Module 11

Project Planning

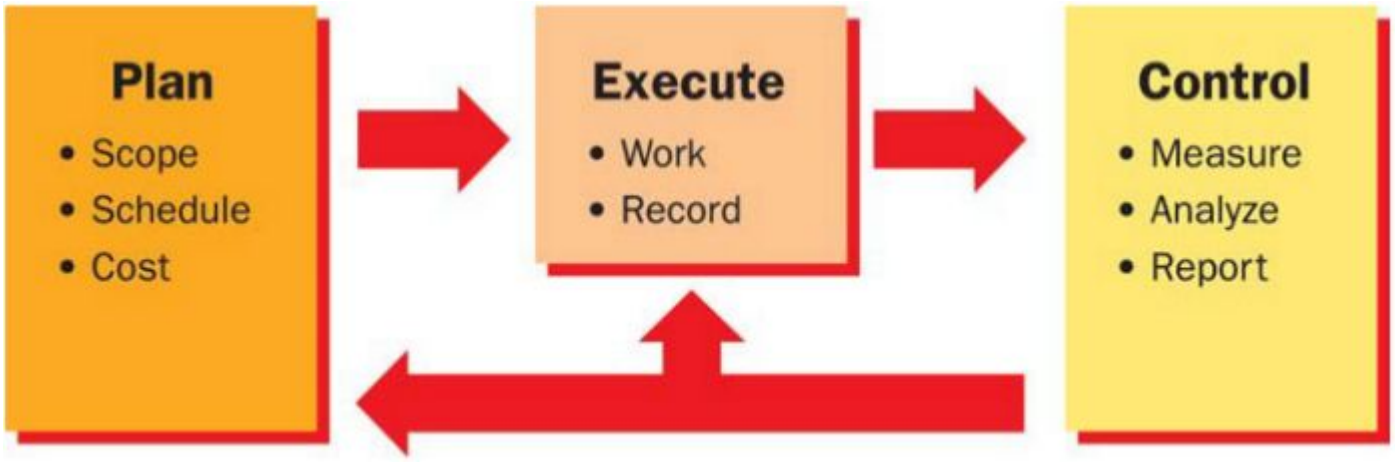
Vasco Amaral
vma@fct.unl.pt

We have finished the part of system modelling with UML but always remember:

Model everything, using the most appropriate formalism, at the right level of abstraction

Managing projects

To manage a given project means to **Plan**, **Execute** and **control** the Work Activity of that same project



To Plan a given project means to understand:

- **What** is the work that must be accomplished(scope) and what are their corresponding activity components (Work Breakdown Structure)
- **Who** is going to execute and manage the work to be done (responsibility matrix)
- **When** is the work going to be done (calendar)
- **Cost** of work, materials and other required resources for its accomplishment

To Execute a given project means:

- to **accomplish** the work that needs to be done according to what was planned;

- and **keep informed** the team and managers.

To Control a given project means:

- Monitor and Report the execution of the management plan pertaining to **scope**, **time** and **cost**, as well as **quality** and **risk**
- Main goal: to keep work performance and its results aligned with the initial plans, within a tolerance margin

Why is Control so challenging?

Project Planning

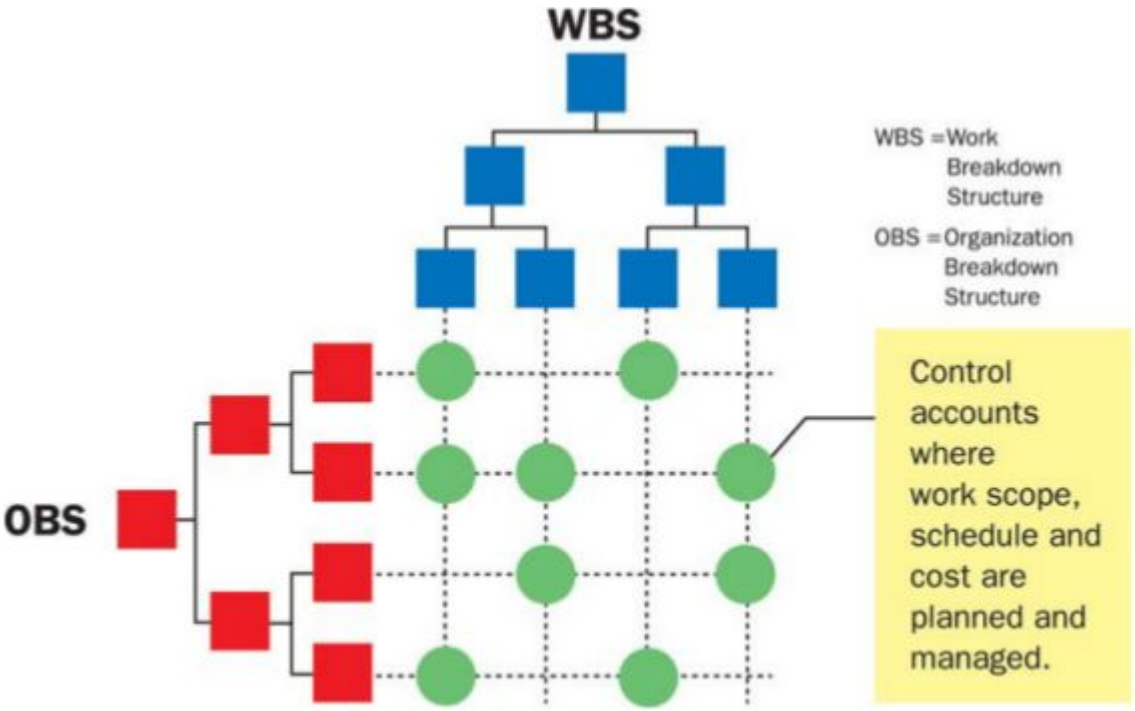
Software project planning

- Identify activities
- Schedule activities
- Assign resources

Activities and resources: scheduling?

Work Breakdown Structure (WBS)

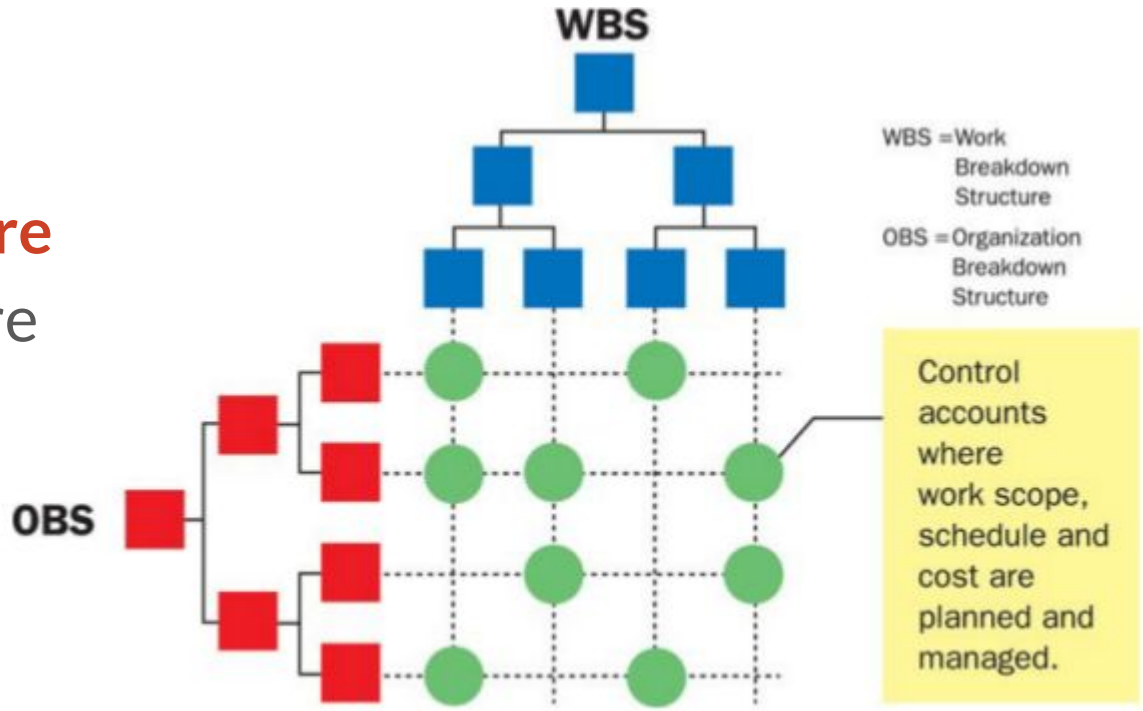
Decomposes the work to be done in a set of activities



Activities and resources: scheduling?

Organization Breakdown Structure

Creates the structure of the organization and is useful for relating elements to the project activity



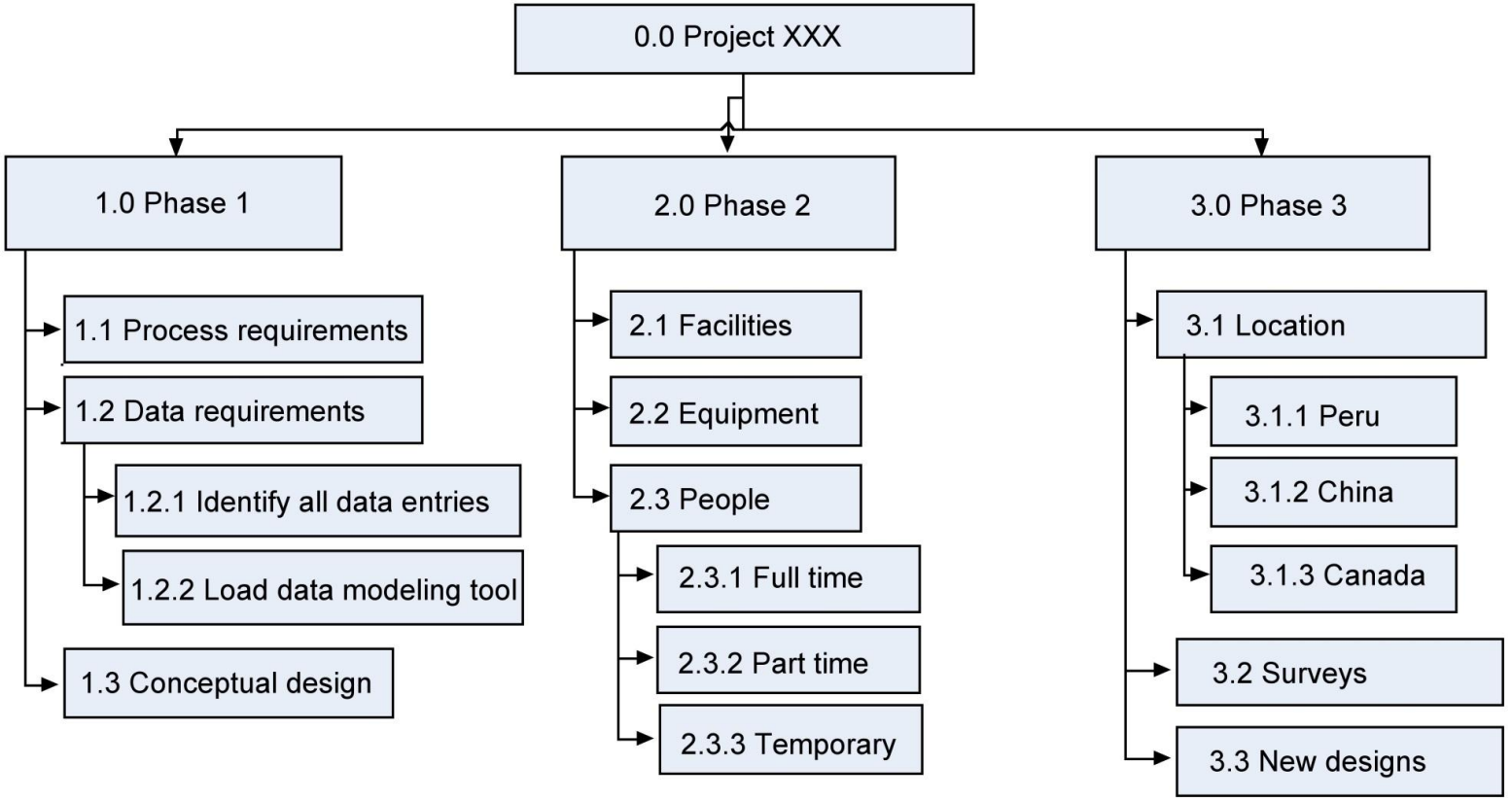
Activity identification

Work Breakdown Structure

Work-Breakdown Structure

- Defines the scope of the project (“to do” list)
- Breaks work down to components
  - Subdivides complex tasks into simpler ones
- Hints on building a WBS
  - Include 100% of the work / deliverables
  - Avoid overlapping elements
  - Plan first for work outcomes, rather than actions
  - Try to get the “sweet spot” of detail
  - Too little makes planning harder
  - Too much hinders the communication role of the WBS

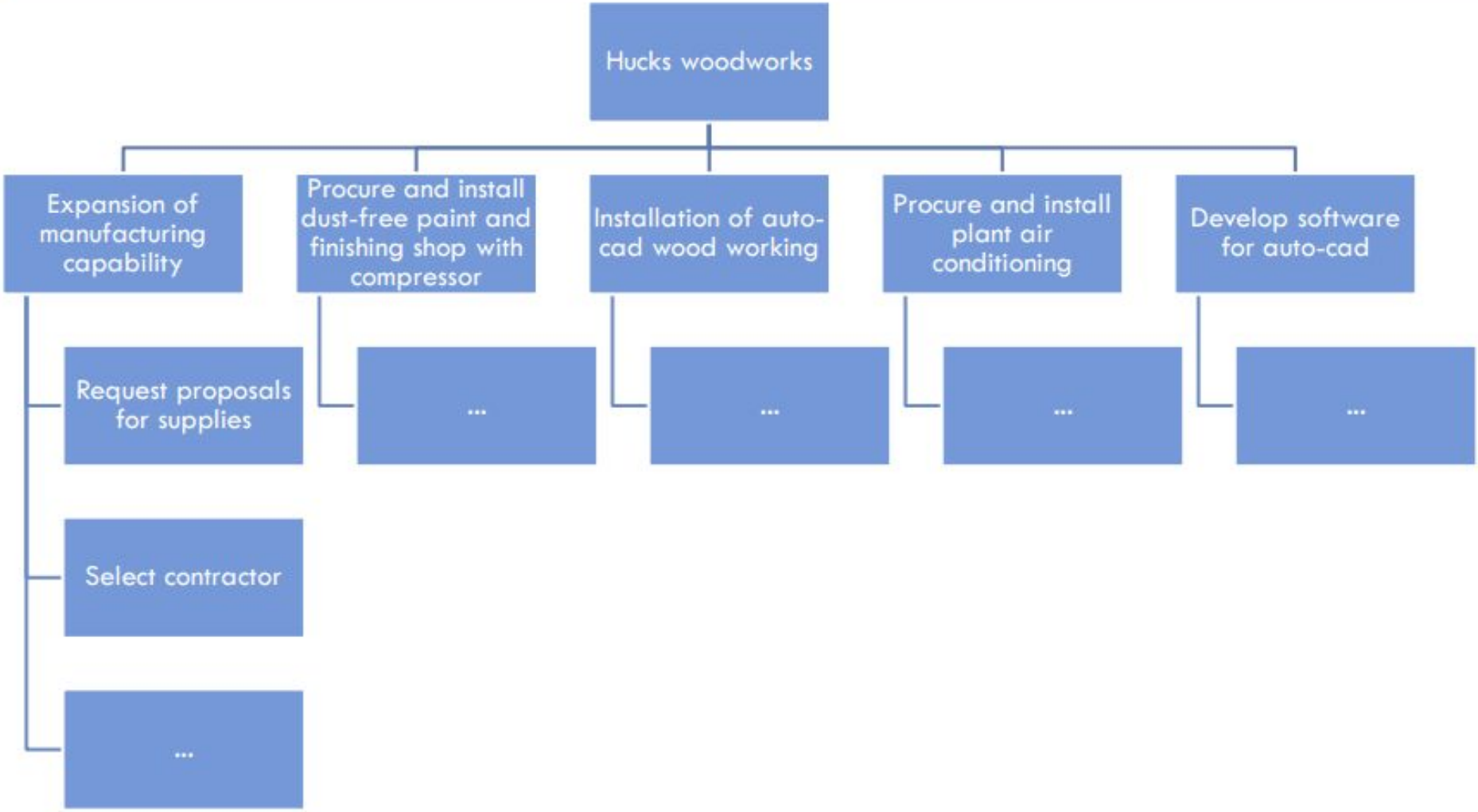
WBS example


0.0 Project XXX
1.0 Phase 1
2.0 Phase 2
3.0 Phase 3
1.1 Process requirements
1.2 Data requirements
1.2.1 Identify all data entries
1.2.2 Load data modeling tool
1.3 Conceptual design
2.1 Facilities
2.2 Equipment
2.3 People
2.3.1 Full time
2.3.2 Part time
2.3.3 Temporary
3.1 Location
3.1.1 Peru
3.1.2 China
3.1.3 Canada
3.2 Surveys
3.3 New designs

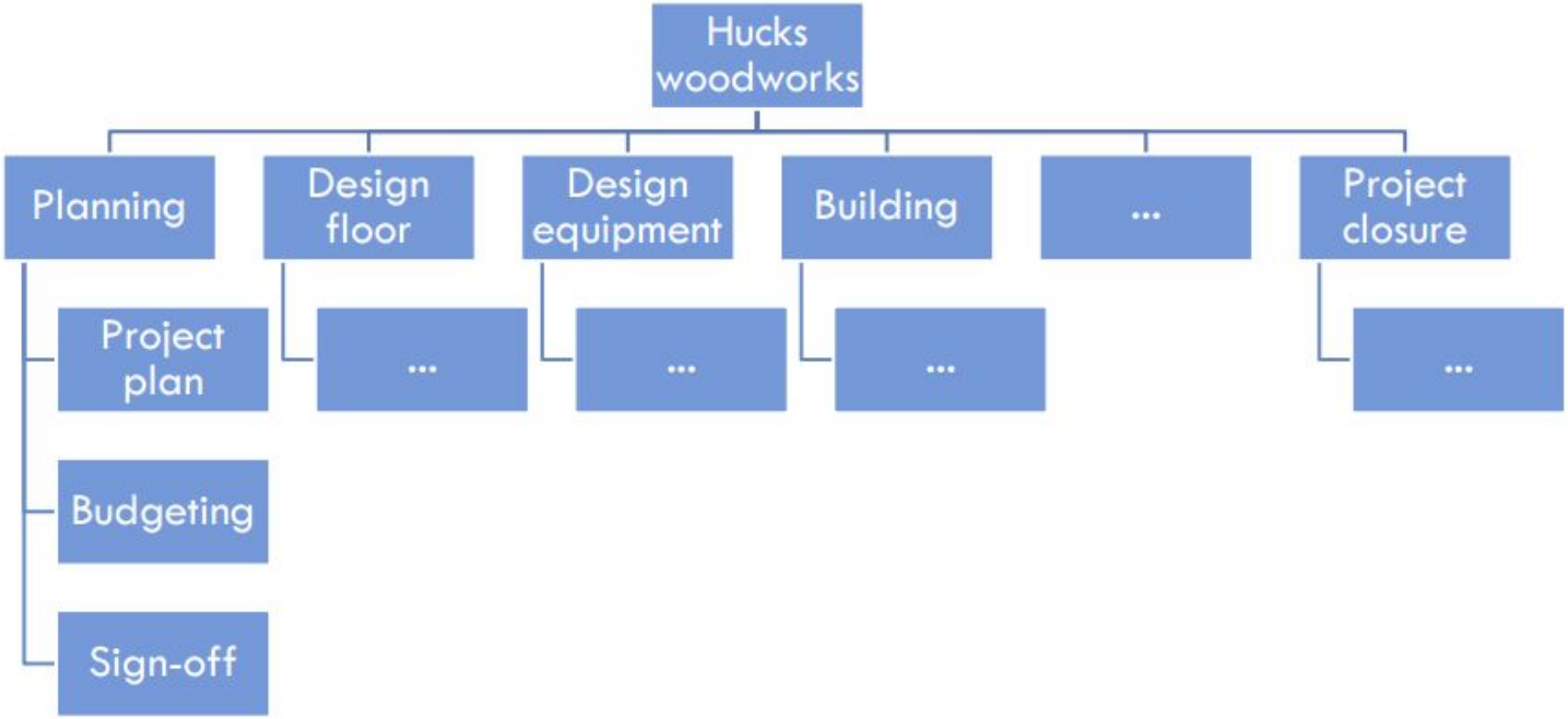
Creating Work-Breakdown Structures

- Several possible **decomposing strategies**
  - **Product oriented**
  - **Process oriented**
  - ...
- The WBS may have a different number of **levels**
  - Enough to facilitate estimates on costs, resources, ...
  - Not too many, to facilitate communication
- The lower level components are **work packages**
  - Must be assigned to individuals, or teams, responsible for delivering them
  - Estimates of time, costs and resources are done at the work package level of granularity

Product oriented work breakdown structure



Process oriented work breakdown structure



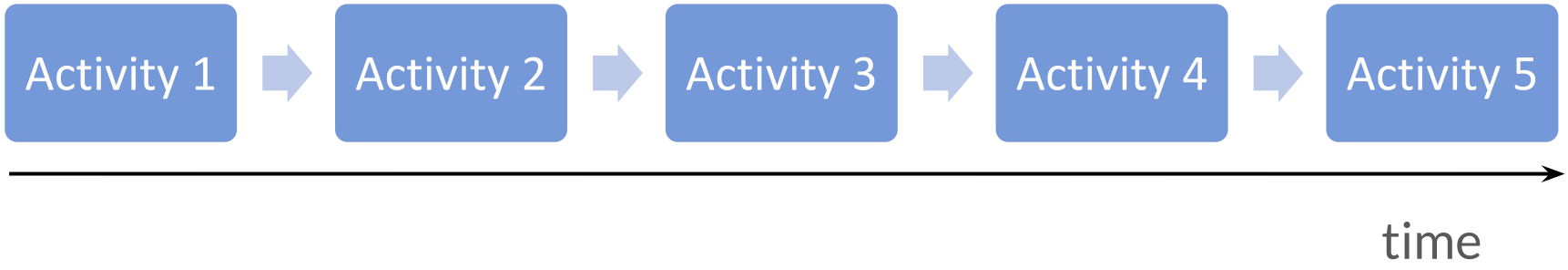
What is the sweet spot for detail?

