Chapter 4

Software Project Management

SE201

Course Learning Outcome

- **CLO4:** Demonstrate project management skills by working effectively in small teams, and cooperatively with other teams

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

Project Management Spectrum

Effective software project management focuses on the four **P**'s: people, product, process, and project

- People: the most important element of a successful project
- Product: the software to be built
- Process: the set of framework activities and software engineering tasks to get the job done
- Project: all work plan required to make the product a reality

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Stakeholders

- Every organization needs to continually improve its ability to **attract, develop, motivate, organize**, and **retain the workforce** needed to accomplish its strategic business objectives, which helps in effective software project management practices.

1- People: Stakeholders

- *Senior managers* who define the business issues that often have significant influence on the project.
- *Project (technical) managers* who must plan, motivate, organize, and control the practitioners who do software work.
- *Practitioners* who deliver the technical skills that are necessary to engineer a product or application.
- *Customers or Product owners* who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- *End-users* who interact with the software once it is released for production use.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Team Leader

- Project management is a people-intensive activity. A team leader needs to have/develop 3 skills:
- *Motivation*. The ability to encourage (by "push or pull") technical people to produce to their best ability.
- *Organization*. The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
- *Ideas or innovation.* The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Team Leader

- An effective project manager emphasizes four key traits:
  - Problem solving skills: diagnose the most relevant technical and organizational issues and systematically structure a solution
  - Managerial Identity: A good project manager must take charge of the project.
  - Achievement: competent manager must reward initiative and accomplishment to optimize the productivity of a project team.
  - Influence and team building: must be able to understand verbal and nonverbal signals and react to the needs of the people

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Software Teams

How to lead?

How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Software Teams

- The "best" team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty.

The following factors must be considered when selecting a software engineering project team structure ...

- Difficulty of the problem to be solved
- Size of the resultant program(s) in lines of code or function points
- Time that the team will stay together (team lifetime)
- Degree to which the problem can be modularized
- Required quality and reliability of the system to be built
- Rigidity of the delivery date
- Degree of sociability (communication) required for the project

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Agile Teams

- In an agile team, adopt characteristics of successful software project teams and avoid toxins that create problems. The agile philosophy stresses **individual competency** coupled with **group collaboration** as critical success factors for the team.

- Avoid Team **"Toxicity"** like:
  - A negatively excited work atmosphere: team members waste energy and lose focus on the objectives of the work
  - High frustration caused by personal, business, or technological factors that cause friction among team members.
  - Poorly defined or improperly chosen process model that becomes a roadblock to accomplishment.
  - Unclear definition of roles resulting in a lack of accountability and resultant finger-pointing.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Agile Teams

- The distribution of skills must be appropriate to the problem.
- Teams are “self-organizing” in order to make effective use of the competencies of each team member and to foster effective collaboration through a software project.
- Agile team gets significant autonomy to make the project management and technical decisions required to get the job done.
- Planning is kept to a minimum, and the team is allowed to select its own approach(e.g., process, methods, tools)

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Team Coordination & Communication

- *Formal, impersonal approaches* include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools, change requests and related documentation, error tracking reports, and repository data.

- *Formal, interpersonal procedures* focus on quality assurance activities applied to software engineering work products. These include status review meetings and design and code inspections.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

1- People: Team Coordination & Communication

- *Informal, interpersonal procedures* include group meetings for information dissemination and problem solving and "collocation of requirements and development staff."

- *Electronic communication* encompasses electronic mail, electronic bulletin boards (real-time chat/forums), and by extension, video-based conferencing systems.

- *Interpersonal networking* includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

2- The Product- Scope

the software to be built, which meets the requirements of a customer.

- **Context.** How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?
- **Information objectives.** What customer-visible data objects are produced as output from the software? What data objects are required for input?
- **Function and performance.** What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

2- The Product- Scope

the software to be built, which meets the requirements of the customer.

The software project scope must be unambiguous and understandable at the management and technical levels.

- A statement of the software scope must be defined. That is, quantitative data (e.g., number of simultaneous users, maximum allowable response time) are stated explicitly, constraints and/or limitations (e.g., product memory size, existing target IT infrastructure, etc.) are noted.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

2- The Product: Problem Decomposition

- Decomposition is an activity at the core of software requirements analysis. Sometimes called *partitioning* or *problem elaboration.*

- Problem decomposition is applied in two major areas:

  (1) the functionality and content (information) that must be delivered and

  2) the process that will be used to deliver it.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014).
Slides copyright 2014 by Roger Pressman.

2- The Product: Problem Decomposition

- A complex problem is partitioned into smaller problems that are more manageable:
  - Software functions are evaluated and refined to provide more details.
  - Similarly, major content or data objects are decomposed into their constituent parts, providing a reasonable understanding of the information to be produced by the software.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014).
Slides copyright 2014 by Roger Pressman.

3- Process

- The problem is to select the process model that is appropriate for the software to be engineered by your project team
- Your team must decide which process model is most appropriate for
  - (1) the customers who have requested the product and the people who will do the work
  - (2) the characteristics of the product itself
  - (3) the project environment in which the software team works
- When a process model has been selected, the team then defines a preliminary project plan based on the set of process framework activities
- Once the preliminary plan is established, process decomposition begins: Define a task set for each software engineering activity

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

3- Process: Melding the Problem and the Process

Project planning begins with the application of the process to the problem. Each function to be engineered by your team must pass through the set of framework activities that have been defined for your software organization.

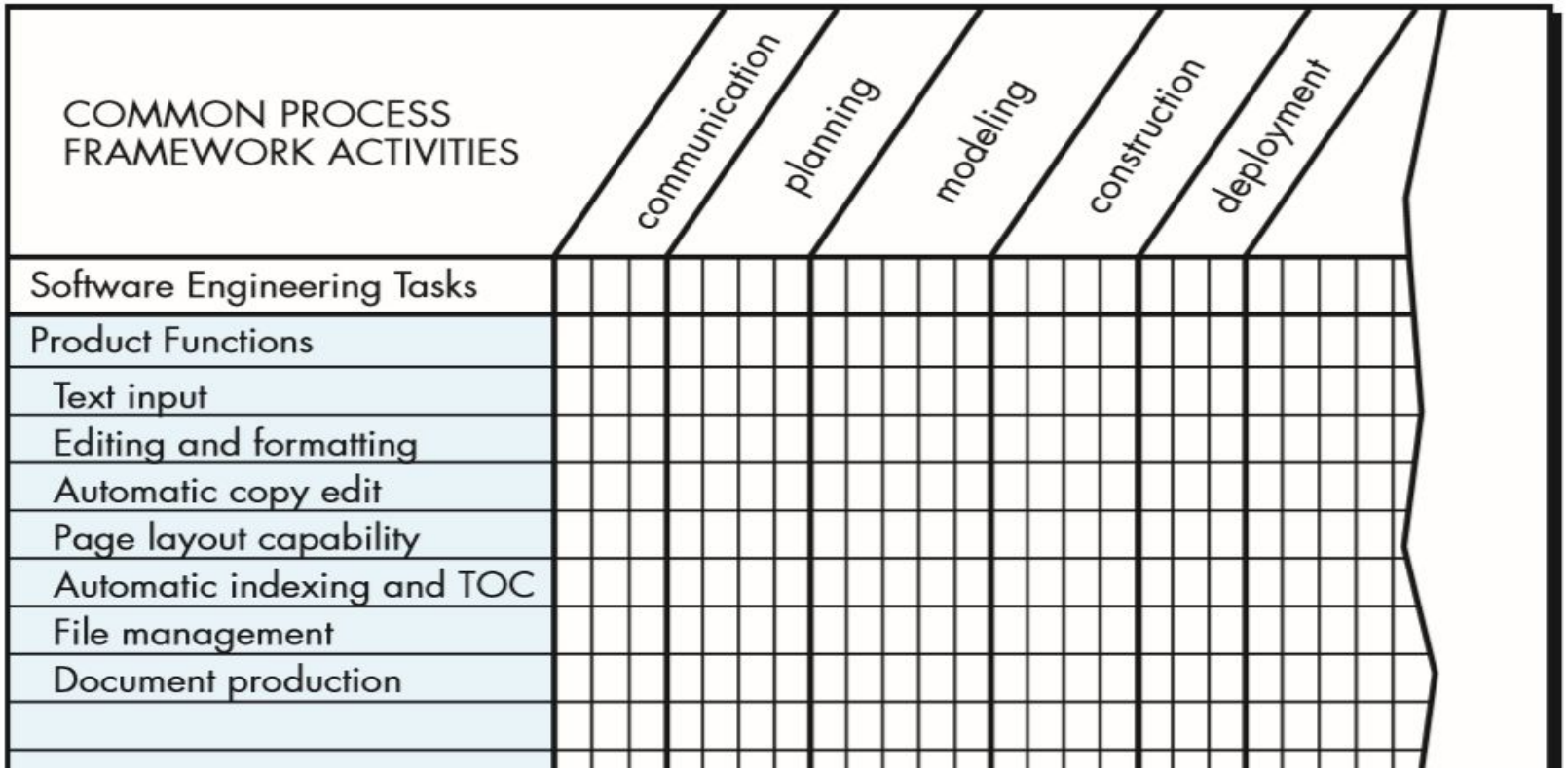
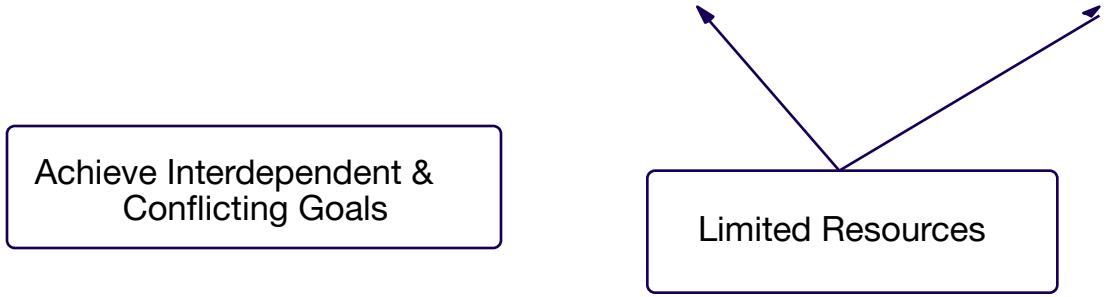


Figure 31.1 Each major product function (the figure notes functions for a **word-processing software**)

4- Project

Almost all software products are obtained via *projects*.

Project Concern = *Deliver on time* and *within budget*



The Project Team is the primary Resource!

4- Signs of Projects Risk

- Ten signs that indicate that project is In *trouble whe*n:

1. Software people don't understand their customer's needs.
2. The product scope is poorly defined.
3. Changes are managed poorly.
4. The chosen technology changes.
5. Business needs change [or are ill-defined].
6. Deadlines are unrealistic.
7. Users are resistant.
8. Sponsorship is lost [or was never properly obtained].
9. The project team lacks skills.
10. Managers [and practitioners] avoid best practices and lessons.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014). Slides copyright 2014 by Roger Pressman.

4- Common-Sense Approach for Projects to Avoid problem

Start on the right foot

- Involves detailed understanding of project
- setting realistic objectives & expectations
- Selecting the right team
- Facilitating the team

Maintain Momentum

- Provide incentives
- Reduce bureaucracy and give autonomy to team members but with supervision. senior management should do everything possible to stay out of the team's way.

Track progress.

- For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using technical reviews) as part of a quality assurance activity.

Make smart decisions

- When possible, use existing software components
- Choose standard approaches to keep it simple
- Avoid risks and allocate more time than needed for complex/risky tasks

Conduct a ~~postmortem~~ post-delivery analysis.

- Compare planned and actual schedule
- Establish a consistent mechanism for extracting lessons learned for each project.

4- To Get to the Essence of a Project using W5HH
(Suggested by Barry Boehm)

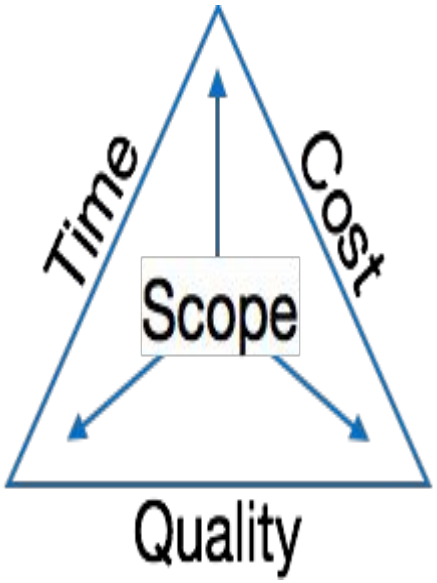
- Why is the system being developed?
- What will be done?
  - Answer to this question establishes the task set required for the project
- When will it be done?
  - this question helps the team establish a project schedule by identifying when tasks have to be conducted and when milestones are to be reached
- Who is responsible?
  - this question establishes roles and responsibility of each team member
- Where are they organizationally located?
  - Answer to this question indicates that all roles and responsibilities are not limited to the software team itself, the customers, users and stakeholders also have responsibilities
- How will the job be done technically and managerially?
- How much of each resource (e.g., people, software, tools, database) will be needed?

Part II
Project Estimation and Scheduling

Triple Constraints for Software Projects

It is an essential part of a software organization to deliver a **quality product,** keeping the **cost** within client's budget constraint and deliver the project as **scheduled.**

There are several factors, both internal and external, which may impact this triple constraint triangle. Any of the three factors can severely impact the other two.

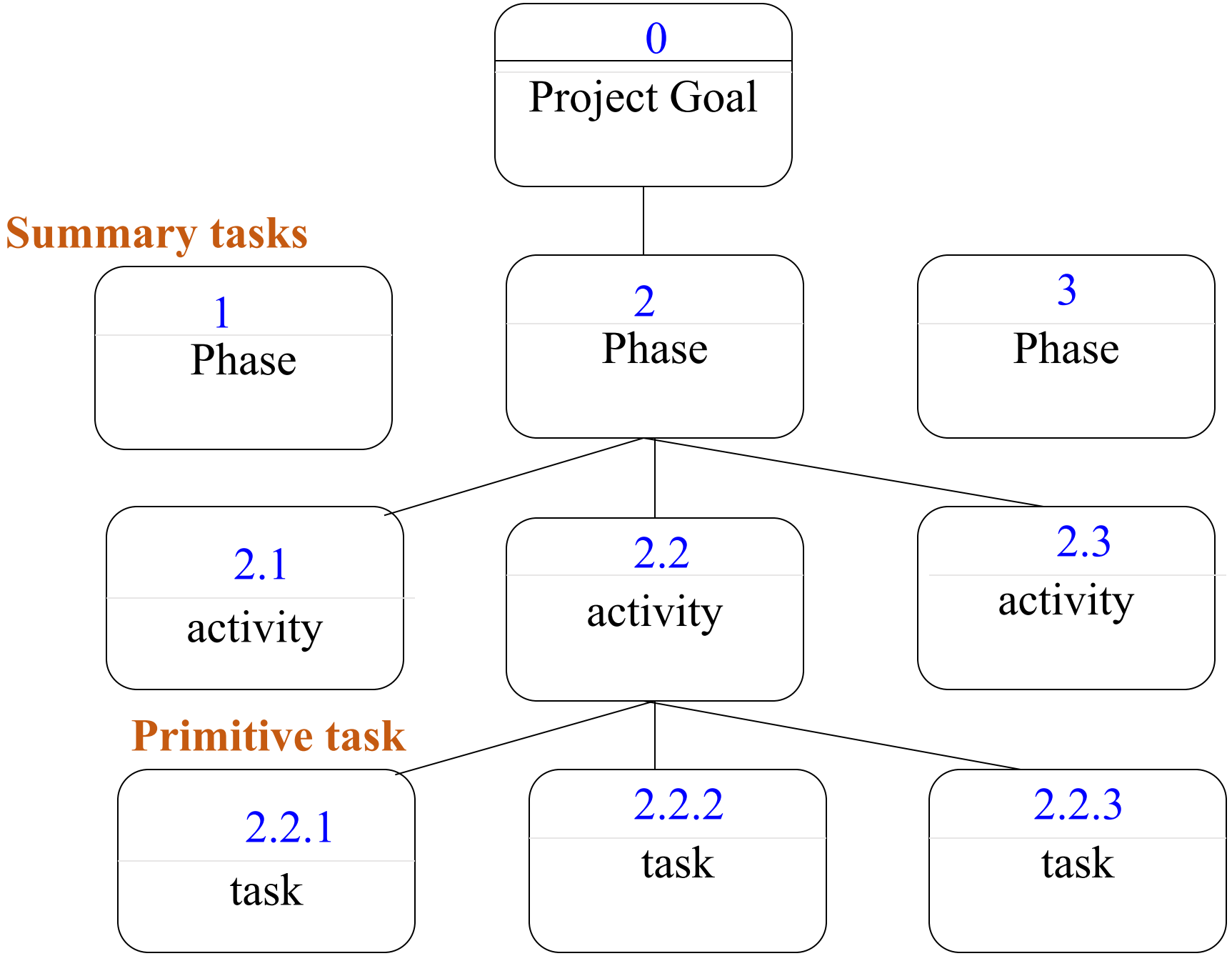


Project Estimation - Scheduling

- Project Scheduling in a project refers to the roadmap of all activities to be done in a specified order and with a time slot allotted to each activity.
- Project managers define various tasks, and project milestones and arrange them keeping various factors in mind.

