# Software Engineering (CSE 327)

## Lecture 13

## (Planning and Scheduling)

# Planning, Task Scheduling

# Planning

- Split project into *tasks* or *activities* using the chosen SDLC as an anchor
- Create a *Work-Breakdown-Structure* (WBS)
  - breaks the project down into a set of well-defined, discrete tasks
- For each task or subtask, **estimate** the time for completion and assess resources required

# What to estimate?

Ultimate goal

**Assign** a duration (usually, time expressed in working days or hours) to each problem / task / outcome identified in a work breakdown structure (WBS)

# What to estimate?

Normal approach:

1. First estimate size and *complexity* of a given problem, task, or outcome
2. Then use this to estimate the effort / time required for the task

- Note that duration depends also on number of people available to do the work
- Also note that estimation is hard, and generally, for software development, is not done very effectively!

# The meaning of Time and Effort

- People cannot work 100% productively, 100% of their available time:
  - Need to consider interruptions, socializing, email etc.
  - Rule of thumb: productivity of IT people ~70%
    - eg, for a 40-hour work week, assume 28 hours of productive work
    - Varies from person to person
    - FACT (borne out by serious research): productivity decreases as total hours worked per week increases above about 40

# The meaning of Time and Effort – Definitions

- Ideal Time
  - fully productive time on given task without interruptions
- Ideal Effort
  - amount of ideal time it takes to complete a task
- Real Effort
  - Amount of real time taken to complete a task

# The meaning of Time and Effort : Example

- If it would take 20 hours of **ideal time** to write a user manual*, then assume it will take **???** of real time

- *How do we obtain the ideal time estimate?

# The meaning of Time and Effort : Example

- If it would take 20 hours of **ideal time** to write a user manual*, then assume it will take **???** of real time

- *How do we obtain the ideal time estimate?
  - Past experience
  - Measurement of actual time on a small task, multiplied to give estimate for full task
  - Measurement of task according to a reasonable size estimate
  - Magic??!!

# Example

- I can mark 1 exam paper in 10 minutes
- I have 100 papers to mark
- How long will it take?

# Example

- I can mark 1 exam paper in 10 minutes, on average
- I have 100 papers to mark
- How long will it take?

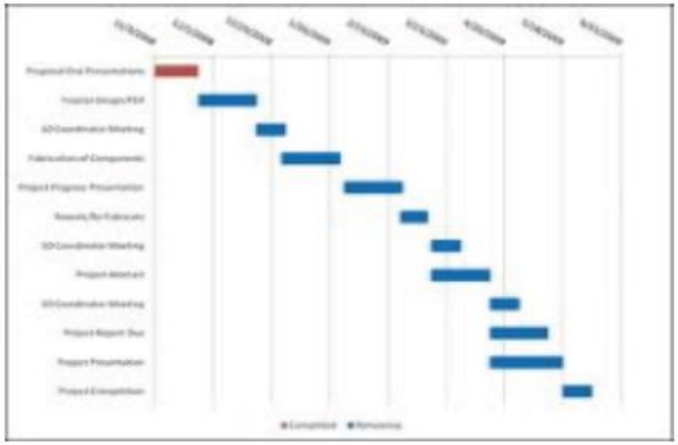
  ---

- Simple answer is 1000 mins = 16 hrs and 40 mins
- But I can't keep up the rate of 1 paper every 10 mins
- Although I will probably only ever take 10 mins to mark a paper, over a day's work I will have time spent away from the direct task – probably 30% of my time {estimated from past experience}
- So, in reality, I will take approx. 24 hrs (= 1000/0.7) to complete the task

# Project Scheduling

Goal:

- Organizing project activities
  in a coherent order



- Organizing the staff allocations
  to activities
  to ensure project is completed in
  minimum time

