# Software Requirements & Specifications

Courtesy:

Roger Pressman, Ian Sommerville &

Prof Rajib Mall

# Representation of Complex Processing System

- The processing logic is that, if certain conditions are satisfied, then do something and check for further conditions and so on.
- In a text form if we write a very complex processing logic it is likely that some of the conditions can be overlooked.
- Semiformal techniques to represent complex processing logic are as follow:
  - **Decision Trees**
  - **Decision Tables**

# Decision Trees:

- Decision trees:
  - Edges of a decision tree represent conditions
  - Leaf nodes represent action to be performed.
- A decision tree gives a graphic view of:
  - Logic involved in decision making
  - Corresponding action taken.

It gives a graphic view of the logic that is involved and the actions that will be taken, if certain logic conditions hold and from the tree we can easily detect if certain conditions are missed.

# Representation of Complex Processing System

A **library membership management software (LMS)** should support the following three options- **new member, renewal, and cancel membership**. When the ***new member*** option is selected, the software should ask the member's name, address and phone number. If proper information is entered, the software should create a membership record for the new member and print a bill for the annual membership charge and the security deposit payable. If the ***renewal*** option is chosen, the LMS should ask the member's name and his membership number and check whether he is valid member. If the member details entered are valid, then the membership expiry date in the membership record should be updated and the annual membership charge payable by the member should be printed. If the membership details entered are invalid, an error message should be displayed. If the ***cancel membership*** option is selected and the name of a valid member is entered, then the membership is cancelled, a cheque for the balance amount due to the member is printed and his membership record is deleted.

# Decision Trees: Library Management System

- A library membership automation software (LMS) should support the following three options:
  - New Member
  - Renewal
  - Cancel membership

  If a user selects create member then there are three options which come out: New member, membership renewal, and cancel membership.
  - When the new member option is selected, then the software asks details about the member:
    - Name,
    - Address,
    - Phone number,etc.

# Decision Trees: Library Management System

- When the new member option is selected, then the software asks details about the member:
  - Name,
  - Address,
  - Phone number,etc.

- If proper information is entered,
  - **A membership record for the member is created,**
  - **A bill is printed for the annual membership charge plus the security deposit payable.**

# Decision Trees: Library Management System

- If the **renew** option is chosen,
  - LMS asks the member's name and his membership number
    - Checks whether he is valid member.
  - If the name represents a valid member,
    - The membership expiry date is updated and the annual membership is bill printed,
    - Otherwise an error message is displayed.

# Decision Trees: Library Management System

- If the **cancel membership option is selected** and the name of a valid member is entered,
  - The membership is cancelled,
  - A cheque for the balance amount due to the member is printed
  - The membership record is deleted.

**If any other option is chosen, then there is a print error message.**

# Decision Table: Library Management System

- A decision table shows the decision-making logic and the corresponding action taken in a tabular or a matrix from.
- Decision tables represent:
  - Which variables are to be tested
  - What actions are to be taken if the conditions are true
  - The order in which decision making is performed.
- The upper rows of the table specify the variables or conditions to be evaluated and lower rows specify the actions to be taken when an evaluation test is satisfied.

# Decision Table: Library Management System

- A column in the table is called a rule.
- A Rule implies
  - If certain condition combination is true, then the corresponding action is executed.

# Decision Table: Library Management System

| • Conditions | | | | | |
|---|---|---|---|---|---|
| Valid selection | NO | YES | YES | YES | |
| New member | -- | YES | NO | NO | |
| Renewal | -- | NO | YES | NO | |
| Cancellation | | -- | NO | NO | YES |
| • Actions | | | | | |
| Display error message | | | | | |
| Ask member's name etc. | | | | | |
| Build customer record | -- | -- | | -- | |
| Generate bill | -- | | | -- | |
| Ask membership details | | -- | | | |
| Update expiry date | -- | -- | -- | | |
| Print cheque | | -- | -- | -- | |
| Delete record | -- | -- | -- | | |

# Example:

- A student grading system promotes the students to the next session on the basis of the university rules. A student has to complete subjects of 52 credits in a session and maintain 4.00 SGPA to take admission in the next session. Otherwise, a fresh admission will be allowed in the same course. Represent this information in the decision table and decision tree.

| **Conditions** | **Condition Entry** | | |
|---|---|---|---|
| Credits >= 52 | No | Yes | Yes |
| SGPA >= 4 | --- | No | Yes |
| **Actions** | **Action Entry** | | |
| Incomplete Credits | ✓ | | |
| Promotion | | | ✓ |
| Readmission | | ✓ | |

# Example:

- Develop a decision tree and decision table for the following:
  - If the flight is more than half-full and ticket cost is more than Rs. 3000, free meals are served unless it is a domestic flight. The meals are charged on all domestic flights.

| **Conditions** | | | | | |
|---|---|---|---|---|---|
| Domestic Flight | Yes | No | No | No | |
| Ticket Cost > 3000 | --- | No | Yes | Yes | |
| Occupancy>50% | --- | --- | No | Yes | |
| **Actions** | | | | | |
| Free meals | | | | ✓ | |
| Charged meals | ✓ | ✓ | ✓ | | |

# Software Project Management

Courtesy:

Roger Pressman, Ian Sommerville &

Prof Rajib Mall

# Software Project Management

- Effective management of a software project is crucial to the success of any software development project.
- Several project in the past failed
  - Not for want of competent technical professionals,
  - Neither for lack of resources, but…
  - Due to the use of faulty project management practices.
- Hence, there is need to learn the latest software project management techniques
- Bob Hughes, Mike Cotterell, and Rajib Mall, "*Software Project Management*", Tata MCGraw-Hill, 2011.
- The main goal of the software project management is to enable a group of developers to work effectively towards the successful completion of a project.

# Objectives:

- To Investigate why management of software projects is much more complex than managing many other types of projects.
- To outline the main responsibilities and activities of a software project manager.
- To Illustrate the project planning activities.
- To discuss estimation and scheduling techniques.
- To provide an overview of the risks and configuration management activities.

# Software Project management complexities:

- The main factors contributing to the complexity of managing a software project, as identified by ***Brooks*** are as follows:
- **Invisibility:** Invisibility of software makes it difficult to assess the progress of a project.
- **Changeability:** Frequent changes to the requirements arise due to changes in the business practices, changes to the hardware or underlying software.
- **Complexity:** Millions of functions (parts) interact with each other in many ways- data coupling, serial and concurrent runs, state transition, control dependency, file sharing etc.

# Software Project management complexities:

- The main factors contributing to the complexity of managing a software project, as identified by ***Brooks*** are as follows:
- **Uniqueness:** Unlike projects in other domain, the every software projects much different from the others.
- **Exactness of the solutions:** Reusing parts of one software product in another is difficult.
- **Team oriented and intellect-intensive work:** Each member needs to typically interact, review, and interface with several other members.

# Responsibilities of a Software Project Manager:

- Project managers take responsibilities for ***project proposal writing***, ***project cost estimation, scheduling, project staffing, software process tailoring, project monitoring and control, software configuration management, risk management, managerial report writing and presentation, and interfacing with clients.***
- We can broadly classify a project manager's varied responsibilities into following two major categories:
  - ***Project Planning***
  - ***Project monitoring and control***

# Responsibilities of a Software Project Manager:

- ***Project Planning*** involves estimating several characteristics of a project and then planning the project activities based on these estimates made.
- The focus of ***project monitoring and control*** activities is to ensure that the software development proceeds as per plan.
- Three ***skills*** that are most critical to successful project management are the following:
  - ***Knowledge of project management techniques***
  - ***Decision taking capabilities***
  - ***Previous experience in managing similar projects***

# Project Planning:

- The project manager performs the following activities during project planning:
  - ***Estimation***
    - ***Cost :*** *How much is it going to cost to develop the software product**?***
    - ***Duration:*** *How long is it going to take to develop the product**?***
    - ***Effort:*** *How much effort would be necessary to develop the product**?***
  - ***Scheduling:***
  - ***Staffing:***
  - ***Risk Management:***
  - ***Miscellaneous Plans:*** *Includes quality assurance plan, and configuration management plan, etc.*

# Project Planning:


**Figure: Precedence ordering among planning activities**

- In the **sliding window planning** technique, starting with an initial plan, the project is planned more accurately over a number of stages.

# Metrics for Project Size Estimation:

- The project size is measure of the problem complexity in terms of the effort and time required to develop the product.
- Accurate estimation of project size is central to satisfactory estimation of all other project parameters such as effort, completion time, and total project cost.
- The size of the project is obviously not
  - the number of bytes that the source code occupies,
  - neither is it the size of the executable code.
- Two metrics are popularly being used to measure size….
  - ***Lines of Code (LOC) and***
  - ***Function Point (FP)***

# Lines of Code (LOC):

- Simplest among all metrics available to measure project size.
- Measures the size of a project by counting the number of source instructions in the developed program.
  - Ignored comment lines and header lines.
- Accurate estimation of the LOC count at the beginning of a project is a very difficult task.
  - May be possible by using some form of systematic guess work.
  - LOC of the leaf level modules are small enough to be predicted.
  - By adding the estimate for all leaf level modules together, project manager compute total size.

# Problem with LOC metric:

- **LOC is a measure of coding activity alone:**
  - The implicit assumption made by the LOC Metric that *the overall product development effort is solely determined from the coding effort alone is flawed.*
- **LOC count depends on the choice of specific instructions:**
  - Even for the same programming problem, different programmers might come up with programs having very different LOC counts.
  - Example: One programmer might write *several instruction on single line, whereas another might split* a single instruction across several lines.
  - This situation does not improve, even if *language tokens* are counted instead of lines of code.

# Problem with LOC metric:

- **LOC measure correlates poorly with the quality and efficiency of the code:**
  - *Larger code size does not imply better quality of code or higher efficiency.*
  - *A piece of poor and sloppily written piece of code can have larger number of source instructions than a piece that is efficient and has been thoughtfully written.*
- **LOC metric penalizes use of higher-level programming languages and code reuse:**
  - *Programmer consciously uses several library routines, then the LOC count will be lower.*
  - *Smaller program size -----→ lower program effort ----→ Discourage code reuse by developers.*

# Problem with LOC metric:

- **LOC metric measures the lexical complexity of a program and does not address the more important issues of logical and structural complexities:**
  - *Two programs with equal LOC counts*
  - *A program with complex logic require much more effort than a program with simple logic.*
- **It is very difficult to accurately estimate LOC of the final program from problem specification:**
  - *From a project manager's perspective, the biggest shortcoming of the LOC metric is that the LOC count is very difficult to estimate during project planning stage, and can only be accurately computed after the software development is complete.*

# Function Point (FP) Metric:

- Function point metric was proposed by Albrecht and Gaffney.
- The function point metric is based on the idea that a software product supporting many features would certainly be of larger size than a product with less number of features.
- The size of software product is directly dependent on the number of different high-level functions or features it supports.
- Different features may take very different amounts of efforts to develop. Hence, just determining the number of functions to be supported may not yield very accurate result

# Function Point (FP) Metric:

- Function point metric
  - Counts the number of input and output data items and
  - The number of files accessed by the function.

- **Albrecht** postulated that in addition to the number of basic functions that a software performs, its **size also depends on the number of files and the number of interfaces associated with it.**

# Function Point (FP) Metric Computation:

- The size of the software product is computed using different characteristics of the product identified in it's requirements specifications.
- It is computed using the following three steps:
- **Step 1:** Compute the unadjusted function point (UFP) using a heuristic expression.
- **Step 2:** Refine UFP to reflect the actual complexities of the different parameters used in UFP computations.
- **Step 3:** Compute FP by further refining UFP to account for the specific characteristics of the project that can influence the entire development effort.

# UFP Computation:

- The unadjusted function points (UFP) is computed as the weighted sum of five characteristics of a product as shown:

***UFP = (Number of inputs)*4 + (Number of outputs)*5 + (Number of inquiries)*4 + (Number of files)*10 + (Number of interfaces)*10***

- The weight associated with the five characteristics were determined empirically by Albrecht through data gathered from many projects.
- UFP computed here is the gross indicator of the problem size. Each parameter has been implicitly assumed to be of average complexity.

