.RemoveItem i
   Else
      Beep
End If
End Sub

Private Sub List5_DblClick()
Dim i As Integer
i = List5.ListIndex
If i >= 0 Then
   List5.RemoveItem i
   Else
      Beep
End If
End Sub

Private Sub m2_Click()
If List3.ListCount < Vlac Then
  MsgBox "vbOKOnly, "Confirm !"
  List1.SetFocus
```

```
ElseIf List3.ListCount >= Vlac + 1 And List5.ListCount >= Vlac + 1 Then
Dim er, wr, i, k, l, m, n, o, p, q As Integer
nub = 1
er = Val(List3.ListCount - 1)
wr = Val(List4.ListCount - 1)
ReDim AC(1 To er, 1 To wr) As Integer
list3  = list4 ?
For k = 1 To er
   For l = 1 To wr
      If l > wr Then
        GoTo Jrr
      ElseIf l <= wr Then
      ' list3 <>  list4 list6 = 0
      If List3.List(k) <> List4.List(l) Then
        List6.List(k) = "0": GoTo Prr
        'list3 = list6 = -
        ElseIf List3.List(k) = List4.List(l) Then
             List6.List(k) = "-": GoTo Jrr
      End If
      End If
Prr:
   Next l
Jrr:
Next k
For m = 1 To er
```

```
    ' list6 = 0 = list5 + list6
    If List6.List(m) = "0" Then
      List7.List(m) = Val(List5.List(m)) + Val(List6.List(m))
    list6 = - list7 = -
    ElseIf List6.List(m) = "-" Then
            List7.List(m) = "-"
    End If
Next m
For n = 1 To er
    For p = 1 To wr
        ' list3 list4            If List3.List(n) <> List4.List(p) Then
          GoTo Zxx
        ElseIf List3.List(n) = List4.List(p) Then
                List6.List(n) = List7.List(p)
                List7.List(n) = Val(List5.List(n)) + Val(List6.List(n))
                GoTo Yxx
        End If
Zxx:
    Next p
Yxx:
Next n


ReDim Dl4(1 To wr) As String
For o = 1 To wr
    Dl4(o) = List4.List(o)
```

```
      Mm = Val(List7.List(o))
      For p = 1 To wr
            If Dl4(o) = List4.List(p) Then
              If Mm <= List7.List(p) Then
                Mm = Val(List7.List(p))
                    For q = 1 To er
                          If List3.List(q) = Dl4(o) Then
                            List6.List(q) = Mm
                            List7.List(q) = Val(List6.List(q)) + Val(List5.List(q))
                          End If
                    Next q
              End If
            End If
      Next p
Next o
For r = 1 To er
      List8.AddItem "-"
      List9.AddItem "-"
      List10.AddItem "-"
      List11.AddItem "-"
      List15.AddItem "-"
Next r
List9.List(er) = List7.List(er)
List8.List(er) = List6.List(er)
es = er - 1
```

```
et = wr - 1
For s = 0 To es
     For t = 0 To et
          list3
          If List3.List(er - s) <> List4.List(wr - t) Then
            GoTo Zxy
list3 = list6 = list7
         ElseIf List3.List(er - s) = List4.List(wr - t) Then
                List9.List(wr - t) = List8.List(er - s)
                List8.List(wr - t) = Val(List9.List(wr - t)) - Val(List5.List(wr - t))
                GoTo Zxy
        End If
Zxy:
     Next t
Next s
List11.List(er) = Val(List7.List(er)) - Val(List7.List(er))
For u = 1 To er
     List10.List(u) = Val(List9.List(u)) - Val(List7.List(u))
Next u
For v = 1 To er
     For w = 1 To wr
          If List3.List(v) = List4.List(w) Then
            List11.List(w) = Val(List6.List(v)) - Val(List7.List(w))
            GoTo ker
          End If
```

```
ker:
    Next w
Next v
For x = 1 To er
    List12.List(x) = List3.List(x)
    List13.List(x) = List5.List(x)
Next x
For y = 1 To wr
    List14.List(y) = List4.List(y)
Next y
For z = 1 To er
    If List10.List(z) = "0" And List11.List(z) = "0" Then
      List15.List(z) = "Critical Path"
    Else
      List15.List(z) = "***"
    End If
Next z
ReDim Limit(1 To er) As Single
Limit(1) = 7200 / er
Dim Art As Integer
ReDim DtaN(1 To er) As String
'input duration
For Art = 1 To er
  If Art > er Then
    GoTo Sat:
```