# Agile Example – Tasks, Dependencies and Duration

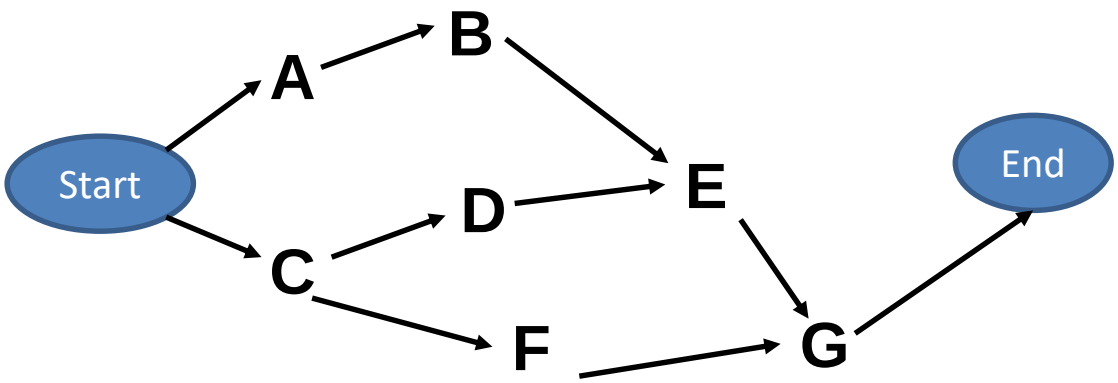
| Task Id. | Task Description | Dependencies | Duration (hours) |
|---|---|---|---|
| A | Select appropriate database server (may need spiking) | | 4 |
| B | Install database server for coding | A | 3 |
| C | Analyze the reporting requirements | | 5 |
| D | Design the required module | C | 4 |
| E | Programming the required module | B,D | 5 |
| F | Design test data for reporting | C | 3 |
| G | Verify the correctness of report | E,F | 2 |

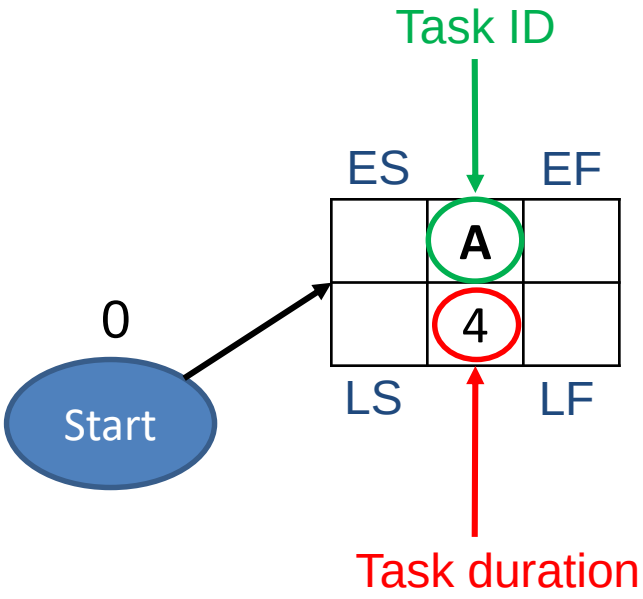
# Agile Example – Tasks, Dependencies and Duration

| Task Id. | Task Description | Dependencies | Duration (hours) |
|---|---|---|---|
| A | Select appropriate database server (may need spiking) | | 4 |
| B | Install database server for coding | A | 3 |
| C | Analyze the reporting requirements | | 5 |
| D | Design the required module | C | 4 |
| E | Programming the required module | B,D | 5 |
| F | Design test data for reporting | C | 3 |
| G | Verify the correctness of report | E,F | 2 |

Total number of hours = 26 hrs → 3.25 days work [Ideal time]
OR → 4.6 days work [realistic time]

| Task ID | Dependencies | Duration |
|---|---|---|
| A | | 4 |
| B | A | 3 |
| C | | 5 |
| D | C | 4 |
| E | B, D | 5 |
| F | C | 3 |
| G | E, F | 2 |