For scheduling a project, it is necessary to -

- Break down the project tasks into smaller, manageable tasks
- Logically order them to ensure a smooth evolution between tasks. The hierarchical decomposition of project into phases, activities, and tasks and their sequences is called “work breakdown structure (WBS)”. Some tasks are performed in parallel whereas others follow one another sequentially

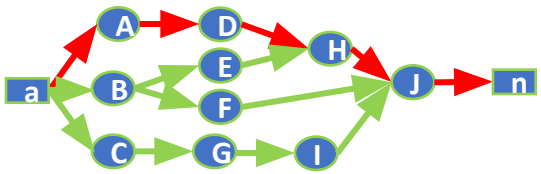

0
Project Goal
Summary tasks
1
Phase
2
Phase
3
Phase
2.1
activity
2.2
activity
2.3
activity
Primitive task
2.2.1
task
2.2.2
task
2.2.3
task

Project Estimation - Scheduling

When the WBS is finalised:

- Estimate the required timeframe for each task
- Divide time into work-units (man-hours, man-days)
- Assign adequate number of work-units for each task
- Assign resources to each task.
- Calculate total time required for the project from start to finish

Project Estimation - Scheduling



Determine the critical path

- The critical path on a Precedence Diagram is represented by the sequence of connected activities that produces the longest overall time period. In other words, the minimum time required to finish the project can be estimated by considering the longest path in the activity graph (the critical path).

- All nodes and activities within this sequence are referred to as being “on” the critical path.
- Any activity on the critical path, if delayed in completion, delays the entire project.
- Nodes not on critical path however can be delayed without delaying the final completion of the project.
- Nodes not on the critical path contain slack/float time and allows the project manager some flexibility in scheduling.

Example

Project Management: Precedence Diagramming Method (PDM)

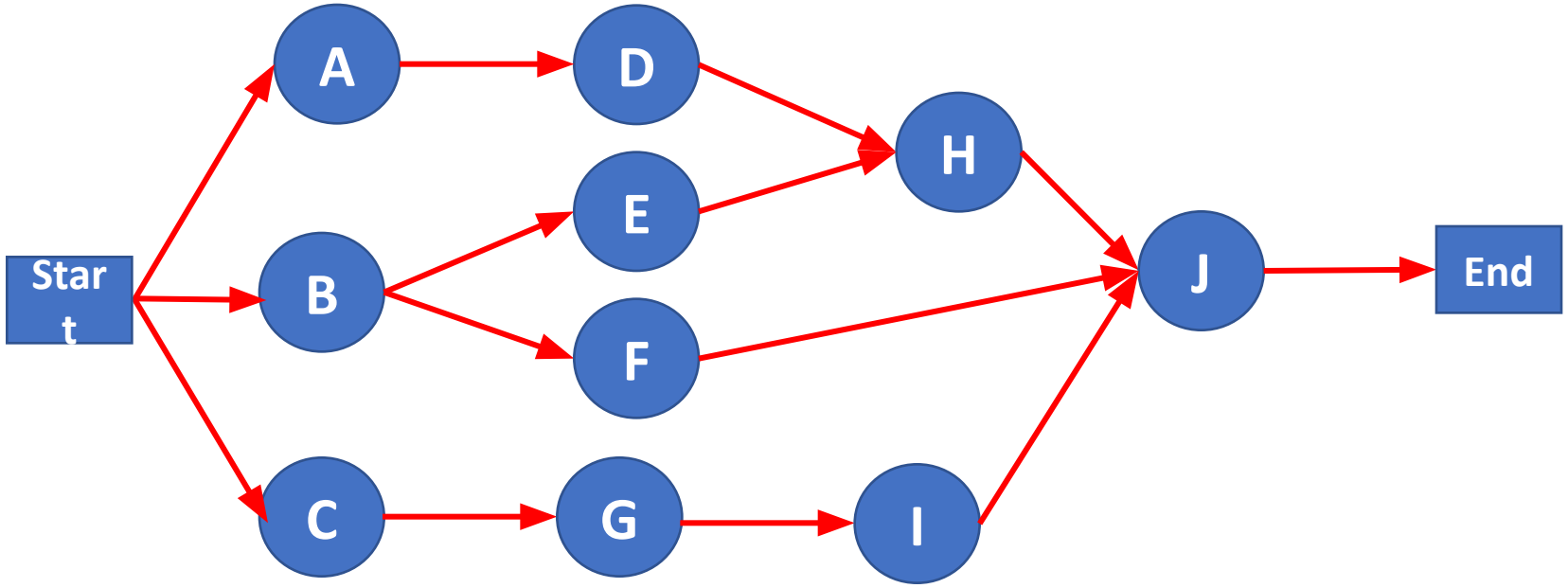
- Dr. Lahouari Ghouti, PhD

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

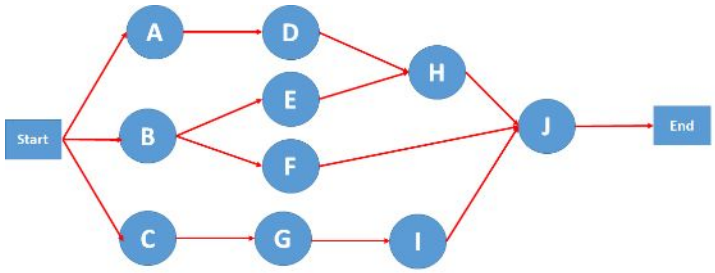
Project Tasks: Precedence Diagramming Method (PDM)

Activity	Predecessor	Duration (e.g., Days)
A	---	1
B	---	2
C	---	2
D	A	4
E	B	5
F	B	4
G	C	6
H	D, E	6
I	G	2
J	F, H, I	3

Project Tasks: Precedence Diagramming Method (PDM)



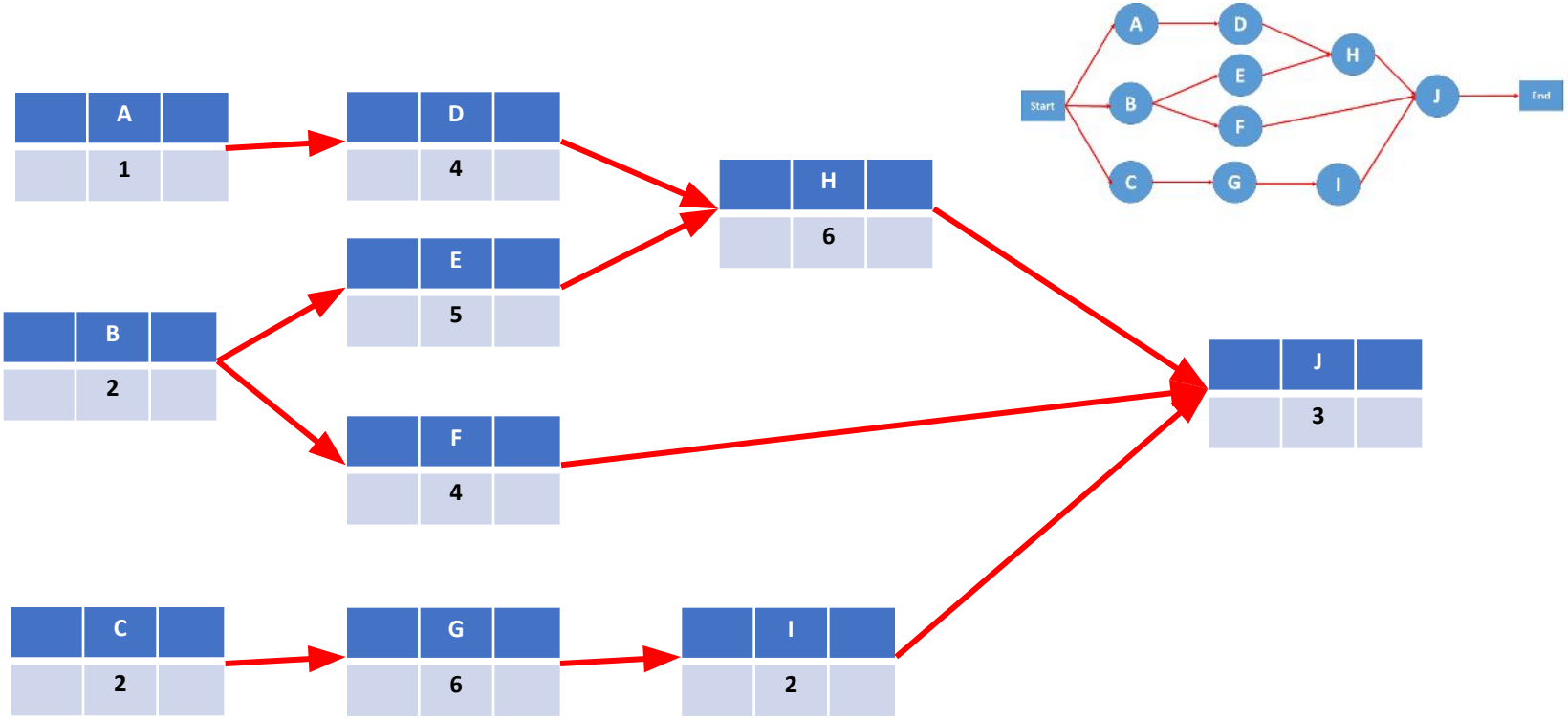
Project Tasks: Precedence Diagramming Method (PDM)



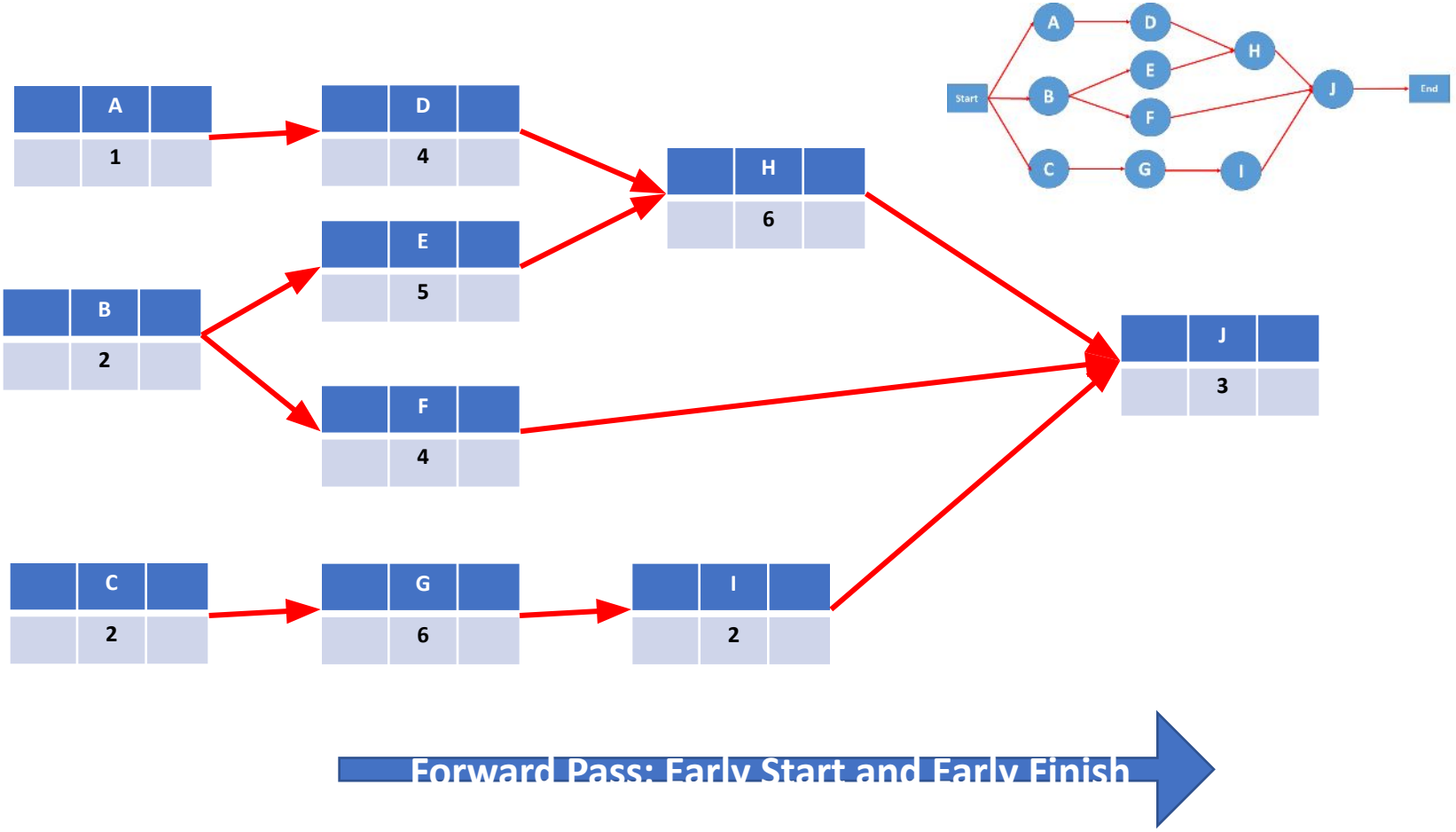
Activity	Predecessor	Duration (e.g., Days)
A	---	1
B	---	2
C	---	2
D	A	4
E	B	5
F	B	4
G	C	6
H	D, E	6
I	G	2
J	F, H, I	3

Early Start	Task	Early Finish
Late Start	**Duration**	**Late Finish**

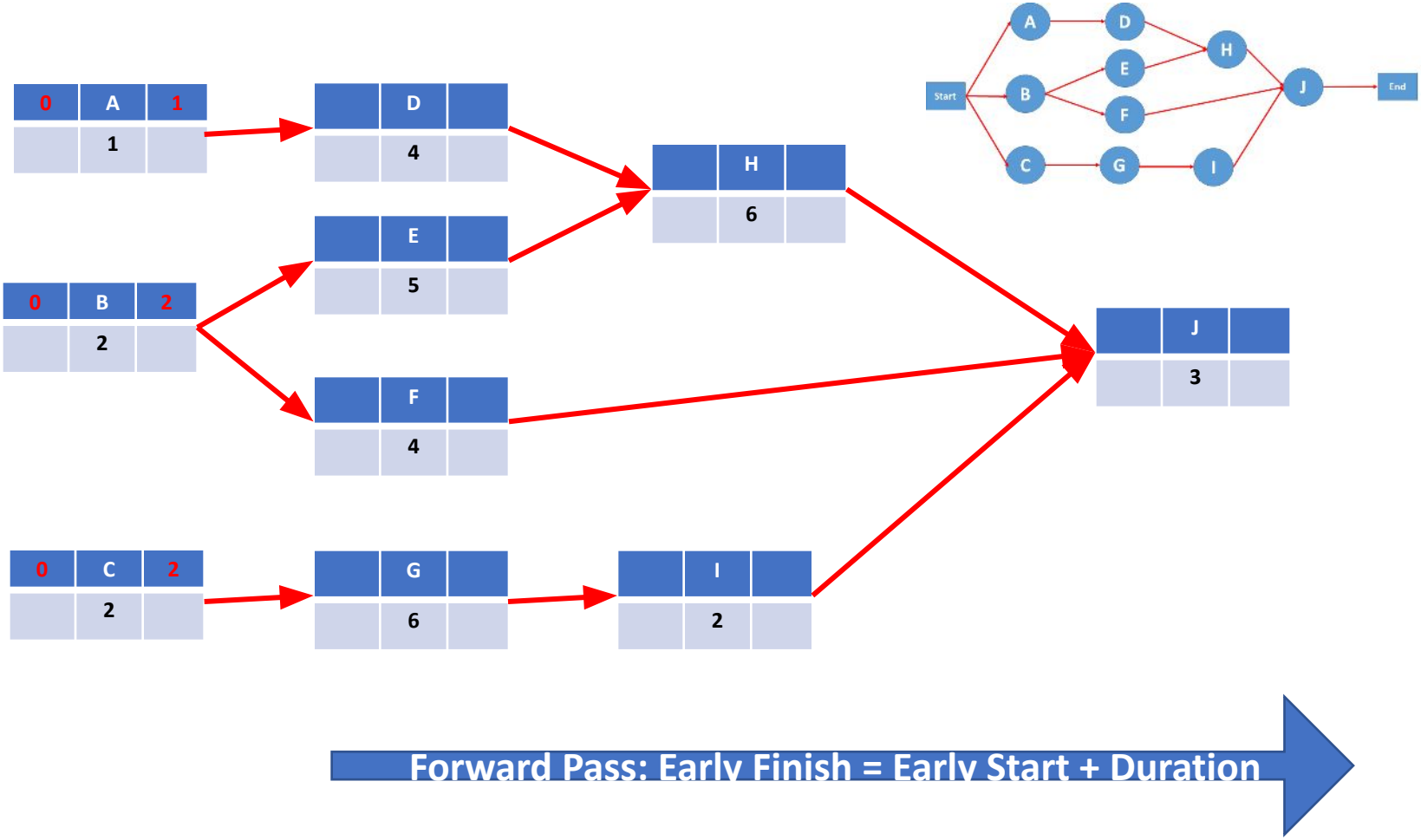
Project Tasks: Precedence Diagramming Method (PDM)