- The 8/80 rule (of thumb)
  - No work package should be less than 8 hours or more than 80 hours
- Groups of tasks, or activities, once complete, should correspond to the completeness of the corresponding upper level

How should we schedule activities?

Naïve project scheduling

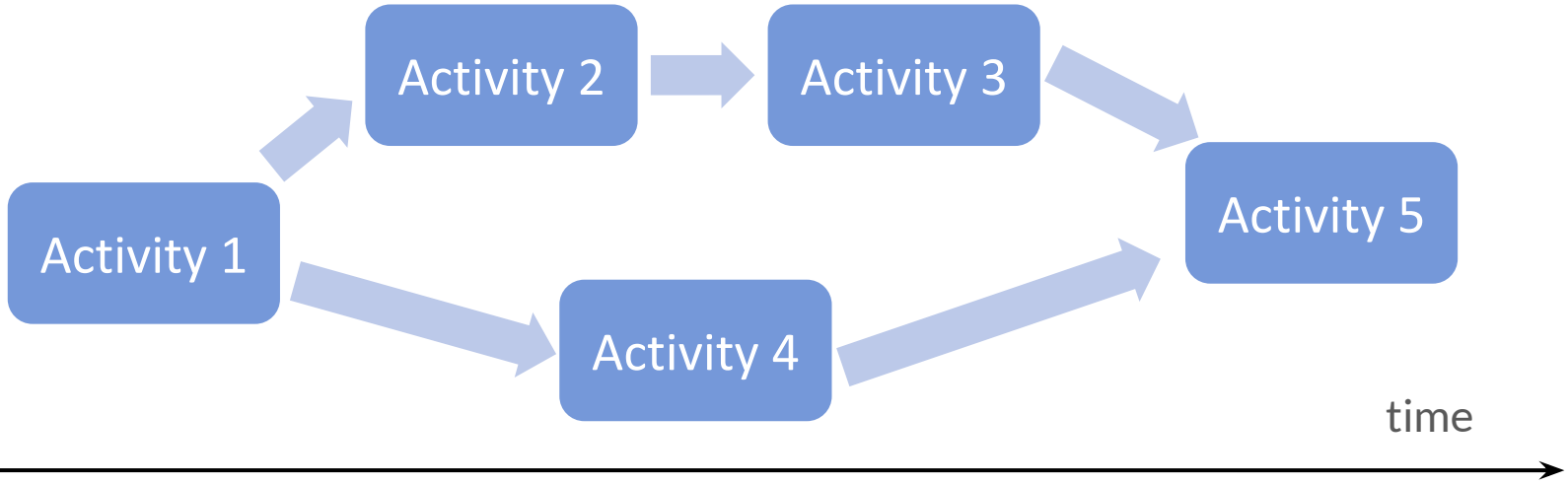
Project: A sequence of interconnected activities to achieve a certain goal



Problem: Longest possible completion schedule

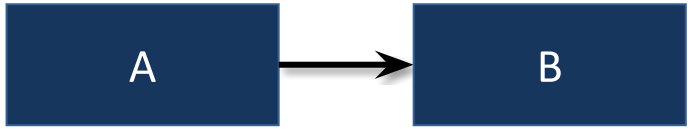
Networked project plan

- Build a network of relationships among activities, so that activities precedences can be established



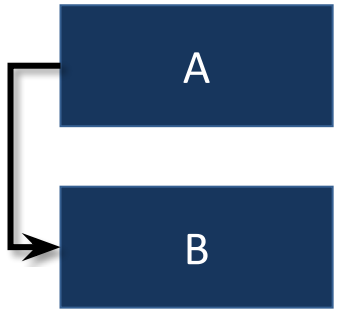
Which activities must finish, before a particular new activity starts?

Dependency relationship (FS)



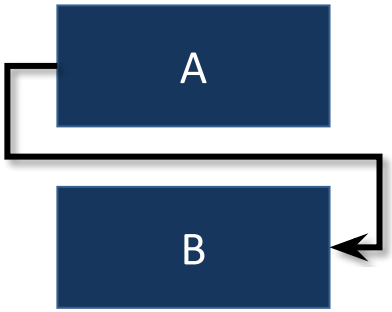
- **F**inish to **S**tart relationship
  - As soon as A finishes, B can start
- Example
  - A is a data collection activity
  - B is a data storing activity
  - As soon as we finish data collection, we can start data storing
- Recommended as default dependency in early planning

Dependency relationship (SS)



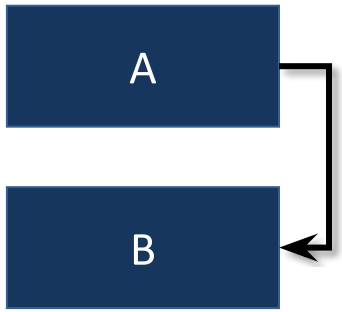
- **S**tart to **S**tart relationship
  - As soon as A starts, B can start
- Example
  - A is a data collection activity
  - B is a data storing activity
  - Data storing cannot start before data collection starts
- Use for compressing activities in time

Dependency relationship (SF)



- **S**tart to **F**inish relationship
  - As soon as A starts, B can finish
- Example
  - A is a new system running
  - B is an old system running
  - As soon as the new system is running, the old system may be discontinued
- Used for just in time scheduling (relatively uncommon)

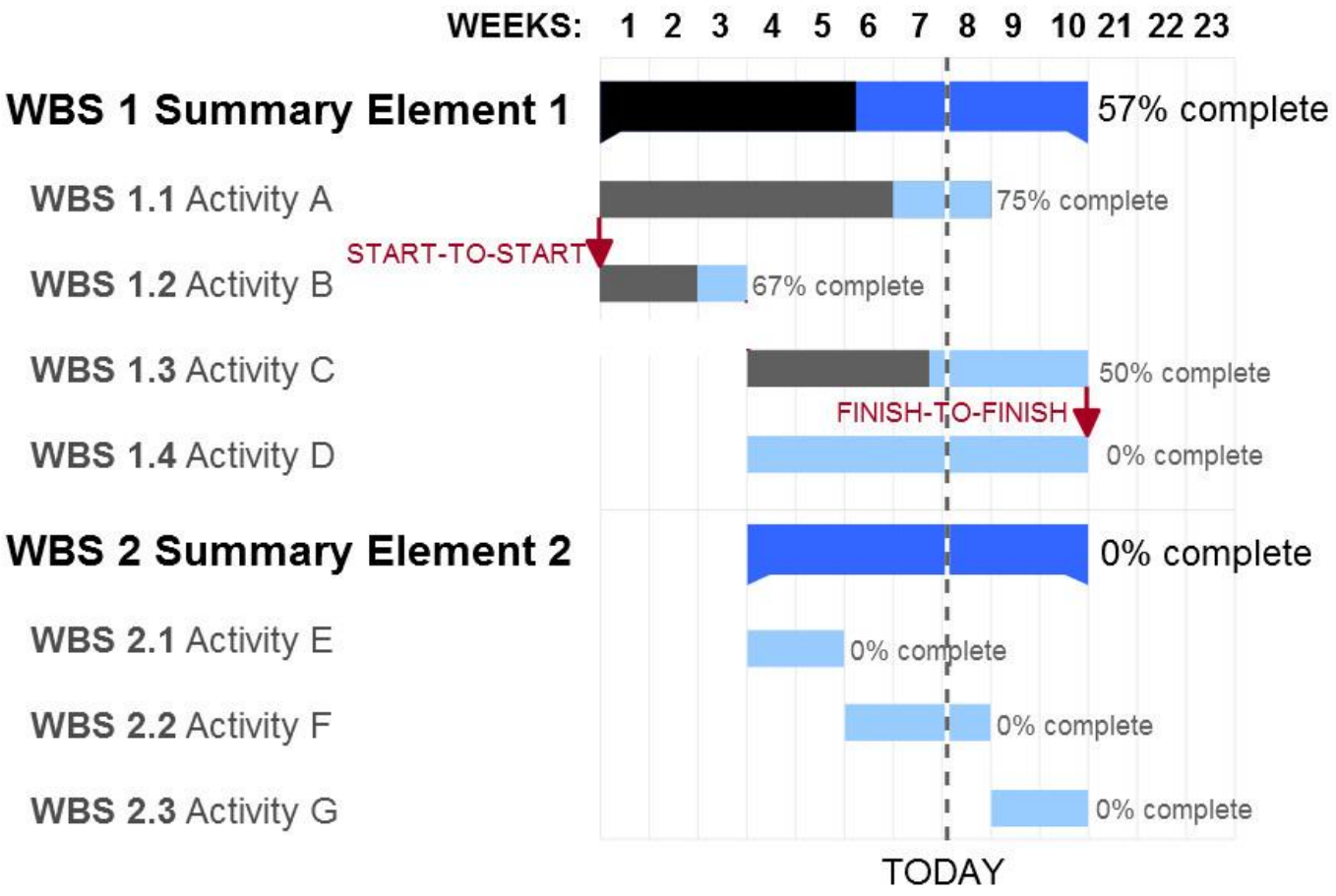
Dependency relationship (FF)



- **F**inish to **F**inish relationship
  - As soon as A finishes, B can finish
- Example
  - A is a data collection activity
  - B is a data storing activity
  - Only when we finish collecting data, can we finish storing it as well
- To preserve network connectivity, **SS** should be used along with **FF**

Gantt charts

Gantt Charts (illustration)

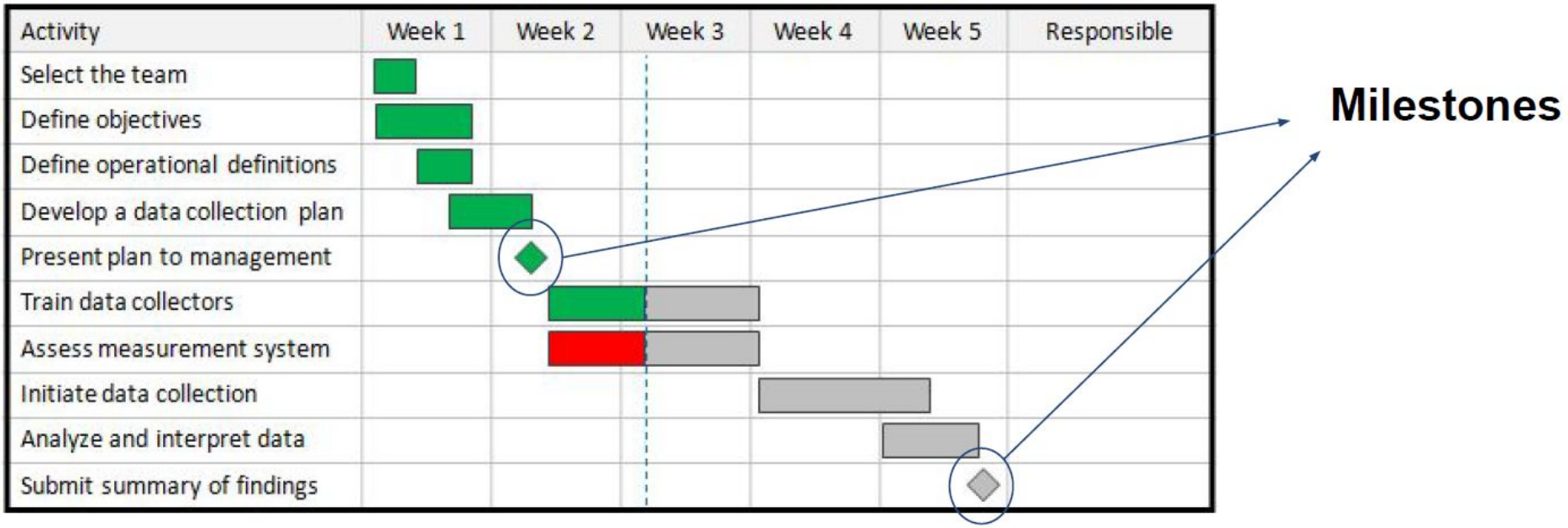


Gantt charts limitations

- Lack of detail on task precedence
  - But some tools do support precedence mechanisms such as those explained before
- No support for helping the project manager
- Defining the shortest possible completion schedule
- Allocating resources effectively

Project Milestones

A milestone is a logical end to a stage in the project for progress reviewing.
It is **documented** (report, email, etc), **summarizes work done** and is **associated with one task or groups of tasks**.



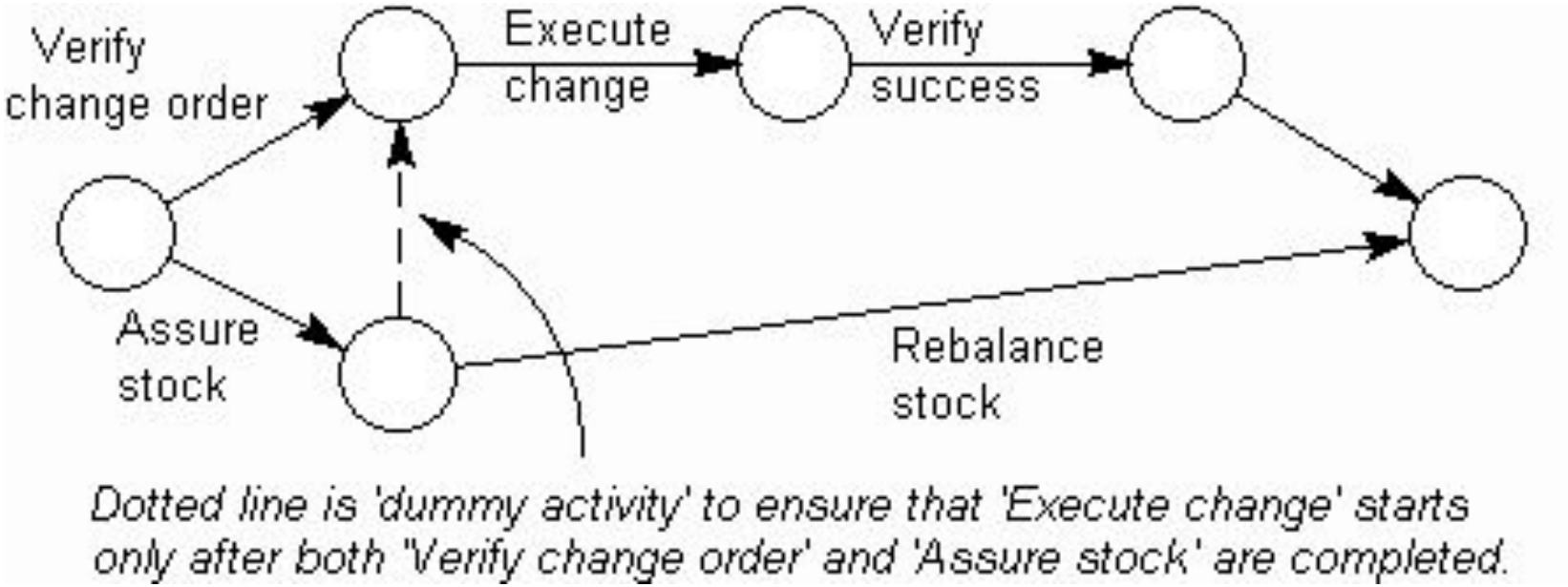
Activity on Arrow (AOA) diagrams

(The most common of these is the Pert diagram)

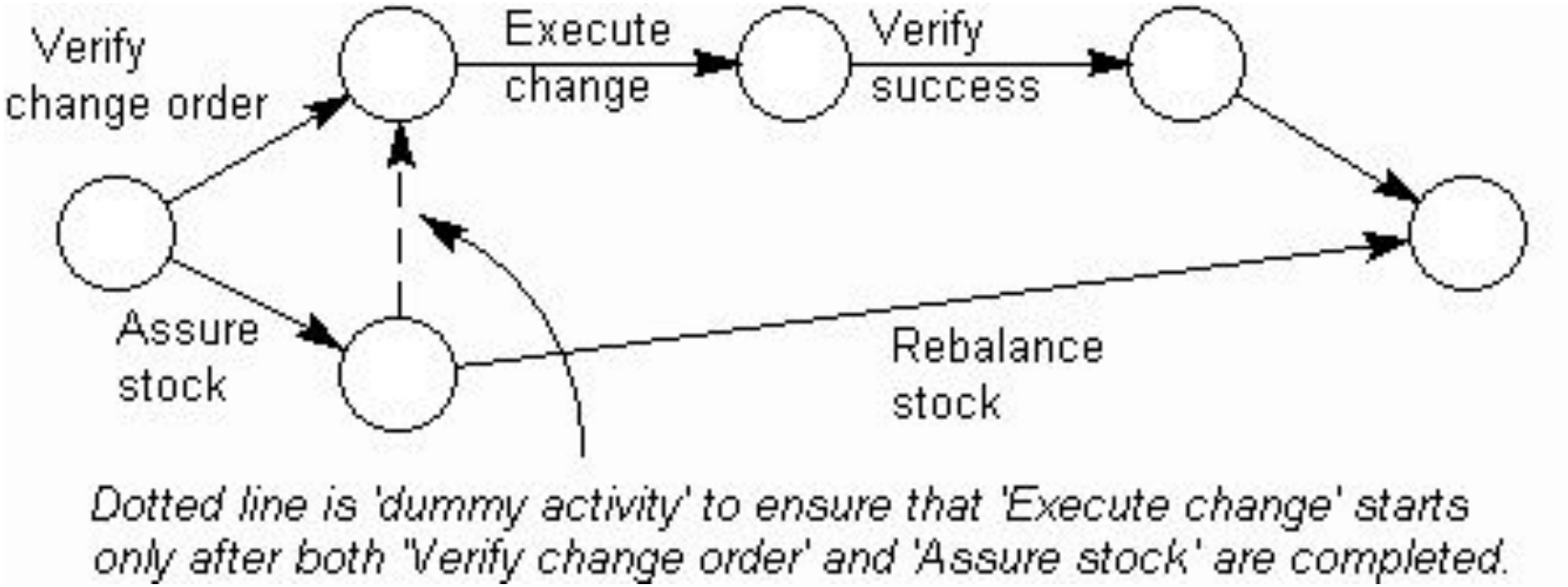
Activity on Arrow diagrams

- Detailed project planning & control
- Support for project schedule analytics
  - Identification of the first possible moment for completing an activity
  - Support for earliest completion date computation
  - Comparison of alternative detailed scheduling
  - Project scheduling control