# Refine Parameters:

- Some input values may be extremely complex, some very simple, etc.
- In order to take this issue into account, UFP is refined by taking into account the complexities of the parameters of UFP computation.
- The complexity of each parameter is graded into three broad categories- Simple, Average, and Complex.

| Types | Simple | Average | Complex |
|---|---|---|---|
| **Input (I)** | 3 | 4 | 6 |
| **Output (O)** | 4 | 5 | 7 |
| **Inquiry (E)** | 3 | 4 | 6 |
| **Number of files (F)** | 7 | 10 | 15 |
| **Number of interfaces** | 5 | 10 | 15 |

**Refinement of Function Point Entities**

# Refine UFP based on complexity of the overall Project:

- In the final step, several factors that can impact the overall project size are considered to refine the UFP computed in step 2.
- Parameters that can influence the project size include high transaction rates, response time requirements, scope for reuse, etc.
- Albrecht identified 14 parameters that can influence the development efforts.
- Each of these 14 parameters is assigned a value from 0 (not present or no influence) to 6 (strong influence).
- The resulting numbers are summed, yielding the total ***degree of influence (DI)***.
- A ***technical complexity factor*** for the project is computed and ***the TCF is multiplied with UFP to yield FP***

# Refine UFP based on complexity of the overall Project:

| Requirement for reliable backup and recovery |
|---|
| Requirement for data communication |
| Extent of distributed processing |
| Performance requirements |
| Expected operational environment |
| Extent of online data entries |
| Extent of multiscreen or multi-operation online data input |
| Extent of online updating of master files |
| Extent of complex inputs, outputs, online queries and files |
| Extent of complex data processing |
| Extent that currently developed code can be designed for reuse |
| Extent of conversion and installation included in the design |
| Extent of multiple installations in an organization and variety of customer organization |
| Extent of change and focus on ease of use |

**Table: Function Point Relative Complexity Adjustment Factors**

# Refine UFP based on complexity of the overall Project:

- The Technical complexity Factor express the overall impact of the corresponding project parameters on the development effort.
- ***TCF (Technical Complexity Factor) / CAA (Complexity Adjustment Attribute)*** is computed as ***0.65 + 0.01 * DI.*** As DI can vary from 0 to 84, TCF can vary from 0.65 to 1.49.
- Finally, FP is given as the product of UFP and TCF i.e. ***FP = UFP * TCF.***

# Example:

- ***Determine the function point measure of the size*** of the following supermarket software. A supermarket needs to develop the following software to encourage regular customers. For this, the customer needs to supply his/her residence address, telephone number, and driving license number. Each customer who register for this scheme is assigned *a unique customer number (CN)* by the computer. Based on the generated CN, a clerk manually prepares a customer identity card after getting market manager's signature on it. A customer can present his customer identity card to the check out staff when he makes any purchase. In this case, the value of his purchase is credited against his CN. At the end of each year, the supermarket intends to award a surprise gifts to 10 customers who make the highest total purchase over the year. Also, it intends to award a 22 caret gold coin to every customer whose purchase exceeded Rs. 10,000. The entries against the CN are reset on the last day of every year after the prize winner's lists are generated. Assume that various project characteristics determining the complexity of software development to be average.

# Project Estimation Techniques:

- The different parameters of a project that need to be estimated include-
  - Project size,
  - Effort required to complete the project,
  - Project duration, and
  - Cost.
- These parameters helps in quoting an appropriate project cost, to form the basis for the resource planning and scheduling.
- Categories as:
  - Empirical estimation techniques
  - Heuristic techniques
  - Analytical estimation techniques

# Empirical Estimation Techniques:

- Based on making an educated guess of the project parameters.
- Prior experience with development of similar products is helpful.
- Easy to use and give reasonably accurate estimates.
- Although this approach is based on common sense and subjective decisions, over the years different activities involved in estimation have been formalized.
  - Expert Judgement
  - Delphi Techniques

# Empirical Estimation Techniques: Expert Judgement

- Expert judgement is widely used size estimation technique.
- Experts makes an educated guess about the problem size after analyzing the problem thoroughly.
- Expert estimates cost of the different components (modules or subsystems) then combines the estimates for the individual modules to arrive at the overall estimates.

# Empirical Estimation Techniques: Expert Judgement

- The outcome of the expert judgement technique is subject to human errors and individual bias.
- There is a possibility that the experts may overlooked some factors.
- There is a possibility that the an expert making an estimate may not have relevant experience and knowledge of all aspects of a project.
- Due to these factors, the size estimation arrived at by the judgement of a single expert may be far from being accurate.

# Empirical Estimation Techniques: Delphi Cost Estimation

- It is carried out by a team comprising a group of experts and a coordinator.
- The coordinator provides each estimator with a copy of the software requirements specification (SRS) document and a form for recording his cost estimate.
- Estimators complete their individual estimates anonymously and submit them to the coordinator.
- The coordinator prepares the summary of the response of all the estimators, and also includes any unusual rationale noted by any of the estimators.
- The prepared summary information is distributed to the estimators. Based, on summary the estimators re-estimates.

# Heuristic Techniques:

- Heuristic techniques assume that the relationships that exist among the different project parameters can be satisfactorily modelled using suitable mathematical expressions.
- Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the values of the independent parameters in the corresponding mathematical expression.
- Heuristics estimation models can be broadly categorized as – Single variable and multivariable models.

# Heuristic Techniques:

- Single variable estimation models assume that various project characteristics can be predicted based on a single previously estimated basic (independent) characteristics of the software such as its size.
- Assumes that the relationship between a parameter to be estimated and the corresponding independent parameter can be characterized by an expression of the following form:
- ***Estimated parameters*** = $c_1 \times e^{d_1}$
- Where, e represents a characteristic of the software that has already been estimated (independent variable). ***Estimated parameter*** is the dependent parameter to be estimated.
- The dependent parameter to be estimated could be effort, project duration, staff size, etc.

# Heuristic Techniques:

- ***Estimated parameters*** = $c_1 \times e^{d_1}$
- Where, e represents a characteristic of the software that has already been estimated (independent variable). ***Estimated parameter*** is the dependent parameter to be estimated.
- The dependent parameter to be estimated could be effort, project duration, staff size, etc., $c_1$ and $d_1$ are constants.
- The value of the constant $c_1$ and $d_1$ are usually determined using data collected from past projects.
- The COCOMO model is an example of single variable cost estimation model.

# Heuristic Techniques:

- A multivariable cost estimation model assumes that a parameter can be predicted based on the values of more than one independent parameters.
- ***Estimated parameters*** = $c_1 \times p^{d_1} + c_1 \times p^{d_1} + \ldots..$
- Where, $p_1, p_2, \ldots.$ are the basic (independent characteristics of the software already estimated and $c_1, c_2, d_1, d_2, \ldots$ *are constants. Values of these constants are usually determined from an analysis of historical data.*
- Multivariable estimation models are expected to give more accurate estimates compared to the single variable models, since a project parameter is typically influenced by several independent parameter to different extent.

# Analytical Estimation Technique

- Analytical estimation techniques derive the required results starting with certain basic assumption regarding a project.
- Unlike empirical and heuristic techniques, analytical techniques do have certain scientific basis.
- Halstead's Software Science is useful for estimating software maintenance efforts.

# COCOMO- A Heuristic Estimation Technique

- COnstructive COst estimation MOdel (COCOMO) was proposed by Boehm.
- COCOMO prescribes a three stage process for project estimation:
  - Basic COCOMO
  - Intermediate COCOMO
  - Complete COCOMO
- In the first stage, an initial stage estimate is arrived. Over the next two stages, the initial estimate is refined to arrive at a more accurate estimate.

# COCOMO- A Heuristic Estimation Technique

- Boehm postulated that any software development project can be classified into one of the following three categories based on the development complexity- **organic, semidetached, and embedded.**
- Based on the category of a software development project, he gave different sets of formulas to estimate the effort and duration from the size estimate.
- To classify a project into the identified categories, need to consider not only the characteristics of the product but also those of the **development team and development environment.**
- Brooks states- **Application : Utility : System :: 1:3:9**

# COCOMO- A Heuristic Estimation Technique

- **Organic:** If the project deals with developing a well understood application program, the size of development team is reasonably small, and the team members are experienced in developing similar kind of projects.
- **Semidetached:** If the development team consists of mixture experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.
- **Embedded:** If the software being developed is strongly coupled to hardware, or if stringent regulations on the operational procedures exist. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.

# COCOMO- A Heuristic Estimation Technique

- **What is Person-month?**
- One person month is the effort an individual can typically put in a month.
- Person-month is considered to be an appropriate unit for measuring effort, because developers are typically assigned to a project for certain number of months.

# Project Scheduling:

- The scheduling problem in essence, consists of deciding which tasks would be taken up when and by whom.
- The first step in scheduling a software project involves identifying all the activities necessary to complete the project.
  - Schedule converts action plan into operating time table
  - Basis for monitoring and controlling project
  - Based on Work Breakdown Structure (WBS)
- The project plan must be developed to the level of showing dates when each activity should start and finish and when and how much of each resource will be required.
  - Once the plan has been refined to this level of detail we call it a project schedule.

# Project Scheduling:

- In order to schedule the project activities, a software project manager needs to do the following:
  - Identify all the major activities that need to be carried out to complete the project.
  - Break down each activity into tasks.
  - Determine the dependency among different tasks.
  - Establish the estimates for the time durations necessary to complete the tasks.
  - Represent the information in the form of an activity network.
  - Determine task starting and ending dates from the information represented in the activity network.
  - Determine the critical path. A critical path is a chain of tasks that determines the duration of the project.
  - Allocate resources to the tasks.

# Work Breakdown Structure

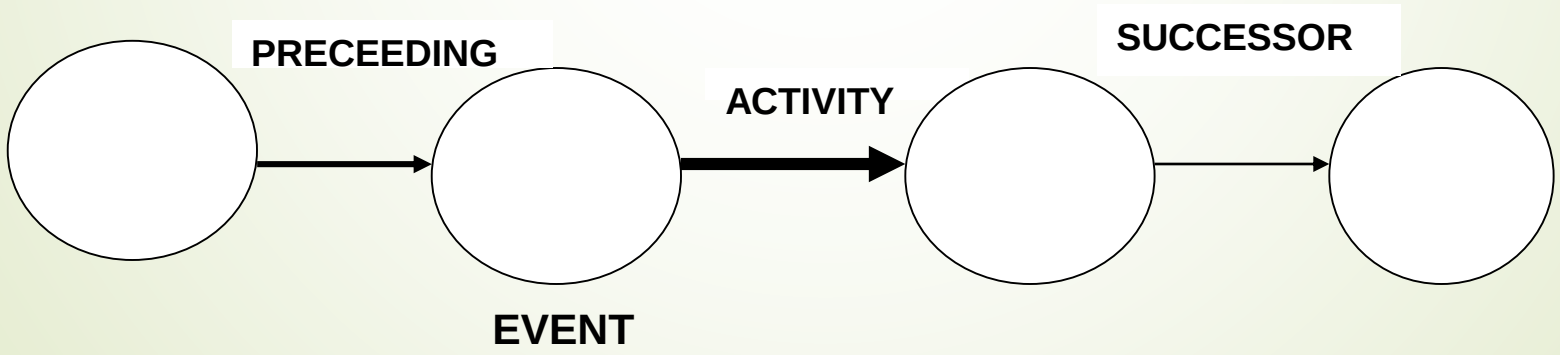
- Work breakdown structure (WBS) is used to recursively decompose a given set of activities into smaller activities.
- Tasks are the lowest level work activities in a WBS hierarchy. They also form the basic units of work that are allocated to the developer and scheduled.
- The decomposition of the activities is carried out until….
  - A leaf-level subactivity (a task) requires approximately two weeks to develop.
  - Hidden complexities are exposed, so that the job to be done is understood and can be assigned as a unit of work to one of the developers.
  - Opportunities for reuse of existing software components is identified.

# Activity Networks:

- An activity network shows the different activities making up a project, their estimated durations, and their interdependencies.
- Graphical portrayal of activities and event.
- Shows dependency relationships between tasks/activities in a project.
- Clearly shows tasks that must precede (precedence) or follow (succeeding) other tasks in a logical manner.
- Clear representation of plan – a powerful tool for planning and controlling project.

# DEFINITION OF TERMS IN A NETWORK

- **Activity :** any portions of project (tasks) which required by project, uses up resource and consumes time – may involve labor, paper work, contractual negotiations, machinery operations **Activity on Arrow (AOA) showed as arrow, AON – Activity on Node**
- **Event :** beginning or ending points of one or more activities, instantaneous point in time, also called 'nodes'
- **Network:** Combination of all project activities and the events



## Merge and Burst Events

One or more activities can start and end simultaneously at an event (Figure 8.3 a, b).

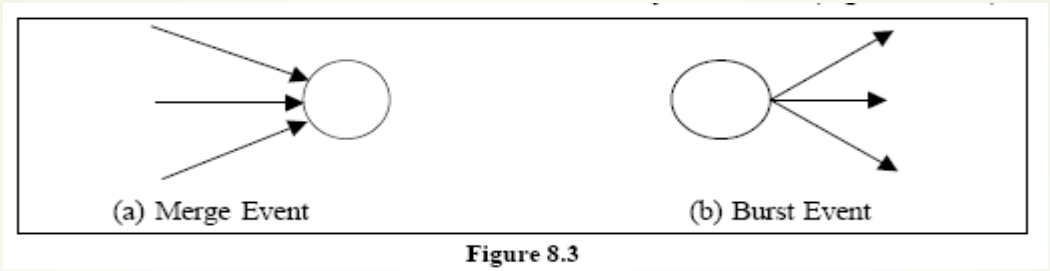



Figure 8.3

## Preceding and Succeeding Activities

Activities performed before given events are known as *preceding activities* (Figure), and activities performed after a given event are known as *succeeding activities*.

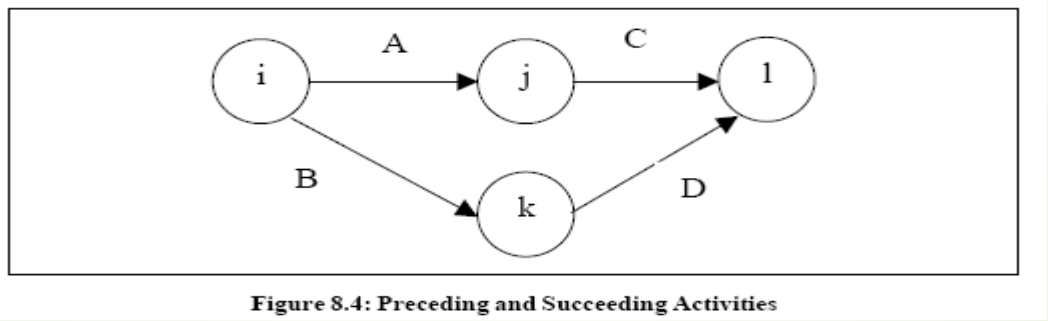



Figure 8.4: Preceding and Succeeding Activities

Activities A and B precede activities C and D respectively.