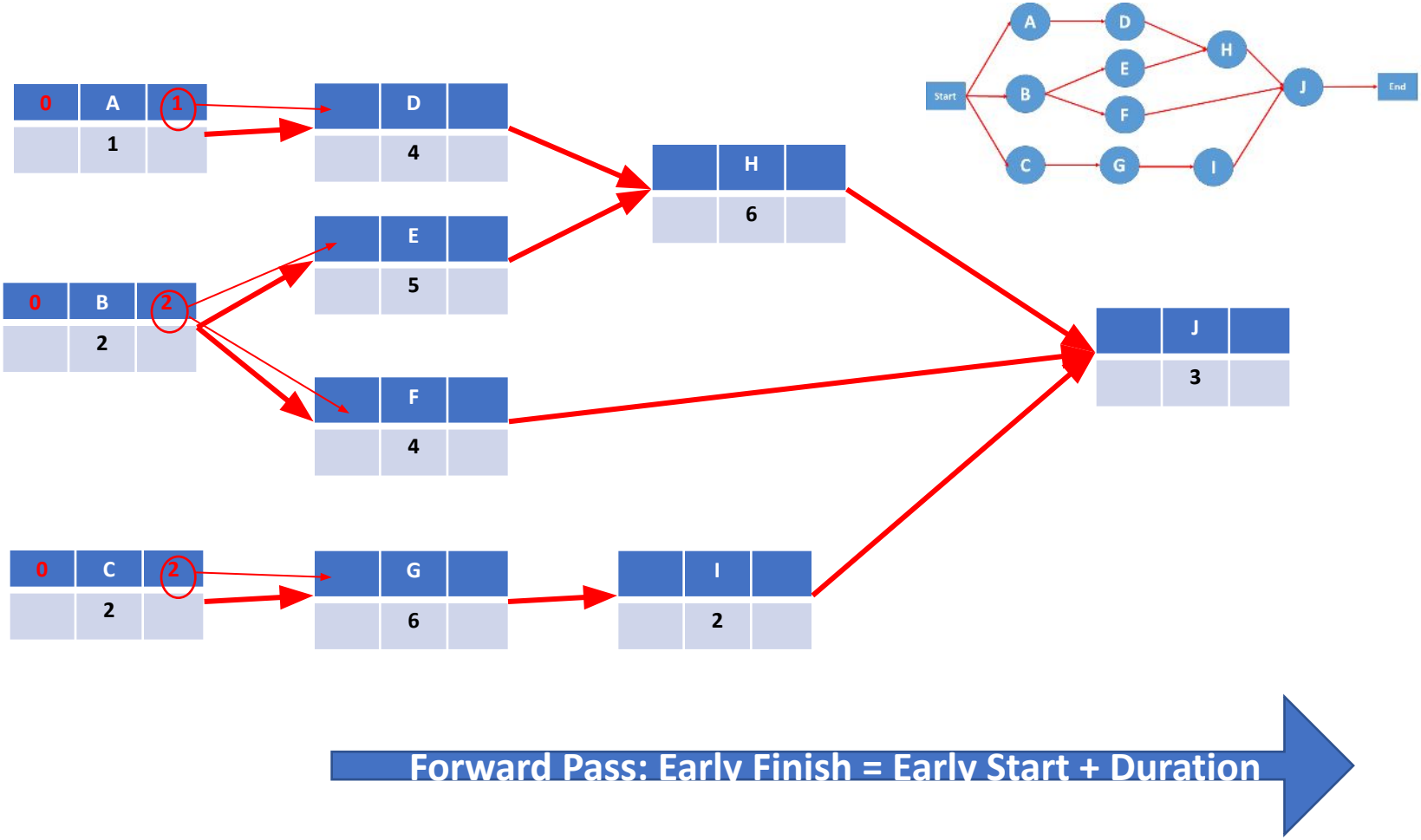
Project Tasks: Precedence Diagramming Method (PDM)



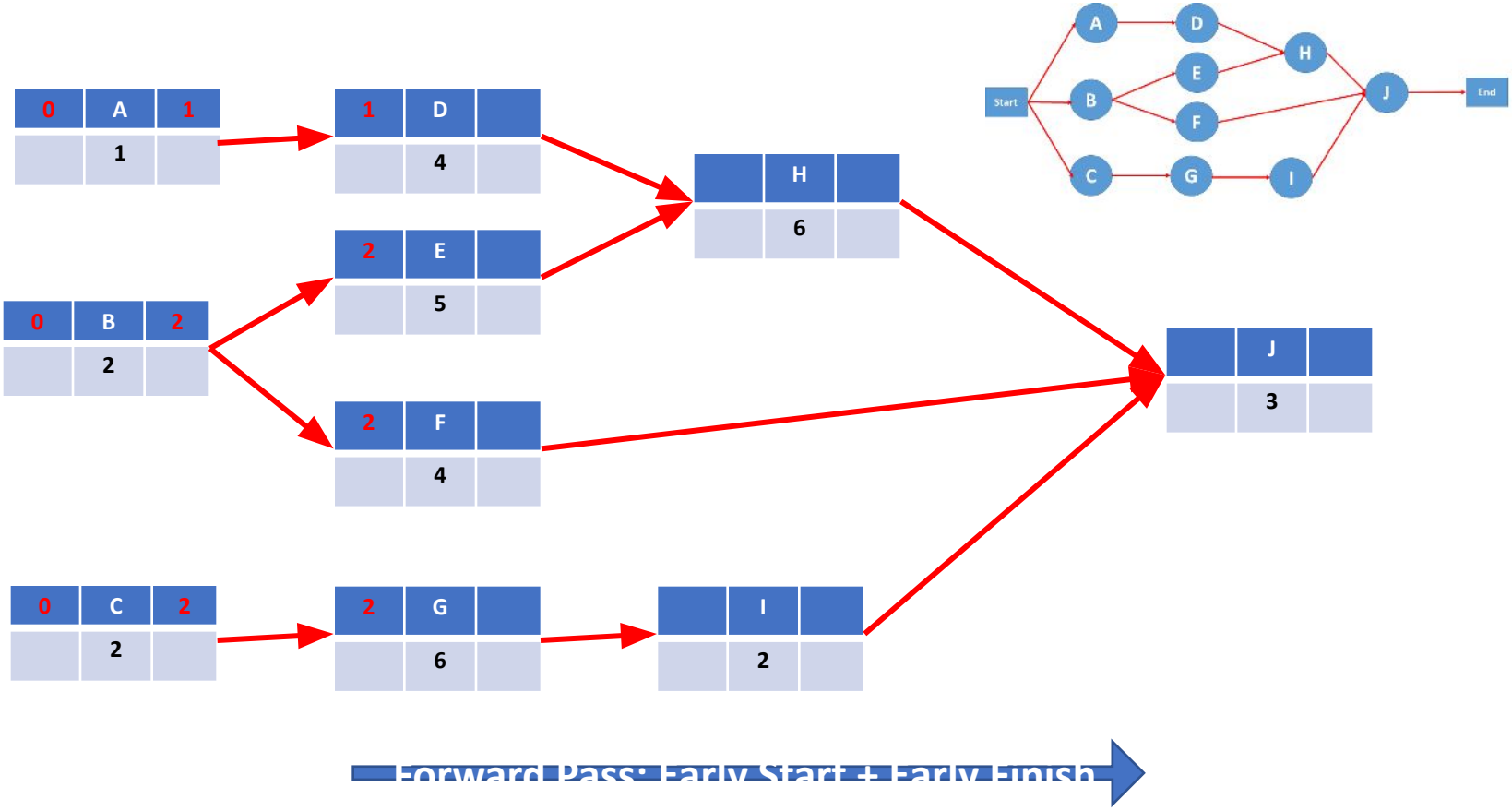
Project Tasks: Precedence Diagramming Method (PDM)



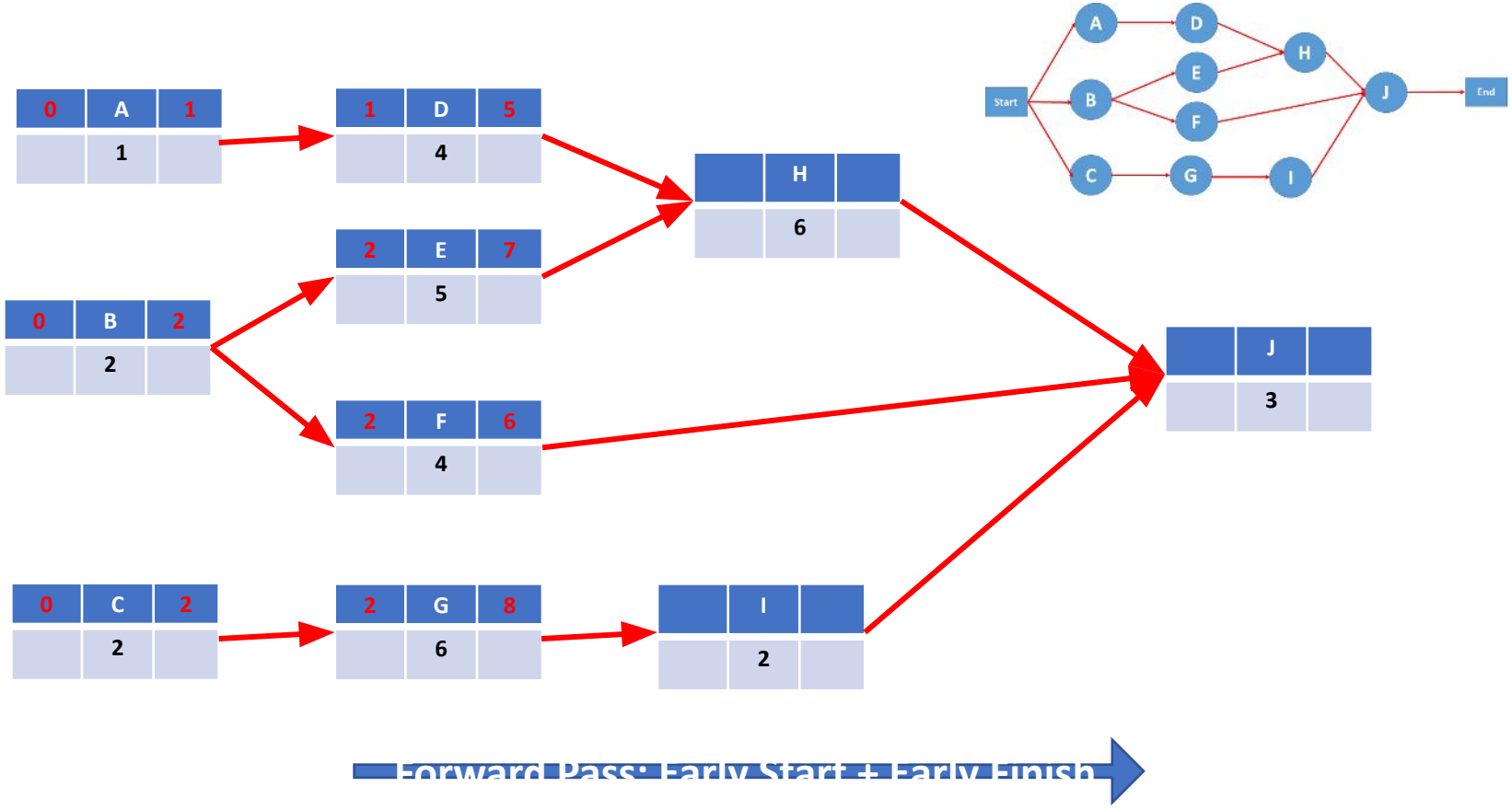
Project Tasks: Precedence Diagramming Method (PDM)



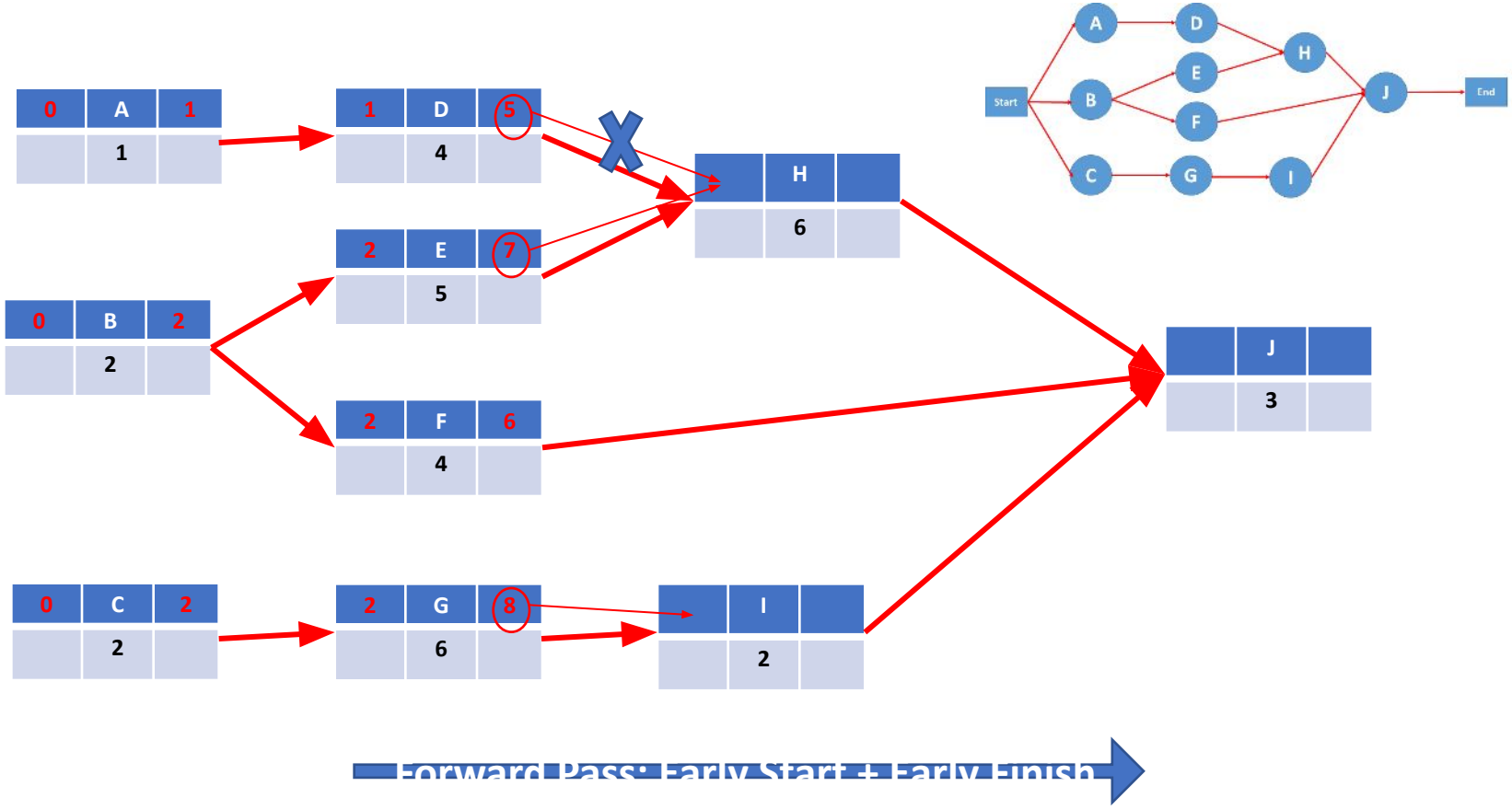
Project Tasks: Precedence Diagramming Method (PDM)



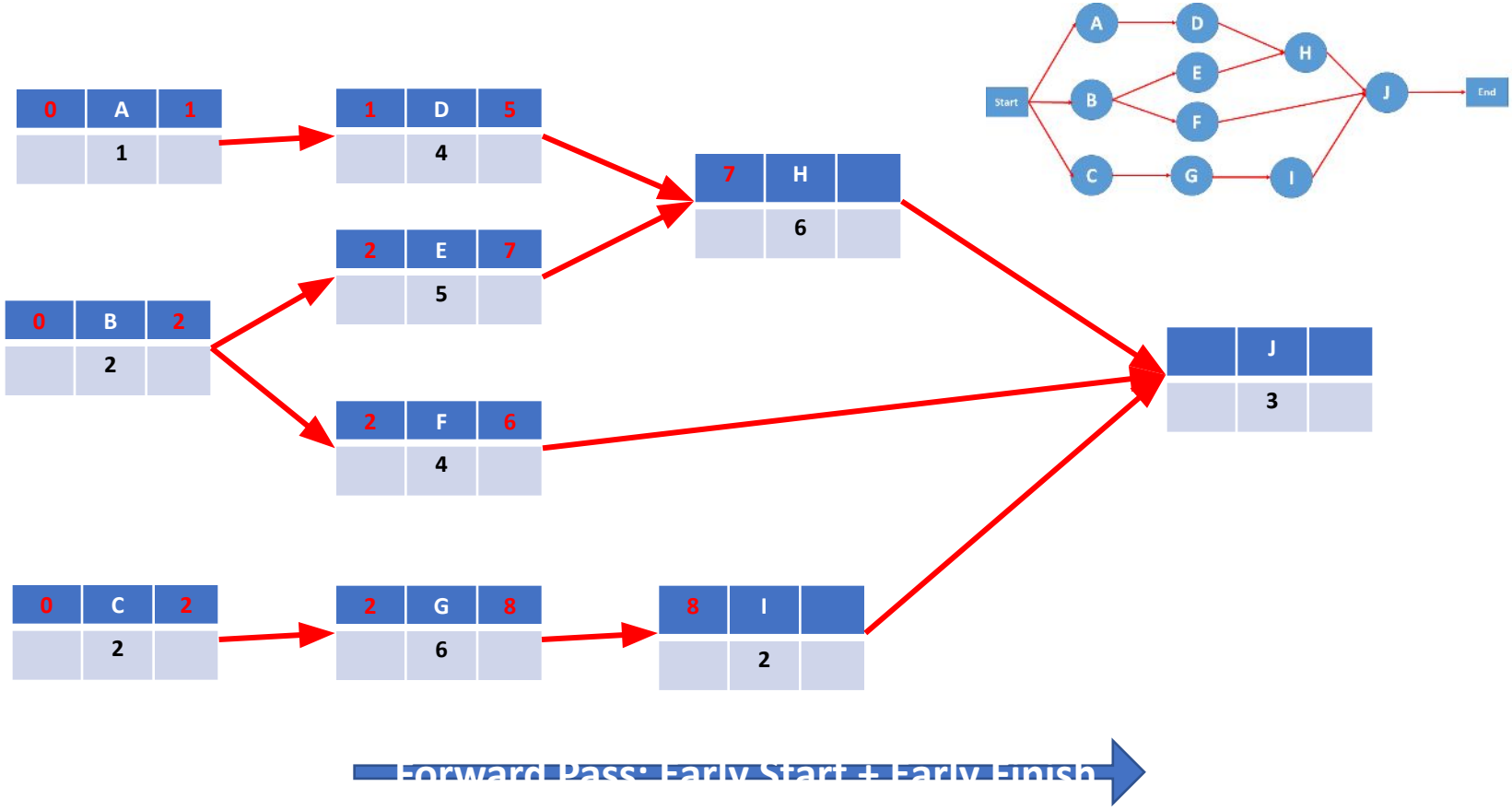
Project Tasks: Precedence Diagramming Method (PDM)