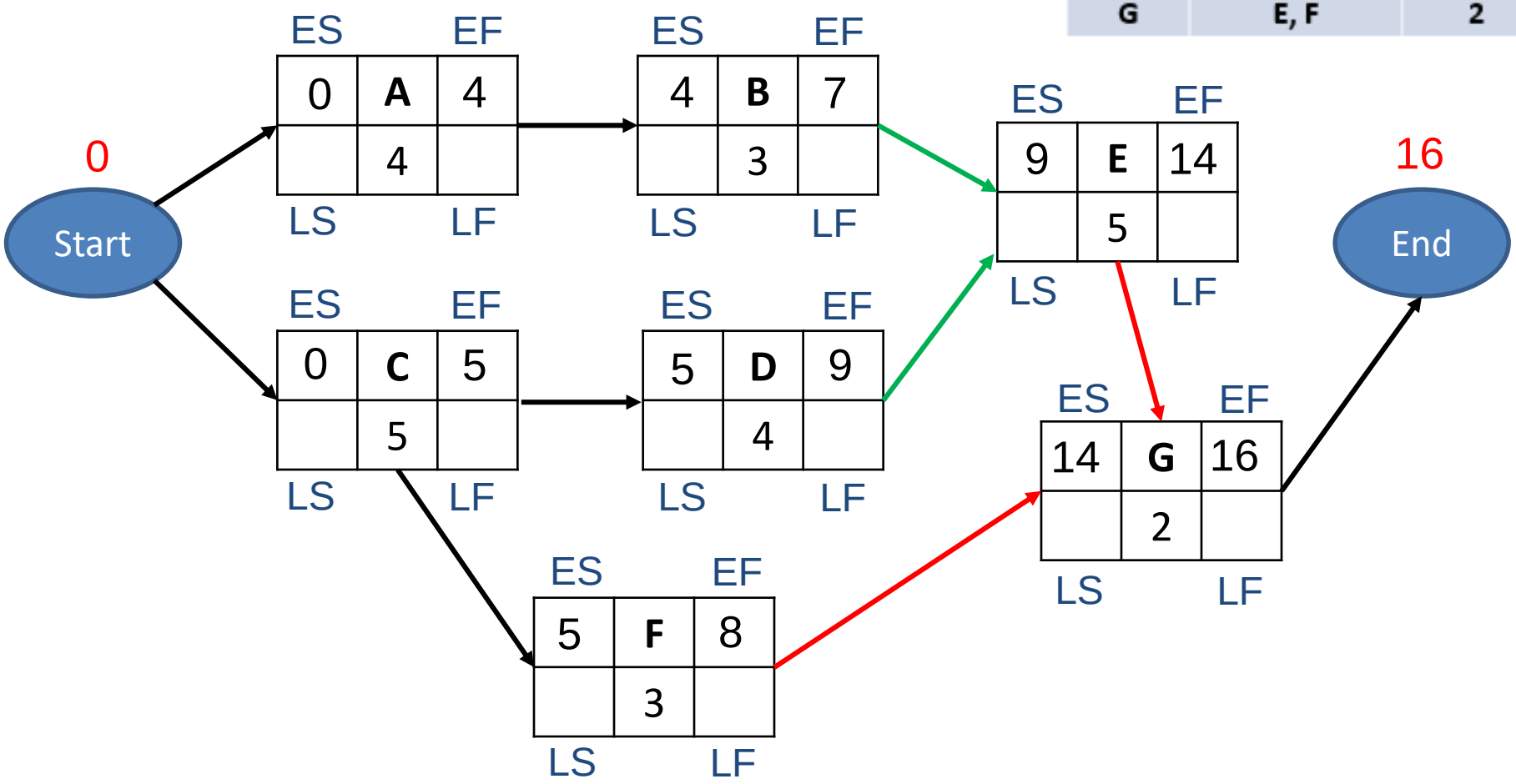



# EF = ES + Task Duration

Early Start of an activity = Early Finish of predecessor activity

Early Start of an activity = **Greater** Early Finish time from **multiple** predecessor activities

| Task ID | Dependencies | Duration |
|---|---|---|
| A | | 4 |
| B | A | 3 |
| C | | 5 |
| D | C | 4 |
| E | B, D | 5 |
| F | C | 3 |
| G | E, F | 2 |