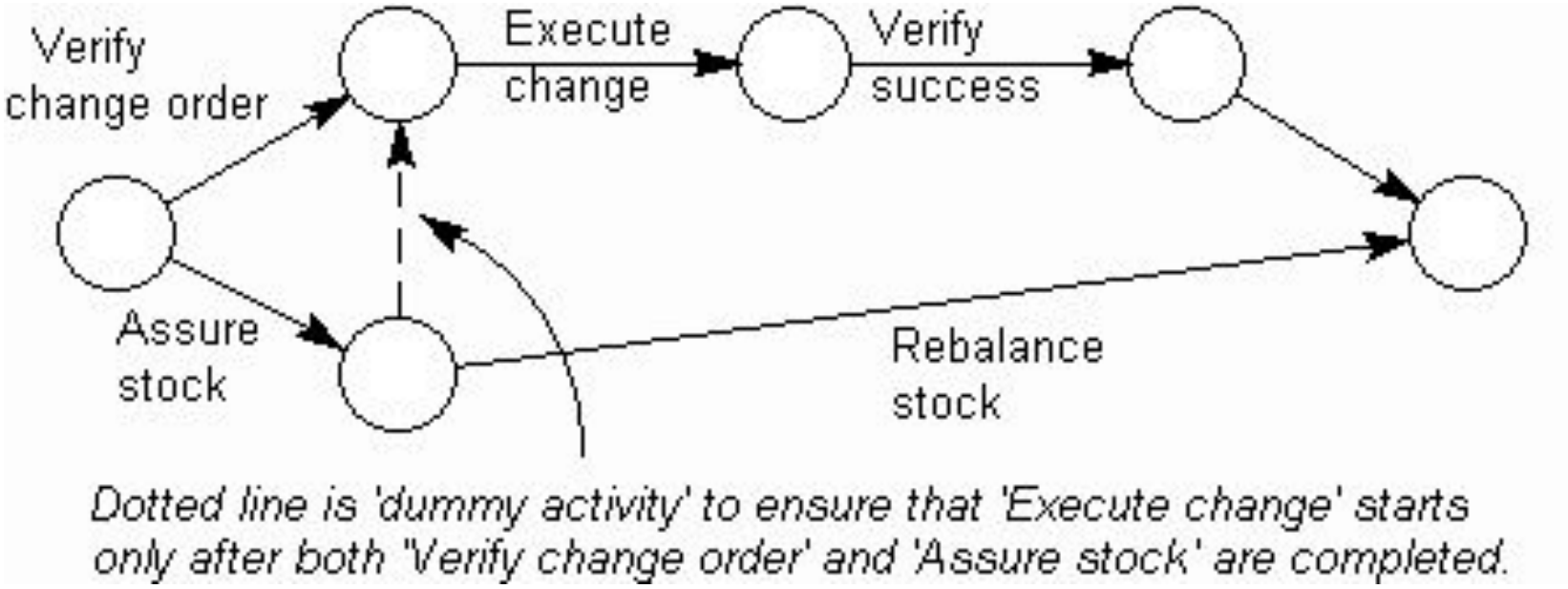
Arrows represent activities execution



Nodes represent the start (or end) of activities

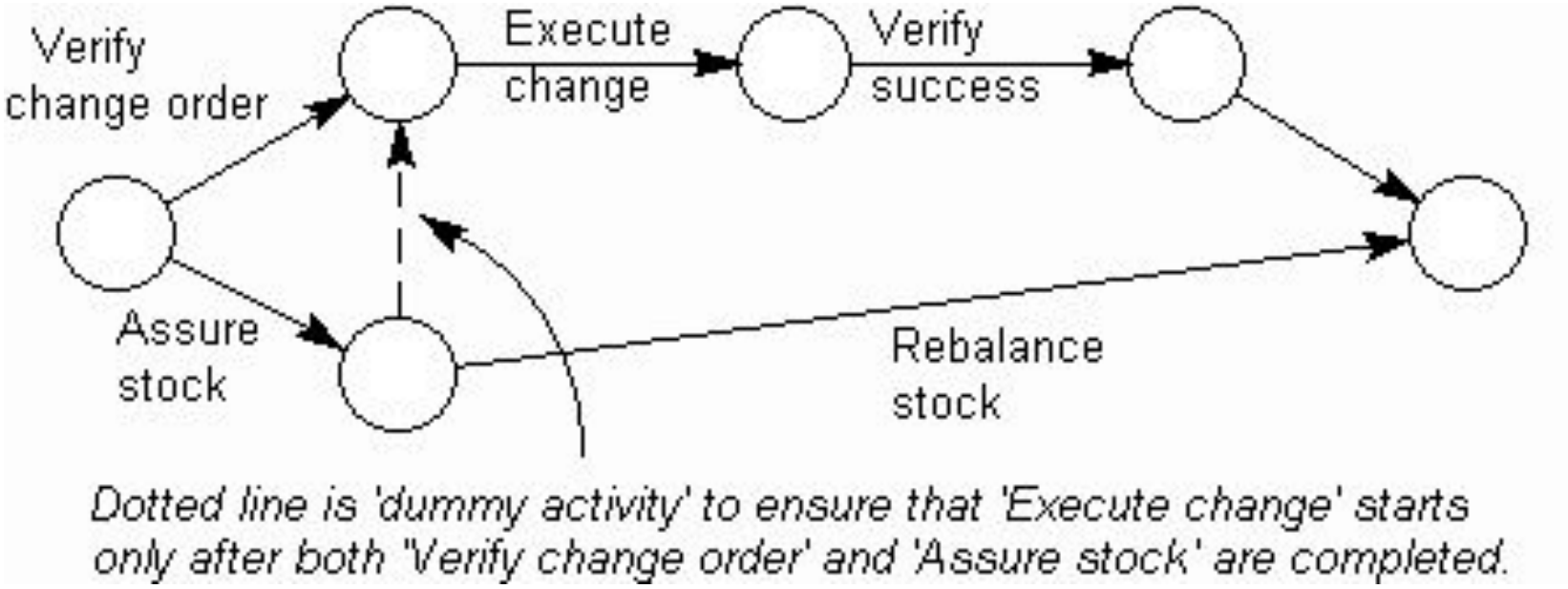


Dashed arrows represent fictitious ("dummy") activities with null execution time



used for specifying pre-requisite relationships, in order to preserve network integrity

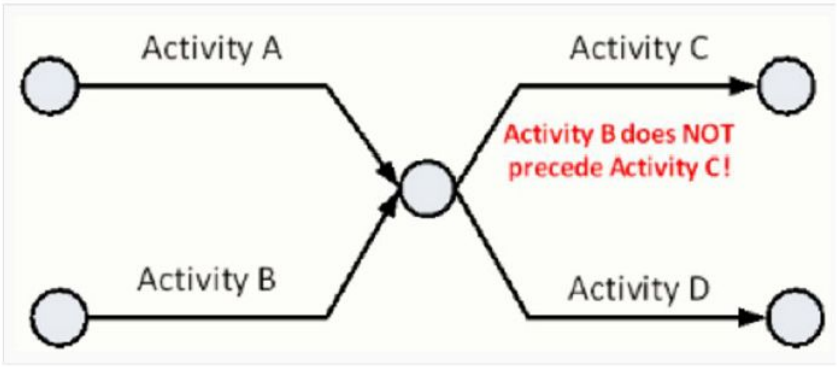
The node event “occurs” after all the inbound activities finish

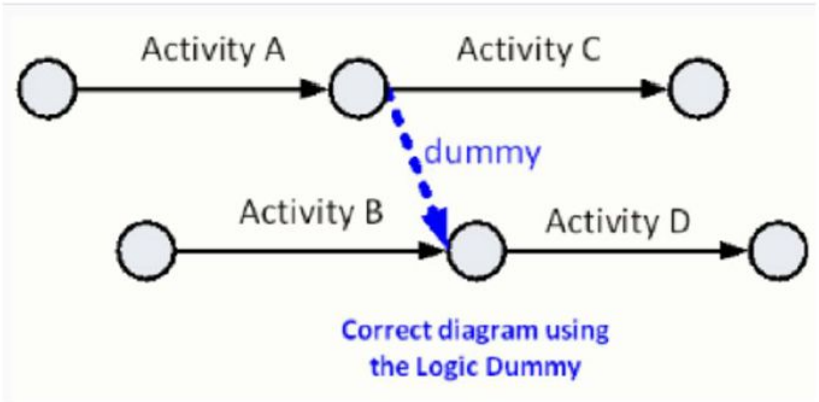


Dotted line is 'dummy activity' to ensure that 'Execute change' starts only after both "Verify change order' and 'Assure stock' are completed.

Dummy activities revisited

Activity	Prior Activity
A	None
B	None
C	A
D	A, B





Representing timing in AOA diagrams

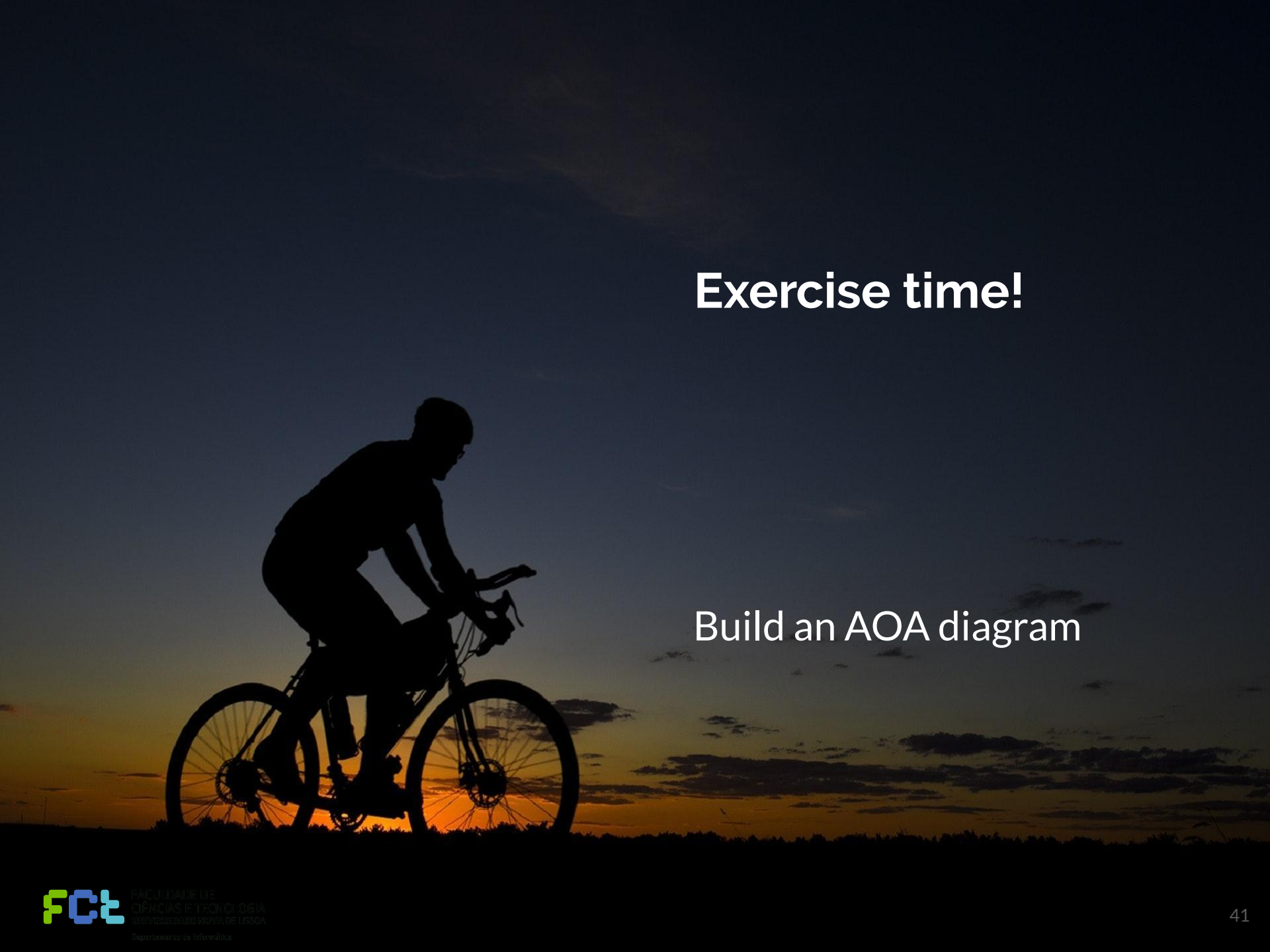
A – Activity
T – Time to complete activity (= EFT-EST = LFT–LST)
EST – Earliest Start Time
EFT – Earliest Finish Time
LST – Latest Start Time
LFT – Latest Finish Time

Algorithm for building AOA diagrams

1. Identify and list all activities
2. Assign each activity a unique id
3. Identify and list the dependencies among activities
4. Design a preliminary network
5. Estimate activities durations
6. Add activities durations to the network
7. Compute early start times
8. Compute late start times
9. Fine tune the network
10. Assign resources

What is the Critical Path of a project?

- The longest network path formed by activities where EST = LST
- This is called the **critical path**
  - Any deviation on the duration of activities in the **critical path** will have a direct impact on the whole network, i.e., on the whole project schedule)


Exercise time!
Build an AOA diagram

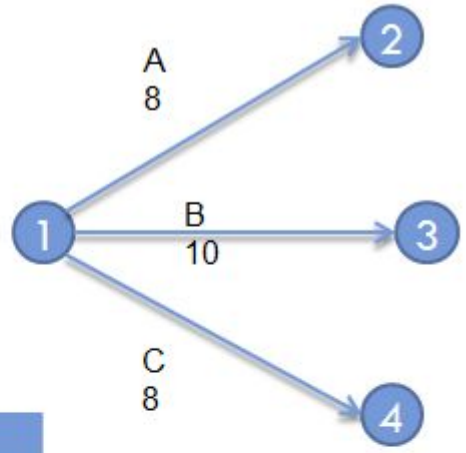
Challenge

Starting from a list of **activities**, their **precedences** and **durations**, build an AOA diagram

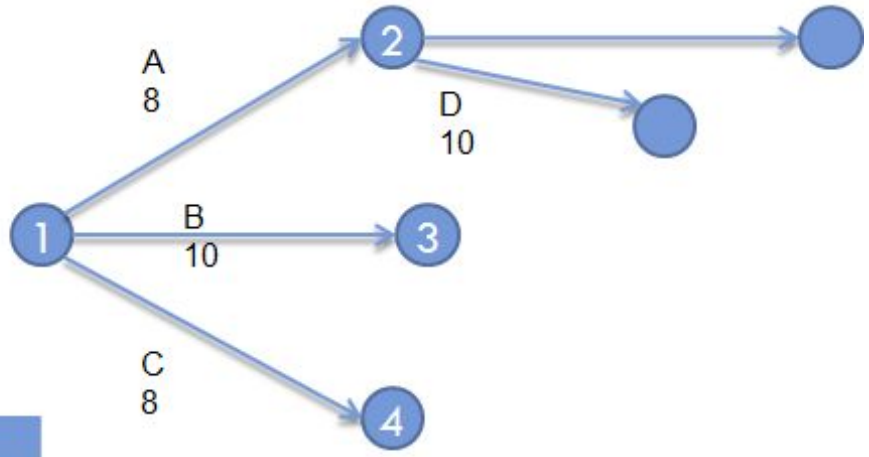
Activity	Predecessors	Duration (days)
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	D, B	17
G	C	18
H	C	14
I	F,G	9

1

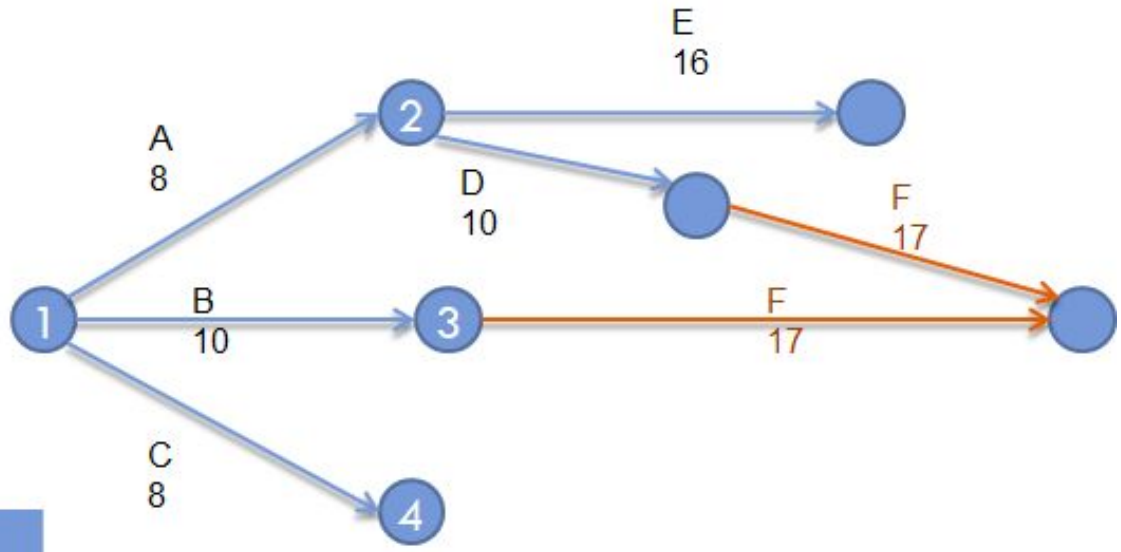
Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	D,B	17
G	C	18
H	C	14
I	F,G	9



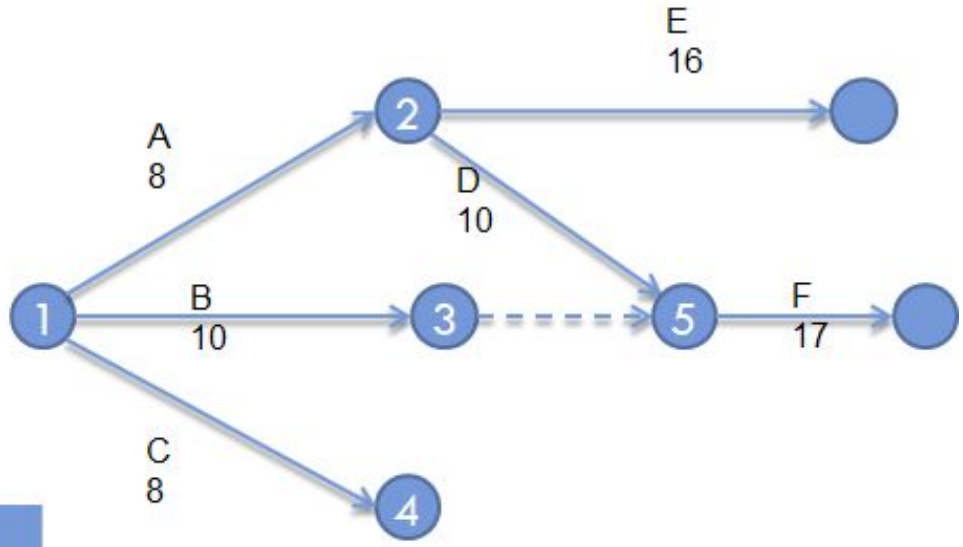
Act	Pred	Dur
A	**-**	**8**
B	**-**	**10**
C	**-**	**8**
D	A	10
E	A	16
F	D,B	17
G	C	18
H	C	14
I	F,G	9