```
    ElseIf Art <= er Then
    DtaN(Art) = List5.List(Art)
    frmDisplay.Label4(Art - 1).Caption = DtaN(Art)
    frmDisplay.Label4(Art - 1).BackStyle = 0
    If List5.List(Art) = List5.List(Art + 1) Then
      frmDisplay.Label4(Art - 1).Caption = List5.List(Art + 1)
      frmDisplay.Label4(Art - 1).BackStyle = 0
    End If
   End If
Next Art
Sat:
'not main
Dim Pol2 As Integer
Dim Jat As Integer
Dim Rtg, pol As Integer
     pol = Val(List7.List(er))
     Pol2 = (pol / 10) + 1
Dim aa As Integer
ReDim DtDu(1 To Pol2 + 1) As Integer
ReDim Par(1 To Pol2 + 1) As Integer
'if pol < 200
If pol < 200 Then
   For l = 1 To Pol2 + 1
      DtDu(l) = 10 * l
      frmDisplay.Label5(l - 1).Caption = DtDu(l)
```

```
        frmDisplay.Label5(l - 1).BackStyle = 0
        On Error GoTo hjk:
    Next l
hjk:
    Par(1) = 9600 / Pol2
    'plot vertical line
    For aa = 1 To Pol2 - 1
        If aa > Pol2 Then
        GoTo Cft:
        ElseIf aa <= Pol2 Then
        frmDisplay.Line10(aa).X1 = 2160 + Par(aa)
        frmDisplay.Line10(aa).X2 = 2160 + Par(aa)
        frmDisplay.Line10(aa).Y1 = 360
        frmDisplay.Line10(aa).Y2 = 8160
        frmDisplay.Line10(aa).BorderWidth = 2
        frmDisplay.Line10(aa).BorderColor = vbBlue
        frmDisplay.Label5(aa - 1).FontBold = True
        frmDisplay.Label5(aa - 1).AutoSize = True
        frmDisplay.Label5(aa - 1).ForeColor = RGB(255, 0, 0)
        frmDisplay.Label5(aa - 1).Top = 600
        frmDisplay.Label5(aa - 1).Left = (2160 + Par(aa)) - ((Par(1) / 2) + 150)
        Par(aa + 1) = Par(aa) + Par(1)
        On Error GoTo Cft:
        End If
    Next aa
```

```
Cft:
'if pol >  200
ElseIf pol >= 200 Then
   Pol2 = (pol / 20) + 1
   For l = 1 To Pol2 + 1
      DtDu(l) = 20 * l
      frmDisplay.Label5(l - 1).Caption = DtDu(l)
      frmDisplay.Label5(l - 1).BackStyle = 0
      On Error GoTo jjk:
   Next l
jjk:
   Par(1) = 9600 / Pol2
   For aa = 1 To Pol2 - 1
      If aa > Pol2 Then
      GoTo jft:
      ElseIf aa <= Pol2 Then
      frmDisplay.Line10(aa).X1 = 2160 + Par(aa)
      frmDisplay.Line10(aa).X2 = 2160 + Par(aa)
      frmDisplay.Line10(aa).Y1 = 360
      frmDisplay.Line10(aa).Y2 = 8160
      frmDisplay.Line10(aa).BorderWidth = 2
      frmDisplay.Line10(aa).BorderColor = vbBlue
      frmDisplay.Label5(aa - 1).BackStyle = 0
      frmDisplay.Label5(aa - 1).FontBold = True
      frmDisplay.Label5(aa - 1).AutoSize = True
```