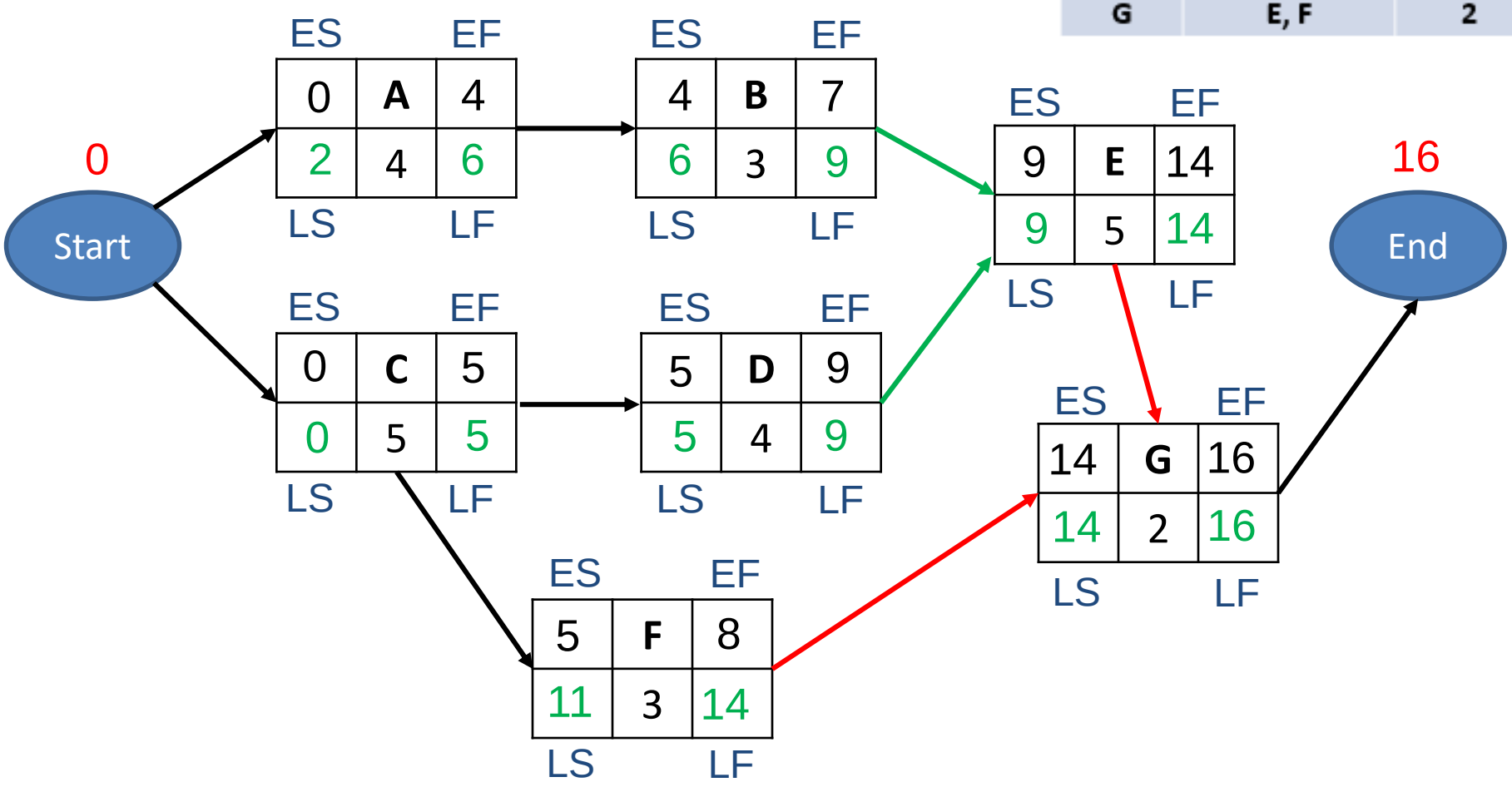


# LS = LF - Task Duration

Late Finish of an activity = Late Start of successor activity

Late Finish of an activity = **Earlier** (least) Late Start time from **multiple** successor activities

| Task ID | Dependencies | Duration |
|---|---|---|
| A | | 4 |
| B | A | 3 |
| C | | 5 |
| D | C | 4 |
| E | B, D | 5 |
| F | C | 3 |
| G | E, F | 2 |