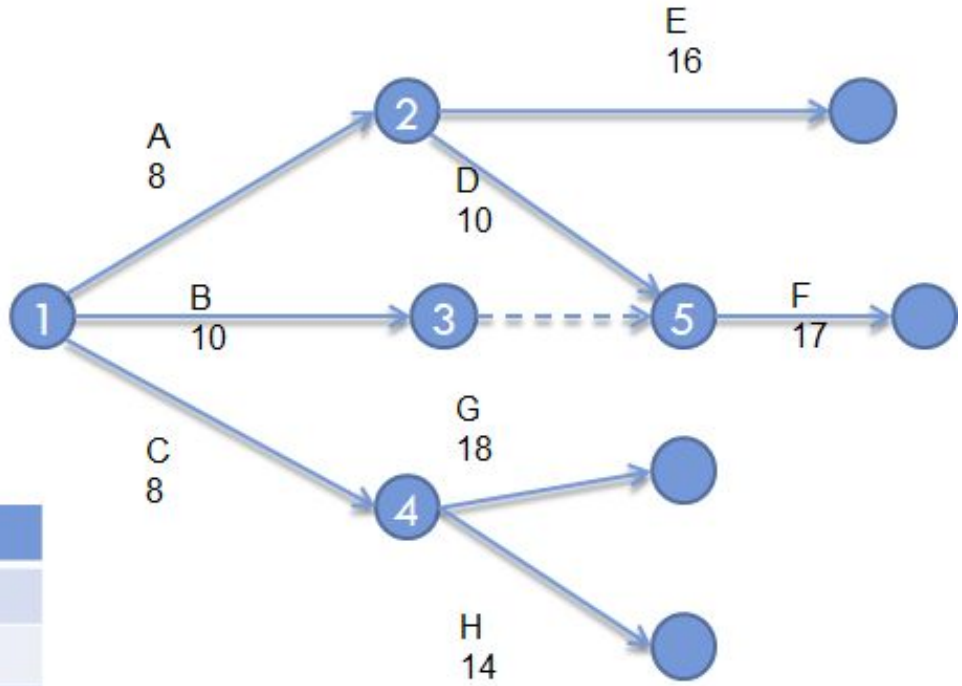
Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	**A**	**10**
E	**A**	**16**
F	D,B	17
G	C	18
H	C	14
I	F,G	9



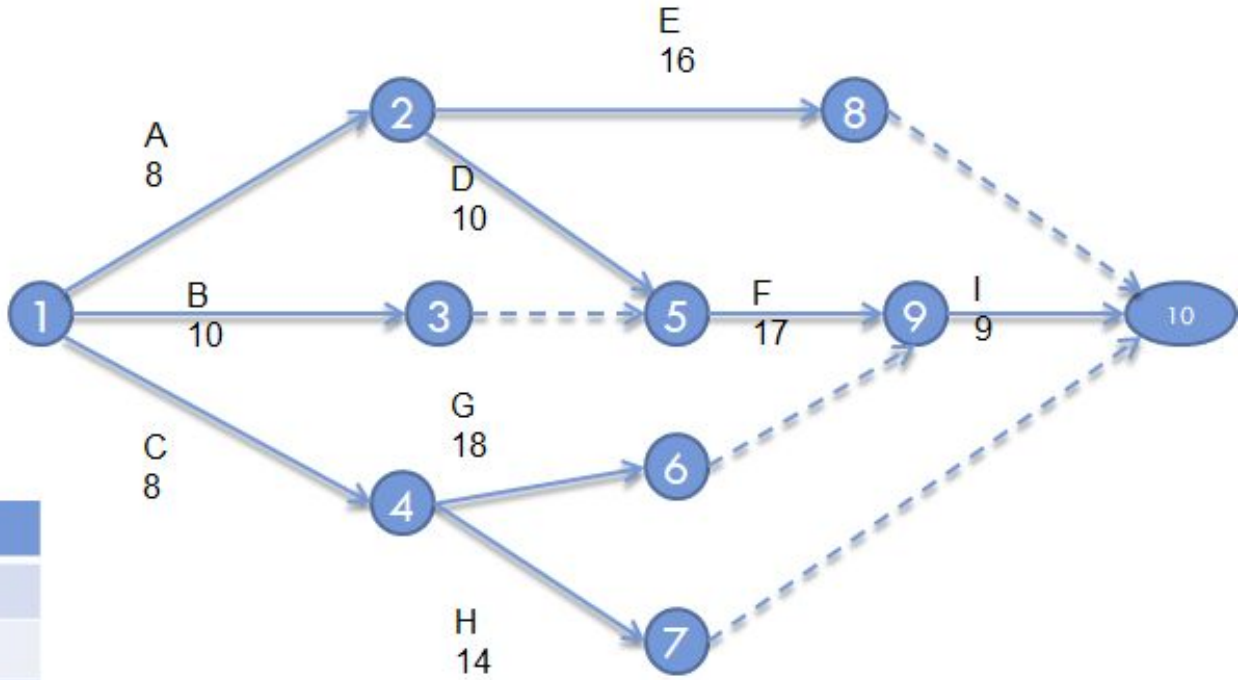
Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	**D,B**	**17**
G	C	18
H	C	14
I	F,G	9



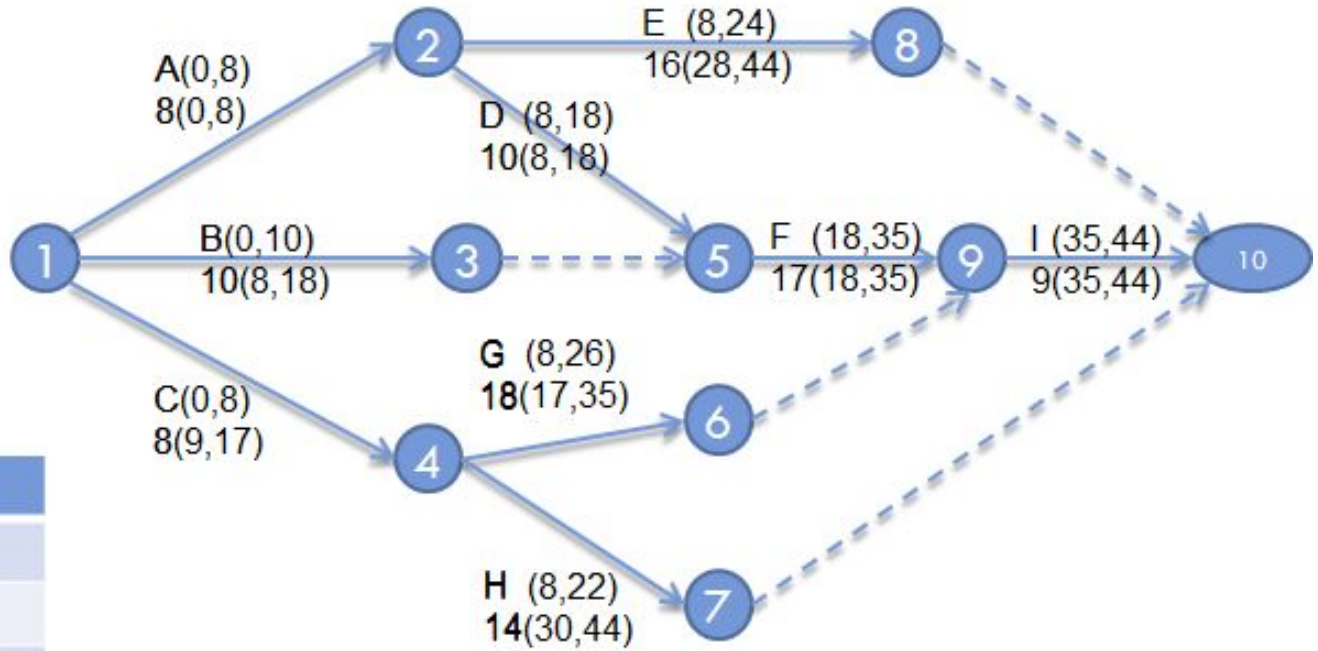
Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	**D,B**	**17**
G	C	18
H	C	14
I	F,G	9



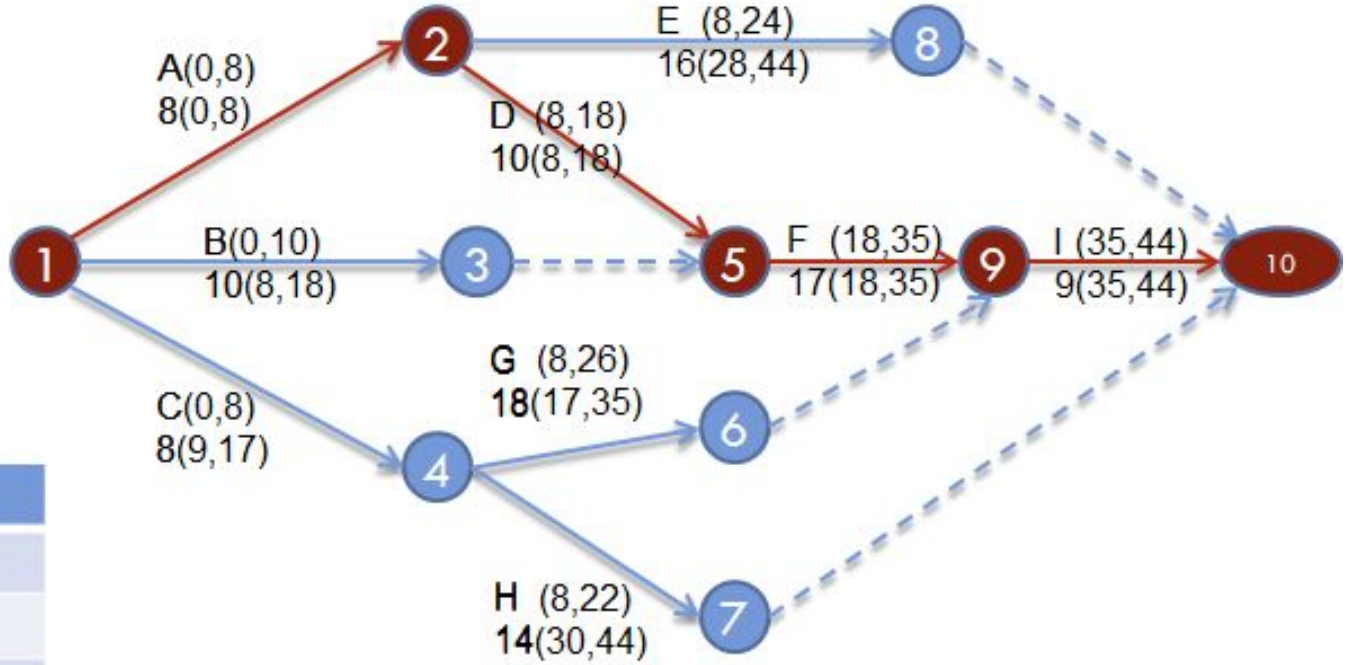
Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	D,B	17
G	**C**	**18**
H	**C**	**14**
I	F,G	9



Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	D,B	17
G	C	18
H	C	14
I	**F,G**	**9**



Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	D,B	17
G	C	18
H	C	14
I	F,G	9



Act	Pred	Dur
A	-	8
B	-	10
C	-	8
D	A	10
E	A	16
F	D,B	17
G	C	18
H	C	14
I	F,G	9

A-E -> 8 + 16 = 24

A-D-F-I -> 8 + 10 +17 + 9 = 44

B-F-I -> 10 + 17 + 9 = 36

C-G-I -> 8 + 18 + 9 = 33

C-H-> 8 + 14 = 22

Limitations of AOA Diagrams

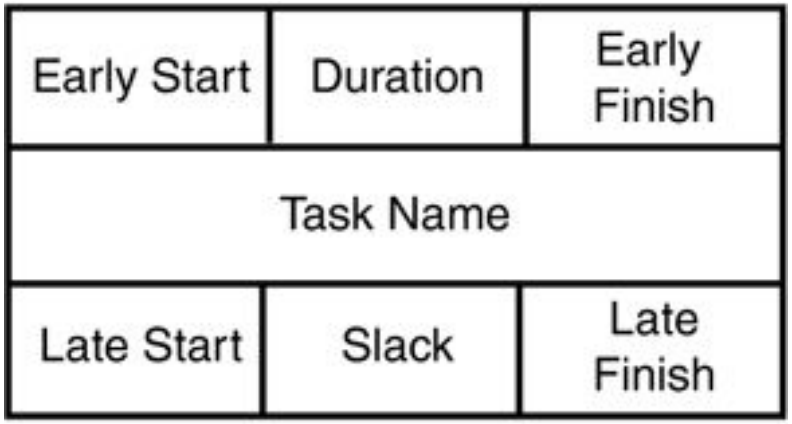
- Need to create “dummy” activities for preserving network integrity
- Only simple dependences can be represented
- Accidental complexity makes this technique hard to apply when projects include many activities

Activity On Node Diagrams

The Precedence Diagram Method

Activity On Node Diagrams

- Each activity is represented in an **activity node**



Duration = Early Finish – Early Start = Late Start – Late Finish
Slack = Late Start – Early Start = Late Finish – Early Finish

- Dependency relationships represented by **arrows**


Exercise time!
Build an AON diagram

Creating an AON

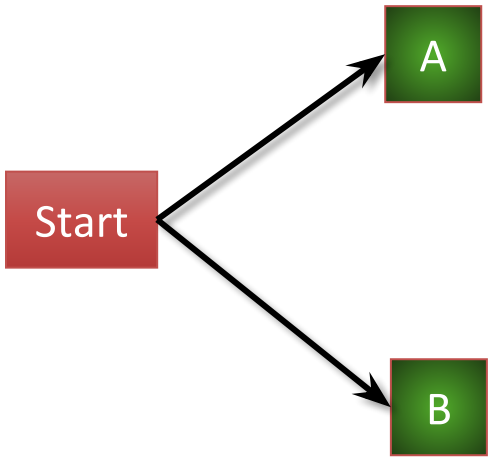
Actividade	Predecessores
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J

Begin by creating a Start node

Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J

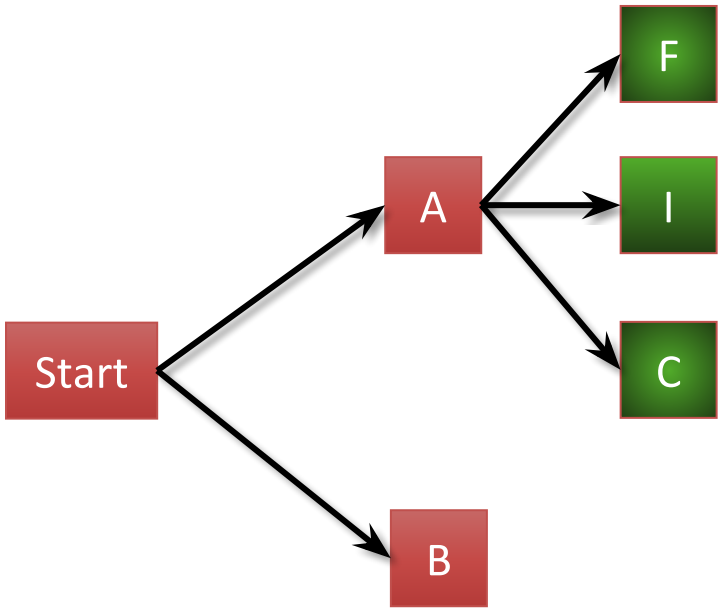
A and B can start from the beginning of the project

Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J



When B finishes, D and E can Start

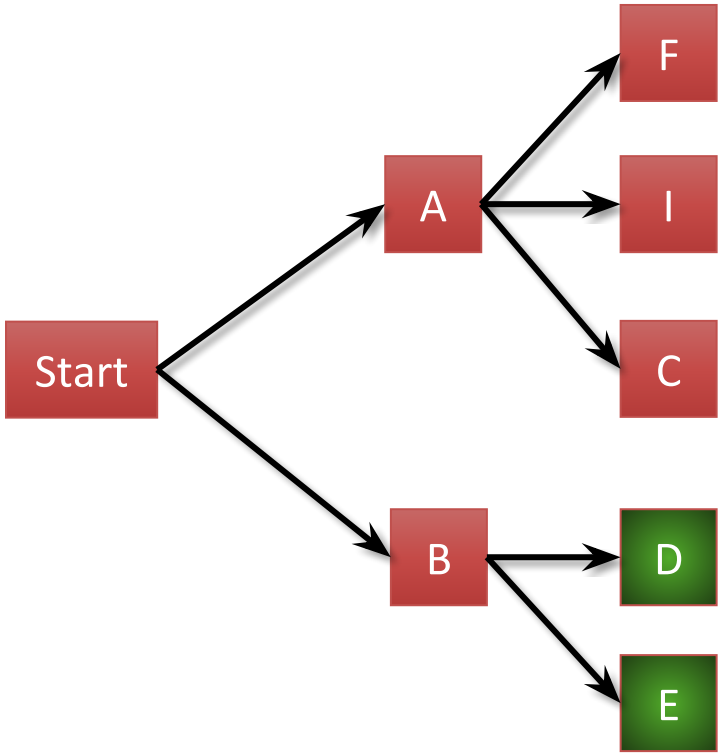
Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J



Note: A is a ***burst*** activity: more than one activity may start once A finishes.

When B finishes, D and E can Start

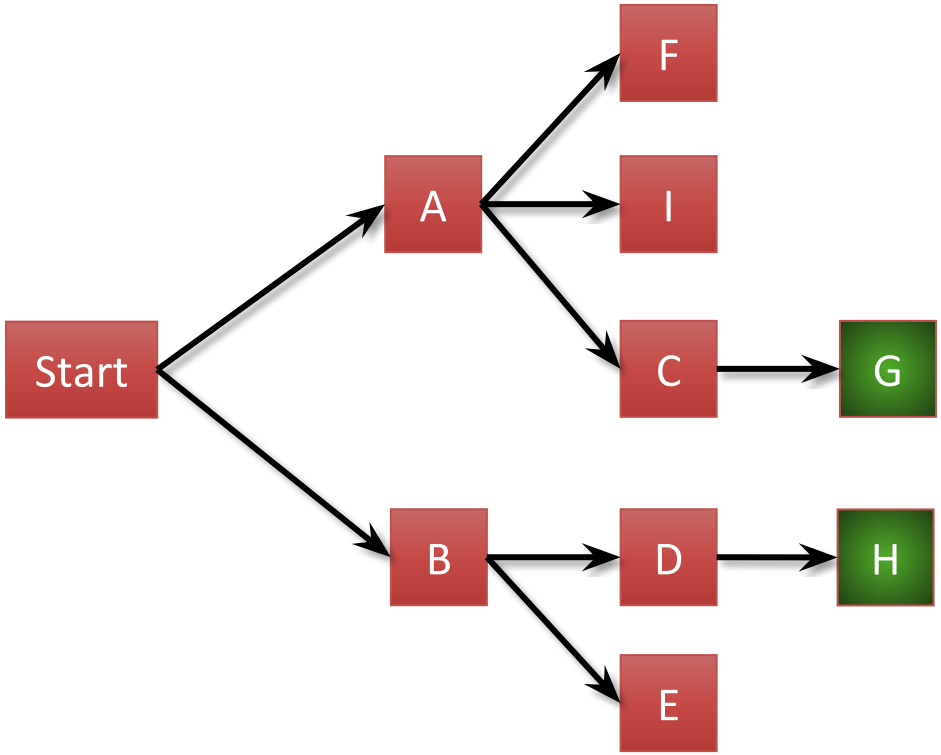
Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J



Note: B is a ***burst*** activity: more than one activity may start once B finishes.