```
        frmDisplay.Label5(aa - 1).ForeColor = RGB(255, 0, 0)
        frmDisplay.Label5(aa - 1).Top = 600
        frmDisplay.Label5(aa - 1).Left = (2160 + Par(aa)) - ((Par(1) / 2) + 150)
        Par(aa + 1) = Par(aa) + Par(1)
        On Error GoTo jft:
        End If
    Next aa
jft:
End If
'plot time - schedul diagram
For i = 1 To er + 1
    If i > er Then
    GoTo ggg:
    ElseIf i < er Then
    frmDisplay.Line8(i).X1 = (Val(List6.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).X2 = (Val(List7.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).Y1 = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Line8(i).Y2 = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Line8(i).BorderWidth = 4
    frmDisplay.Line8(i).BorderColor = vbGreen
    Limit(i + 1) = Limit(i) + Limit(1)
    ElseIf i = er Then
    frmDisplay.Line8(i).X1 = (Val(List6.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).X2 = (Val(List7.List(i)) * (9600 / (Pol2 * 10))) + 2160
    frmDisplay.Line8(i).Y1 = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Line8(i).Y2 = (960 + Limit(i)) - (Limit(1) / 2)

    frmDisplay.Line8(i).BorderWidth = 4

    frmDisplay.Line8(i).BorderColor = vbGreen

    End If

Next i

ggg:

'plot horizontal line

For i = 1 To er + 1

    If i > er Then

    GoTo pgg:

    ElseIf i < er Then


    frmDisplay.Line9(i).X1 = 240

    frmDisplay.Line9(i).X2 = 11760

    frmDisplay.Line9(i).Y1 = 960 + Limit(i)

    frmDisplay.Line9(i).Y2 = 960 + Limit(i)

    frmDisplay.Line9(i).BorderWidth = 2

    frmDisplay.Line9(i).BorderColor = vbBlue

    frmDisplay.Label3(i - 1).BackStyle = 0

    frmDisplay.Label3(i - 1).Caption = List3.List(i)

    frmDisplay.Label3(i - 1).AutoSize = True

    frmDisplay.Label3(i - 1).ForeColor = RGB(200, 100, 50)

    frmDisplay.Label3(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)

    frmDisplay.Label3(i - 1).Left = 600

    frmDisplay.Label3(i - 1).FontBold = True
frmDisplay.Label4(i - 1).BackStyle = 0
frmDisplay.Label4(i - 1).FontBold = True
frmDisplay.Label4(i - 1).AutoSize = True
frmDisplay.Label4(i - 1).ForeColor = RGB(255, 0, 100)
frmDisplay.Label4(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)
frmDisplay.Label4(i - 1).Left = 1480
Limit(i + 1) = Limit(i) + Limit(1)
ElseIf i = er Then
frmDisplay.Line9(i).X1 = 240
frmDisplay.Line9(i).X2 = 11760
frmDisplay.Line9(i).Y1 = 960 + Limit(i)
frmDisplay.Line9(i).Y2 = 960 + Limit(i)
frmDisplay.Line9(i).BorderWidth = 2
frmDisplay.Line9(i).BorderColor = vbBlue
frmDisplay.Label3(i - 1).BackStyle = 0
frmDisplay.Label3(i - 1).Caption = List3.List(i)
frmDisplay.Label3(i - 1).AutoSize = True
frmDisplay.Label3(i - 1).ForeColor = RGB(200, 100, 50)
frmDisplay.Label3(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)
frmDisplay.Label3(i - 1).Left = 600
frmDisplay.Label3(i - 1).FontBold = True
frmDisplay.Label4(i - 1).BackStyle = 0
frmDisplay.Label4(i - 1).FontBold = True
frmDisplay.Label4(i - 1).AutoSize = True
frmDisplay.Label4(i - 1).ForeColor = RGB(255, 0, 100)
```

```
    frmDisplay.Label4(i - 1).Top = (960 + Limit(i)) - (Limit(1) / 2)
    frmDisplay.Label4(i - 1).Left = 1480
    End If
Next i
pgg:
Load frmDisplay
frmDisplay.Show
Me.Hide
End If
End Sub
Private Sub m3_Click()
Frame2.Visible = False
Frame1.Visible = True
End Sub
Private Sub m4_Click()
End
End Sub.
```

BIBLIOGRAPHY

1. Hegde, B. K. Production Management: Text and Cases. New Delhi: Prentice-Hall of India Private, 1972.

2. Pinney, William E. Management Science: An Introduction to Quantitative Analysis for Management. NY: Harper & Row, 1982.

3. Prabhu, Vas. Production Management and Control. London: McGraw-Hill, 1986.

4. Thierauf, Robert I. Management Science: A Model Formulation Approach with Computer Applications. Columbus: Charles R. Merrill, 1985.

5. Thompson, Gerald E. Management Science: An Introduction to Modern Quantitative Analysis and Decision Making. NY: McGraw-Hill, 1976.