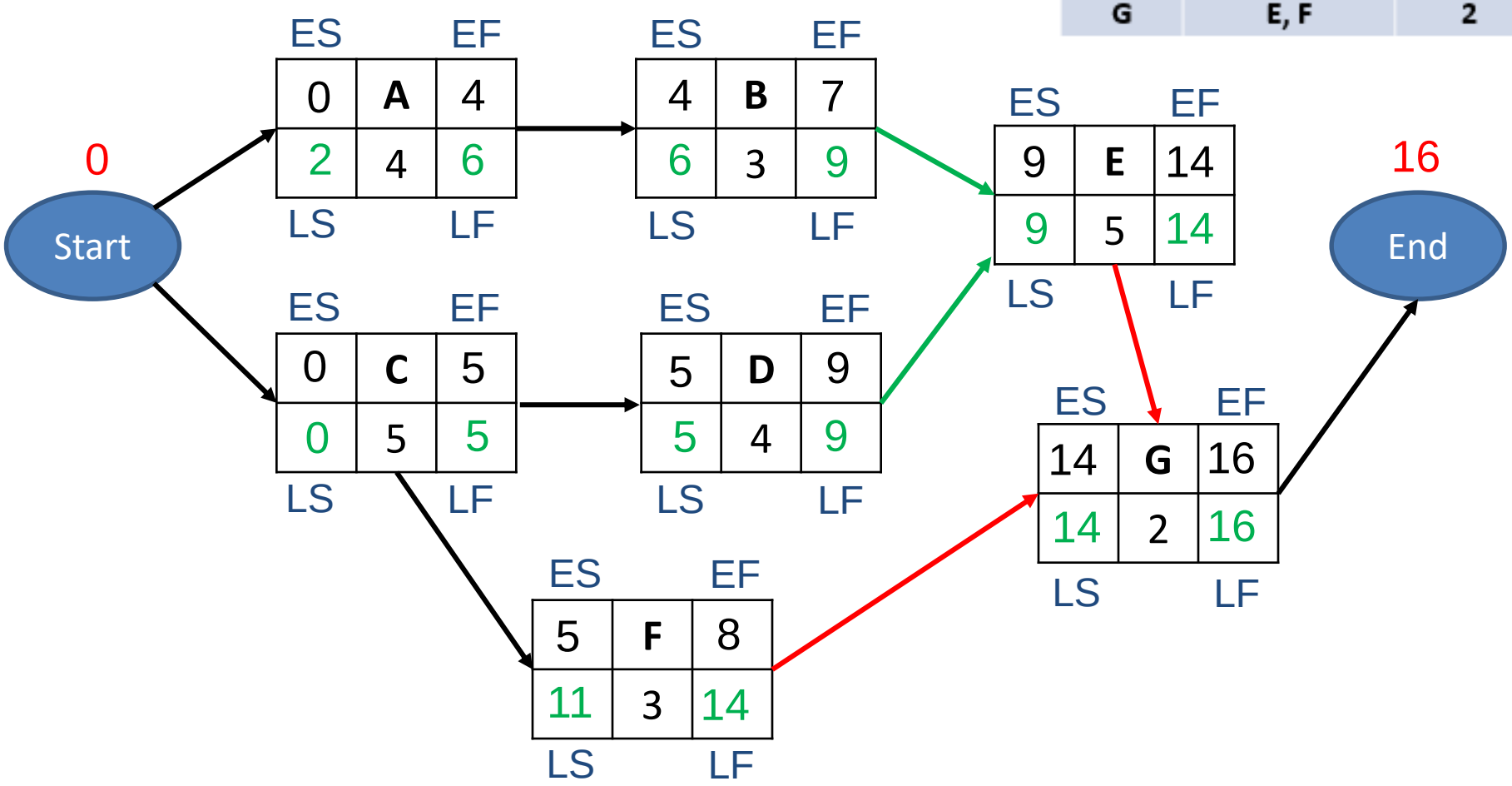
**Float / Slack Time** = (LS – ES) or (LF - EF)

Float time of Task **A**: (2 - 0) or (6 - 4) = 2

Float time of Task **F**: (11 - 5) or (14 - 8) = 6

Float time of Task **D**: (5 - 5) or (9 - 9) = 0

| Task ID | Dependencies | Duration |
|---|---|---|
| A | | 4 |
| B | A | 3 |
| C | | 5 |
| D | C | 4 |
| E | B, D | 5 |
| F | C | 3 |
| G | E, F | 2 |