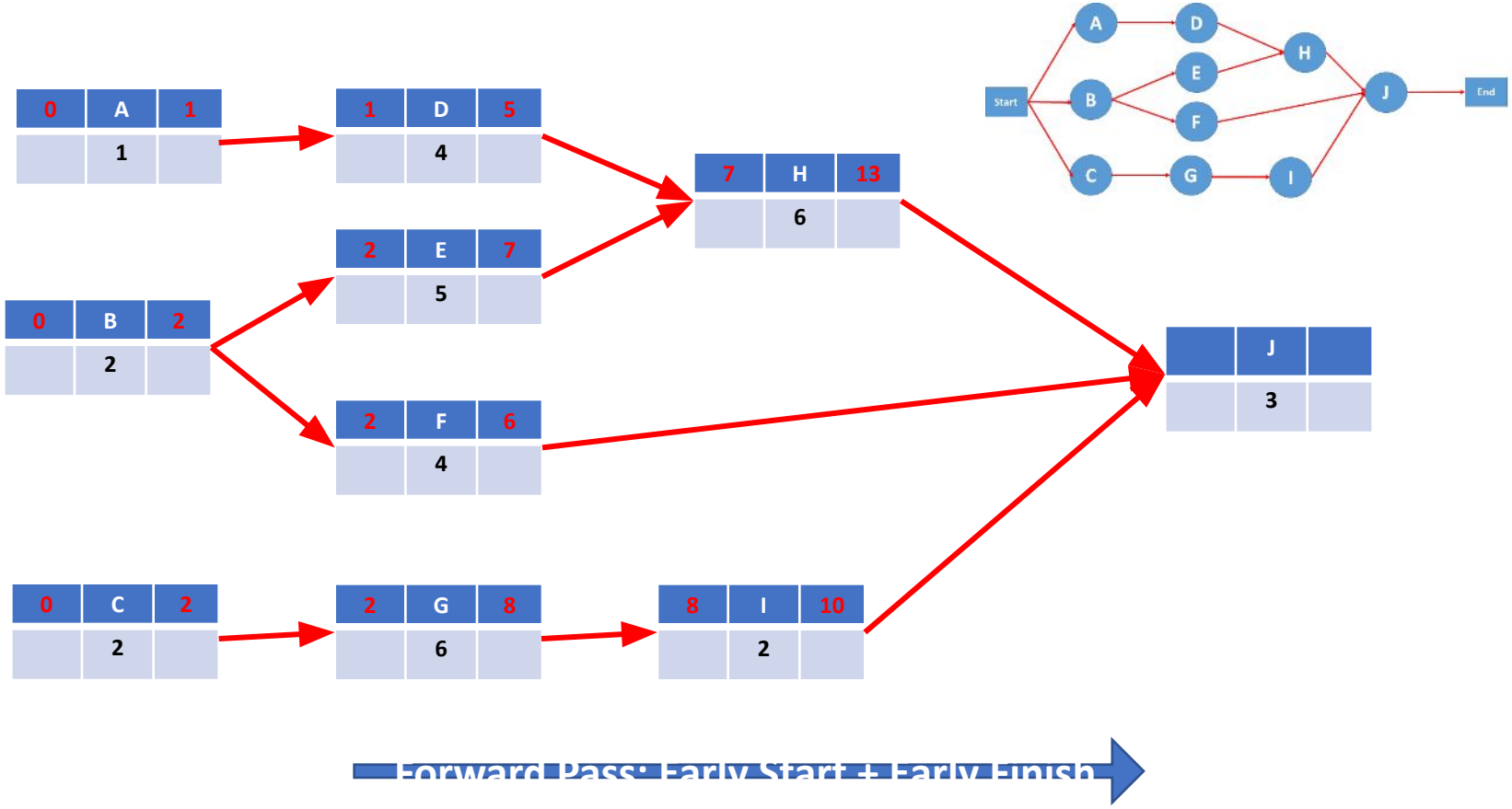
Project Tasks: Precedence Diagramming Method (PDM)



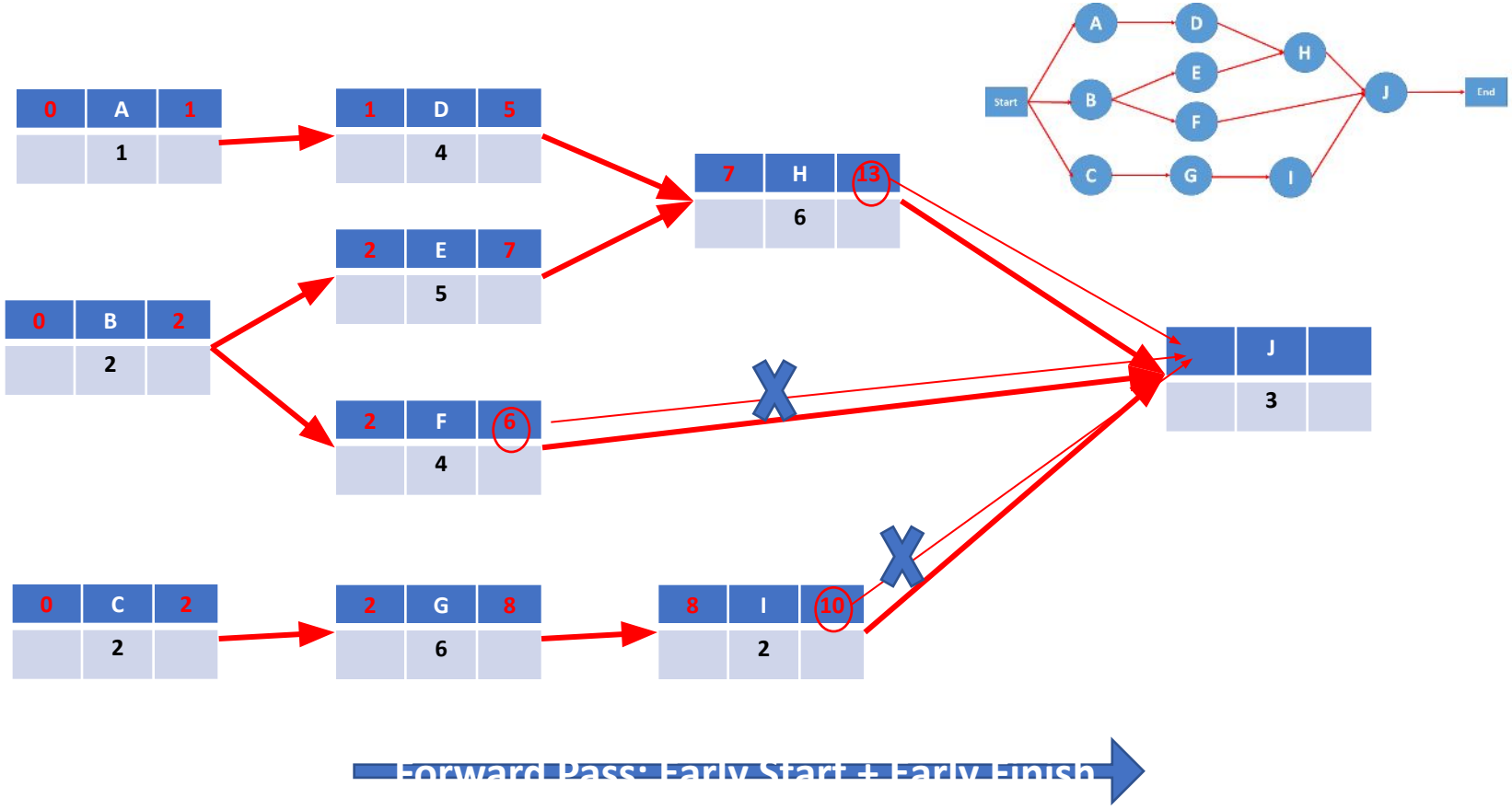
Project Tasks: Precedence Diagramming Method (PDM)



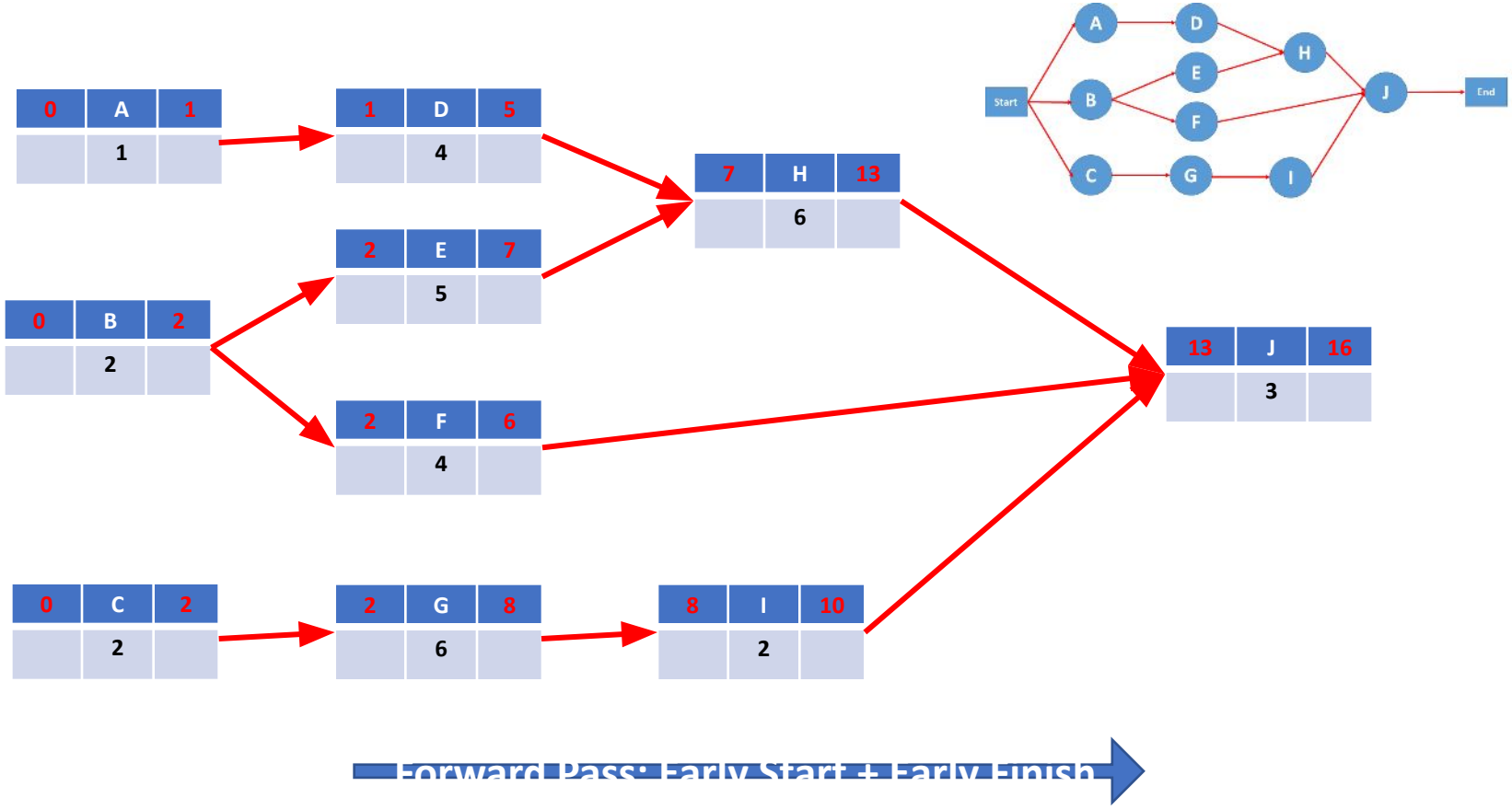
Project Tasks: Precedence Diagramming Method (PDM)



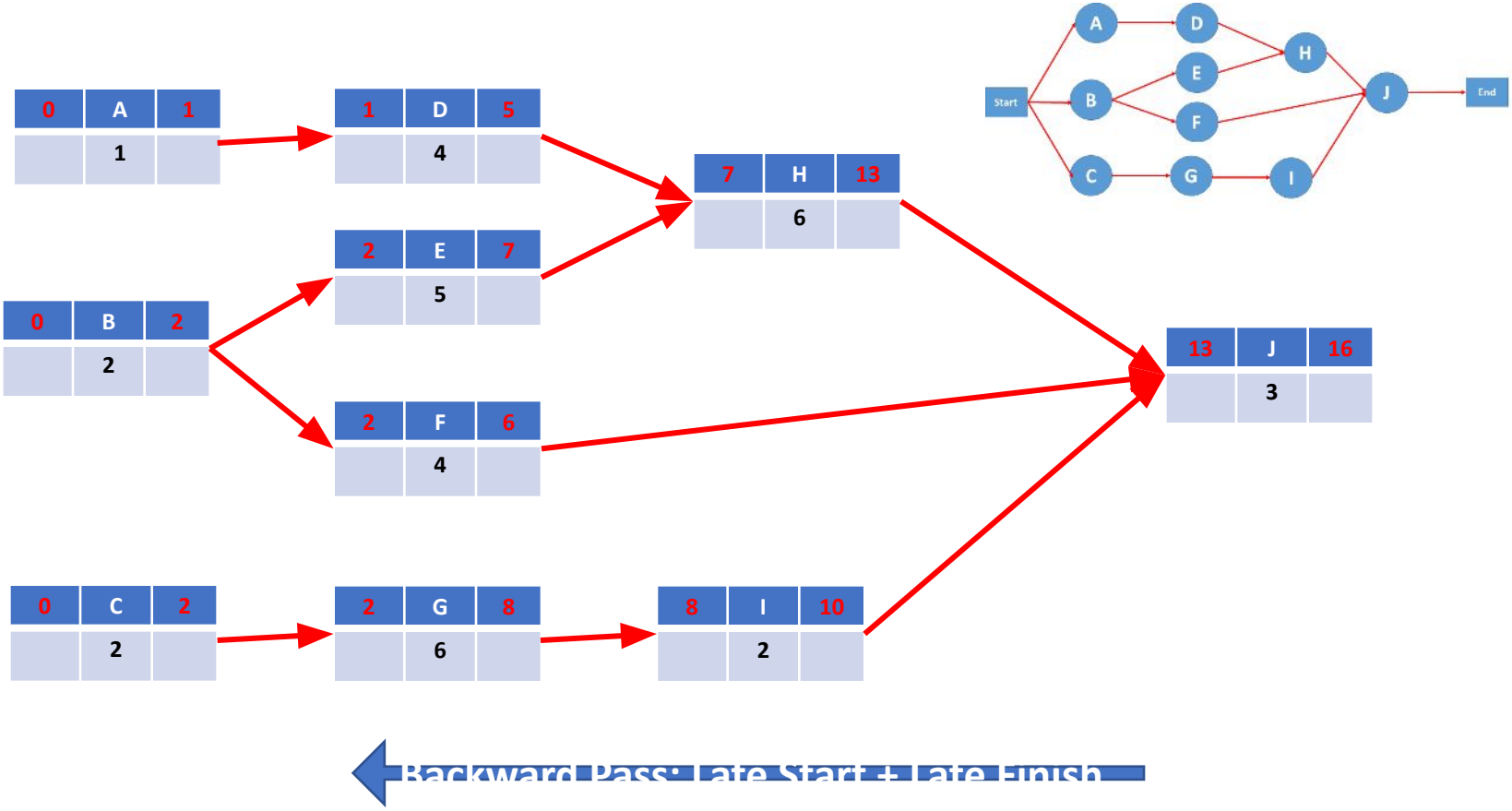
Project Tasks: Precedence Diagramming Method (PDM)



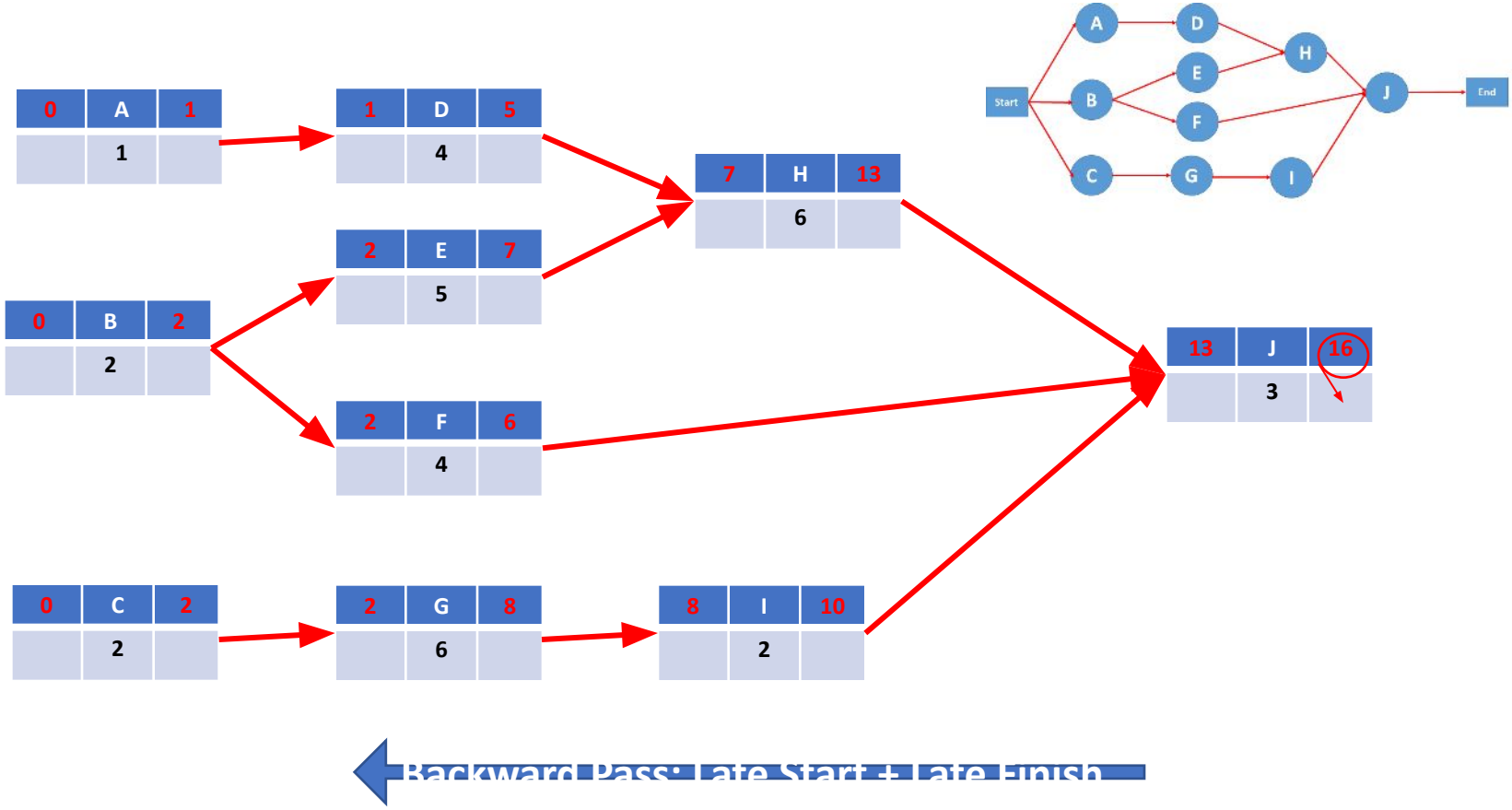
Project Tasks: Precedence Diagramming Method (PDM)