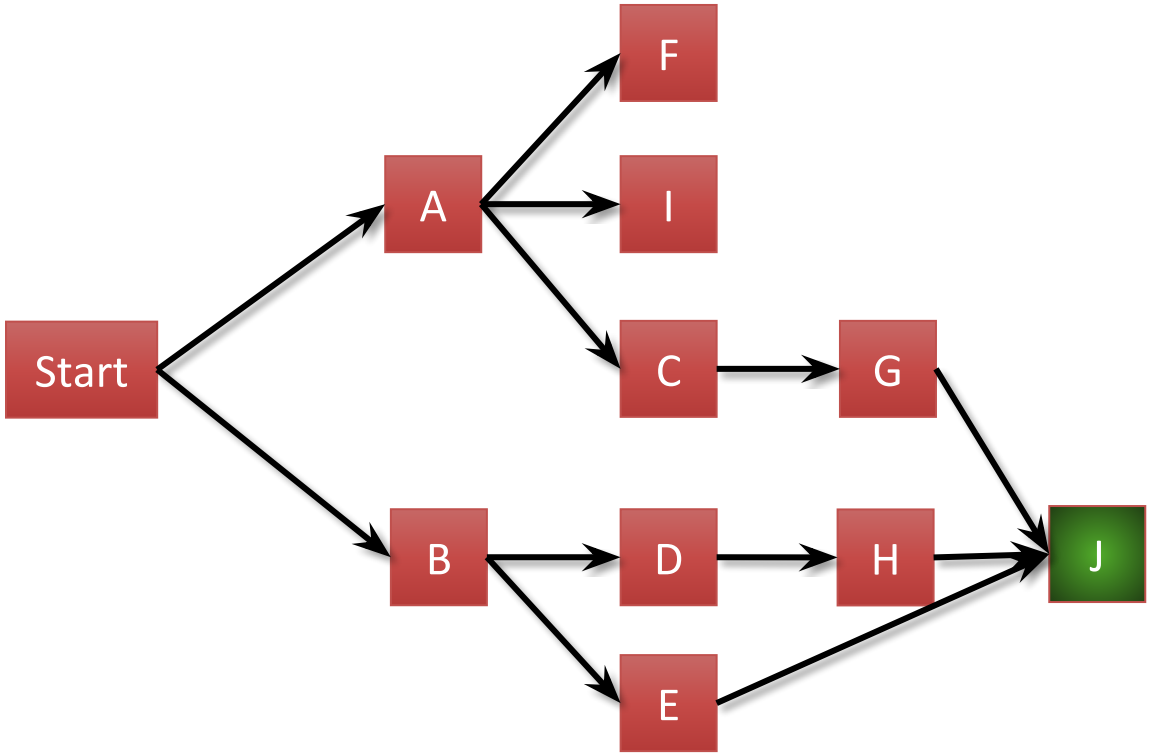
When C finishes, G can start
When D finishes, H can start

Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J



When E, G and H finish, J can start

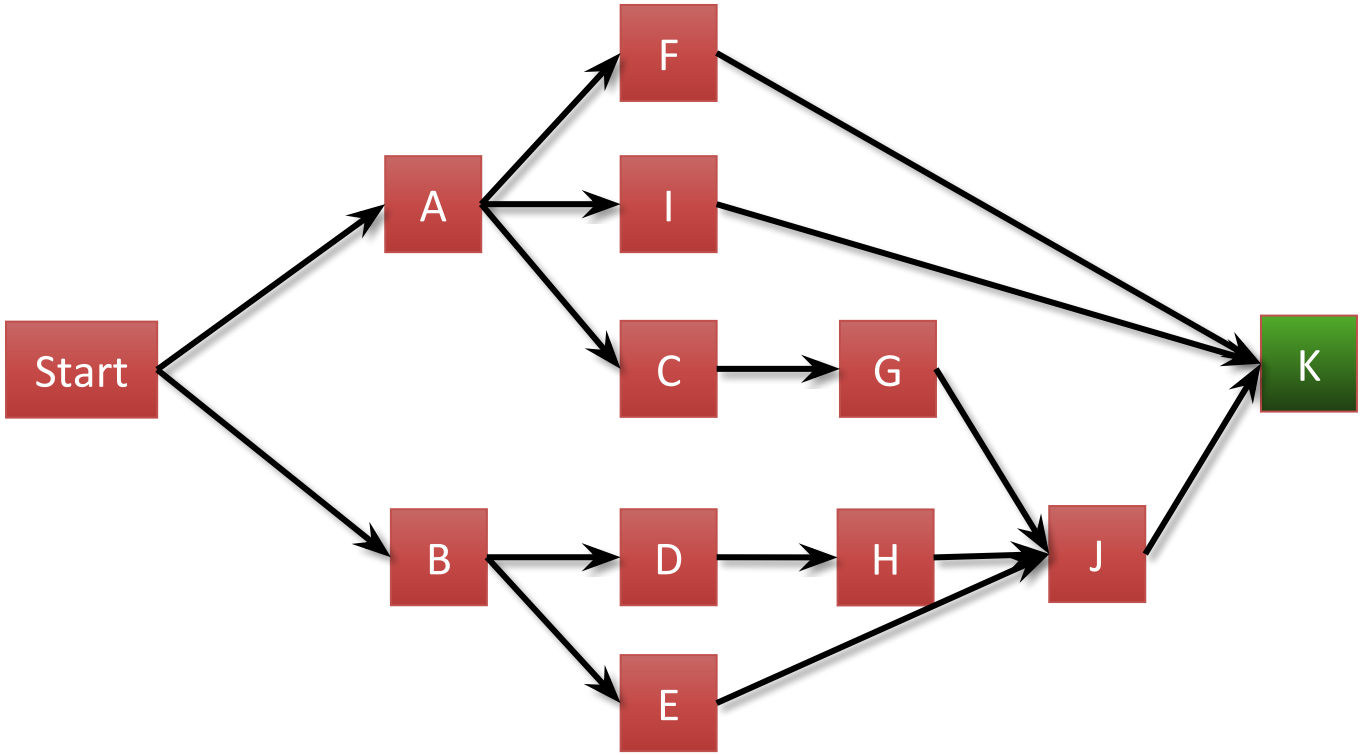
Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J



Note: J is a **fusion** activity: J cannot start before all its precedent activities finish

When F, I and J finish, K can start

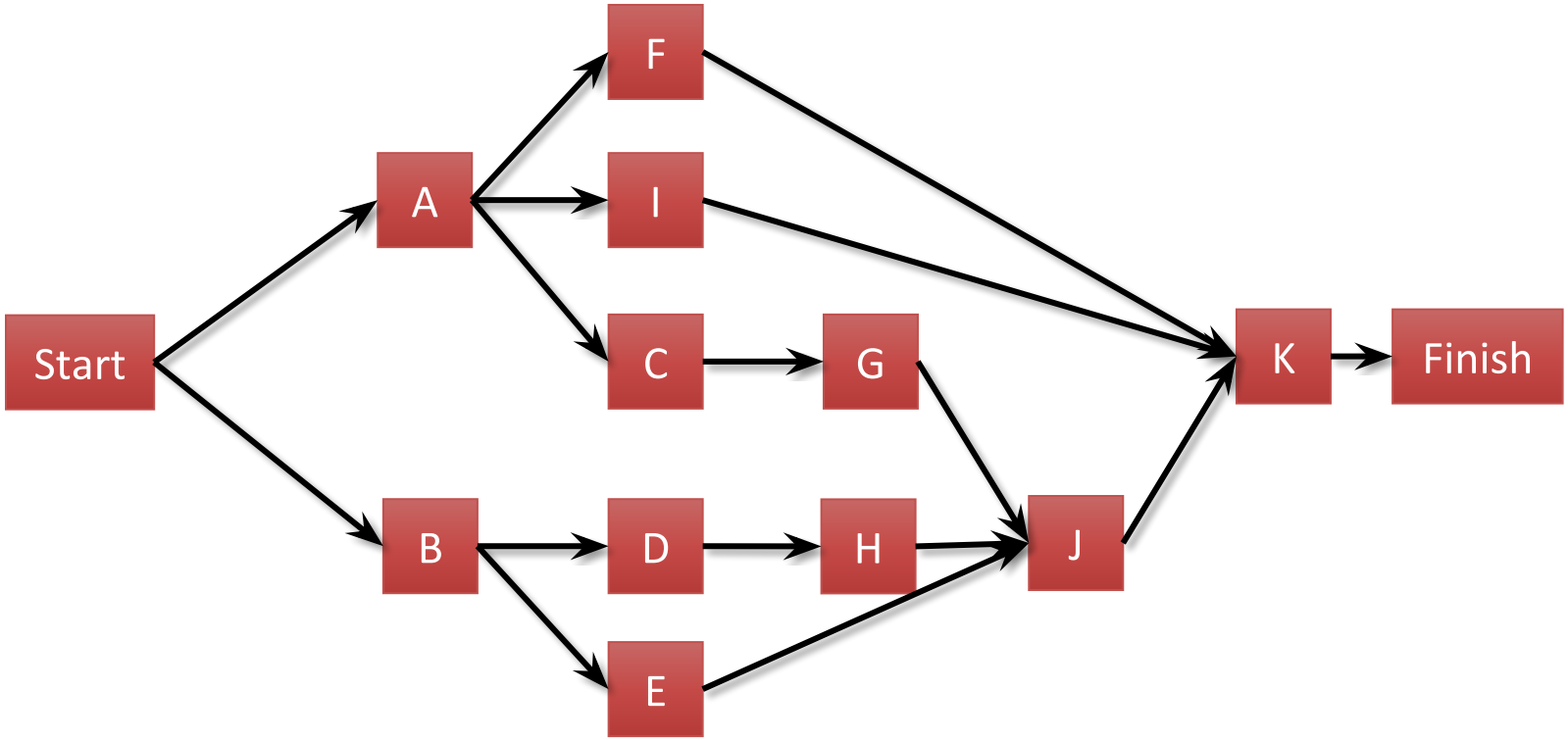
Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J



Note: K is a **fusion** activity: K cannot start before all its precedent activities finish

K was the last activity, end the diagram

Act	Pred
A	-
B	-
C	A
D	B
E	B
F	A
G	C
H	D
I	A
J	E, G, H
K	F, I, J



How long will a project take?

Consider this scenario

- A company decides to reengineer its IT system
- They will need new hardware, network and internet access, along with the corresponding software
- The Project Manager has already made a description of the activities to conduct and the time required for each of them

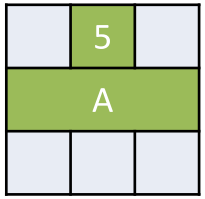
How much time do we need for this project?

How long will it take, considering these estimated durations per activity?

Activity	Predecessors	Duration
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

Activity A has no precedents and takes 5 days

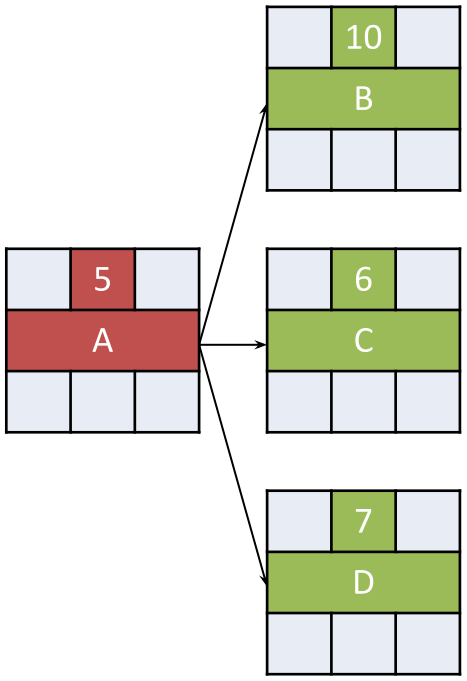
Act	Pred	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

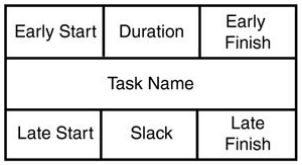


Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

B, C, and D start after A finishes

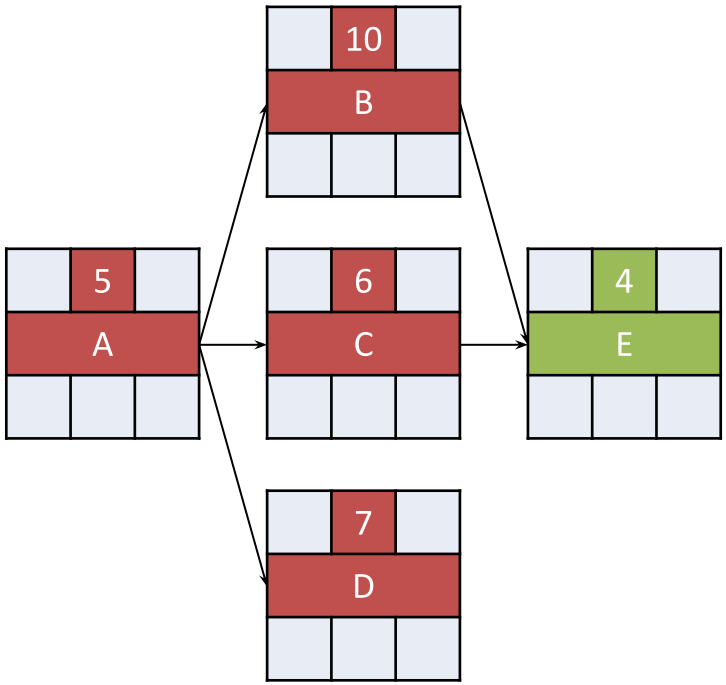
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

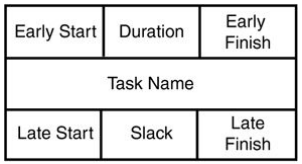




E starts when both B and C finish

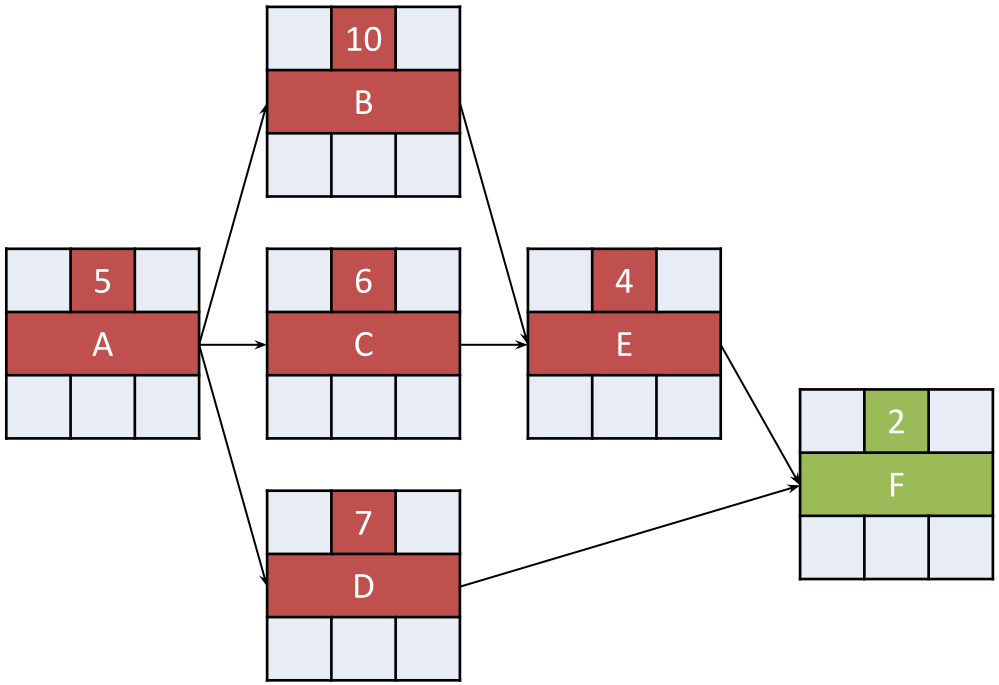
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

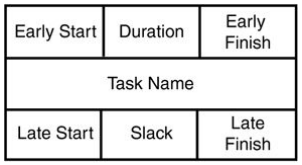




F starts when D and E finish

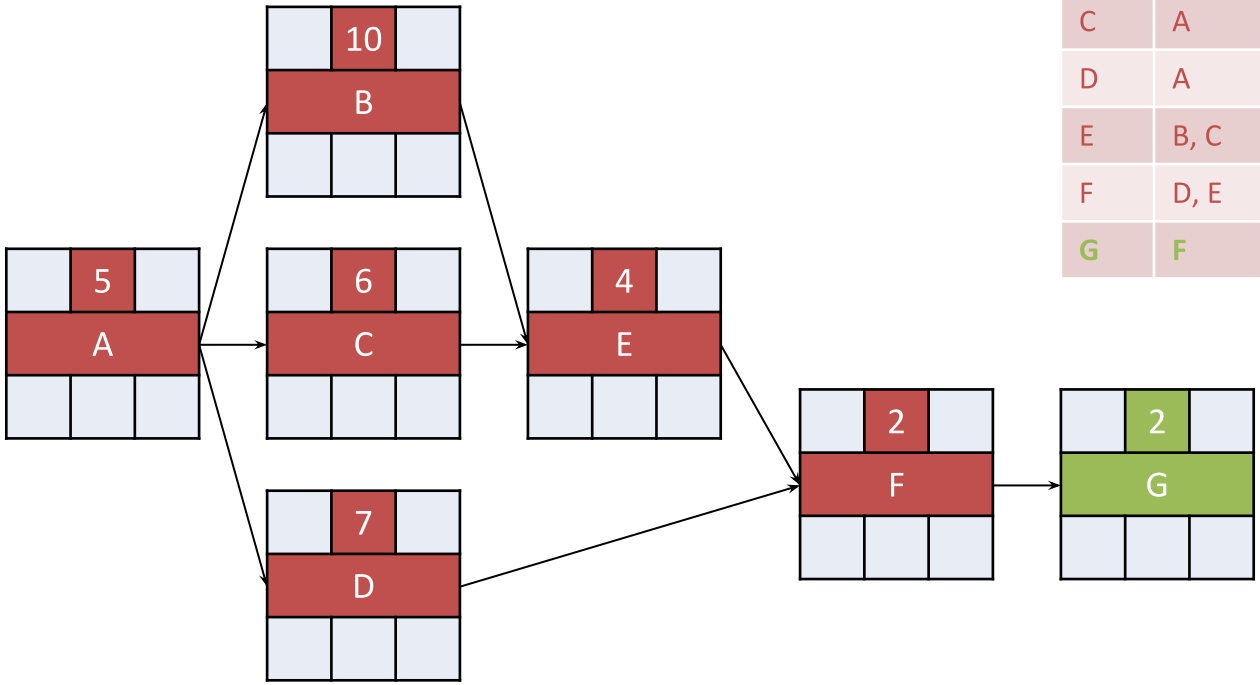
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

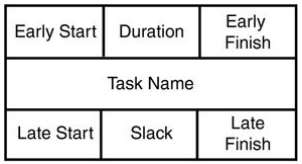




G starts when F finishes

Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

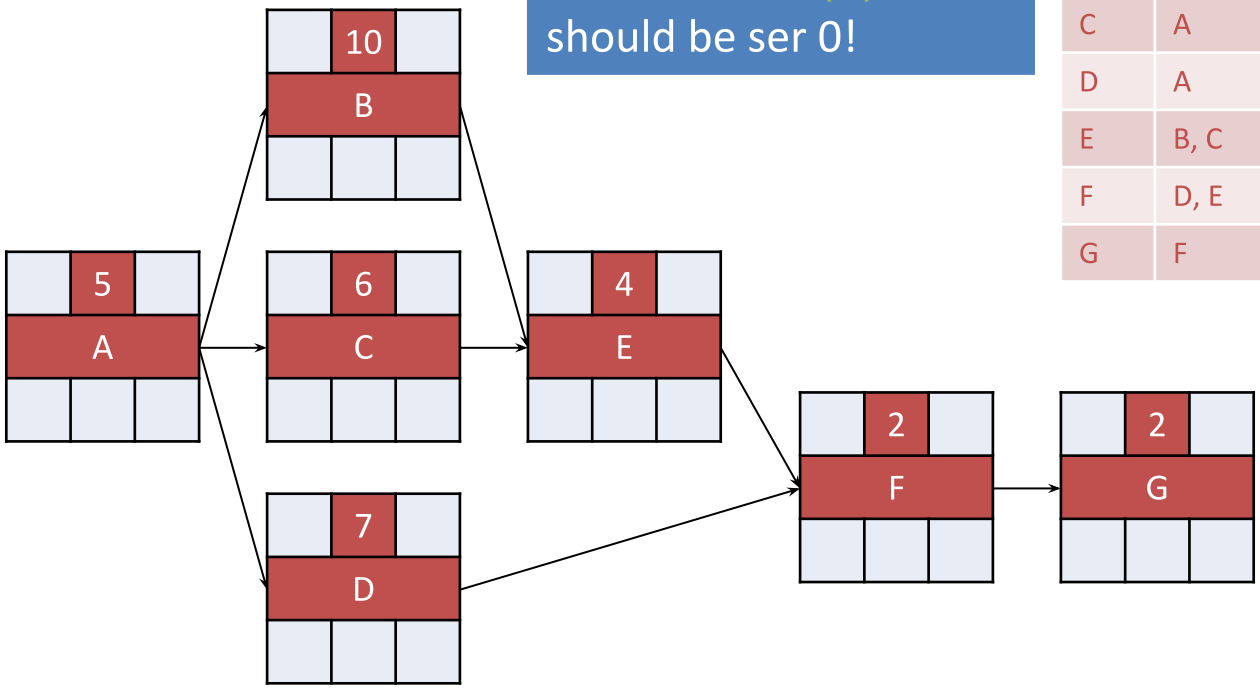




EST(A) = ?
EFT(A) = ?