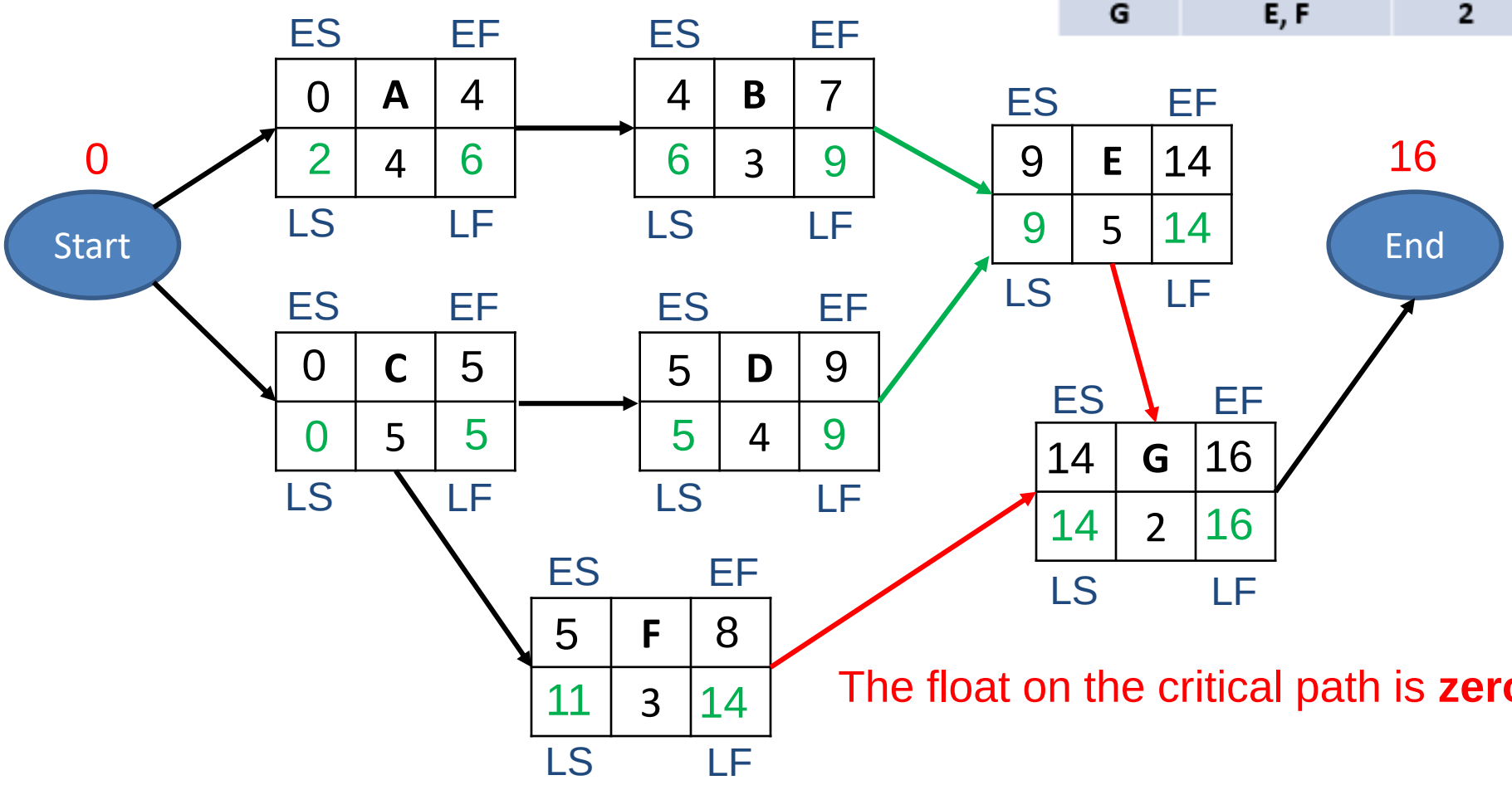


This network diagram has **three paths** from start to end. The paths and their durations are as follows:

- Start -> A -> B -> E -> G -> End {duration: 14 days}
- Start -> C -> D -> E -> G -> End {duration: 16 days}
- Start -> C -> F -> G -> End {duration: 10 days}