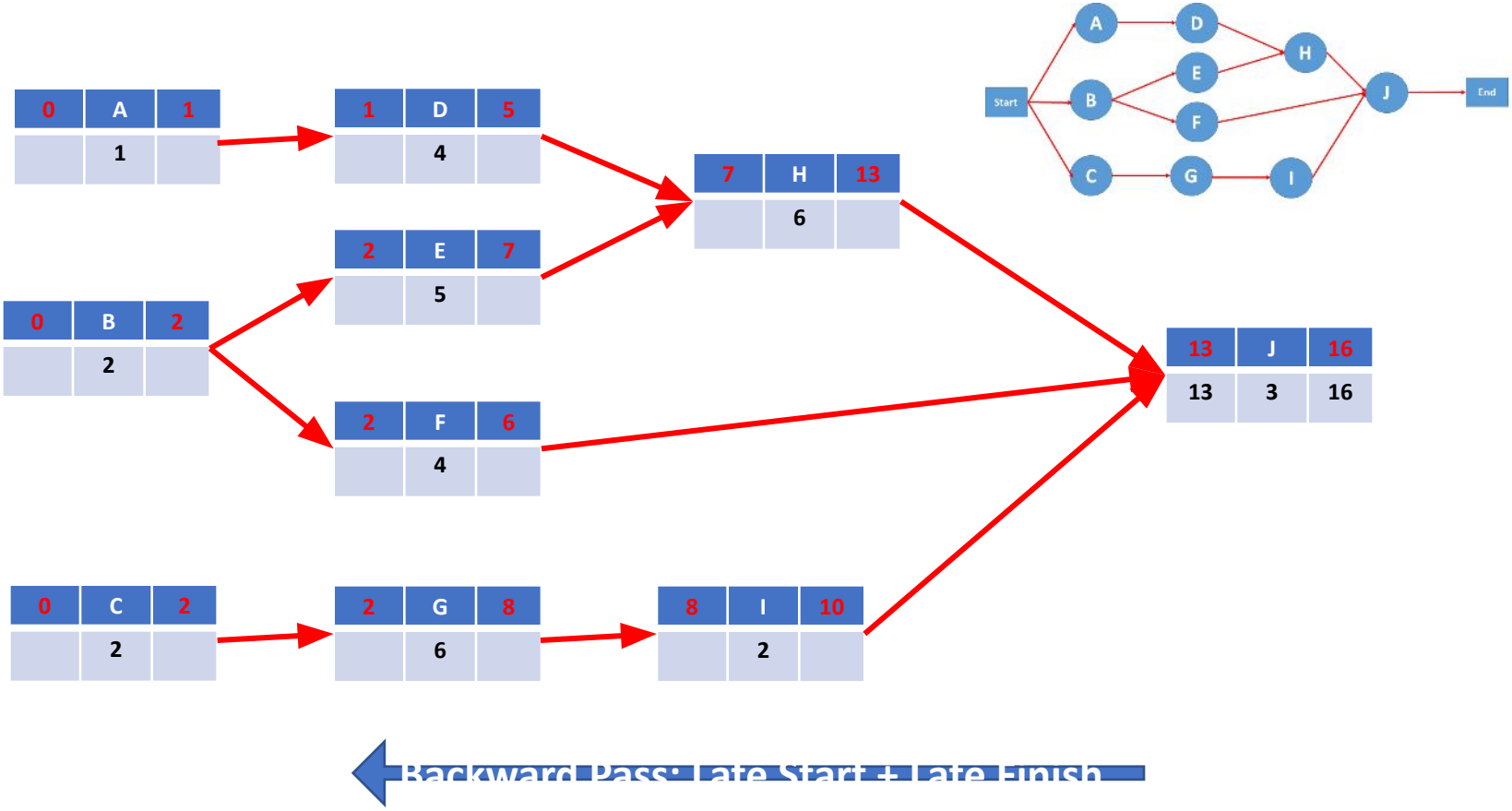
Project Tasks: Precedence Diagramming Method (PDM)



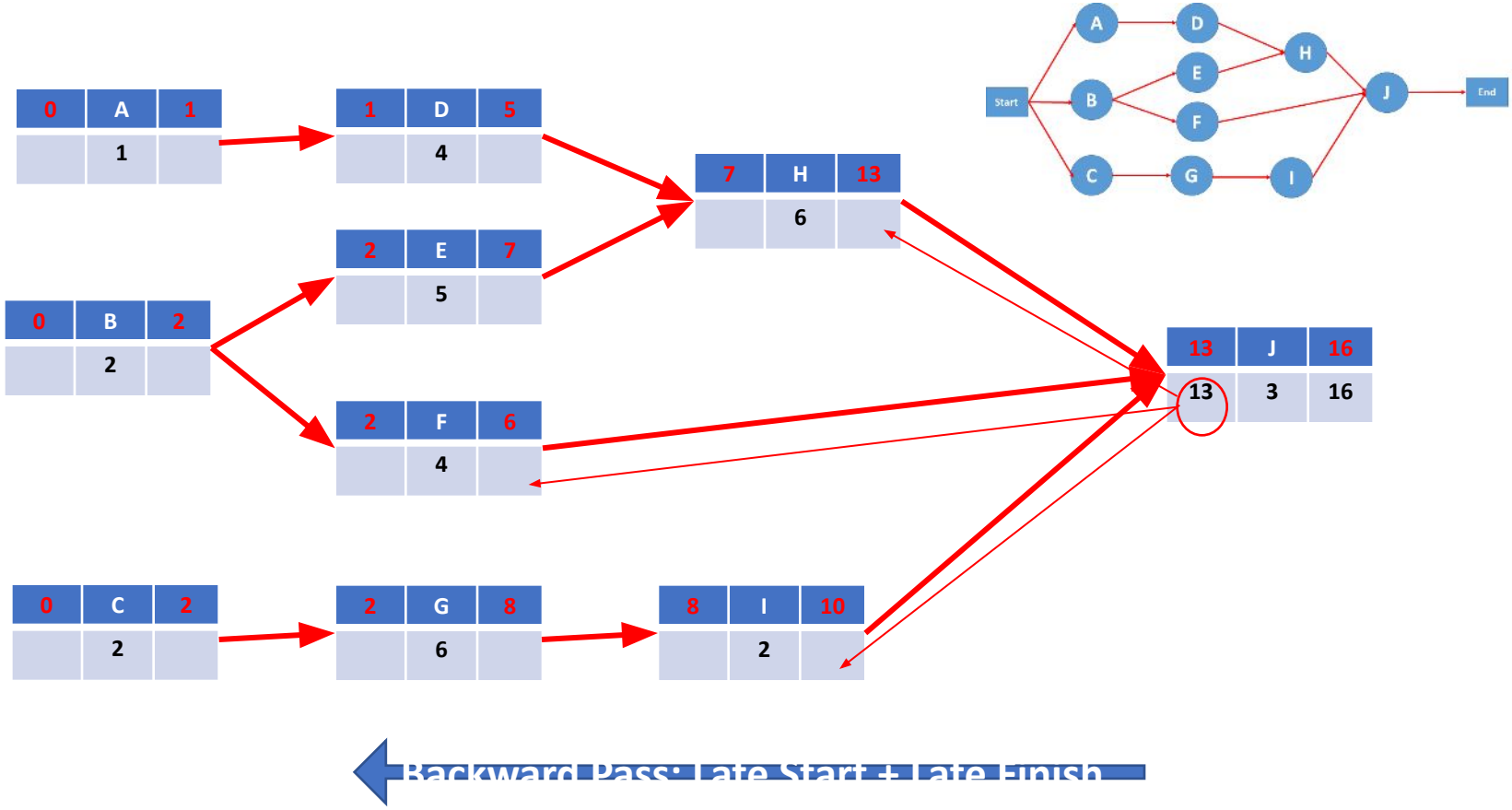
Project Tasks: Precedence Diagramming Method (PDM)



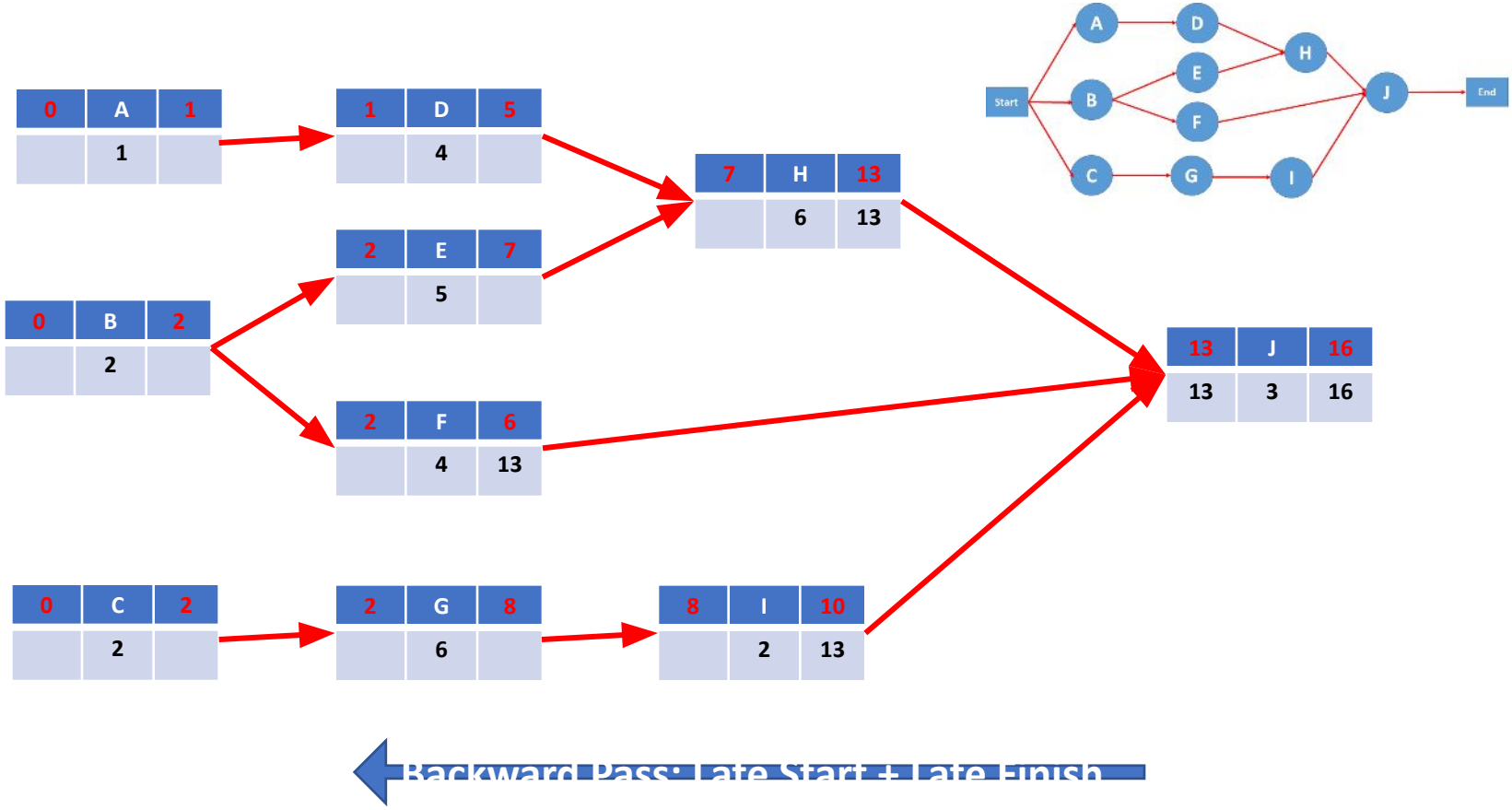
Project Tasks: Precedence Diagramming Method (PDM)



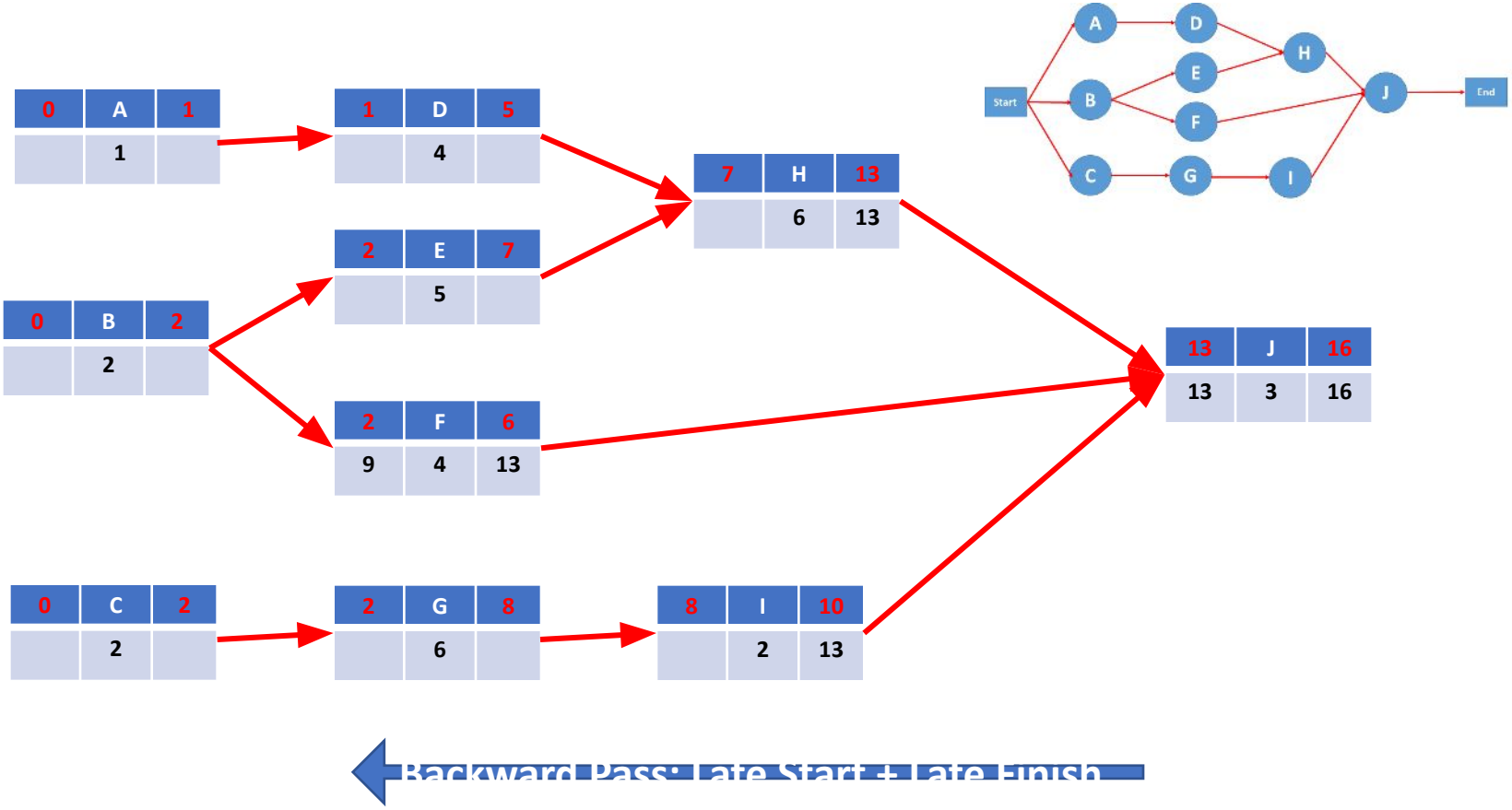
Project Tasks: Precedence Diagramming Method (PDM)