## Dummy Activity

An imaginary activity which does not consume any resource and time is called a ***dummy activity. Dummy activities are simply used to represent a connection between events in*** order to maintain a logic in the network. It is represented by a dotted line in a network, see Figure 8.5.

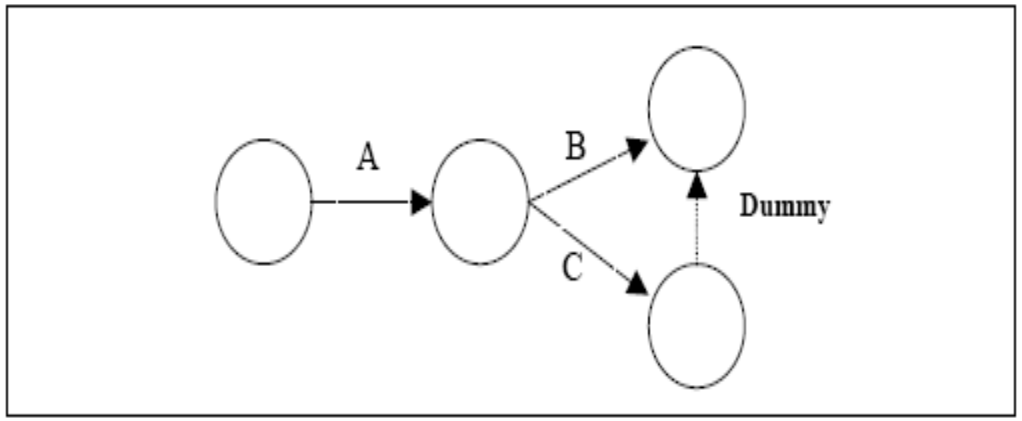



Figure 8.5: Dummy Activity

# ERRORS TO BE AVOIDED IN CONSTRUCTING A NETWORK



**a. Two activities starting from a tail event must not have a same end event.** To ensure this, it is absolutely necessary to introduce a dummy activity, as shown in Figure 8.6.

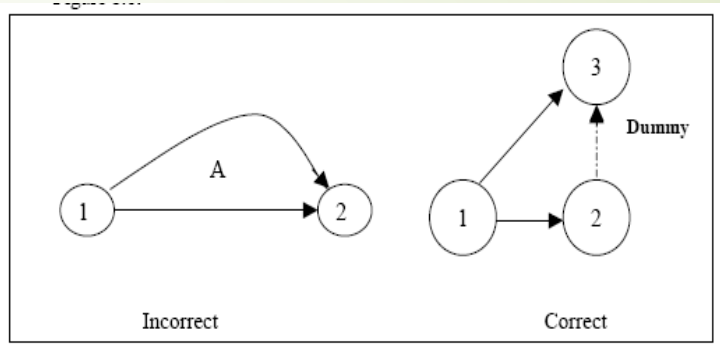


Figure 8.6: Correct and Incorrect Activities

**b. Looping error should not be formed in a network,** as it represents performance of activities repeatedly in a cyclic manner, as shown below in Figure 8.7.

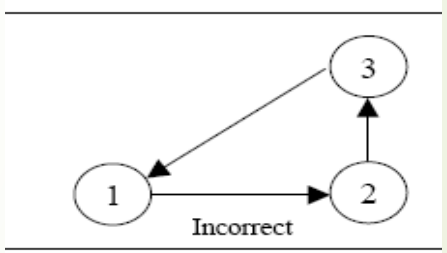


Figure 8.7: Looping Error

**c.** In a network, **there should be only one start event and one ending event** as shown below, in Figure 8.8.

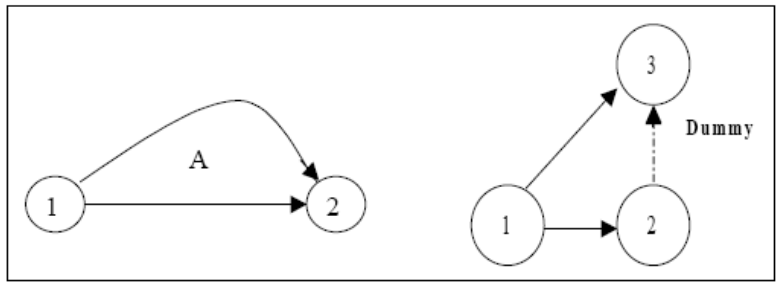
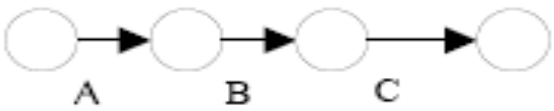


Figure 8.8: Only One Start and End Event

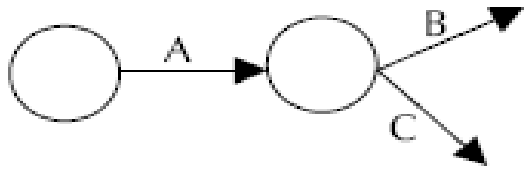
# Some conventions of network diagram are shown in Figure (a), (b), (c), (d) below:

(a)



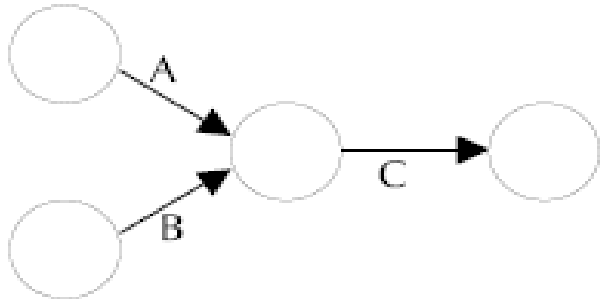

Activity B can be performed only after completing activity A, and activity C can be performed only after completing activity B.

(b)



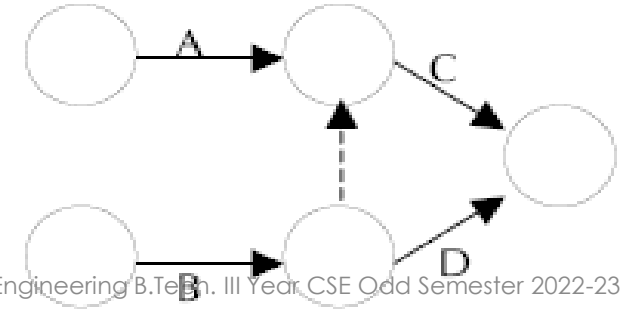

Activities B and C can start simultaneously only after completing A.

(c)




Activities A and B must be completed before start of activity C.

(d)




Activity C must start only after completing activities A and B. But activity D can start after completion of activity B.

# Example 1- A simple network

Consider the list of four activities for making a simple product:

| Activity | Description | Immediate predecessors |
|---|---|---|
| A | Buy Plastic Body | - |
| B | Design Component | - |
| C | Make Component | B |
| D | Assemble product | A,C |

**Immediate predecessors** for a particular activity are the activities that, when completed, enable the start of the activity in question.

# Sequence of activities

- Can start work on activities A and B anytime, since neither of these activities depends upon the completion of prior activities.
- Activity C cannot be started until activity B has been completed
- Activity D cannot be started until both activities A and C have been completed.
- The graphical representation (next slide) is referred to as the PERT/CPM network

# Network of Four Activities

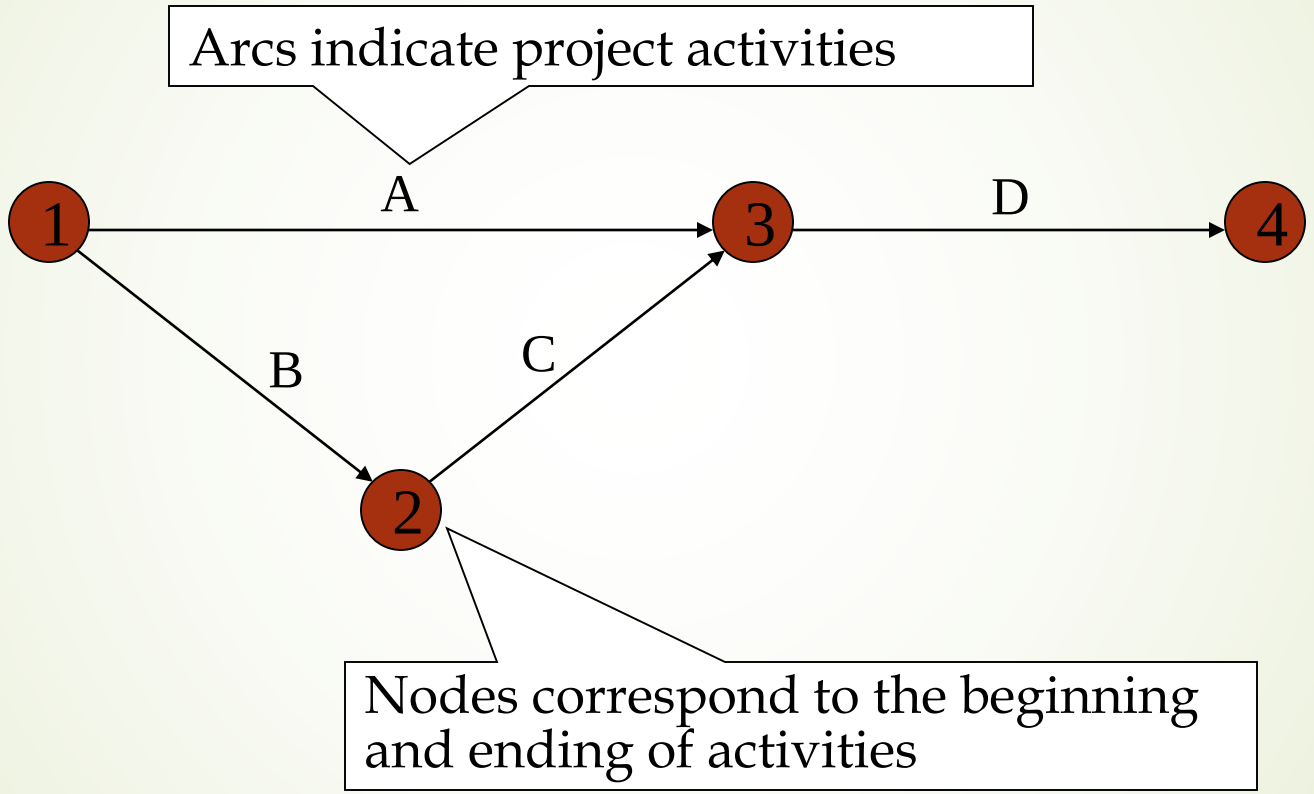


**Table 8.1: Sequence of Activities for House Construction Project**

| Name of the activity | Starting and finishing event | Description of activity | Predecessor | Time duration (days) |
|---|---|---|---|---|
| A | (1,2) | Prepare the house plan | -- | 4 |
| B | (2,3) | Construct the house | A | 58 |
| C | (3,4) | Fix the door / windows | B | 2 |
| D | (3,5) | Wiring the house | B | 2 |
| E | (4,6) | Paint the house | C | 1 |
| F | (5,6) | Polish the doors / windows | D | 1 |