Note: A is the first activity
Therefore EST(A) should be ser 0!

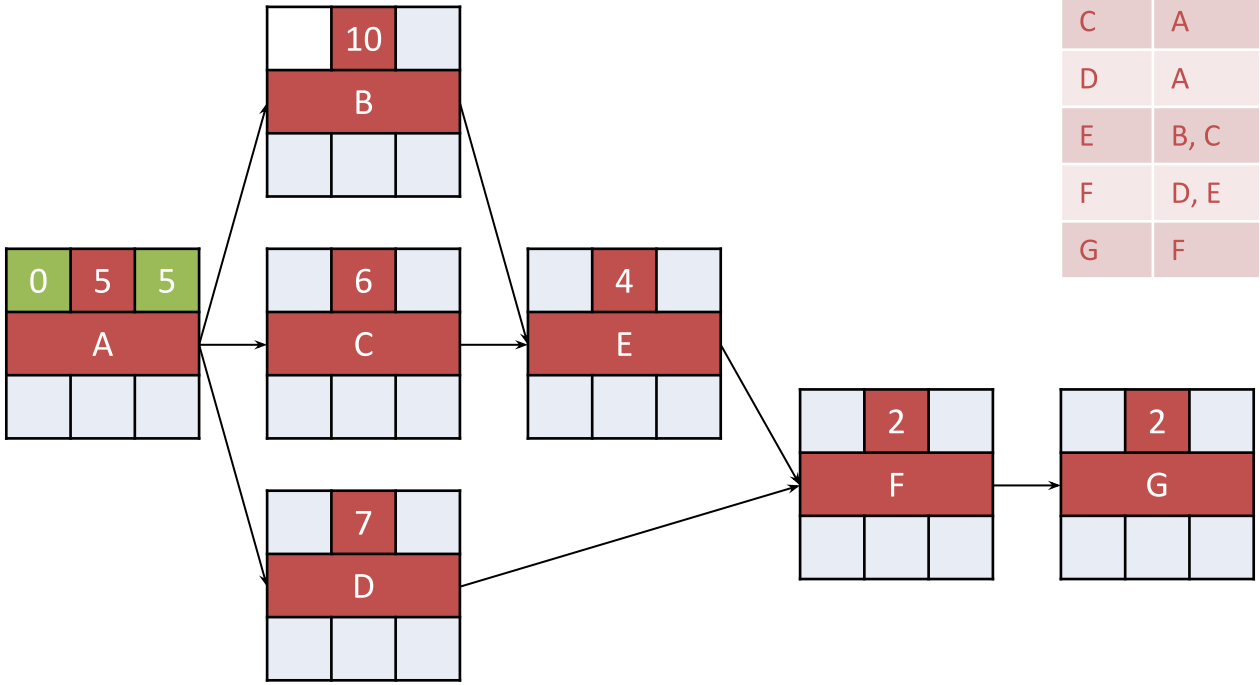
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

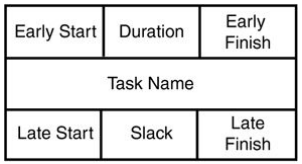


Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

EST(A) ← 0
EFT(A) ← EST(A) + DUR(A)

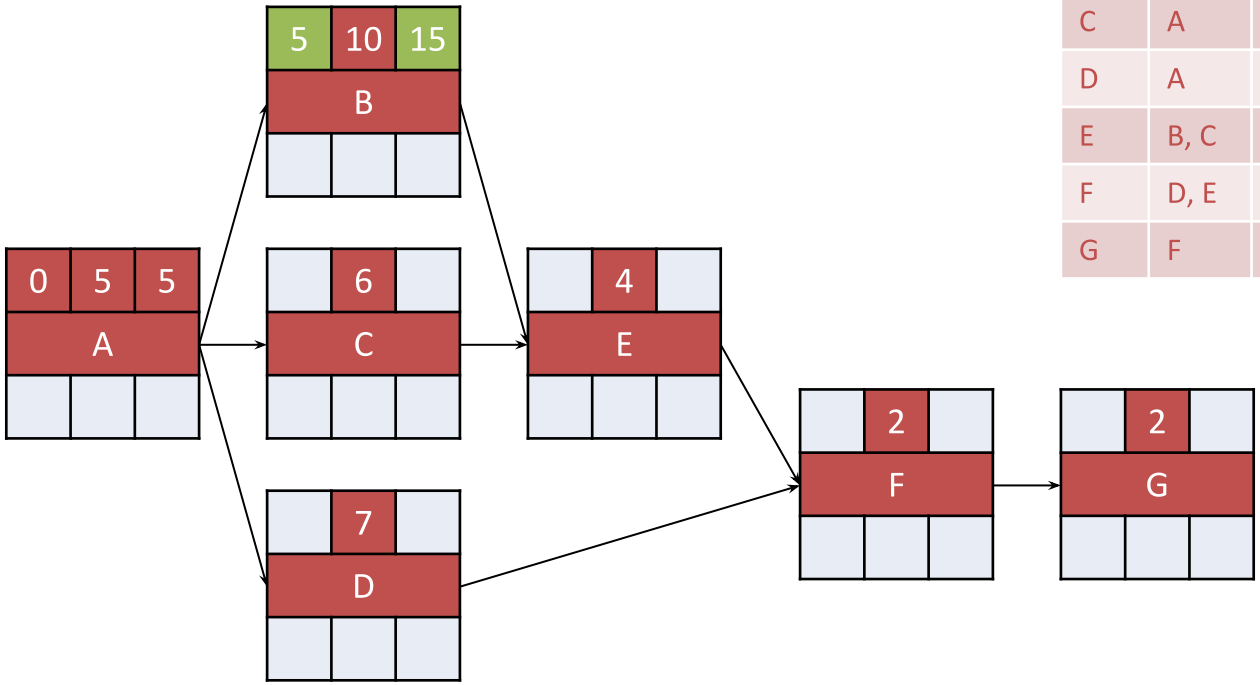
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

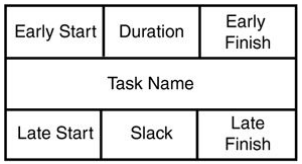




EST(B) ← EFT(A)
EFT(B) ← EST(B) + DUR (B)

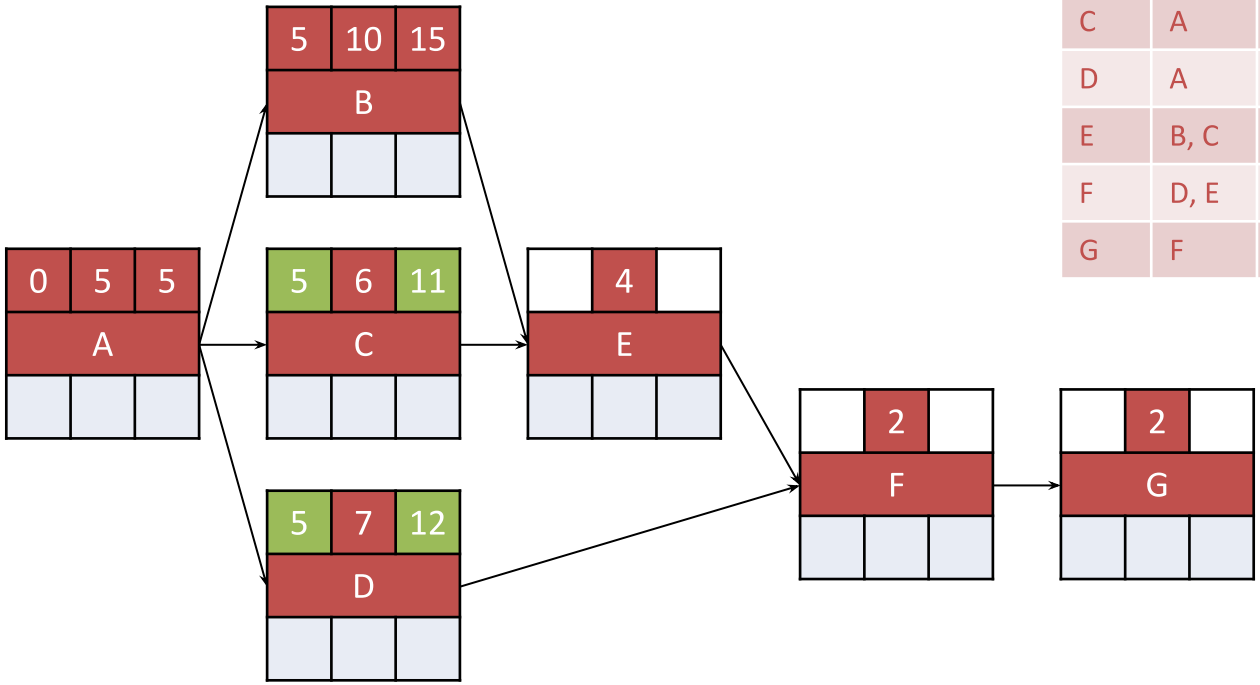
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

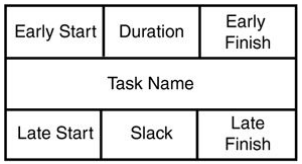




EST(C) ← EFT(A)
EFT(C) ← EST(C) + DUR(C)

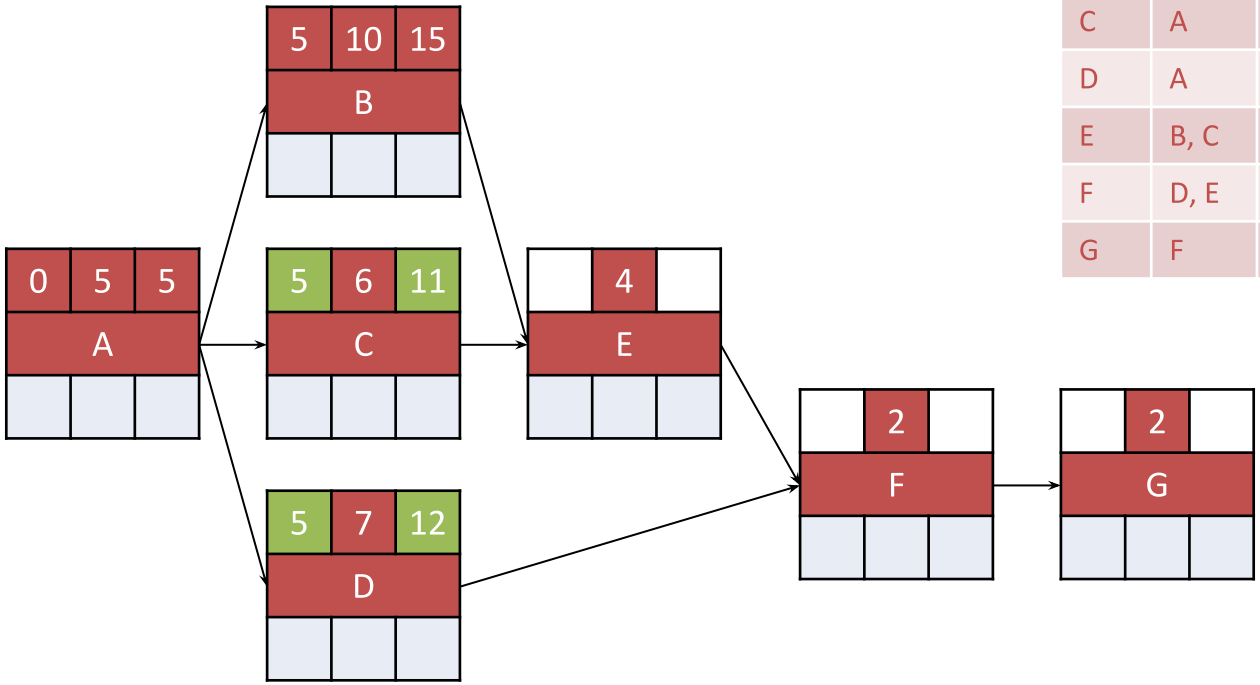
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2





EST(D) ← EFT(A)
EFT(D) ← EST(D) + DUR(D)

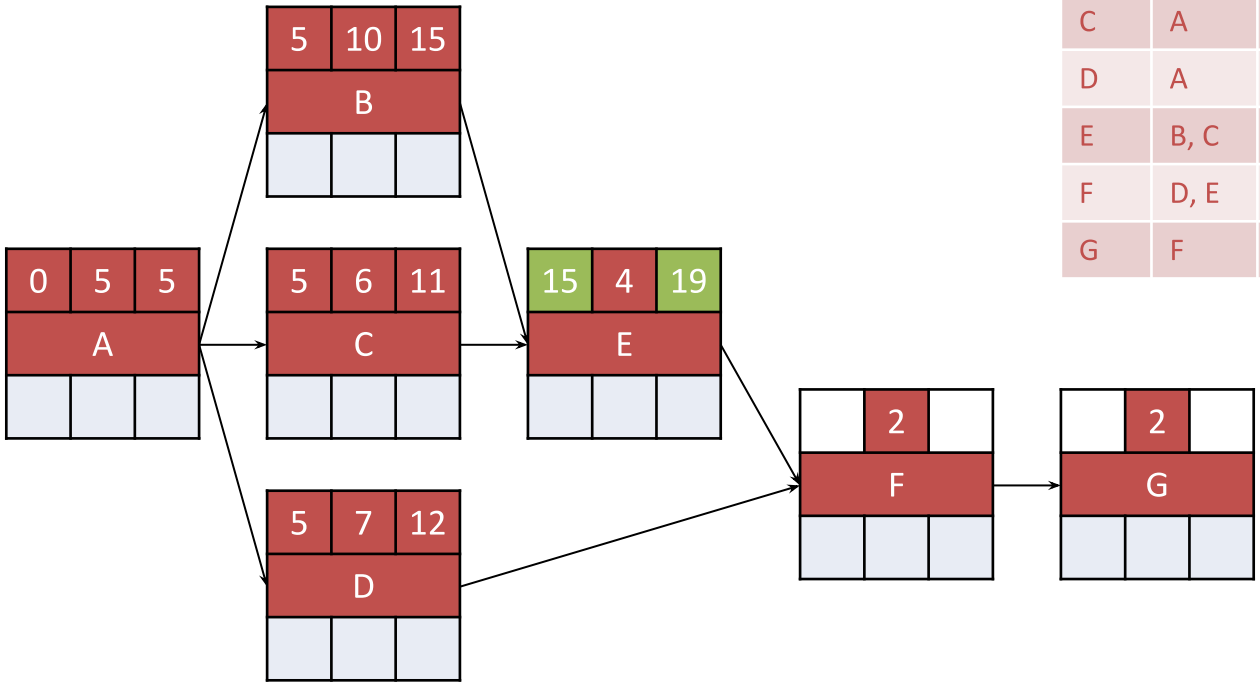
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2



Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

EST(E) ← MAX (EFT(B), EFT(C))
EFT(E) ← EST(E) + DUR(E)

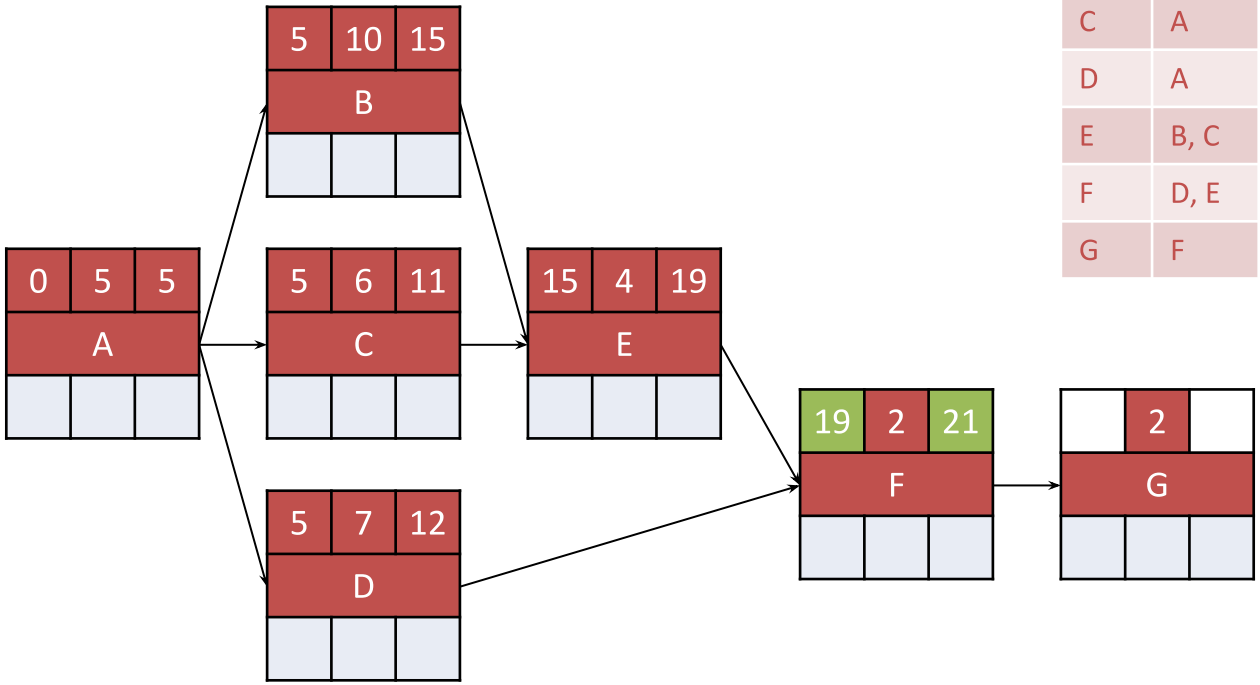
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2



Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

EST(F) ← MAX (EFT(D), EFT(E))
EFT(F) ← EST(F) + DUR(F)

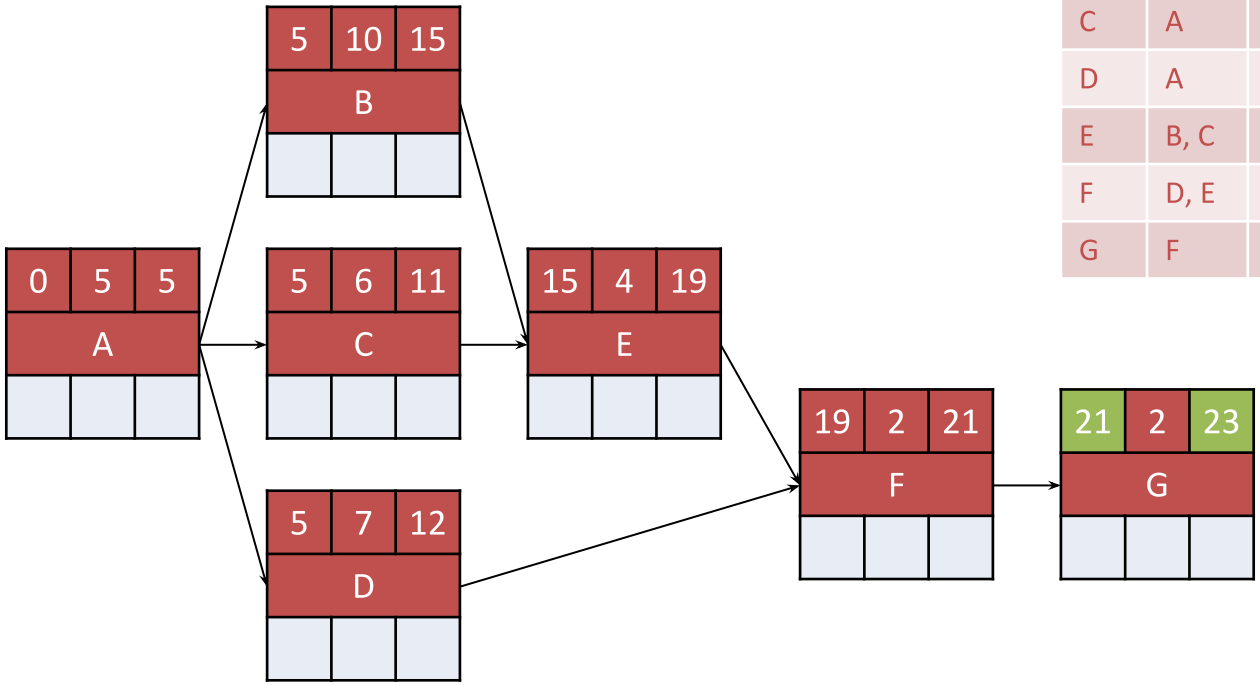
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