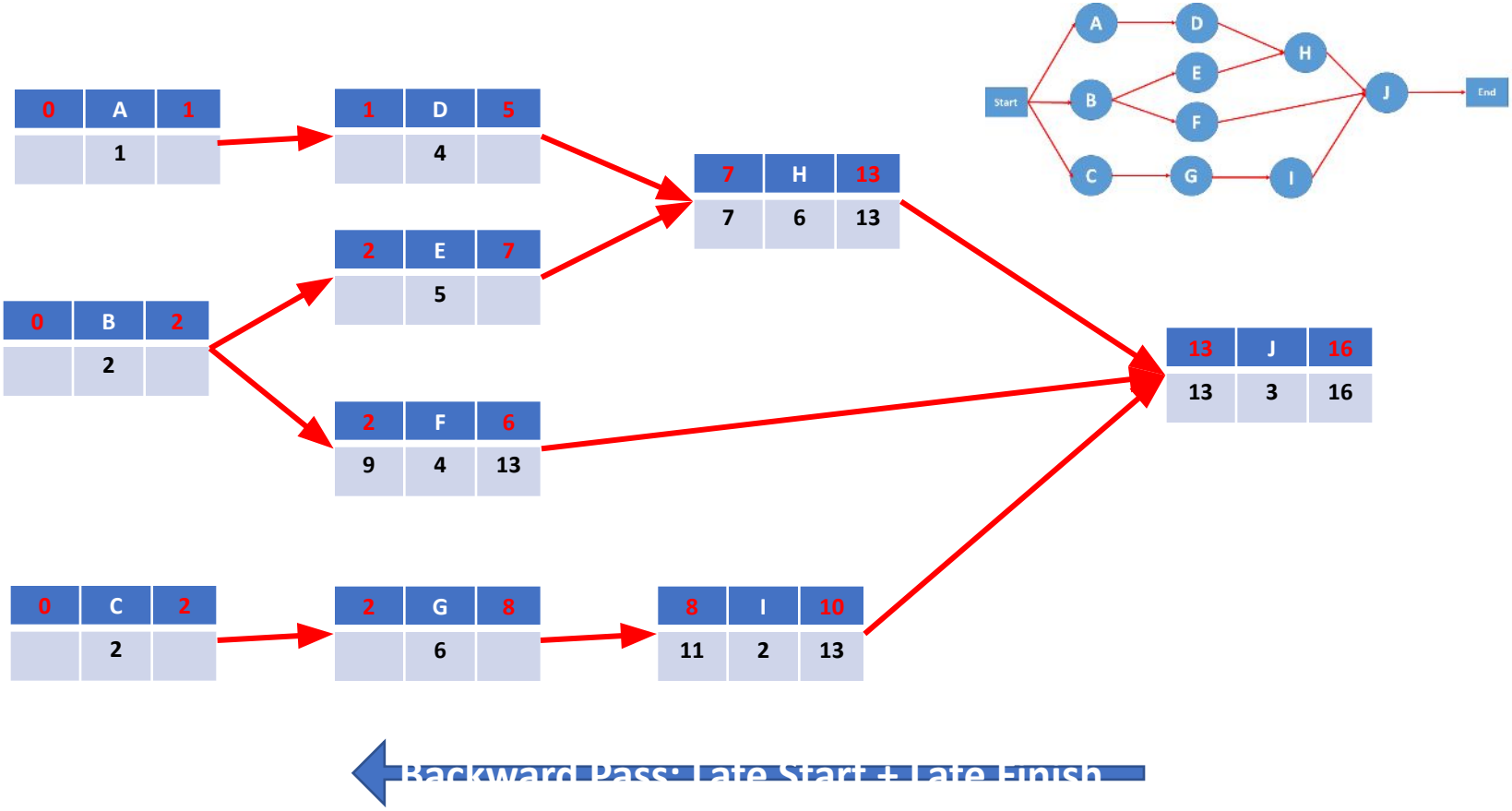
Project Tasks: Precedence Diagramming Method (PDM)



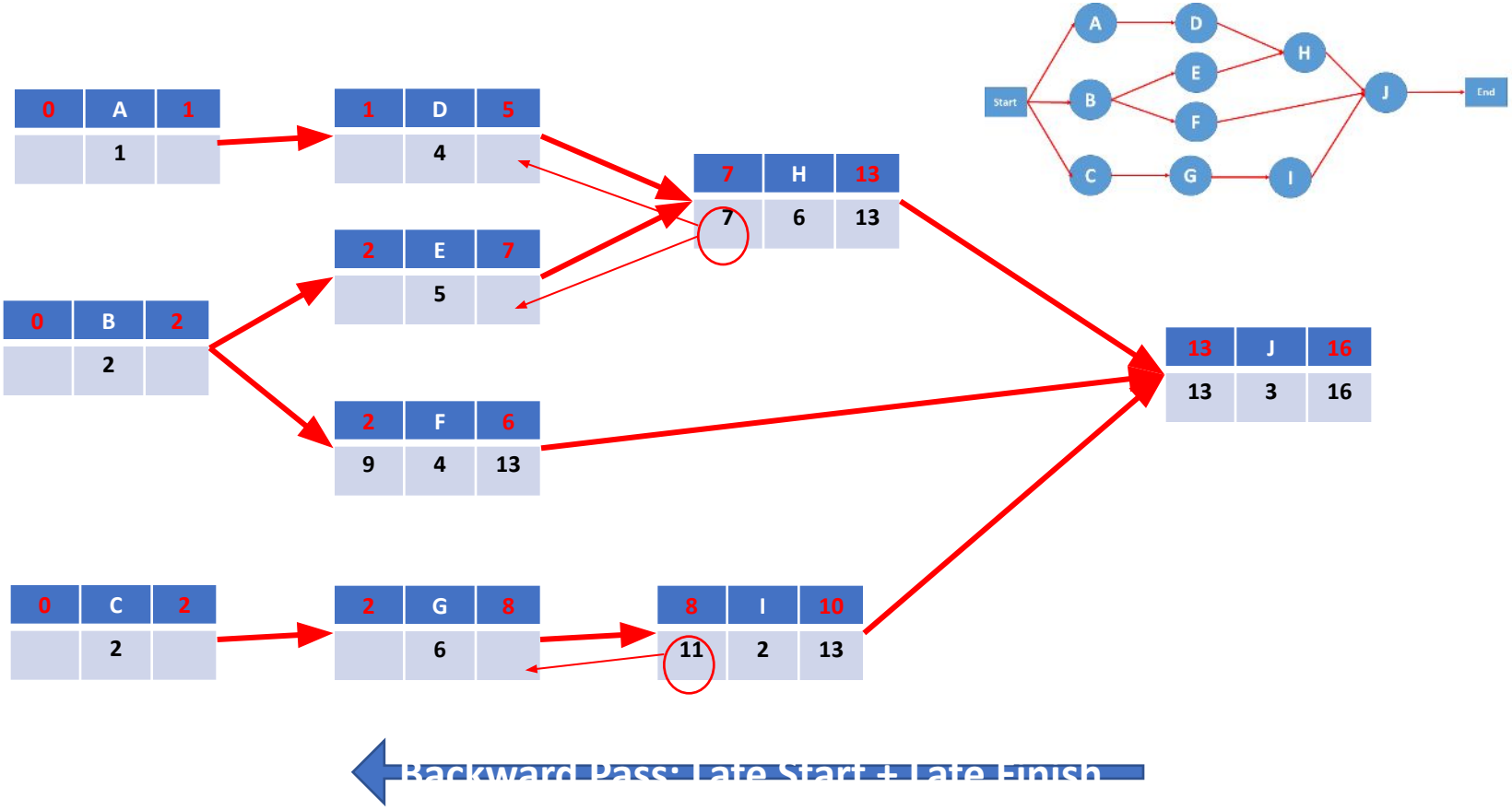
Project Tasks: Precedence Diagramming Method (PDM)



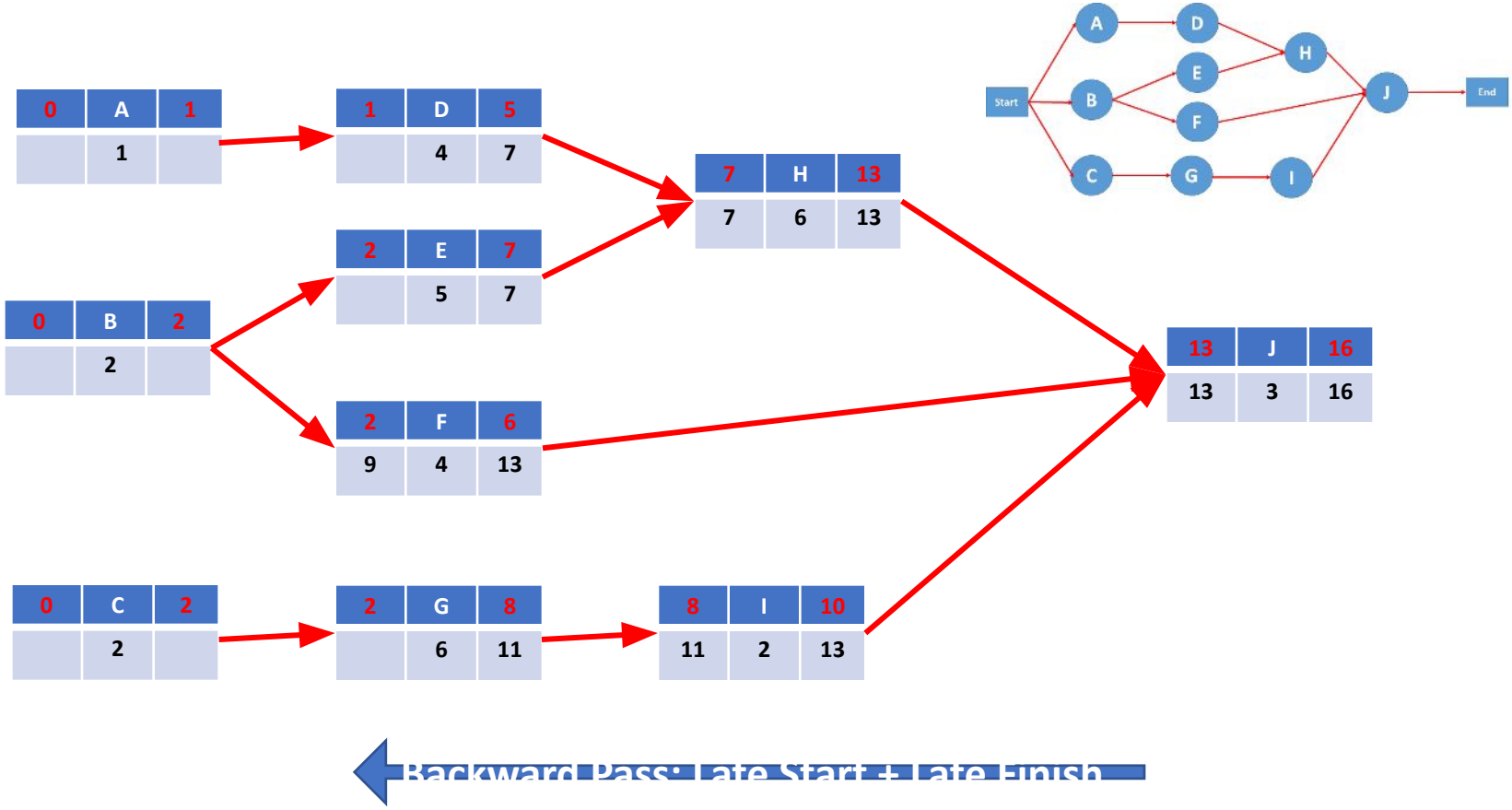
Project Tasks: Precedence Diagramming Method (PDM)



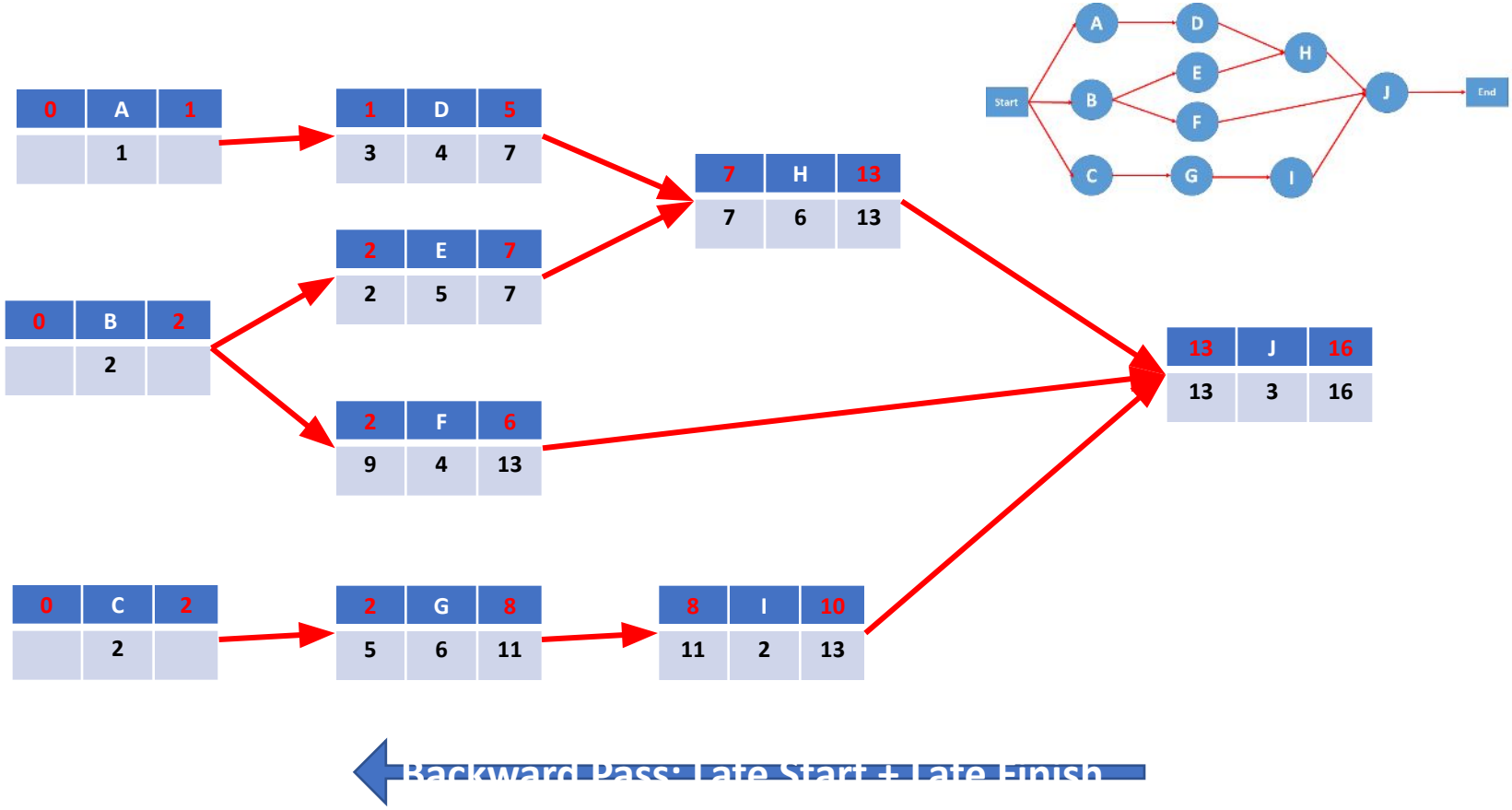
Project Tasks: Precedence Diagramming Method (PDM)



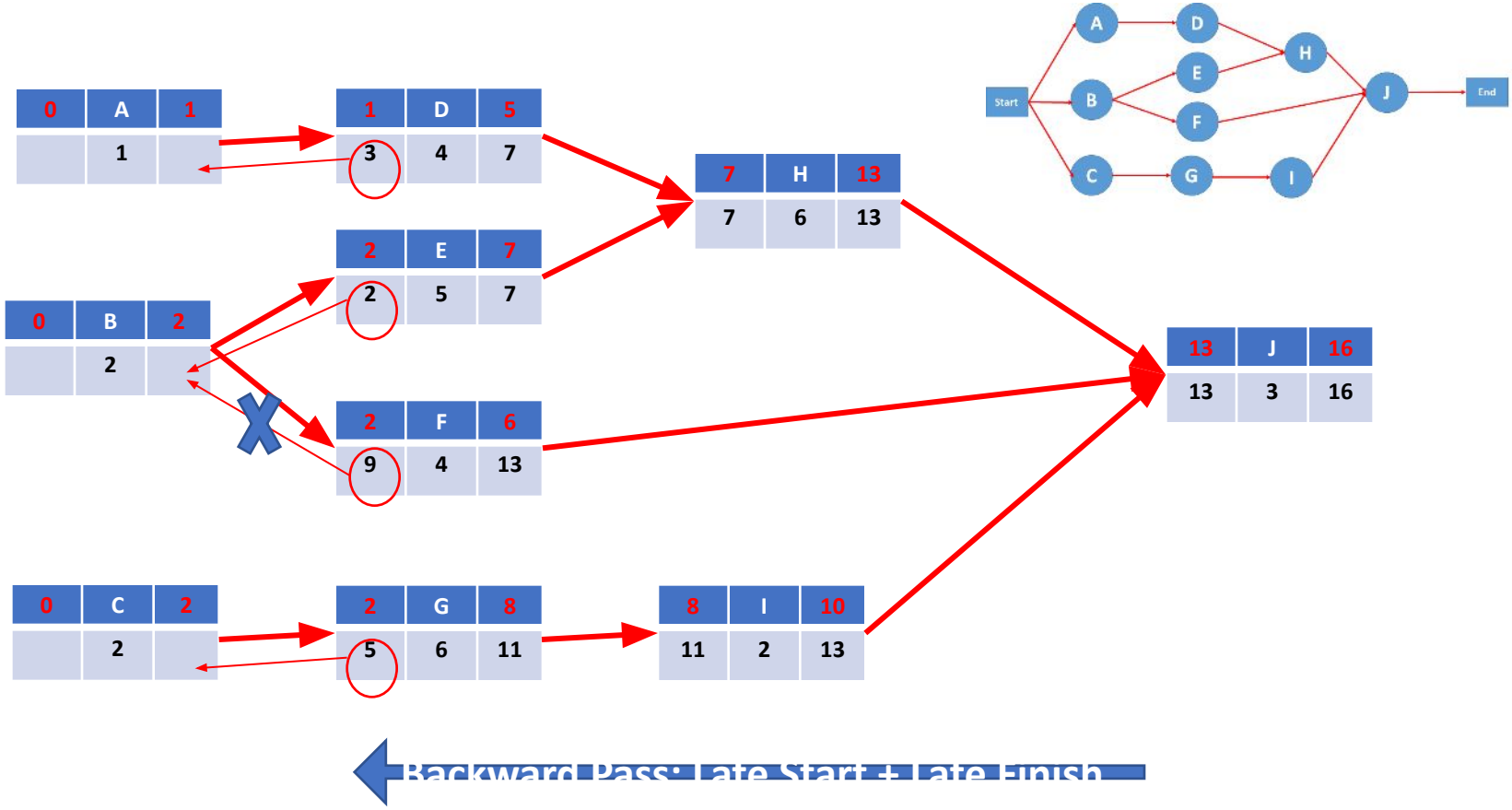
Project Tasks: Precedence Diagramming Method (PDM)