***Solution:***

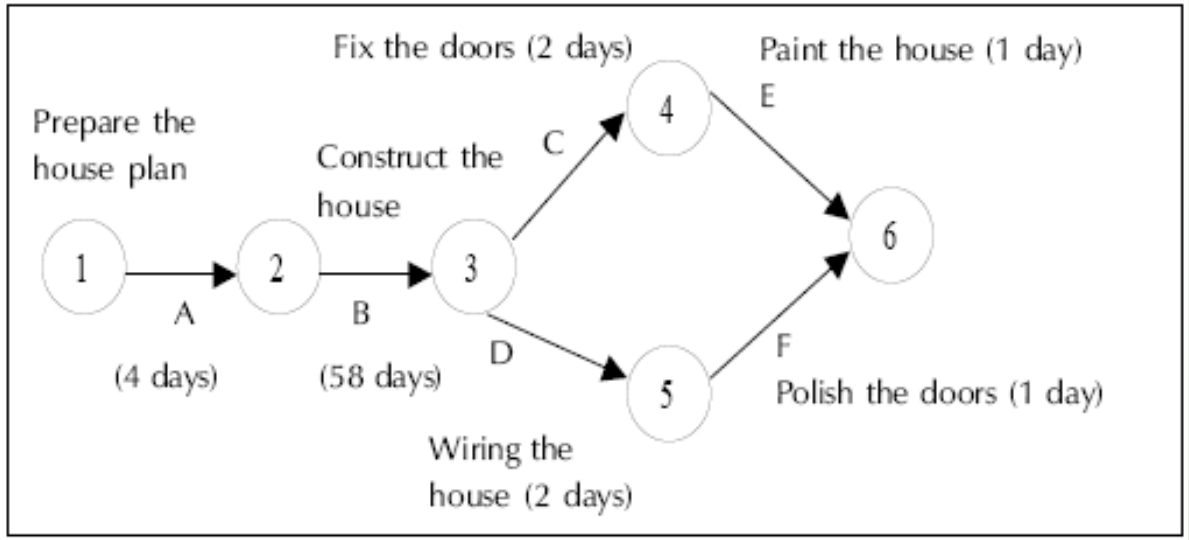



**Figure 8.11: Network diagram representing house construction project.**

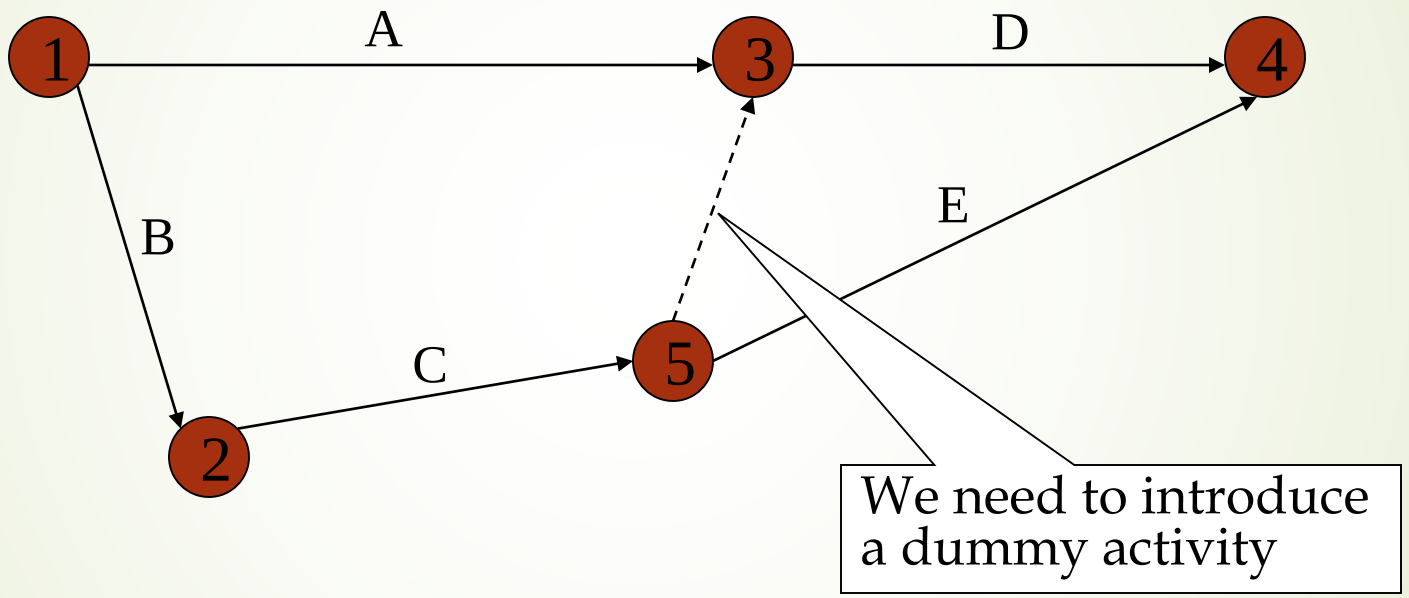
# Example 2

Develop the network for a project with following activities and immediate predecessors:

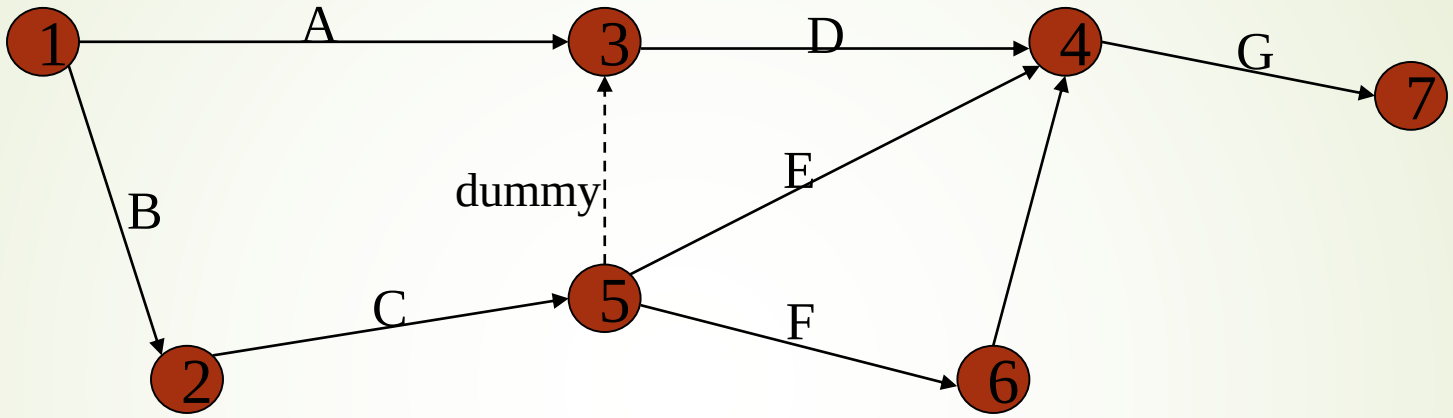
| Activity | Immediate predecessors |
|---|---|
| A | - |
| B | - |
| C | B |
| D | A, C |
| E | C |
| F | C |
| G | D,E,F |

Try to do for the first five (A,B,C,D,E) activities

# Network of first five activities



# Network of Seven Activities




- Note how the network correctly identifies D, E, and F as the immediate predecessors for activity G.
- Dummy activities is used to identify precedence relationships correctly and to eliminate possible confusion of two or more activities having the same starting and ending nodes
- Dummy activities have no resources (time, labor, machinery, etc) – purpose is to PRESERVE LOGIC of the network

# Activity Networks:

- Determine the Activity network representation for the MIS development project for which the relevant data is given in below Table.

| Task Number | Task | Duration | Dependent on Tasks |
|---|---|---|---|
| T1 | Specification | 15 | - |
| T2 | Design database | 45 | T1 |
| T3 | Design GUI | 30 | T1 |
| T4 | Code database | 105 | T2 |
| T5 | Code GUI part | 45 | T3 |
| T6 | Integrate and Test | 120 | T4 and T5 |
| T7 | Write user manual | 60 | T1 |

# Critical Path Method:

- CPM and PERT are operation research techniques.
- A **path** in the activity network graph is any set of consecutive nodes and edges in the graph from starting node to the last node.
- A **critical path** consists of a set of dependent tasks that need to be performed in a sequence and which together take the longest time to complete.
- **A critical task is one with a zero slack time.** A path from the start node to the finish node containing only critical tasks is **called a critical path**.

# Critical Path Method:

- **Minimum Time (MT):** It is the minimum time required to complete the project. It is computed by determining the maximum of all paths from start to finish.
- **Earliest Start (ES):** It is the time of a task is the maximum of all paths from the start to this task. The ES for a tasks is the ES of the previous tasks plus the duration of the preceding tasks.
- **Earliest Finish (EF):** The EF for a task is the sum of the earliest start time of the task and the duration of the task.
- **Latest Start (LS):** It is the difference between MT and the maximum of all paths from this task to the finish. The LS can be computed by subtracting the duration of the subsequent task from the LS of the subsequent task.

# Critical Path Method:

- **Latest Finish (LF):** LF indicates the latest time by which a task can finish without affecting the final completion time of the project. A task completing beyond its LF would cause project delay.
- LF of a task can be obtained by subtracting maximum of all paths from this task to finish from MT.
- **Slack Time (ST):** The slack time (or float time) is the total time that a task may be delayed before it will affect the end time of the project.
- The slack time indicates the flexibility in starting and completion of tasks. ST for a task is LS-ES and can equivalently be written as LF-EF.

# Scheduling with activity time

| Activity | Immediate predecessors | Completion Time (week) |
|---|---|---|
| A | - | 5 |
| B | - | 6 |
| C | A | 4 |
| D | A | 3 |
| E | A | 1 |
| F | E | 4 |
| G | D,F | 14 |
| H | B,C | 12 |
| I | G,H | 2 |
| | | Total …… 51 |

This information indicates that the total time required to complete activities is 51 weeks. However, we can see from the network that several of the activities can be conducted simultaneously (A and B, for example).

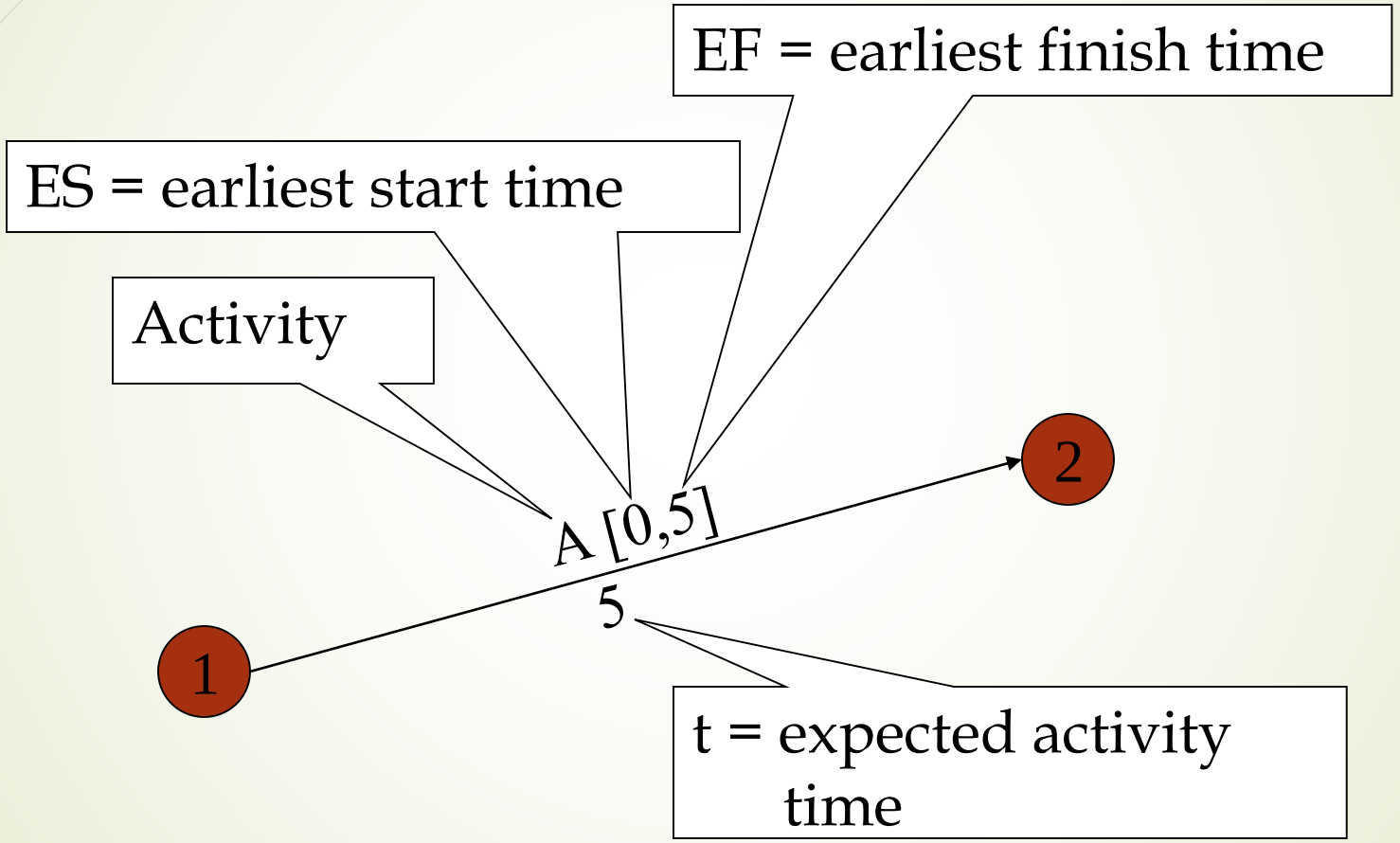
# Earliest start & Earliest finish time :

- We are interested in the longest path through the network, i.e., the critical path.
- Starting at the network's origin (node 1) and using a starting time of 0, we compute an ***earliest start*** (ES) and ***earliest finish*** (EF) time for each activity in the network.
- The expression EF = ES + t can be used to find the earliest finish time for a given activity.