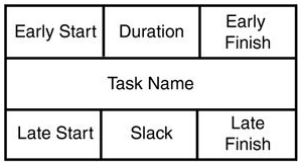


Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

EST(G) ← EFT(F)
EFT(G) ← EST(G) + DUR (G)

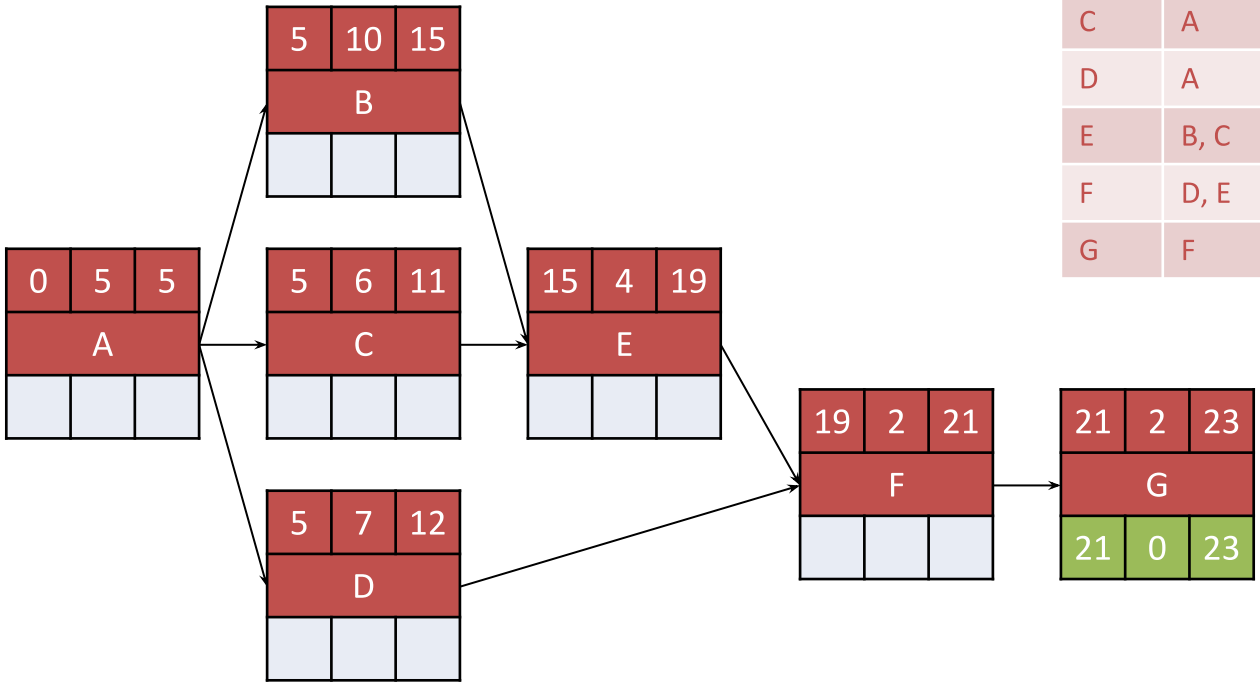
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

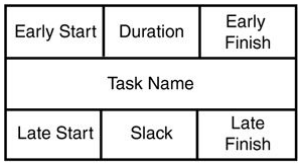




LFT(G) ← EFT(G)
LST(G) ← LFT(G) – DUR(G)
SLACK(G) ← LST(G) – EST(G)

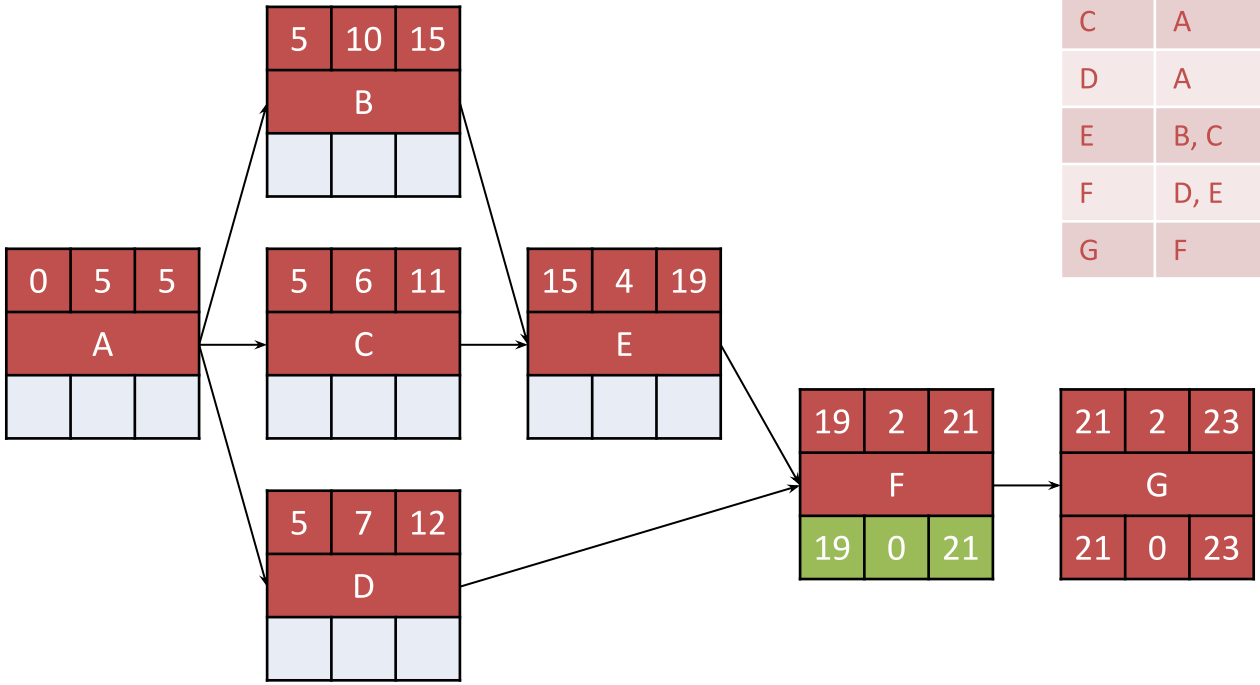
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

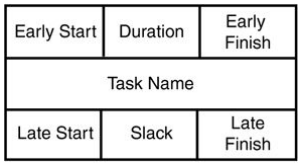




LFT(F) ← LST(G)
LST(F) ← LFT(F) – DUR(F)
SLACK(F) ← LST(F) – EST(F)

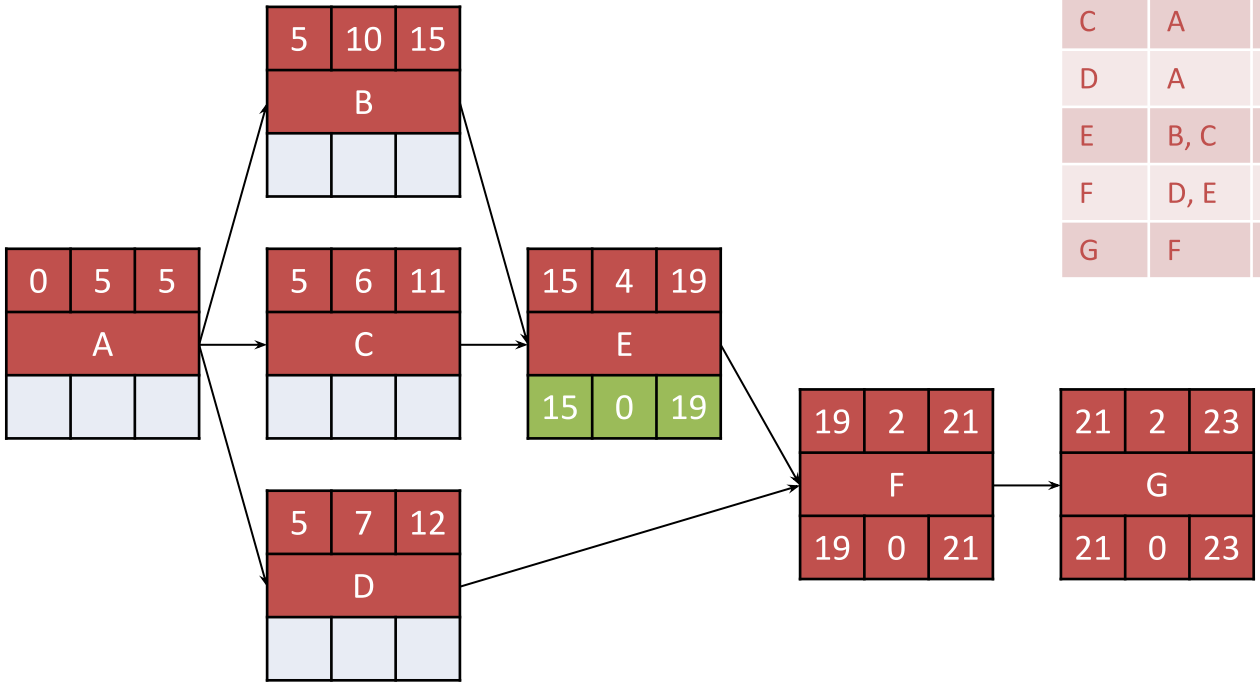
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2





LFT(E) ← LST(F)
LST(E) ← LFT(E) – DUR(E)
SLACK(E) ← LST(E) – EST(E)

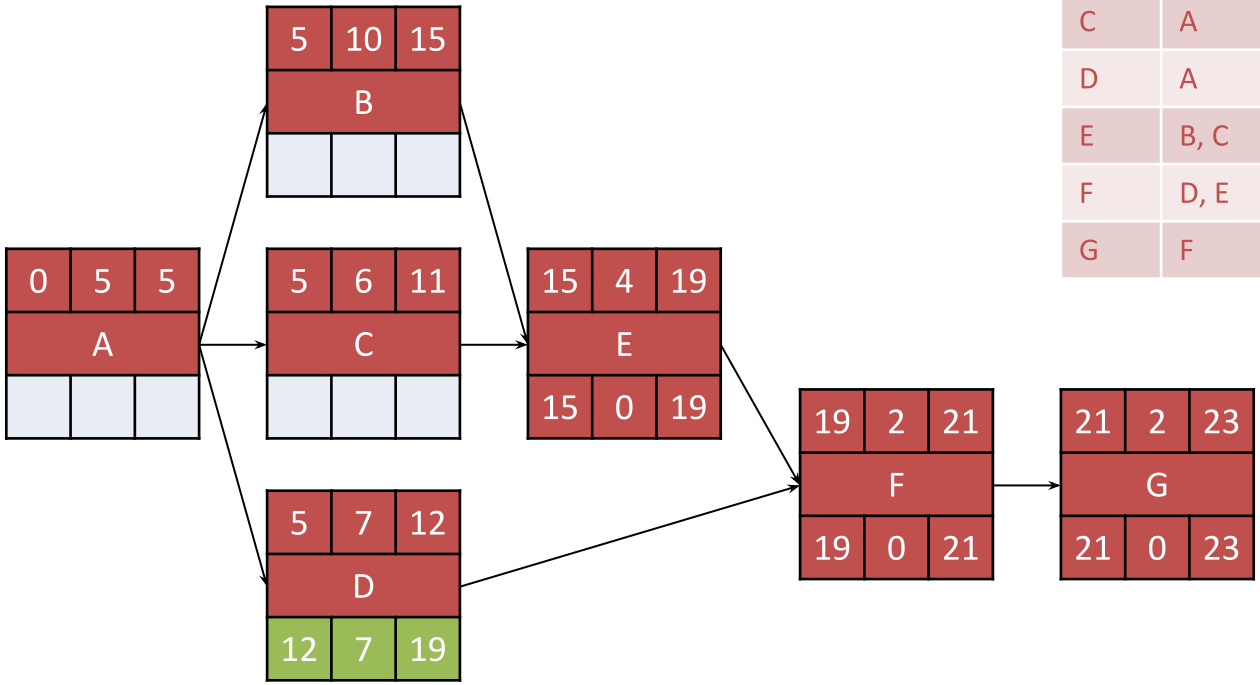
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

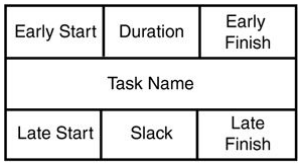


Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

LFT(D) ← LST(F)
LST(D) ← LFT(D) – DUR(D)
SLACK(D) ← LST(D) – EST(D)

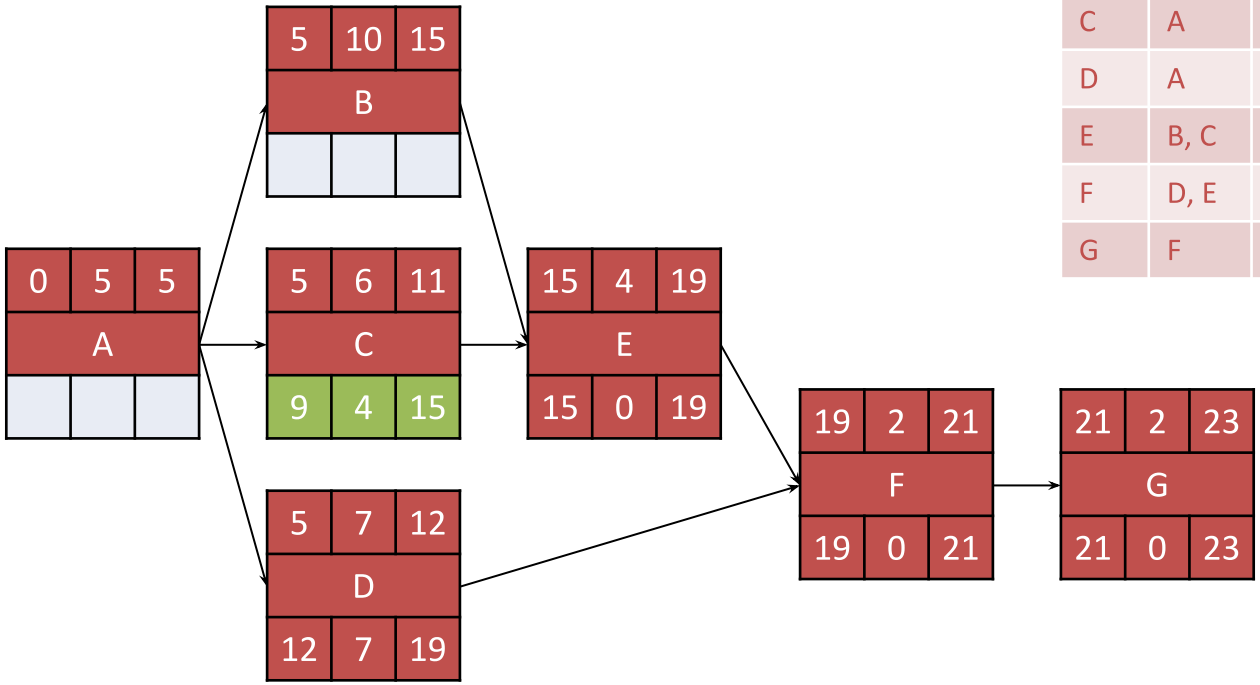
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

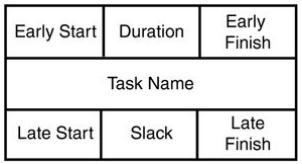




LFT(C) ← LST(E)
LST(C) ← LFT(C) – DUR(C)
SLACK(C) ← LST(C) – EST(C)

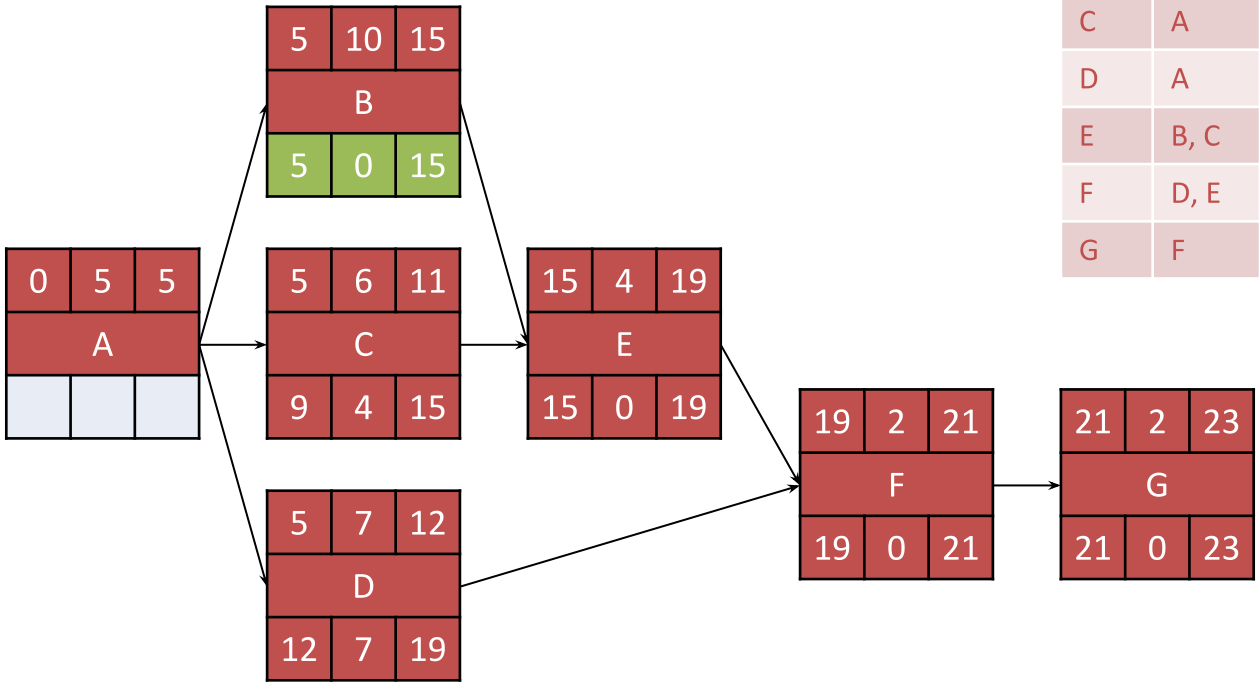
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

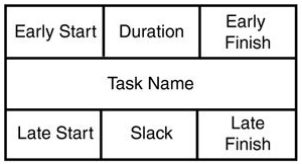




LFT(B) ← LST(E)
LST(B) ← LFT(B) – DUR(B)
SLACK(B) ← LST(B) – EST(B)

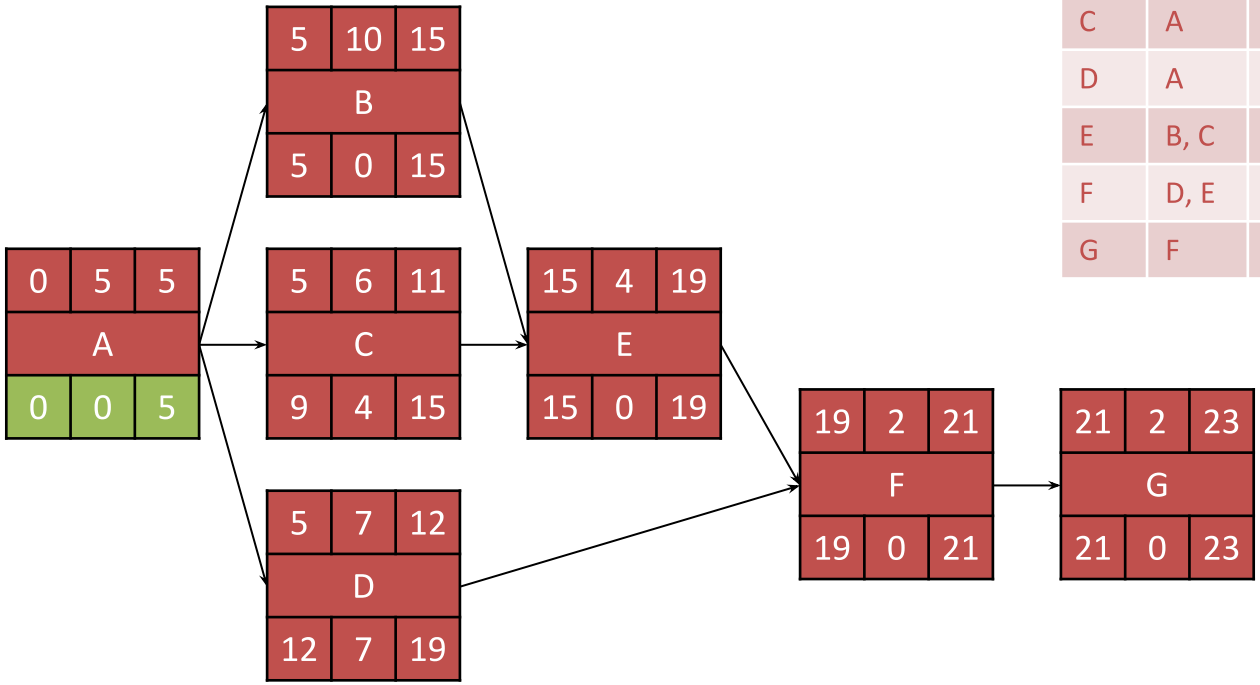
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