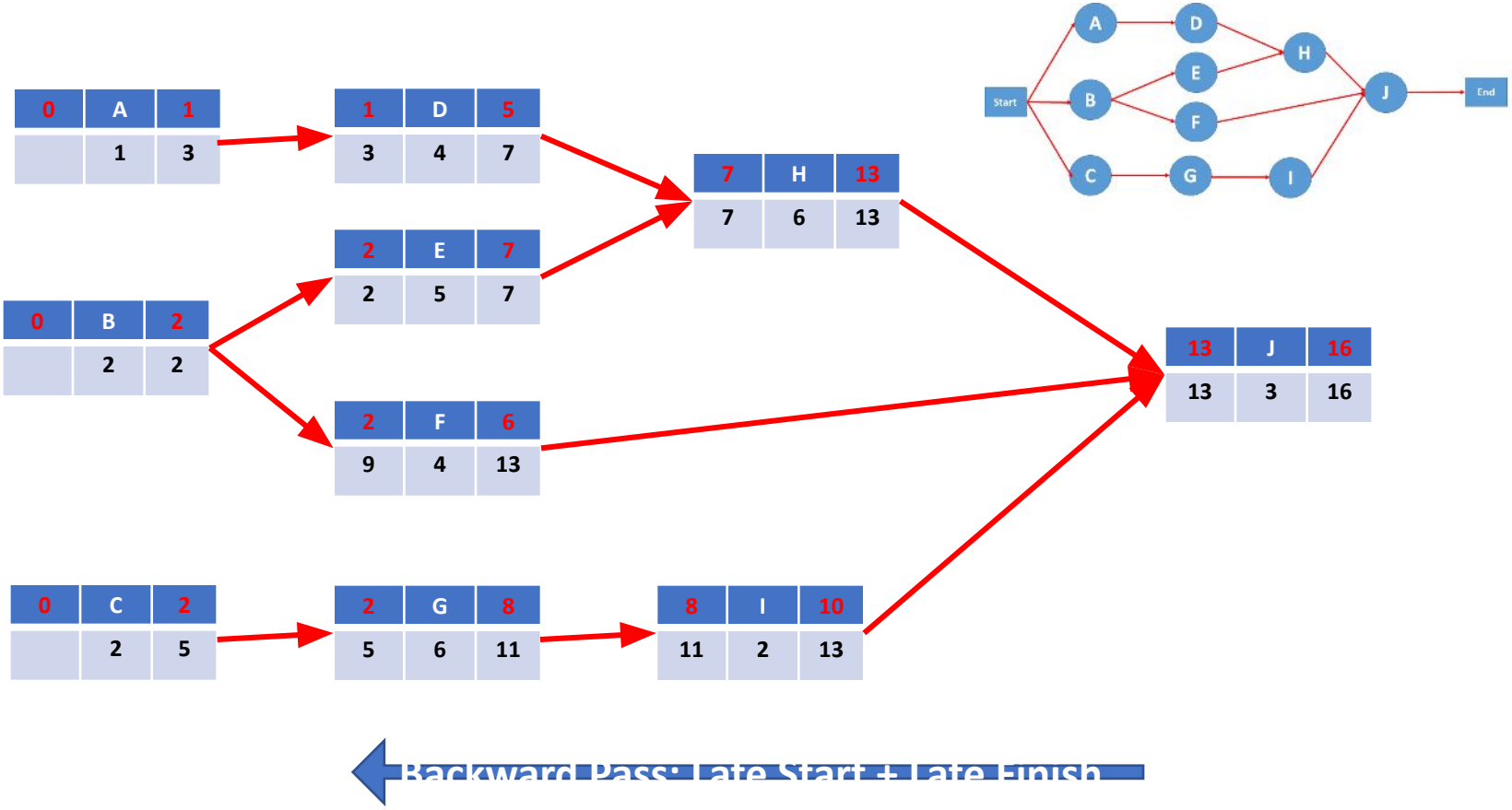
Project Tasks: Precedence Diagramming Method (PDM)



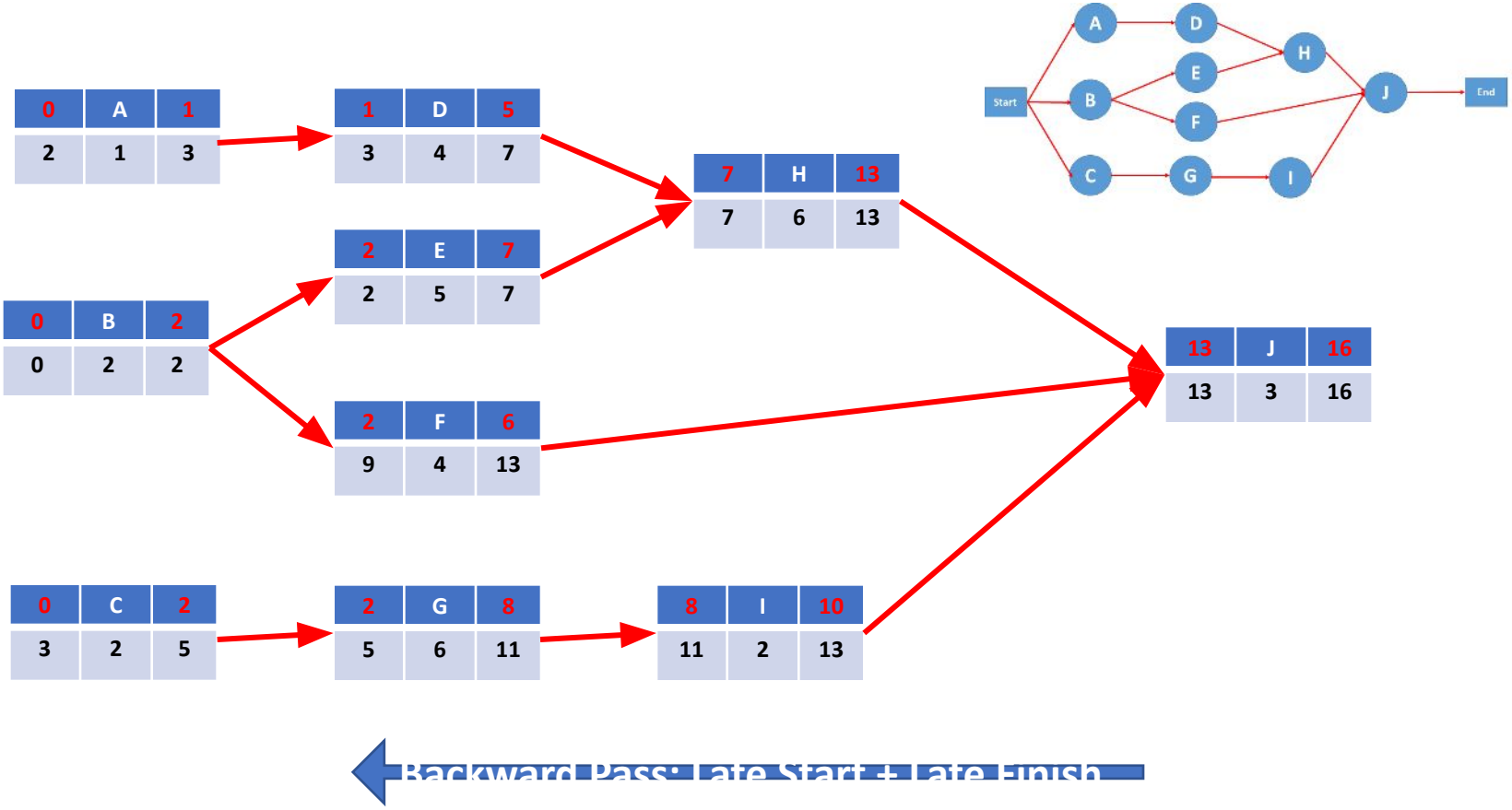
Project Tasks: Precedence Diagramming Method (PDM)



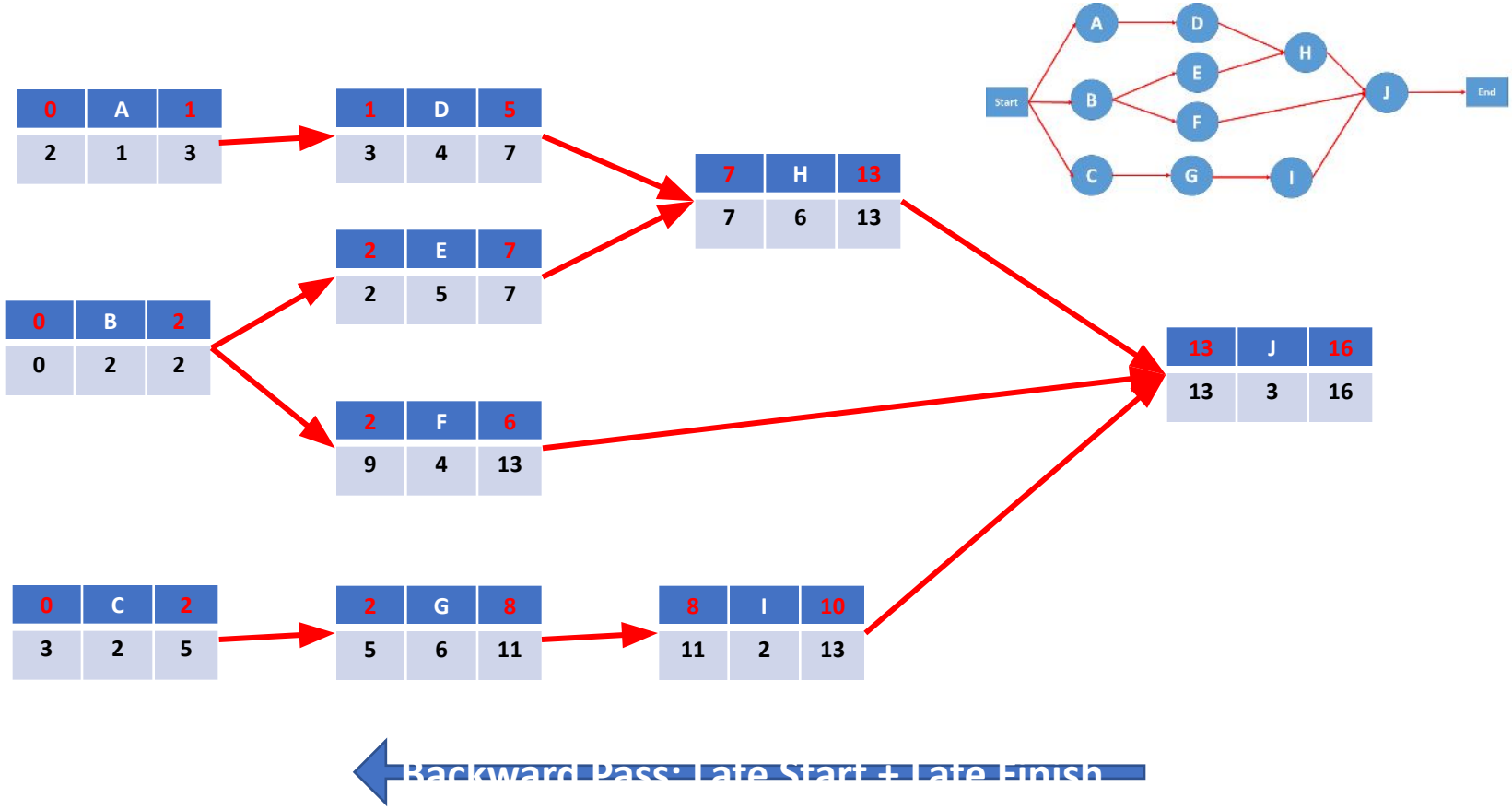
Project Tasks: Precedence Diagramming Method (PDM)



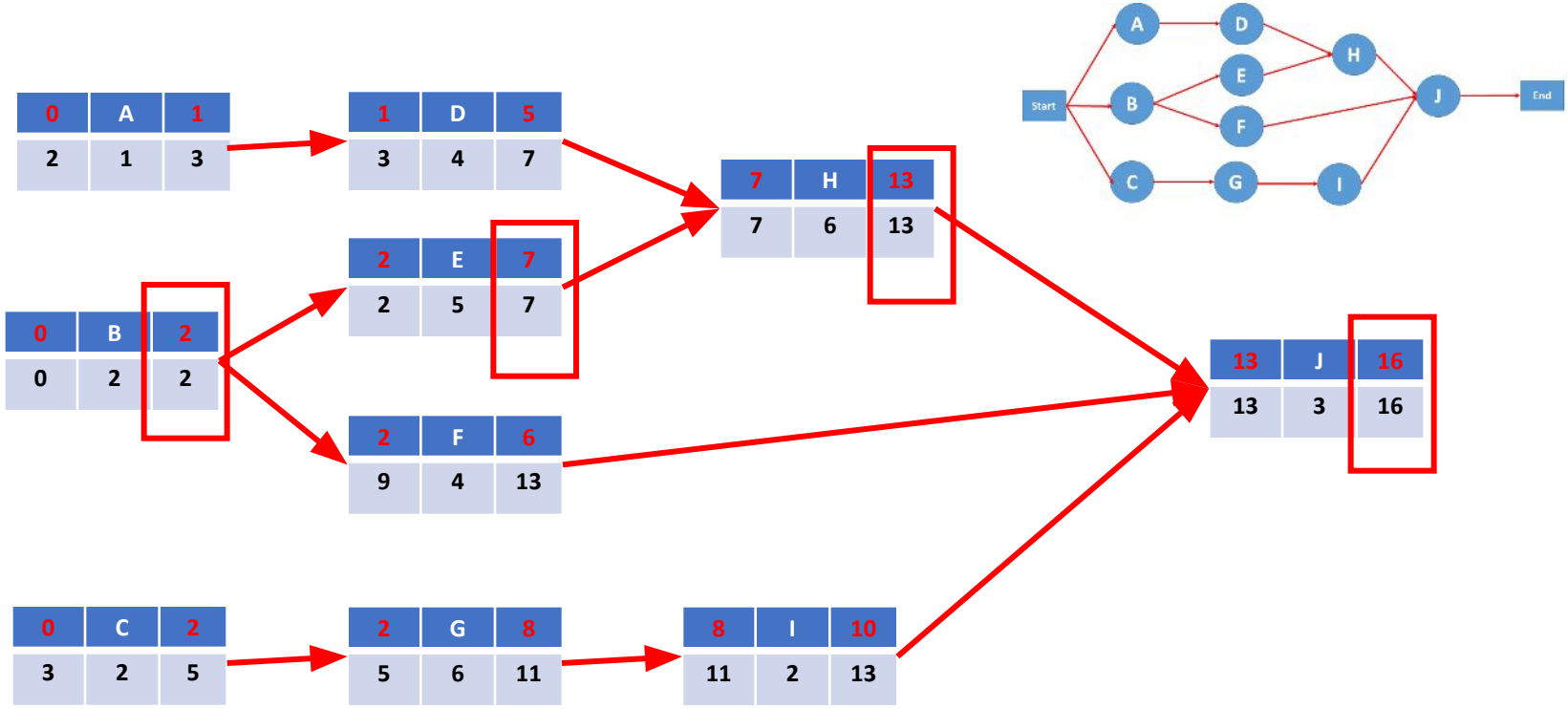
Project Tasks: Precedence Diagramming Method (PDM)



Project Tasks: Precedence Diagramming Method (PDM)