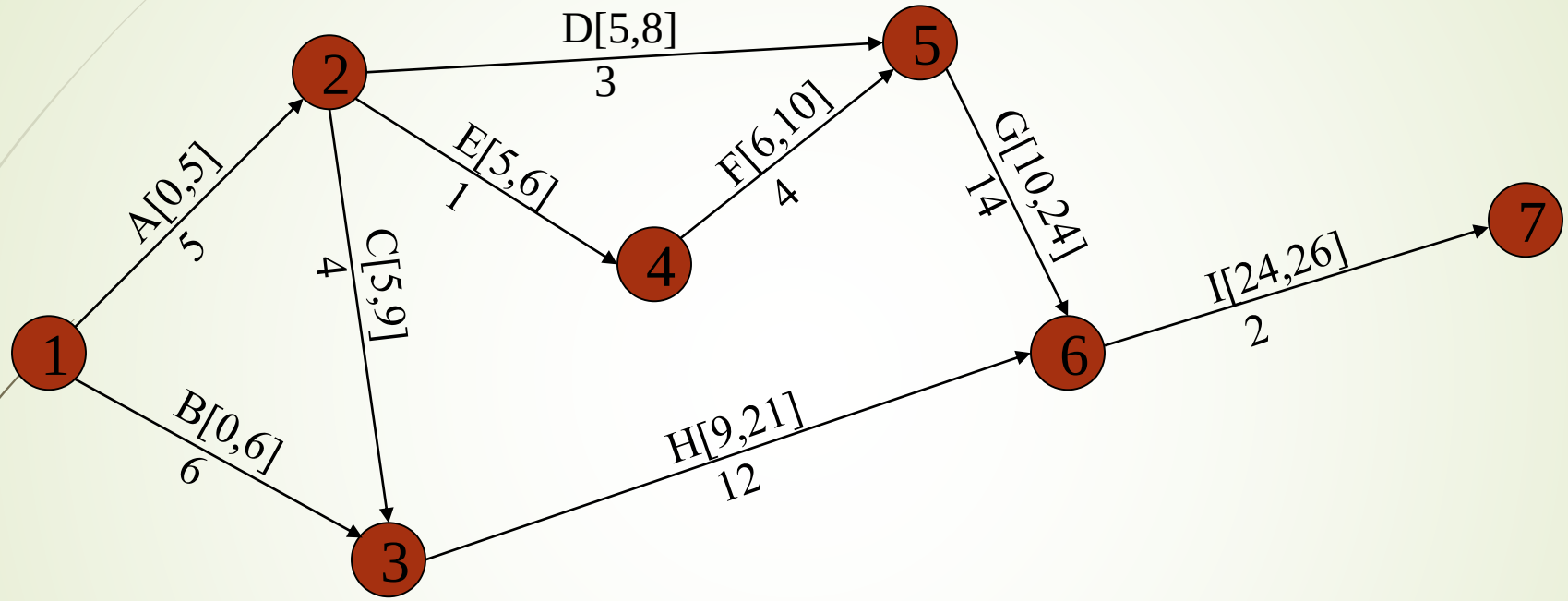
  For example, for activity A, ES = 0 and t = 5; thus the earliest finish time for activity A is

  EF = 0 + 5 = 5

# Arc with ES & EF time



# Network with ES & EF time




**Earliest start time rule:**

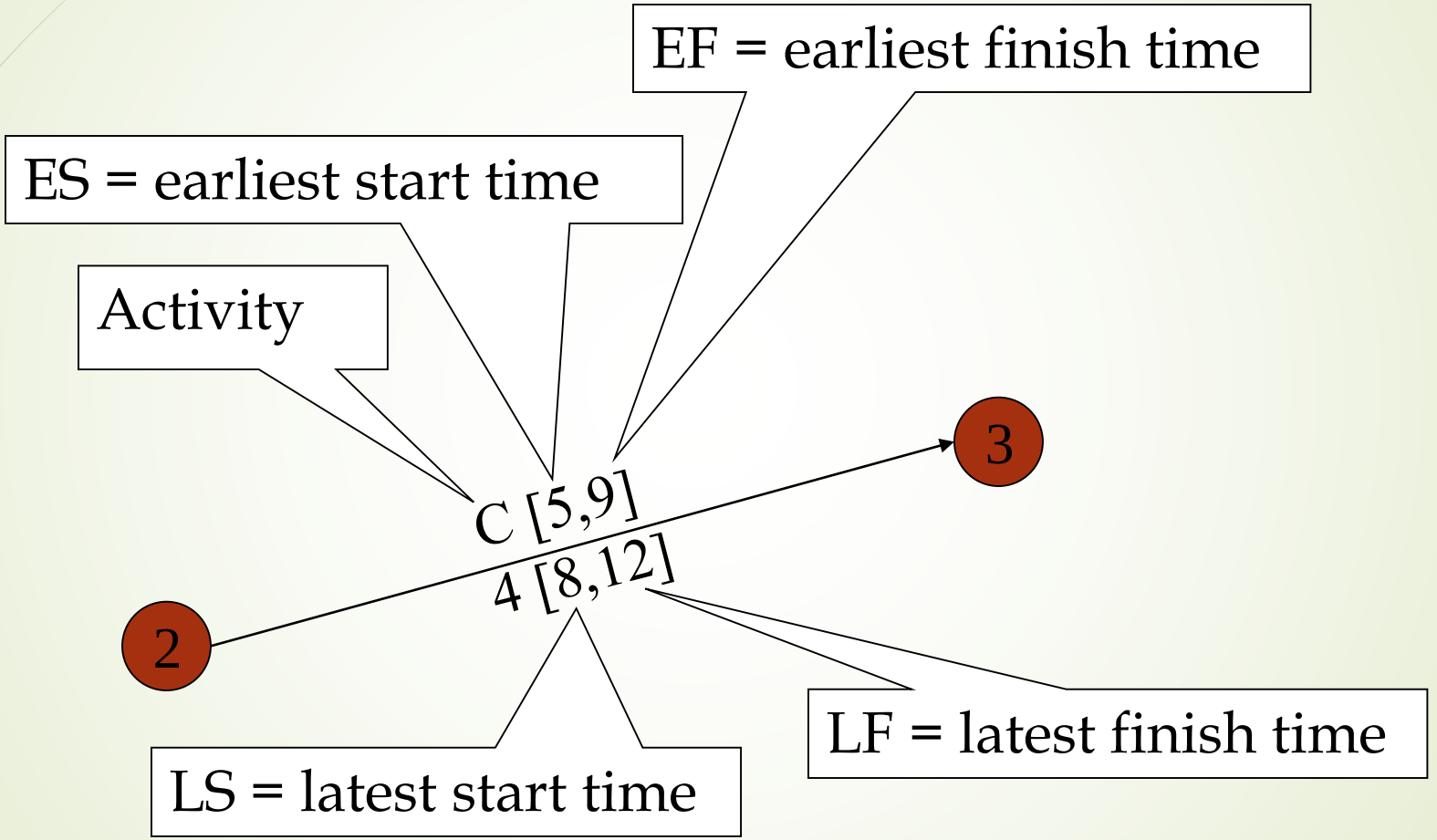
The earliest start time for an activity leaving a particular node is equal to the **largest** of the earliest finish times for all activities entering the node.

# Latest start & latest finish time

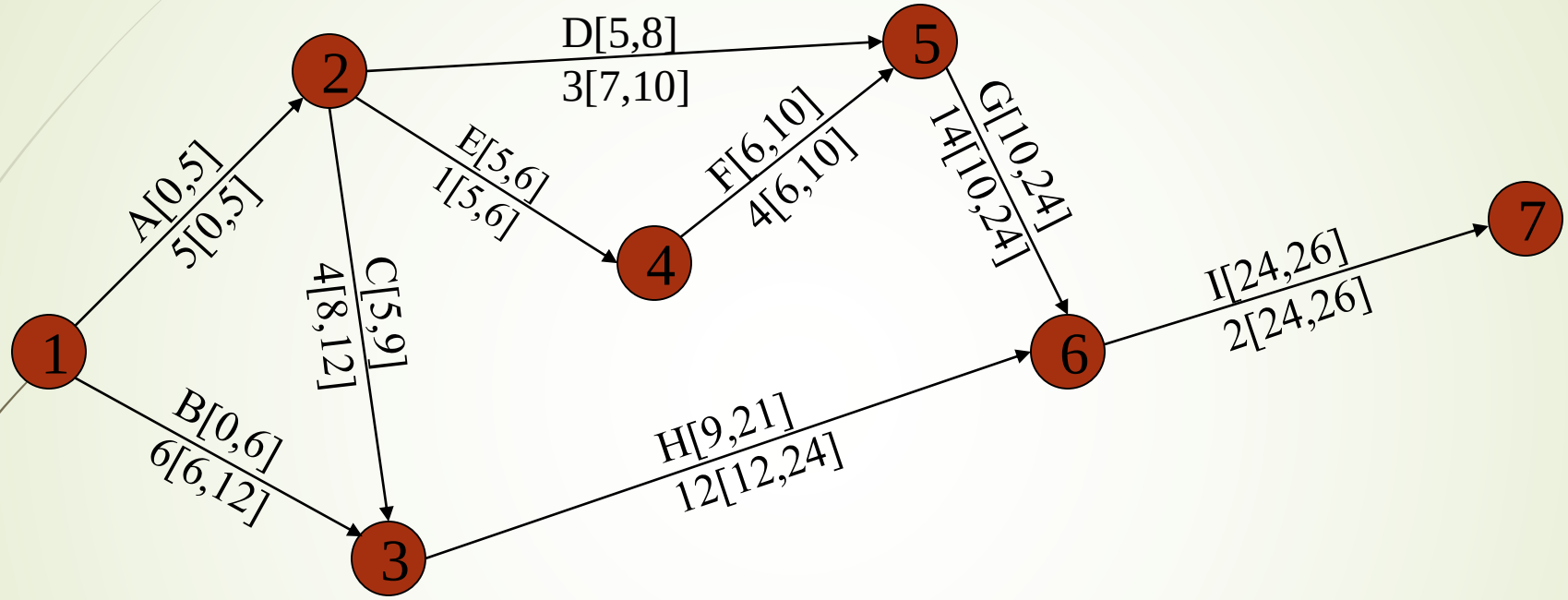
- To find the critical path we need a backward pass calculation.
- Starting at the completion point (node 7) and using a ***latest finish*** time (LF) of 26 for activity I, we trace back through the network computing a ***latest start*** (LS) and latest finish time for each activity
- The expression **LS = LF – t** can be used to calculate latest start time for each activity. For example, for activity I, LF = 26 and t = 2, thus the latest start time for activity I is

  LS = 26 – 2 = 24

# Activity, duration, ES, EF, LS, LF



# Network with LS & LF time




Latest finish time rule:

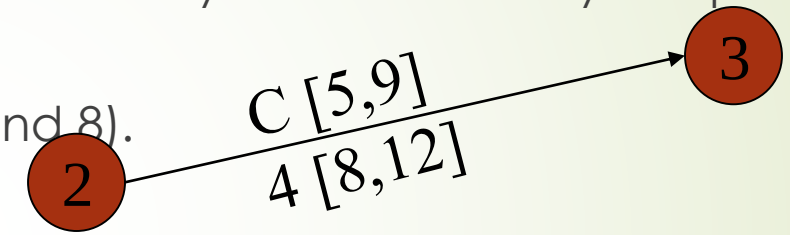
The latest finish time for an activity entering a particular node is equal to the **smallest** of the latest start times for all activities leaving the node.

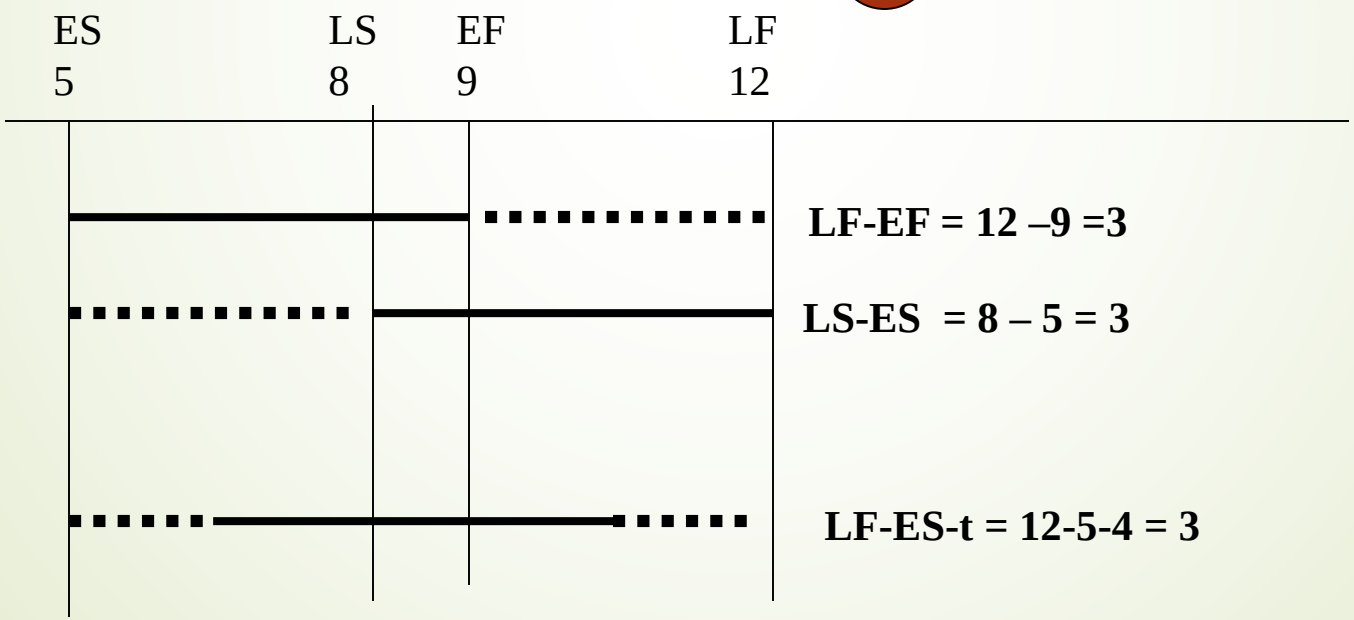
# Slack or Free Time or Float

Slack is the length of time an activity can be delayed without affecting the completion date for the entire project.

For example, slack for C = 3 weeks, i.e Activity C can be delayed up to 3 weeks

(start anywhere between weeks 5 and 8).