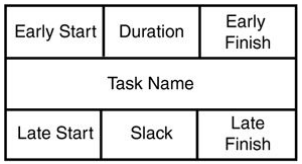




LFT(A) ← MIN(LST(B), LST(C), LST(D))
LST(A) ← LFT(A) – DUR(A)
SLACK(A) ← LST(A) – EST(A)

Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

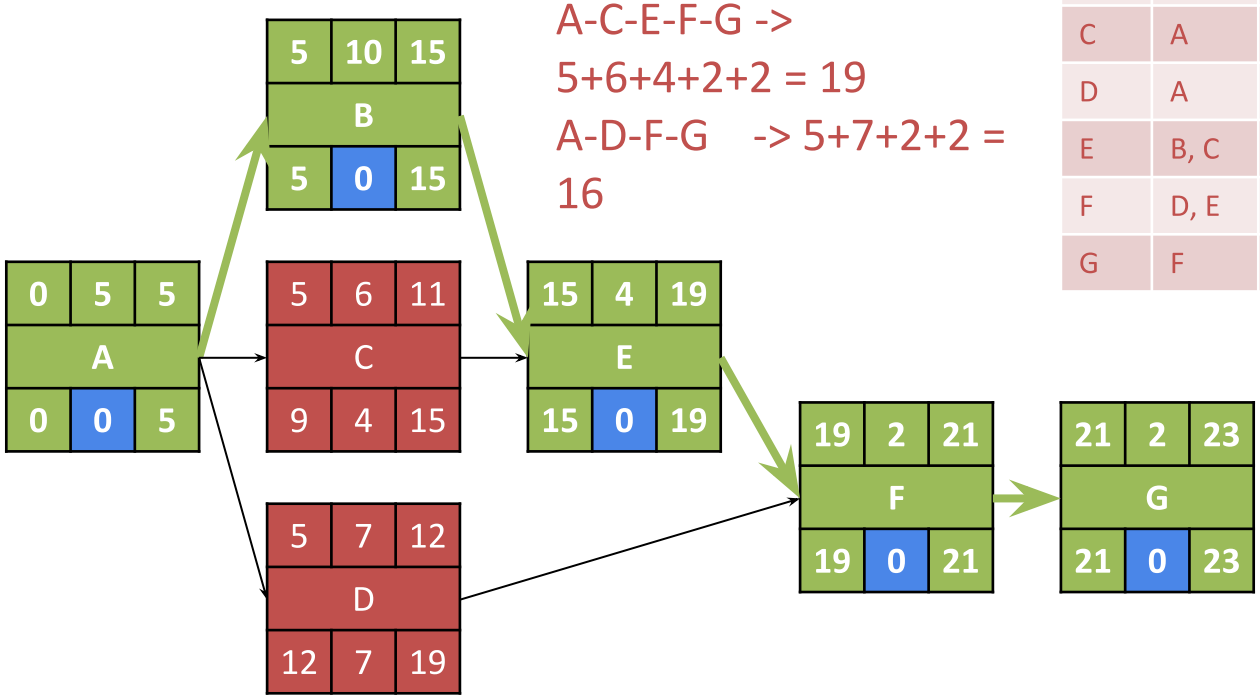




In the critical path, the slack is 0 and the sum of durations is maximized

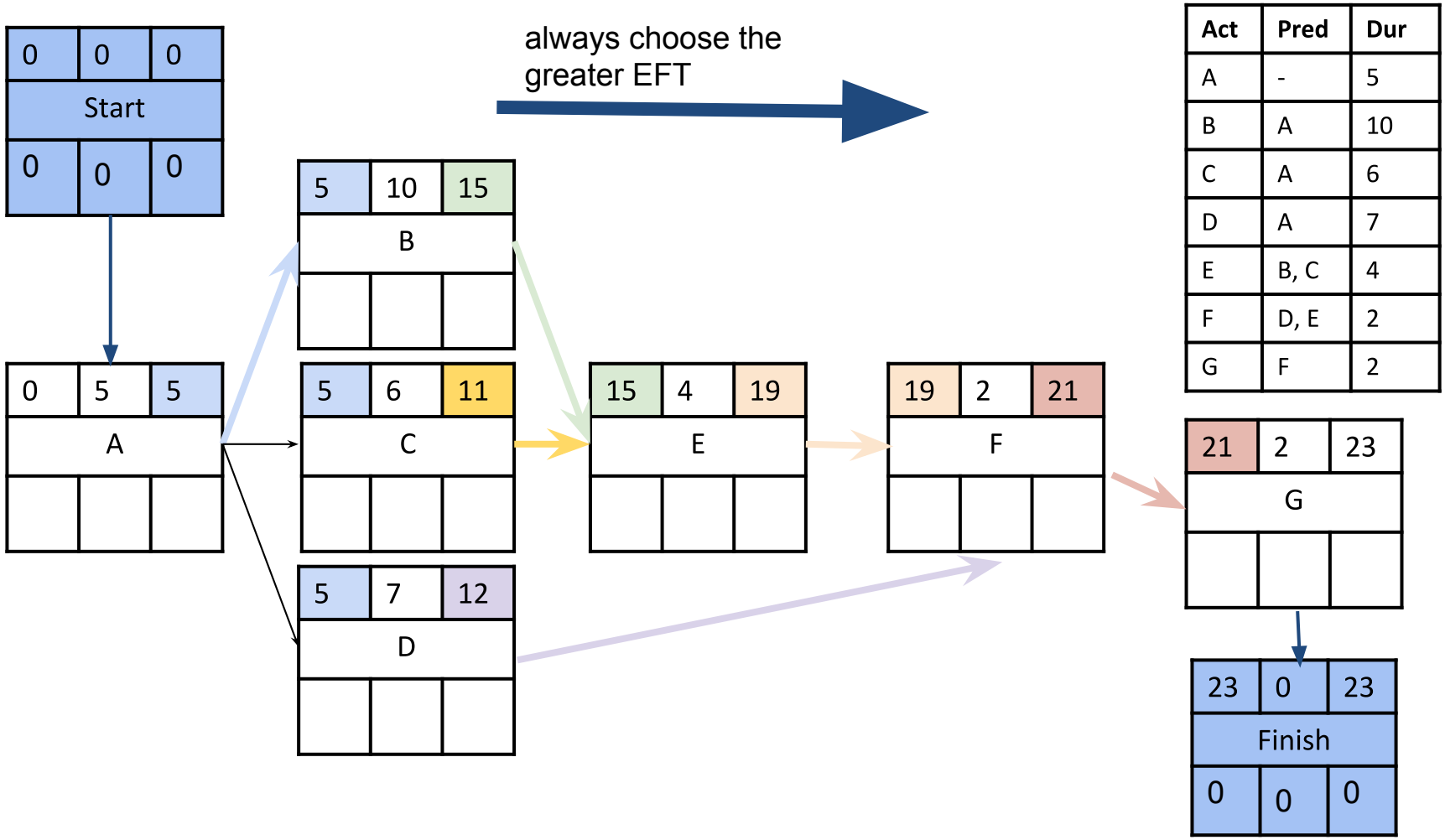
Act	Prec	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

A-B-E-F-G ->
5+10+4+2+2 = 23
A-C-E-F-G ->
5+6+4+2+2 = 19
A-D-F-G -> 5+7+2+2 = 16



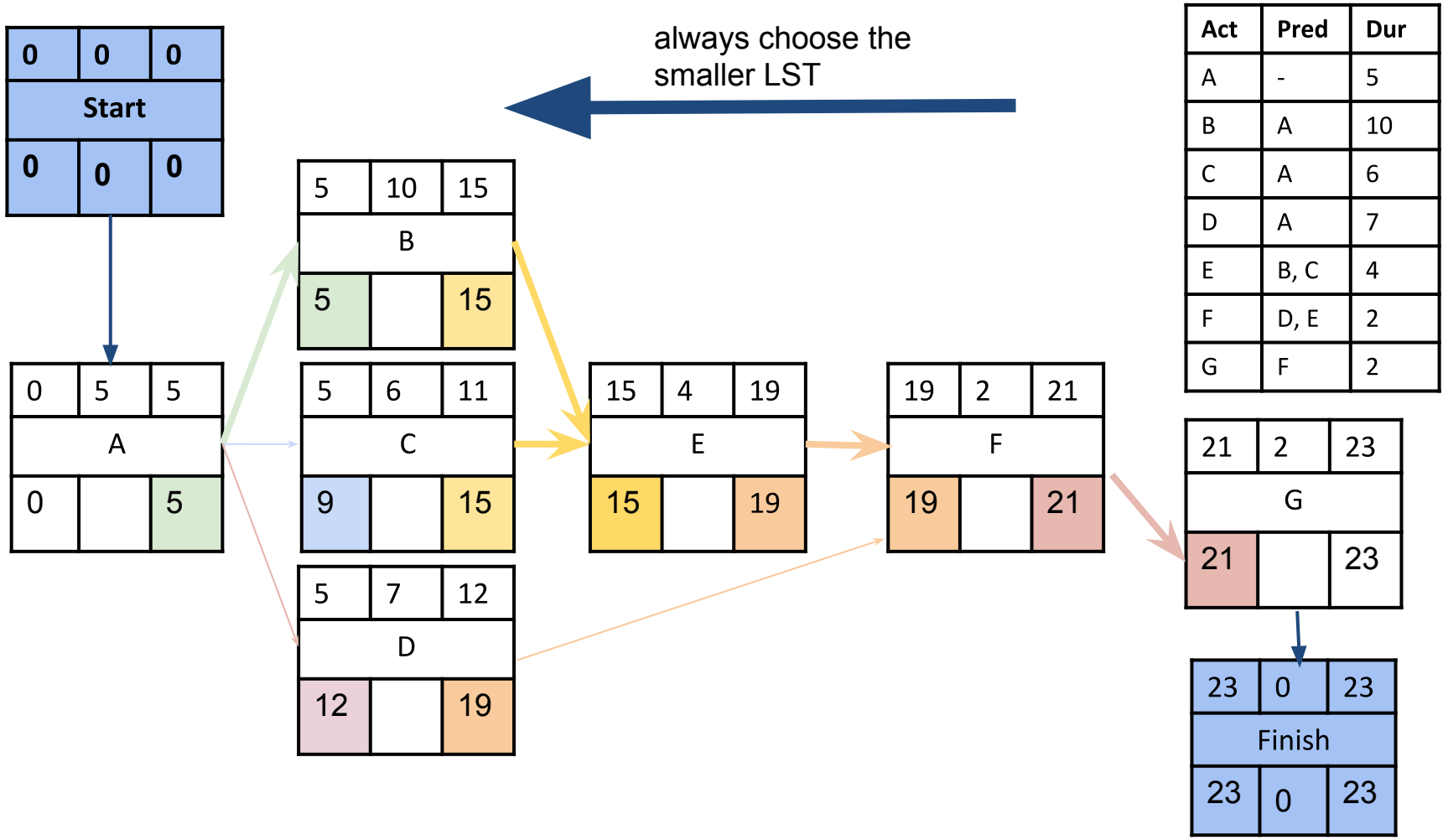
Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

Summary: choose the greater EFT



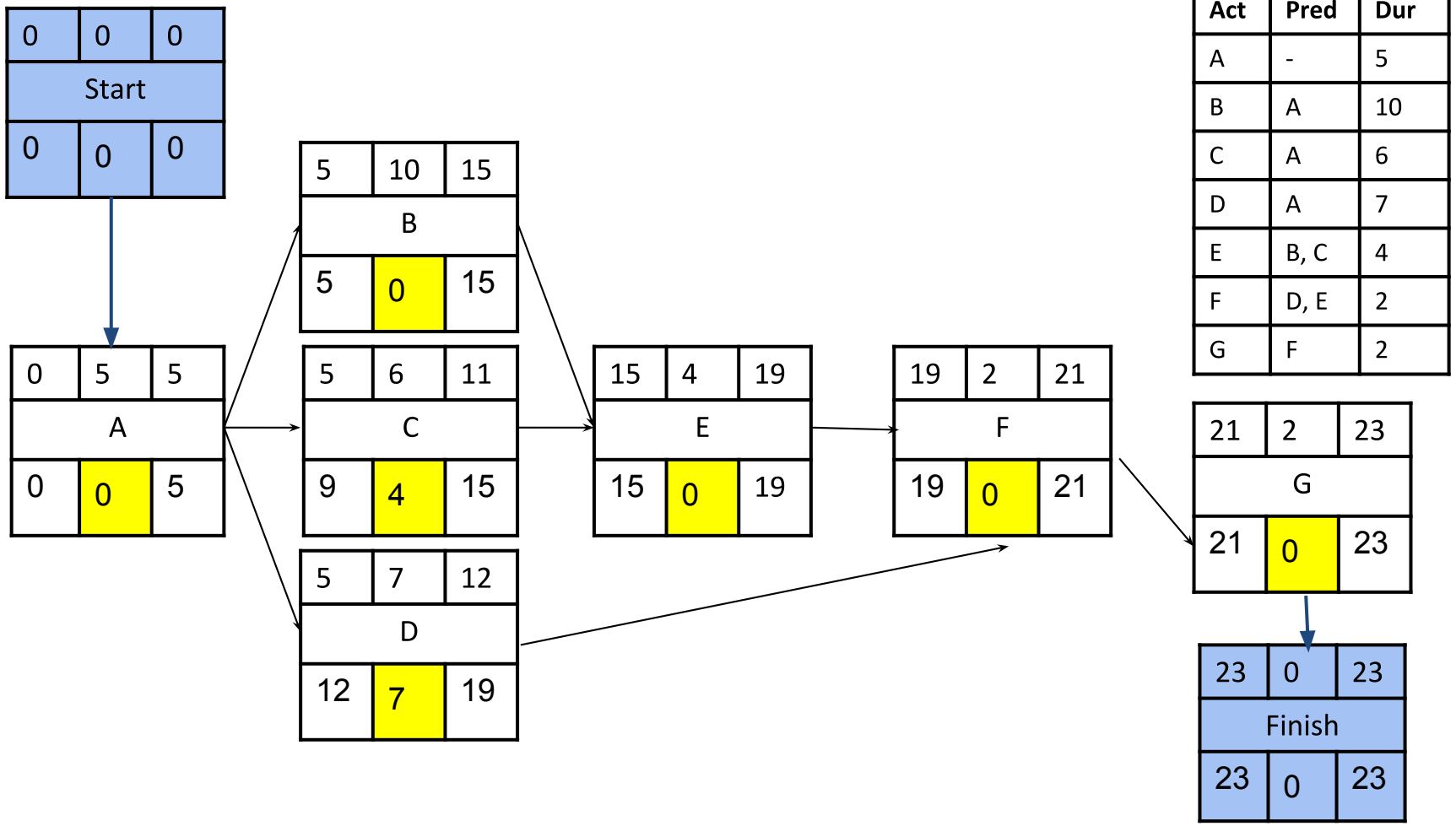
Act	Pred	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

Choose the smaller LST



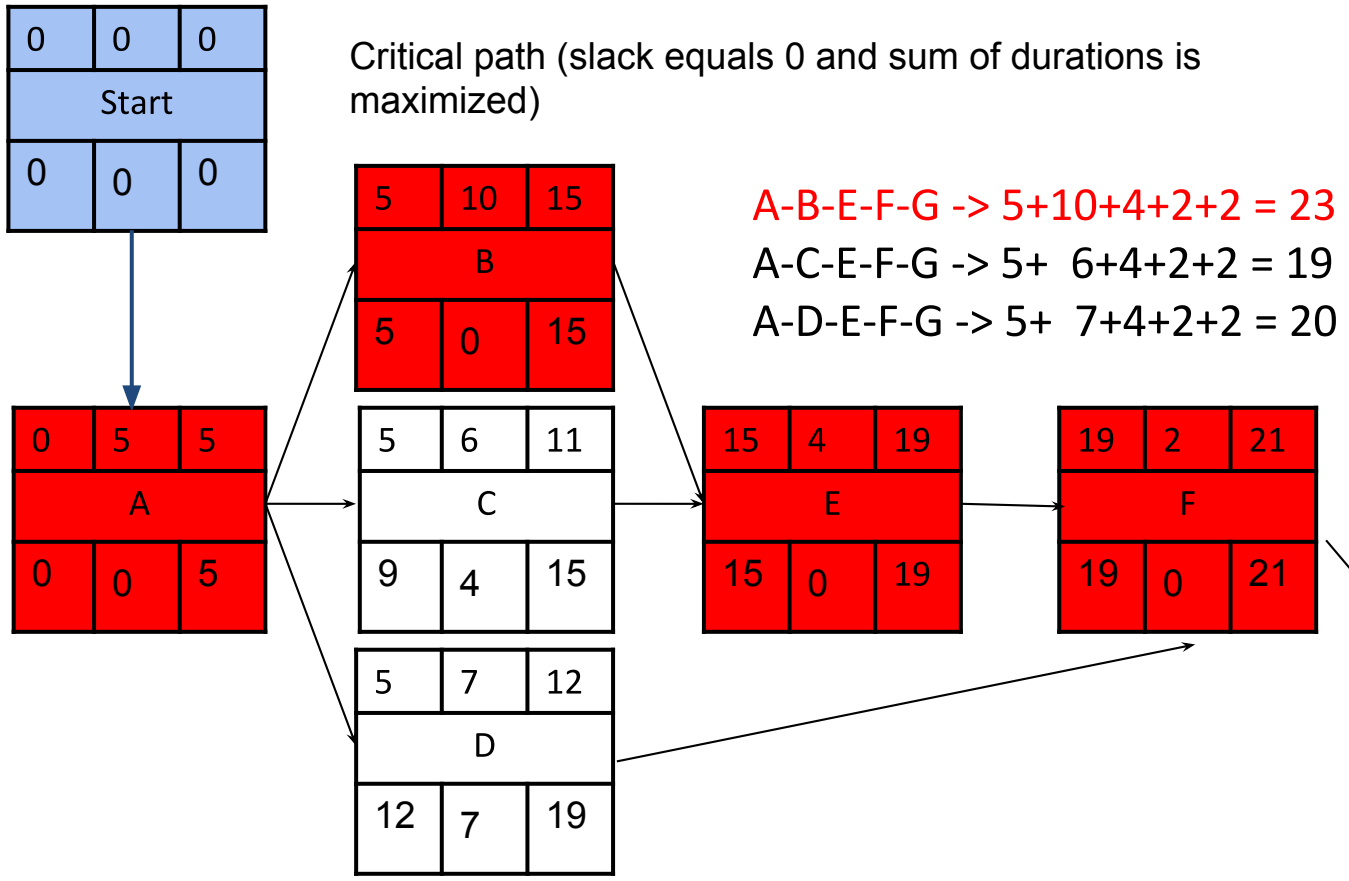
Act	Pred	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

Compute the slack



Act	Pred	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2

Compute the critical path



Critical path (slack equals 0 and sum of durations is maximized)

A-B-E-F-G -> 5+10+4+2+2 = 23

A-C-E-F-G -> 5+ 6+4+2+2 = 19

A-D-E-F-G -> 5+ 7+4+2+2 = 20

Act	Pred	Dur
A	-	5
B	A	10
C	A	6
D	A	7
E	B, C	4
F	D, E	2
G	F	2