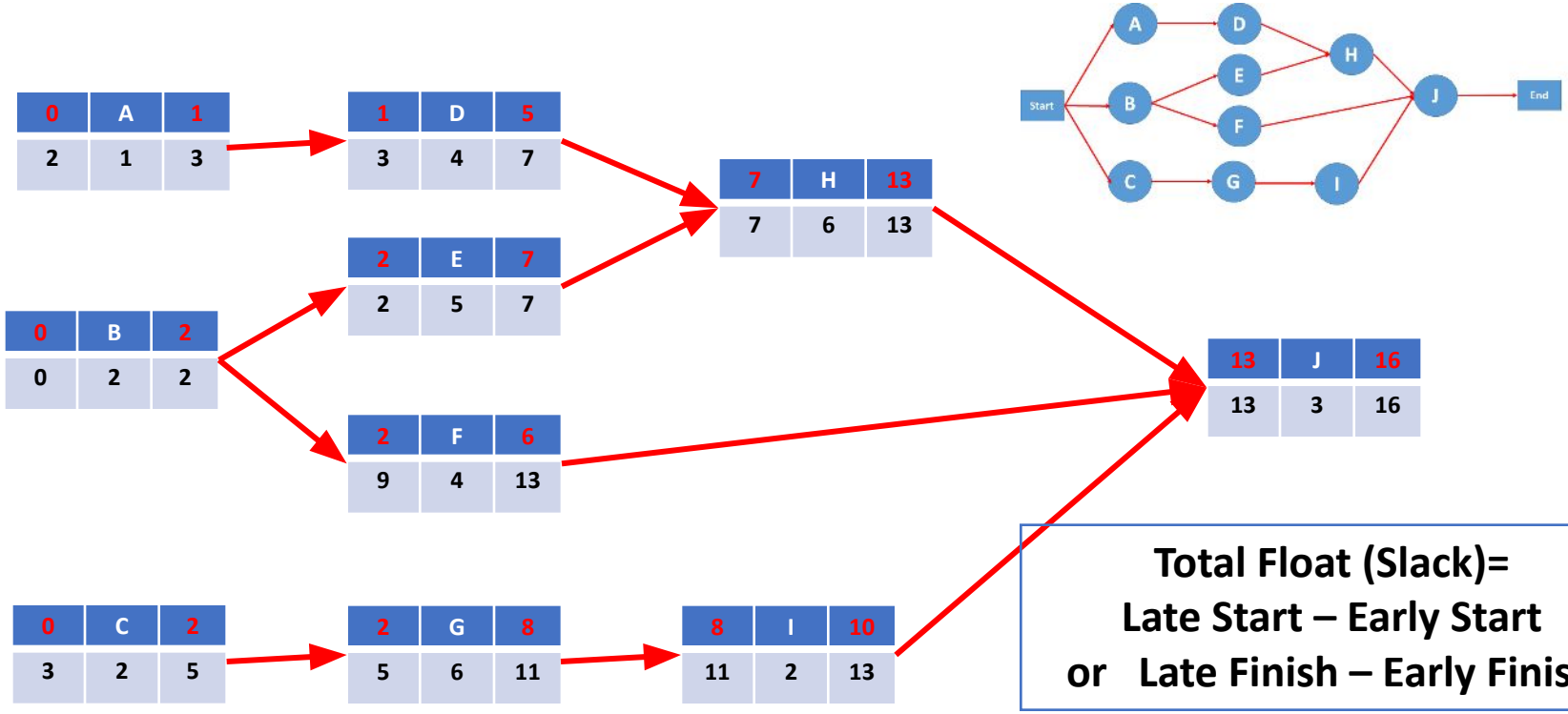


Project Tasks: Precedence Diagramming Method (PDM)



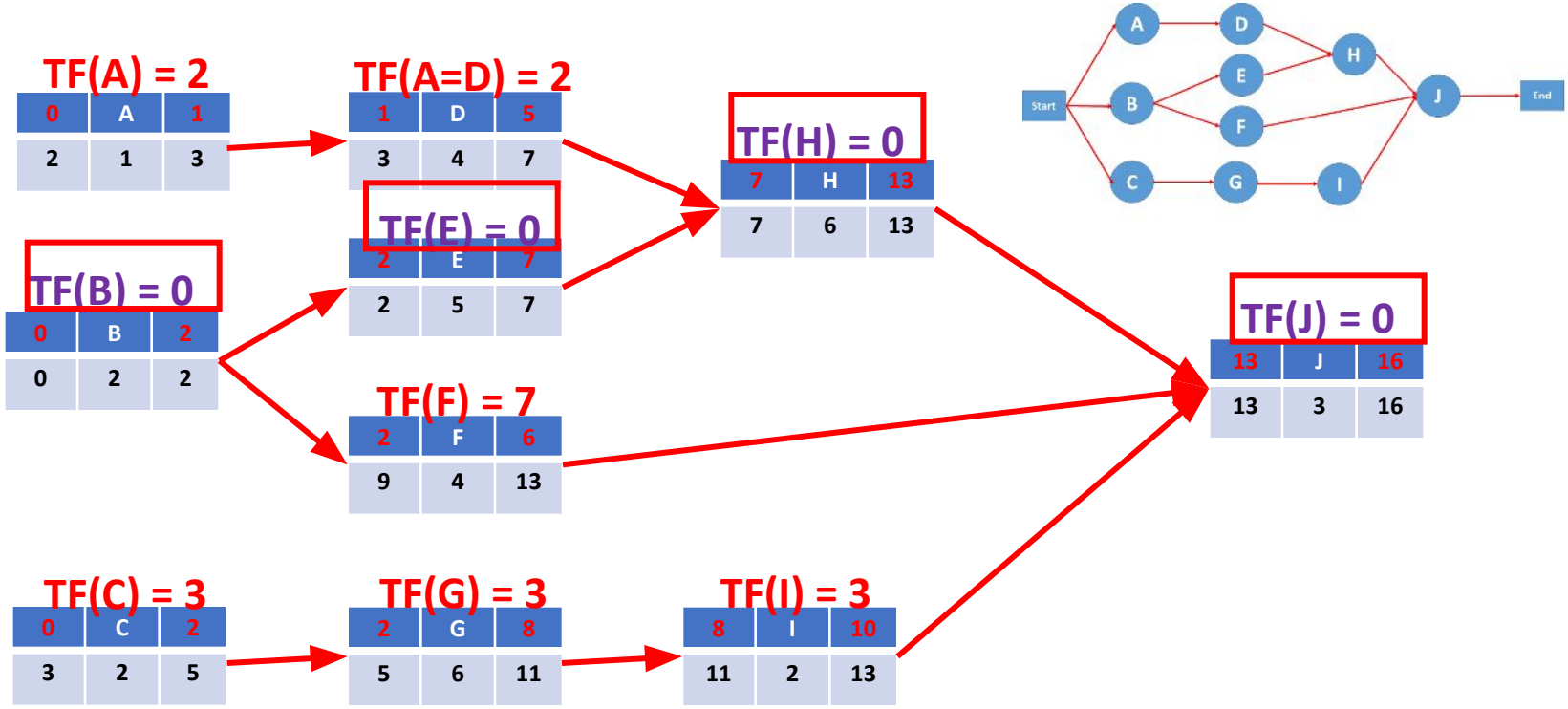
Path: B-E-H-J is a CRITICAL PATH (each task in this path has EF=LF)

Project Tasks: Precedence Diagramming Method (PDM)



Total Float (Definition): Amount of time an activity (task) can be delayed without affecting the total duration of the whole project!

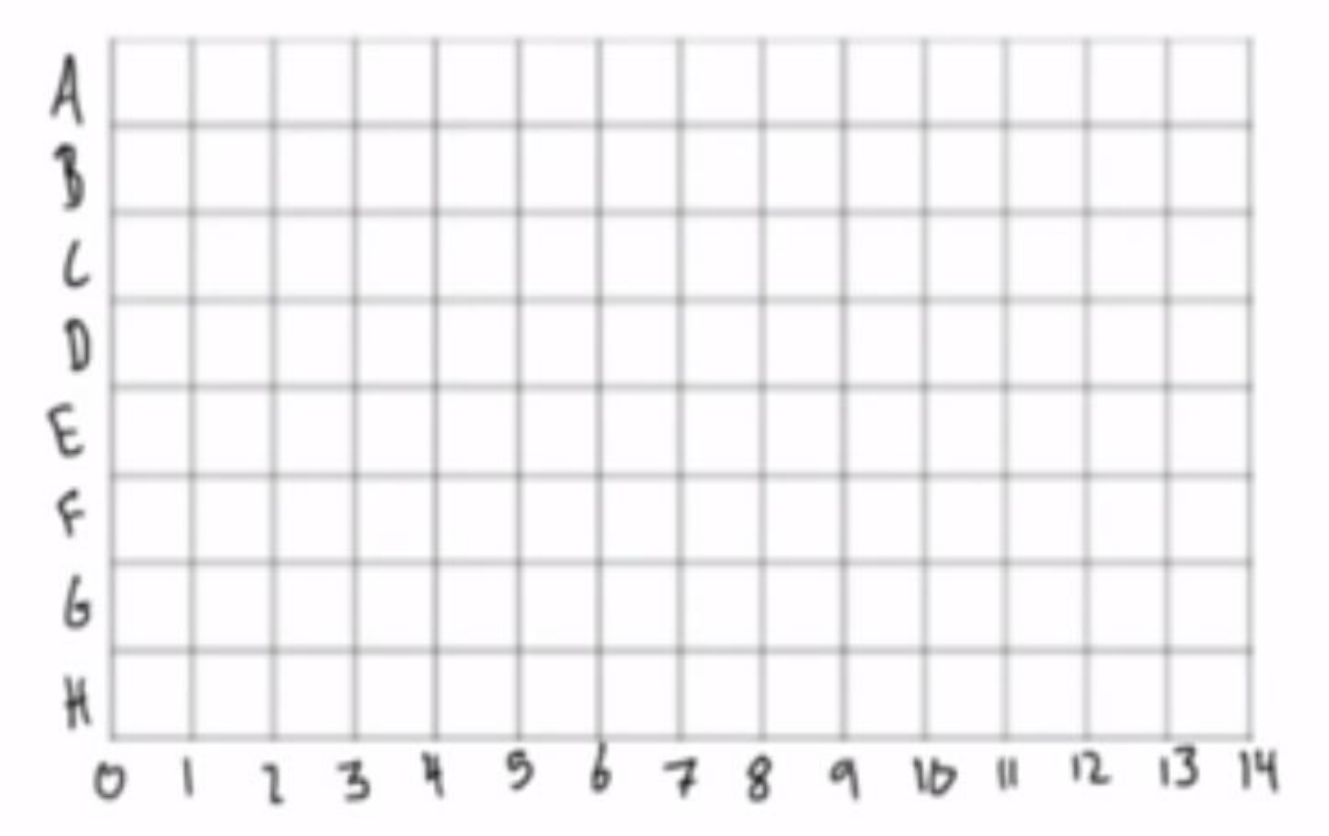
Project Tasks: Precedence Diagramming Method (PDM)



Project Tasks: Gantt chart

Task	Predecessor	Duration
A	------	3
B	A	2
C	A	2
D	B	4
E	C	1
F	C	2
G	D , E	4
H	F , G	3

Project Tasks: Gantt chart



Project Gantt Chart

For 25th October 2023

Assignment

Activity	Predecessor	Duration
A	-	5
B	A	4
C	A	5
D	B	6
E	C	3
F	D,E	4

Task 1: Draw the PDM diagram.
Task 2: Find the early start (ES), finish (EF), late start (LS) and finish (LF) for each task.
Task 3: Find the total slacks (or floats) for each task.

Part III
Risk Management

Topics

- Risk Management
- Communication Management

Risk Management

- Risk management is concerned with identifying risks affecting the project schedule or the quality of the software and drawing up plans to avoid or minimise their effect on a project.
- Software risk management is important because of the inherent uncertainties in software development.
  - These uncertainties stem from loosely defined requirements, requirements changes due to changes in customer needs, difficulties in estimating the time and resources required for software development, and differences in individual skills.
- You have to anticipate risks, understand the impact of these risks on the project, the product and the business, and take steps to avoid these risks.

Risk Classification

There are two dimensions of risk classification

- The type of risk (technical, organizational, ..)
- what is affected by the risk:
  - ***Project risks*** affect schedule or resources e.g. loss of an experienced programmer.
  - ***Product risks*** affect the quality or performance of the software being developed e.g. failure of a purchased component as expected.
  - ***Business risks*** affect the organisation developing or procuring the software e.g. a competitor launching a new product .

Examples of project, product, and business risks

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

The Risk Management Process

- **Risk Identification**

  Identify project, product and business risks;

- **Risk Analysis**

  Assess the likelihood and consequences of these risks;

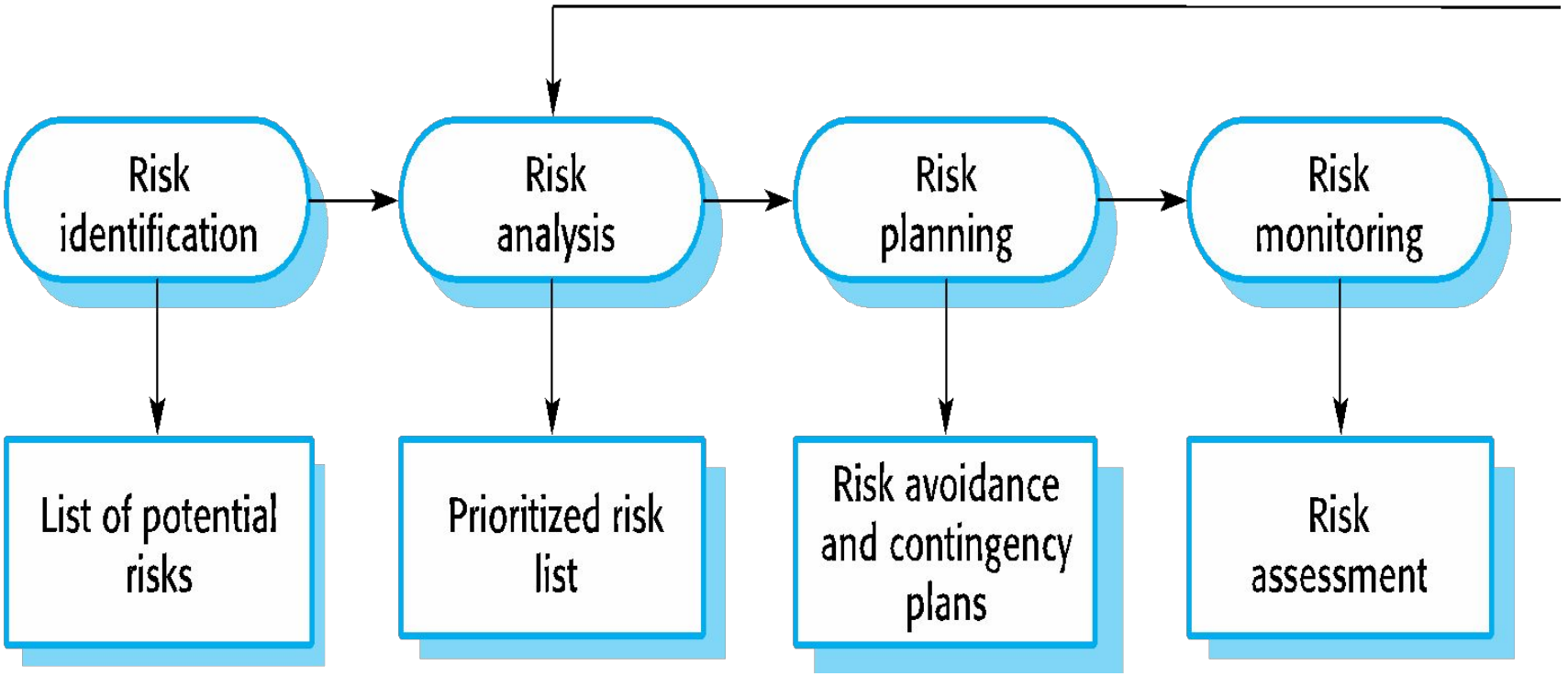
- **Risk Planning**

  Draw up plans to avoid or minimise the effects of the risk;

- **Risk Monitoring**

  Regularly assess the risk and your plans for risk mitigation and revise these when you learn more about the risk

The Risk Management Process



Risk Identification

- First stage of the risk management process.
- May be team activities or based on the individual project manager's experience to identify the most probable or critical risks.
- A checklist of common risks may be used to identify risks in a project
  - Technology risks.
  - Organizational risks.
  - People risks.
  - Requirements risks.
  - Estimation risks.
  - Tools risks

Examples of Different Risk Types

Risk type	Possible risks
Estimation	The time required to develop the software is underestimated. The rate of defect repair is overderestimated. The size of the software is underestimated.
Organizational	The organization is restructured so that different management are responsible for the project. Organizational financial problems force reductions in the project budget.
People	It is impossible to recruit staff with the skills required. Key staff are ill and unavailable at critical times. Required training for staff is not available.
Requirements	Changes to requirements that require major design rework are proposed. Customers fail to understand the impact of requirements changes.
Technology	The database used in the system cannot process as many transactions per second as expected. Reusable software components contain defects that mean they cannot be reused as planned.
Tools	The code generated by software code generation tools is inefficient. Software tools cannot work together in an integrated way.

Risk Analysis

- Consider each identified risk and assess probability and seriousness of each risk.
- Mostly rely on the PM's own judgment and experience of previous projects.
- Probability may be very low(<10%), low, (10-25%), moderate(25-50%), high (50-75%) or very high(>75%).
- Risk consequences might be catastrophic (threaten the survival of the project), serious (would cause major delays), tolerable (delays are within allowed contingency), or insignificant.

Risk Examples with probability and effects estimation

h

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget.	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project.	High	Catastrophic
Key staff are ill at critical times in the project.	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused.	Moderate	Serious
Changes to requirements that require major design rework are proposed.	Moderate	Serious
The organization is restructured so that different management are responsible for the project.	High	Serious
The database used in the system cannot process as many transactions per second as expected.	Moderate	Serious

Risk Examples with probability and effects estimation

Risk	Probability	Effects
The time required to develop the software is underestimated.	High	Serious
Software tools cannot be integrated.	High	Tolerable
Customers fail to understand the impact of requirements changes.	Moderate	Tolerable
Required training for staff is not available.	Moderate	Tolerable
The rate of defect repair is underestimated.	Moderate	Tolerable
The size of the software is underestimated.	High	Tolerable
Code generated by code generation tools is inefficient.	Moderate	Insignificant

Risk Planning

- Consider each identified risk and develop strategies to manage those risks.

- **Avoidance strategies**

  The probability that the risk will arise is reduced

  e.g. dealing with defective components

- **Minimization strategies**

- The impact of the risk on the project or product will be reduced

  e.g. the strategy for staff illnesses

- **Contingency plans**

  If the risk arises, prepare plans to deal with that risk;

Risk Monitoring

- Assess each identified risks regularly to decide whether or not it is becoming less or more probable.
- Also assess whether the effects of the risk have changed.
- Each key risk should be discussed at management progress meetings.

Risk indicators

Risk type	Potential indicators
Estimation	Failure to meet agreed schedule; failure to clear reported defects.
Organizational	Organizational gossip; lack of action by senior management.
People	Poor staff morale; poor relationships amongst team members; high staff turnover.
Requirements	Many requirements change requests; customer complaints.
Technology	Late delivery of hardware or support software; many reported technology problems.
Tools	Reluctance by team members to use tools; complaints about CASE tools; demands for higher-powered workstations.