# Activity schedule for our example

| Activity | Earliest start (ES) | Latest start (LS) | Earliest finish (EF) | Latest finish (LF) | Slack (LS-ES) | Critical path |
|---|---|---|---|---|---|---|
| A | 0 | 0 | 5 | 5 | 0 | Yes |
| B | 0 | 6 | 6 | 12 | 6 | |
| C | 5 | 8 | 9 | 12 | 3 | |
| D | 5 | 7 | 8 | 10 | 2 | |
| E | 5 | 5 | 6 | 6 | 0 | Yes |
| F | 6 | 6 | 10 | 10 | 0 | Yes |
| G | 10 | 10 | 24 | 24 | 0 | Yes |
| H | 9 | 12 | 21 | 24 | 3 | |
| I | 24 | 24 | 26 | 26 | 0 | Yes |

# IMPORTANT QUESTIONS

- **What is the total time to complete the project?**
  - 26 weeks if the individual activities are completed on schedule.
- **What are the scheduled start and completion times for each activity?**
  - ES, EF, LS, LF are given for each activity.
- **What activities are *critical* and must be completed as scheduled in order to keep the project on time?**
  - Critical path activities: A, E, F, G, and I.
- **How long can *non-critical* activities be delayed before they cause a delay in the project's completion time**
  - Slack time available for all activities are given.

# Importance of Float (Slack) and Critical Path

1. Slack or Float shows how much allowance each activity has, i.e how long it can be delayed without affecting completion date of project.

2. Critical path is a sequence of activities from start to finish with zero slack. Critical activities are activities on the critical path.

3. Critical path identifies the minimum time to complete project.

4. If any activity on the critical path is shortened or extended, project time will be shortened or extended accordingly.

# Importance of Float (Slack) and Critical Path (cont)

5. So, a lot of effort should be put in trying to control activities along this path, so that project can meet due date. If any activity is lengthened, be aware that project will not meet deadline and some action needs to be taken.

6. If can spend resources to speed up some activity, do so only for critical activities.

7. Don't waste resources on non-critical activity, it will not shorten the project time.

8. If resources can be saved by lengthening some activities, do so for non-critical activities, up to limit of float.

# Critical Path Method (CPM)

- Use the activity network and determine the ES, EF, LS, LF and slack time value of the every tasks for MIS development project for which the relevant data is given in below Table.

| Task Number | Task | Duration | Dependent on Tasks |
|---|---|---|---|
| T1 | Specification | 15 | - |
| T2 | Design database | 45 | T1 |
| T3 | Design GUI | 30 | T1 |
| T4 | Code database | 105 | T2 |
| T5 | Code GUI part | 45 | T3 |
| T6 | Integrate and Test | 120 | T4 and T5 |
| T7 | Write user manual | 60 | T1 |

# Construct the CPM Network and Find critical path :

**Table 8.5: Project Schedule**

| Activity | Name | Time | Activity | Name | Time (days) |
|---|---|---|---|---|---|
| 1-2 | A | 4 | 5-6 | G | 4 |
| 1-3 | B | 1 | 5-7 | H | 8 |
| 2-4 | C | 1 | 6-8 | I | 1 |
| 3-4 | D | 1 | 7-8 | J | 2 |
| 3-5 | E | 6 | 8-10 | K | 5 |
| 4-9 | F | 5 | 9-10 | L | 7 |

# Construct the CPM Network and Find critical path :

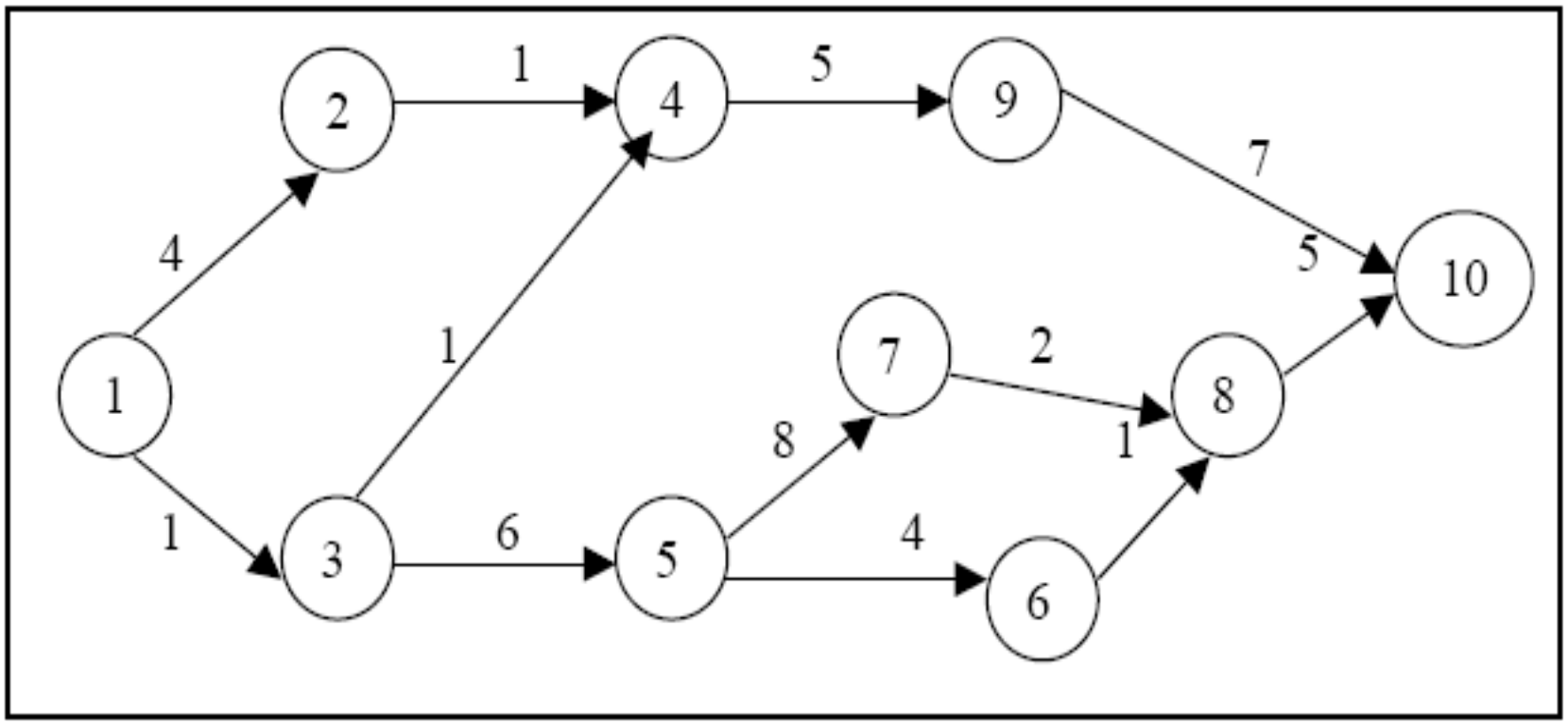



**Figure 8.16: Activity Network Diagram**

Table 8.6: Various Activities and their Floats

| Activity | Activity Name | Normal Time (tij) | Earliest Time (TE) | | Latest Time (TL) | | Total Float |
|---|---|---|---|---|---|---|---|
| | | | Start (ES) | Finish (EF) | Start (LS) | Finish (LF) | |
| 1-2 | A | 4 | 0 | 4 | 5 | 9 | 5 |
| 1-3 | B | 1 | 0 | 1 | 0 | 1 | 0 |
| 2-4 | C | 1 | 4 | 5 | 9 | 10 | 5 |
| 3-4 | D | 1 | 1 | 2 | 9 | 10 | 8 |
| 3-5 | E | 6 | 1 | 7 | 1 | 7 | 0 |
| 4-9 | F | 5 | 5 | 10 | 10 | 15 | 5 |
| 5-6 | G | 4 | 7 | 11 | 12 | 16 | 5 |
| 5-7 | H | 8 | 7 | 15 | 7 | 15 | 0 |
| 6-8 | I | 1 | 11 | 12 | 16 | 17 | 5 |
| 7-8 | J | 2 | 15 | 17 | 15 | 17 | 0 |
| 8-10 | K | 5 | 17 | 22 | 17 | 22 | 0 |
| 9-10 | L | 7 | 10 | 17 | 15 | 22 | 5 |

From the Table , we observe that the activities 1 – 3, 3 – 5, 5 – 7, 7 – 8 and 8 – 10 are critical activities as their floats are zero.

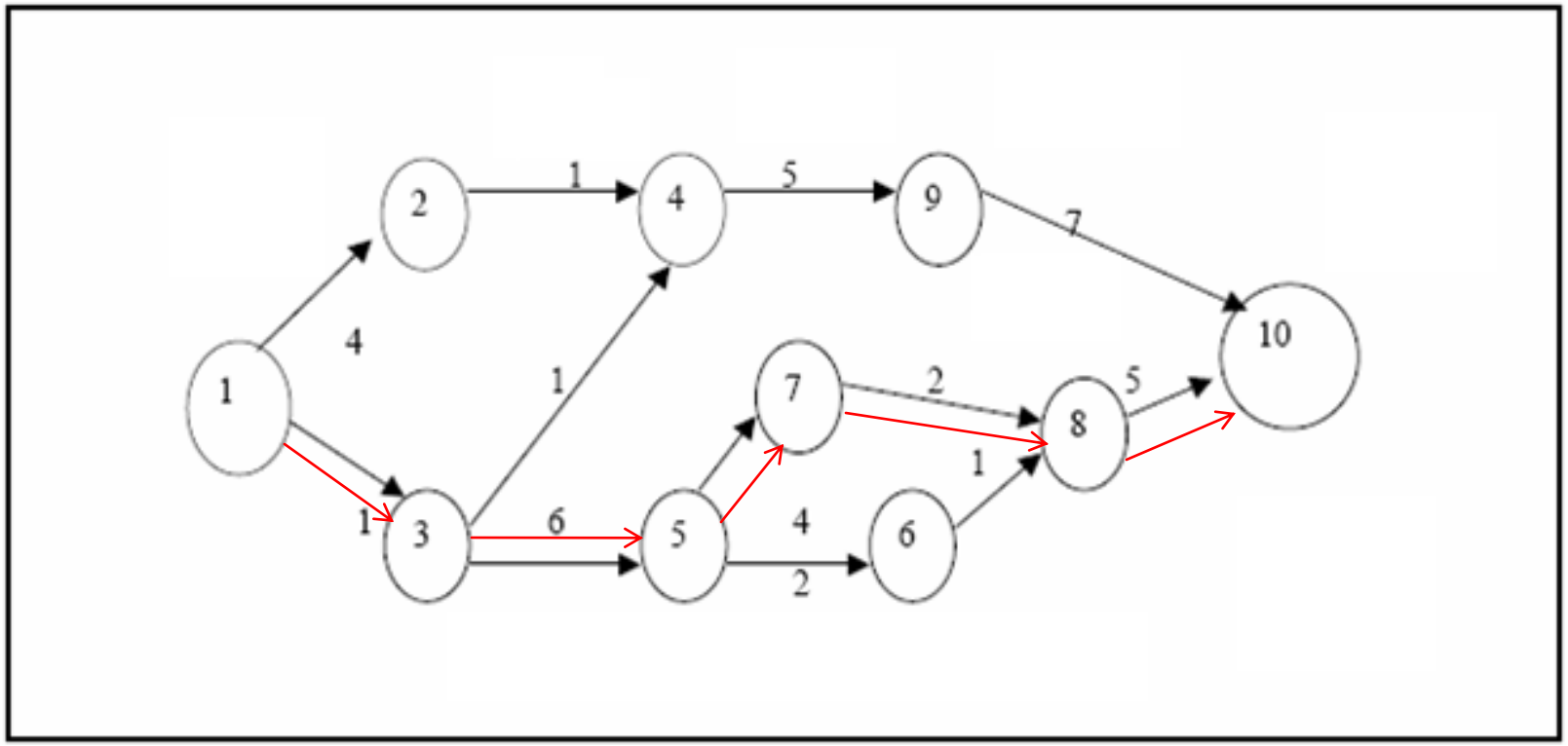



Figure 8.17: Critical Path of the Project

# Project Evaluation Review Technique (PERT):

- The CPM can be used to determine the duration of a project, but does not provide any indication of the probability of meeting that schedule.
- PERT charts can be used to determine the probabilistic times for reaching various project mile stones.
- A PERT chart represents the statistical variations in the project estimates assuming these to be normal distribution.
- Each task is annotated with three estimates: Optimistic (O) or base case, Most likely estimate (M) and Pessimistic or Worst Case (W)

# Project Evaluation Review Technique (PERT):

- In the critical path method, the time estimates are assumed to be known with certainty. In certain projects like research and development, new product introductions, it is difficult to estimate the time of various activities.
- Hence PERT is used in such projects with a probabilistic method using three time estimates for an activity, rather than a single estimate, as shown in Figure

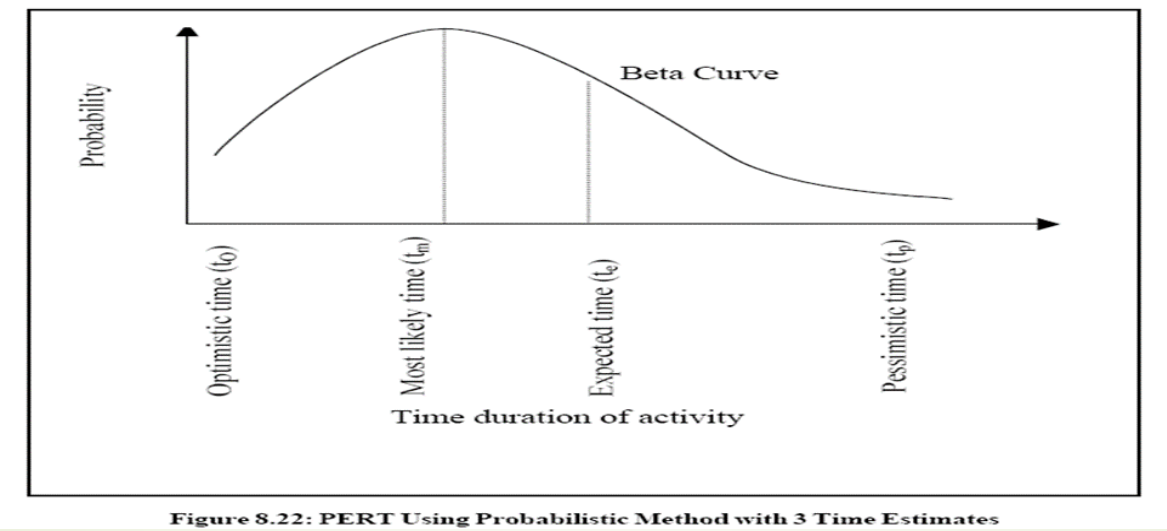



Figure 8.22: PERT Using Probabilistic Method with 3 Time Estimates

# Project Evaluation Review Technique (PERT):

**Optimistic time (To):**

It is the shortest time taken to complete the activity. It means that if everything goes well then there is more chance of completing the activity within this time.

**Most likely time (Tm):**

It is the normal time taken to complete an activity, if the activity were frequently repeated under the same conditions.

**Pessimistic time (Tp):**

It is the longest time that an activity would take to complete. It is the worst time estimate that an activity would take if unexpected problems are faced.

# PERT For Dealing With Uncertainty:

- So far, times can be estimated with relative certainty, confidence.
- For many situations this is not possible, e.g. Research development, new products and projects etc.
- Use 3 time estimates
  - **Tm= m = most likely time estimate,**
  - **To = a = optimistic time estimate,**
  - **Tp = b = pessimistic time estimate, and**

**Expected Value (TE) = (a + 4m + b) /6**

**Variance ($\sigma^2$) (V) = ( ( b – a) / 6 )$^2$**

**Std. Deviation (δ) = SQRT (V)**

# PERT For Dealing With Uncertainty:

- The probability of completing the project within the scheduled time (Ts) or contracted time may be obtained by using the standard normal deviate where Te is the expected time of project completion.
- Probability of completing the project within the scheduled time is,

$$Z_0 = \frac{T_s - T_e}{\sqrt{\Sigma\sigma^2 \text{ in critical path}}}$$

$$P(T \leq T_s) = P(Z \leq Z_0) \text{ (from normal tables)} \quad \ldots$$

## *Example*

An R & D project has a list of tasks to be performed whose time estimates are given in the Table, as follows.

**Table 8.11: Time Estimates for R & D Project**

| Activity i j | Activity Name | $T_0$ | $t_m$ ( in days) | $t_p$ |
|---|---|---|---|---|
| 1-2 | A | 4 | 6 | 8 |
| 1-3 | B | 2 | 3 | 10 |
| 1-4 | C | 6 | 8 | 16 |
| 2-4 | D | 1 | 2 | 3 |
| 3-4 | E | 6 | 7 | 8 |
| 3-5 | F | 6 | 7 | 14 |
| 4-6 | G | 3 | 5 | 7 |
| 4-7 | H | 4 | 11 | 12 |
| 5-7 | I | 2 | 4 | 6 |
| 6-7 | J | 2 | 9 | 10 |

a. Draw the project network.
b. Find the critical path.
c. Find the probability that the project is completed in 19 days. If the probability is less than 20%, find the probability of completing it in 24 days.

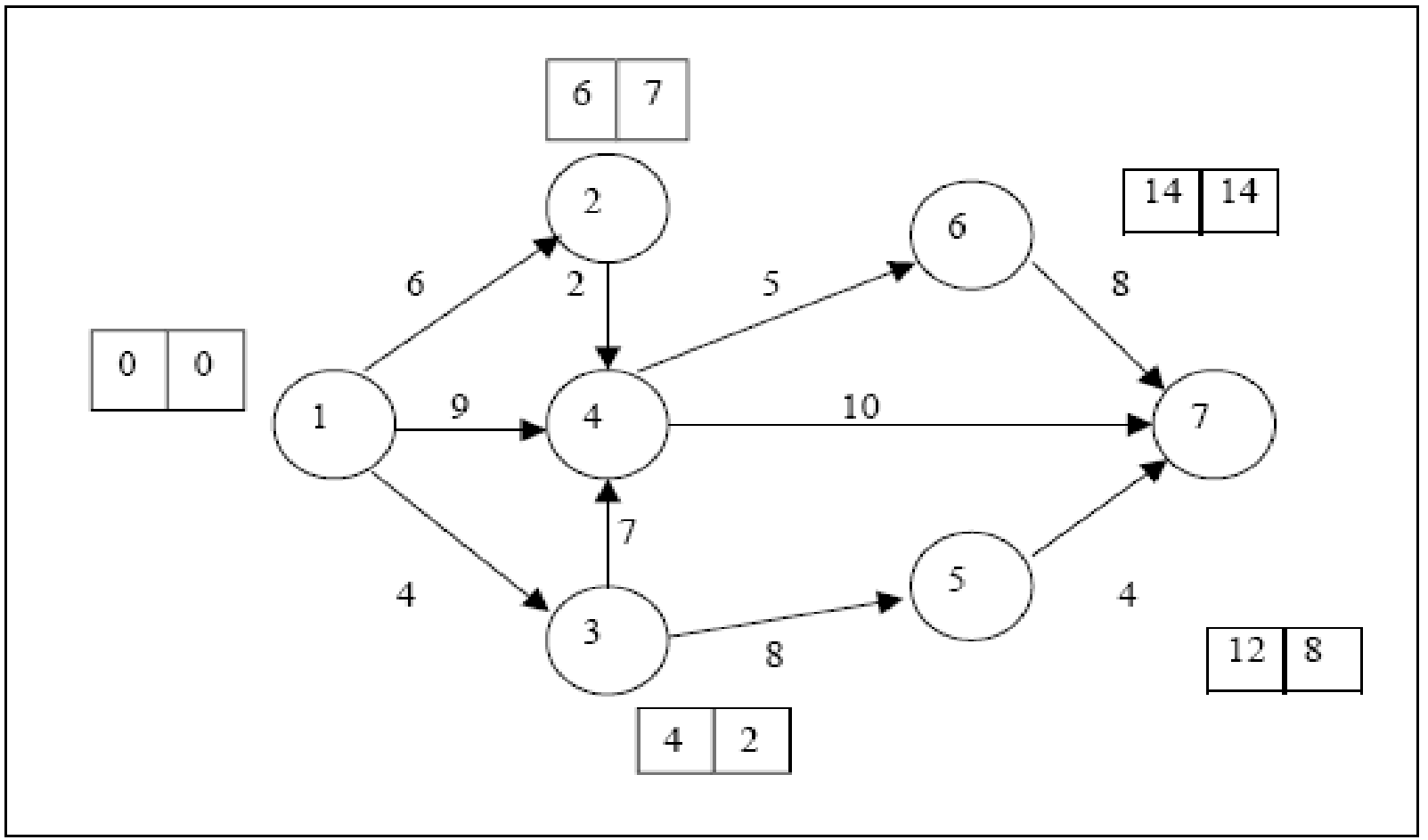



**Figure 8.23: Network Diagram**

Table 8.12: $T_e$ & $s^2$ Calculated

| Activity | $T_o$ | $T_m$ | $T_p$ | $T_e$ | $\sigma^2$ |
|---|---|---|---|---|---|
| 1-2 | 4 | 6 | 8 | 6 | 0.444 |
| 1-3 | 2 | 3 | 10 | 4 | 1.777 |
| 1-4 | 6 | 8 | 16 | 9 | 2.777 |
| 2-4 | 1 | 2 | 3 | 2 | 0.111 |
| 3-4 | 6 | 7 | 8 | 7 | 0.111 |
| 3-5 | 6 | 7 | 14 | 8 | 1.777 |
| 4-6 | 3 | 5 | 7 | 5 | 0.444 |
| 4-7 | 4 | 11 | 12 | 10 | 1.777 |
| 5-7 | 2 | 4 | 6 | 4 | 0.444 |
| 6-7 | 2 | 9 | 10 | 8 | 1.777 |

# The probability of completing the project within 19 days is given by, P (Z< Zo)

To find Zo ,

$$Z_0 = \left(\frac{T_s - T_e}{\sqrt{\Sigma\sigma \text{ in critical path}}}\right)$$

$$= \left(\frac{19-22}{\sqrt{2.777+0.444+1.777}}\right)$$

$$= \left(\frac{-3}{\sqrt{5}}\right) = -1.3416$$

we know, P (Z <Z Network Model o) = z (-1.3416) (from normal tables, z (-1.3416) = 0.0901)
= 0.0901
= 9.01%

Thus, the probability of completing the R & D project in 19 days is 9.01%.

*Since the probability of completing the project in 19 days is less than 20% As in question, we find the probability of completing it in 24 days.*

$$Z_0 = \frac{T_s - T_e}{\sqrt{\Sigma\sigma \text{ in critical path}}}$$

$$= \left(\frac{24-22}{\sqrt{5}}\right) = \left(\frac{2}{\sqrt{5}}\right) = 0.8944 \text{ days}$$

we know, P (Z <Z Network Model ) = z (0.8944) (from normal tables, z (0.8944) = 0.8132)
= 0.8132
= 81.32%

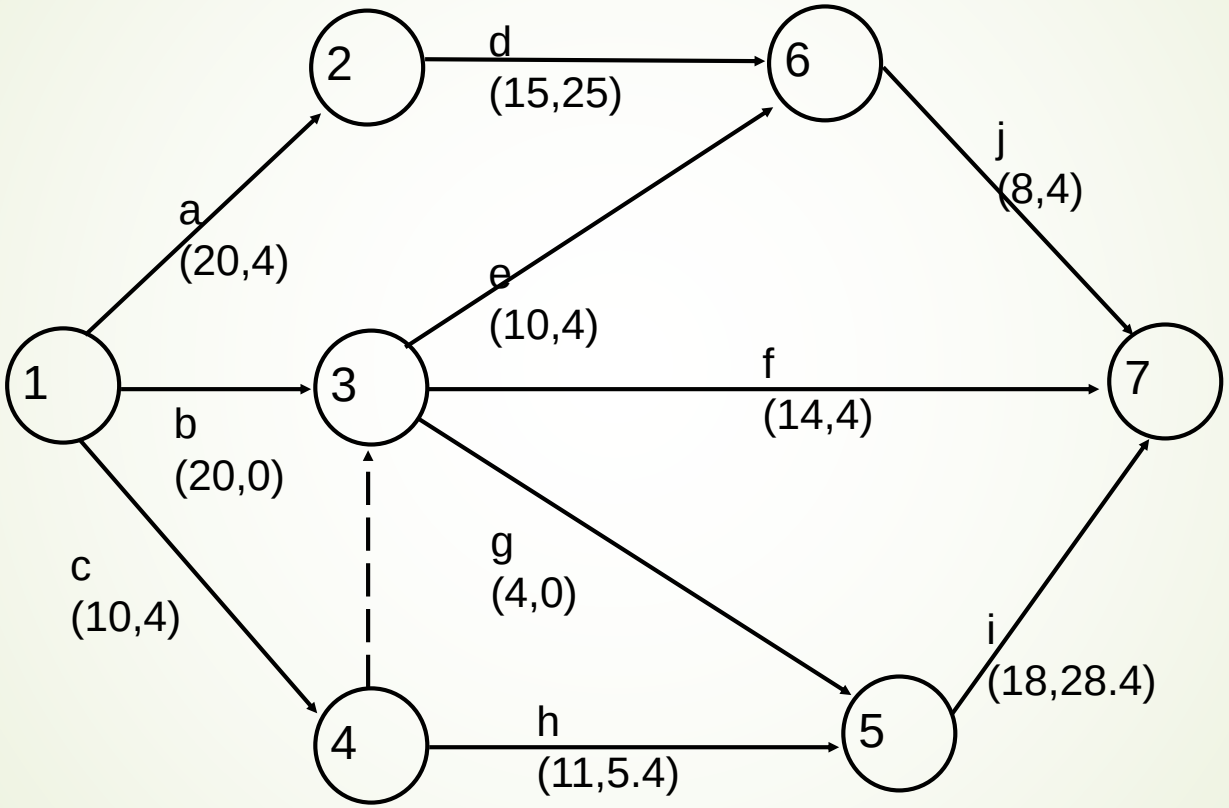
# Precedence And Project Activity Times

| Activity | Immediate Predecessor | Optimistic Time | Most Likely Time | Pessimistic Time |
|---|---|---|---|---|
| a | - | 10 | 22 | 22 |
| b | - | 20 | 20 | 20 |
| c | - | 4 | 10 | 16 |
| d | a | 2 | 14 | 32 |
| e | b,c | 8 | 8 | 20 |
| f | b,c | 8 | 14 | 20 |
| g | b,c | 4 | 4 | 4 |
| h | c | 2 | 12 | 16 |
| I | g,h | 6 | 16 | 38 |
| j | d,e | 2 | 8 | 14 |

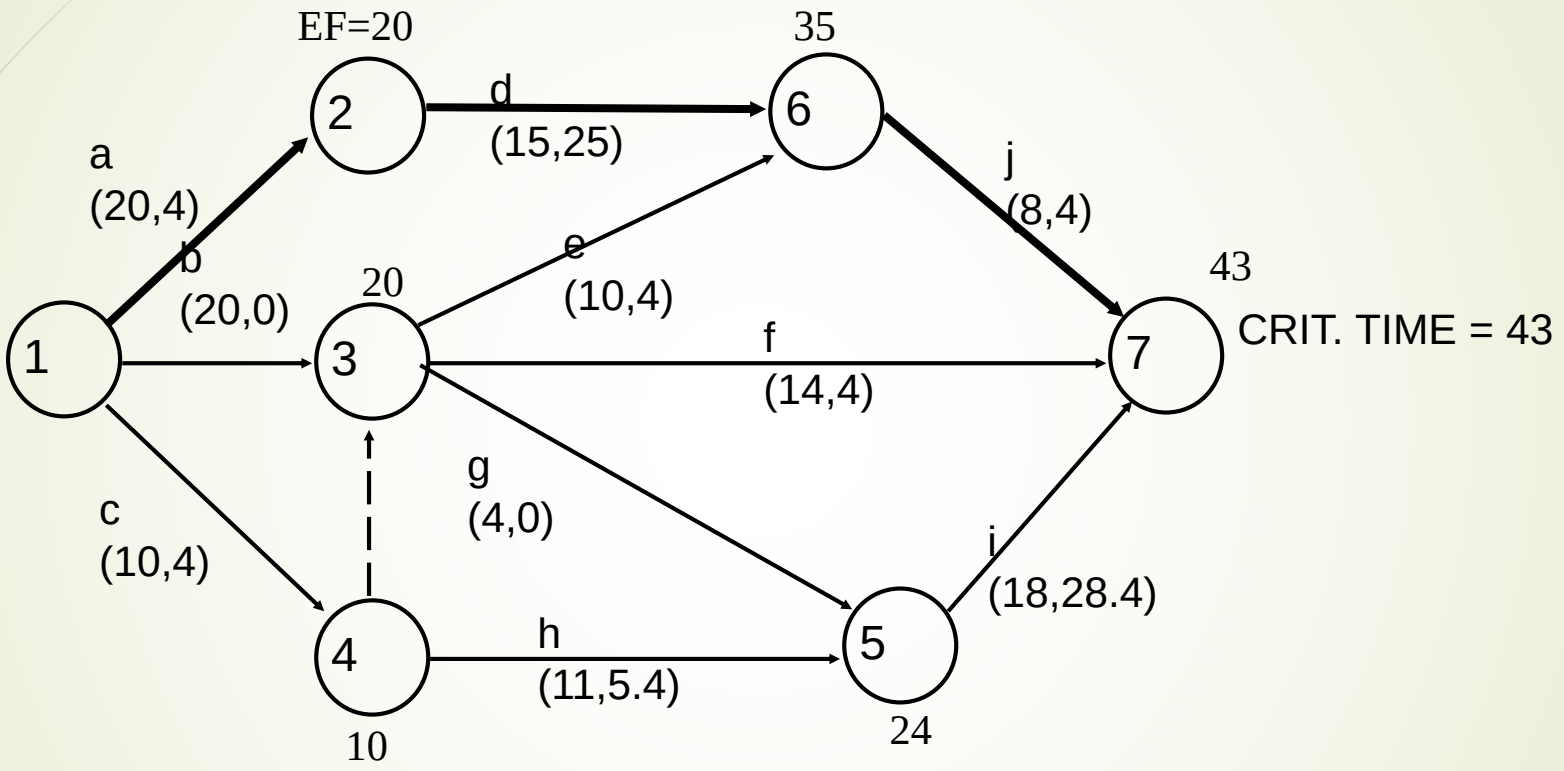
# Precedence And Project Activity Times

| Activity | Immediate Predecessor | Optimistic Time | Most Likely Time | Pessimistic Time | EXP TE | Var V | S.Dev σ |
|---|---|---|---|---|---|---|---|
| a | - | 10 | 22 | 22 | 20 | 4 | 2 |
| b | - | 20 | 20 | 20 | 20 | 0 | 0 |
| c | - | 4 | 10 | 16 | 10 | 4 | 2 |
| d | a | 2 | 14 | 32 | 15 | 25 | 5 |
| e | b,c | 8 | 8 | 20 | 10 | 4 | 2 |
| f | b,c | 8 | 14 | 20 | 14 | 4 | 2 |
| g | b,c | 4 | 4 | 4 | 4 | 0 | 0 |
| h | c | 2 | 12 | 16 | 11 | 5.4 | 2.32 |
| I | g,h | 6 | 16 | 38 | 18 | 28.4 | 5.33 |
| j | d,e | 2 | 8 | 14 | 8 | 4 | 2 |

# The complete network



# Figure 8-13 The complete Network



# Critical Path Analysis (PERT)

| Activity | LS | ES | Slacks | Critical ? |
|---|---|---|---|---|
| **a** | **0** | **0** | **0** | **Yes** |
| b | 1 | 0 | 1 | |
| c | 4 | 0 | 4 | |
| **d** | **20** | **20** | **0** | **Yes** |
| e | 25 | 20 | 5 | |
| f | 29 | 20 | 9 | |
| g | 21 | 20 | 1 | |
| h | 14 | 10 | 4 | |
| i | 25 | 24 | 1 | |
| **j** | **35** | **35** | **0** | **Yes** |

Assume, PM promised to complete the project in the fifty days.
What are the chances of meeting that deadline?
Calculate Z, where

$$Z = (D-S) / \sqrt{V}$$

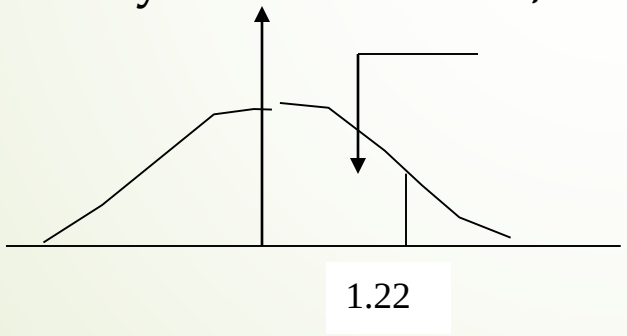
**Example,**

**D = 50; S(Scheduled date) = 20+15+8 =43; V = (4+25+4) =33**

**Z = (50 – 43) / 5.745**

**=** 1.22 standard deviations.

**The probability value of Z = 1.22, is 0.888**



What deadline are you 95% sure of meeting

Z value associated with 0.95 is 1.645

D = S + 5.745 (1.645)
= 43 + 9.45
= 52.45 days

Thus, there is a 95 percent chance of finishing the project by 52.45 days.

# Comparison Between CPM and PERT

| | CPM | PERT |
|---|---|---|
| 1 | Uses network, calculate float or slack, identify critical path and activities, guides to monitor and controlling project | Same as CPM |
| 2 | Uses one value of activity time | Requires 3 estimates of activity time<br>Calculates mean and variance of time |
| 3 | Used where times can be estimated with confidence, familiar activities | Used where times cannot be estimated with confidence.<br>Unfamiliar or new activities |
| 4 | Minimizing cost is more important | Meeting time target or estimating percent completion is more important |
| 5 | Example: construction projects, building one off machines, ships, etc | Example: Involving new activities or products, research and development etc |

# BENEFITS OF CPM / PERT NETWORK

**Consistent framework for planning, scheduling, monitoring, and controlling project.**

- Shows interdependence of all tasks, work packages, and work units.
- Helps proper communications between departments and functions.
- Determines expected project completion date.
- Identifies so-called critical activities, which can delay the project completion time.
- Identified activities with slacks that can be delayed for specified periods without penalty, or from which resources may be temporarily borrowed.
- Determines the dates on which tasks may be started or must be started if the project is to stay in schedule.
- Shows which tasks must be coordinated to avoid resource or timing conflicts.
- Shows which tasks may run in parallel to meet project completion date .

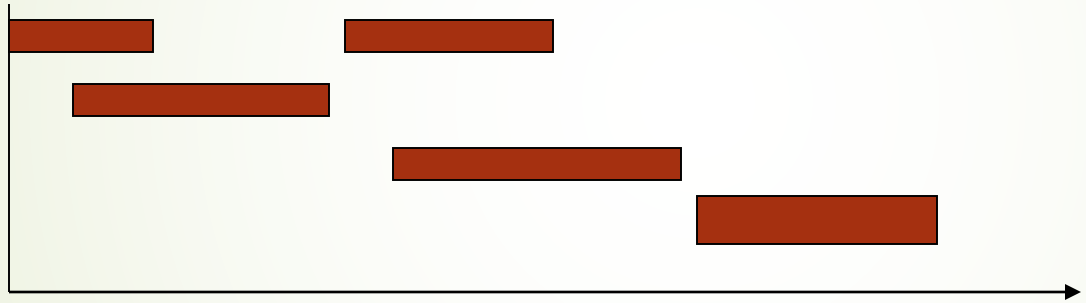
# Gantt Charts:

- Gantt chart has been named after its developer **Henry Gantt.**
- A Gantt chart is a special type of bar chart where each bar represents an activity. The vertical axis lists all the tasks to be performed. The bars are drawn along the y-axis, one for each task.
- The bars are drawn along a time line. The length of each bar is proportional to the duration of time planned for the corresponding activity.
- In the Gantt chart used for SPM, each bar consist of an **unshaded part and a shaded part.** The shaded part of the bar shown the length of time each task is estimated to take. The unshaded part shown slack time.

# Gantt Charts

- Since 1917; Useful for showing work vs time in form of bar charts

e.g.

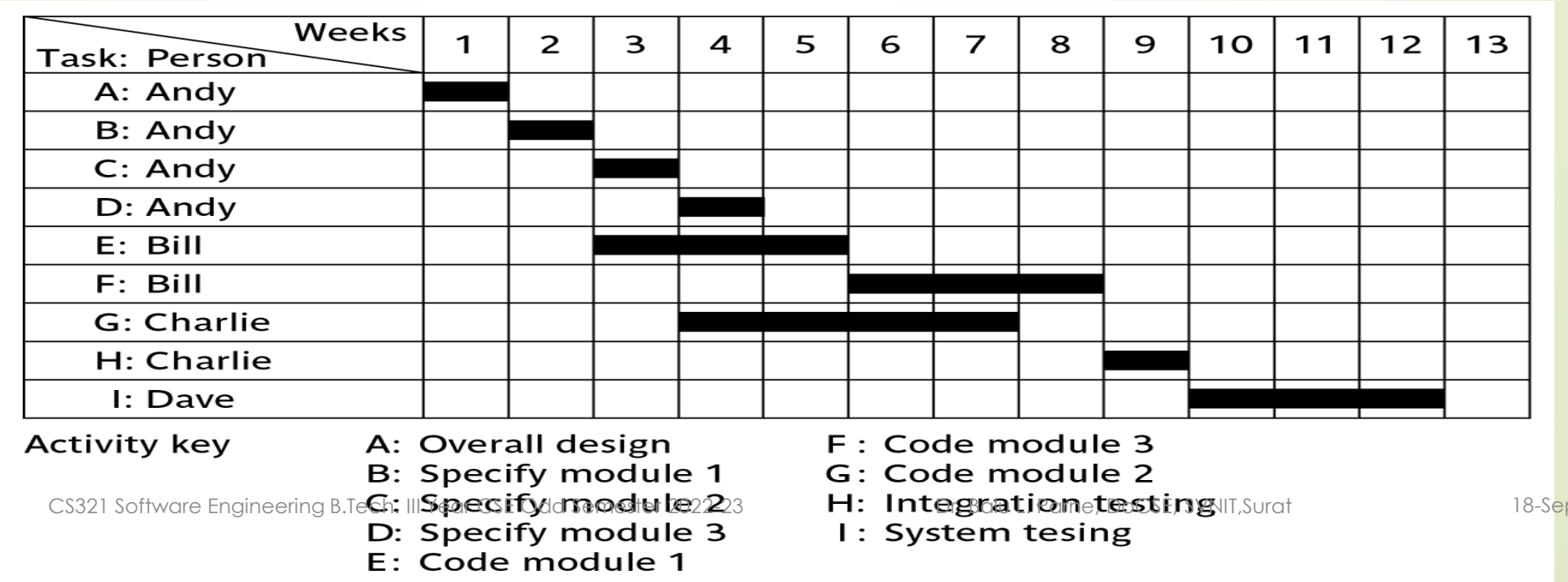


- Can draw directly or from CPM/PERT network

# Gantt Charts and CPM/PERT Networks



- **Gantt chart** representation of project schedule is helpful in **planning and utilization of resources,** while **PERT chart** is useful for **monitoring and timely progress of activities.**
- Even though a lot of info, easy to read and , understand to monitor and follow progress.
  - Not very good for logical constraints
  - Should be used to **COMPLEMENT** networks, not replace



# Home Work:

1. You are required to prepare a network diagram for constructing a 5 floor apartmer
The major activities of the project are given as follows:

| Activity | Description | Immediate Predecessor |
|---|---|---|
| A | Selection of site | - |
| B | Preparation of drawings | - |
| C | Arranging the for finance | A |
| D | Selection of contractor | A |
| E | Getting approval from Govt | A |
| F | Laying the foundation | E |
| G | Start construction | D, F |
| H | Advertise in newspaper | B, C |
| I | Allocation of tenants | G, H |

2. For the problem No.1 the time estimates in days are given. Determine the Time earliest and Time latest, and the critical activities

| Activity | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| Time (days) | 3 | 5 | 7 | 2 | 5 | 20 | 60 | 2 | 10 |

# Thank You.