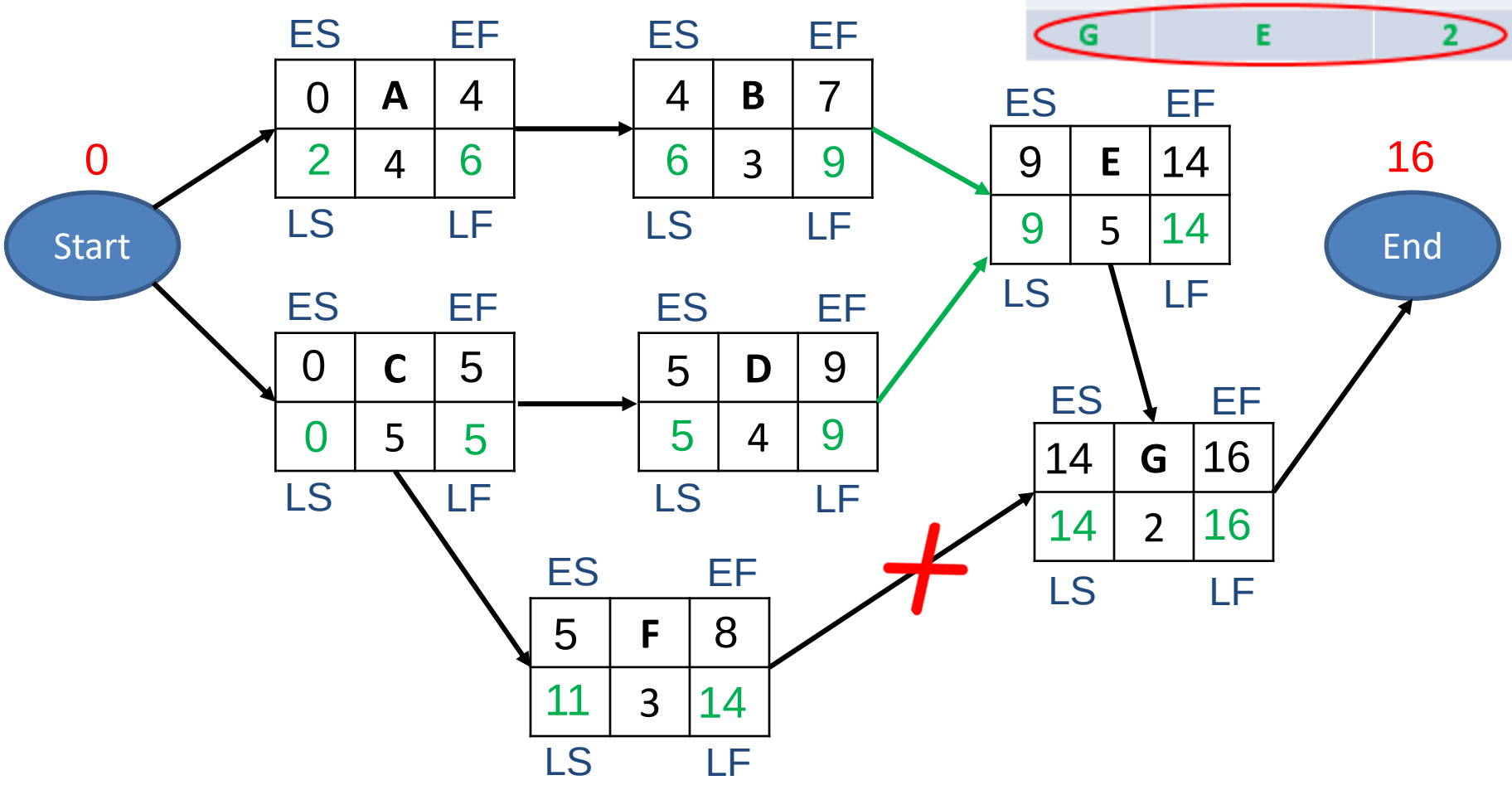
**Critical path** is the duration of the longest path.

| Task ID | Dependencies | Duration |
|---|---|---|
| A | | 4 |
| B | A | 3 |
| C | | 5 |
| D | C | 4 |
| E | B, D | 5 |
| F | C | 3 |
| G | E, F | 2 |




The float on the critical path is **zero**.

| Task ID | Dependencies | Duration |
|---|---|---|
| A | | 4 |
| B | A | 3 |
| C | | 5 |
| D | C | 4 |
| E | B, D | 5 |
| F | C | 3 |
| G | E | 2 |

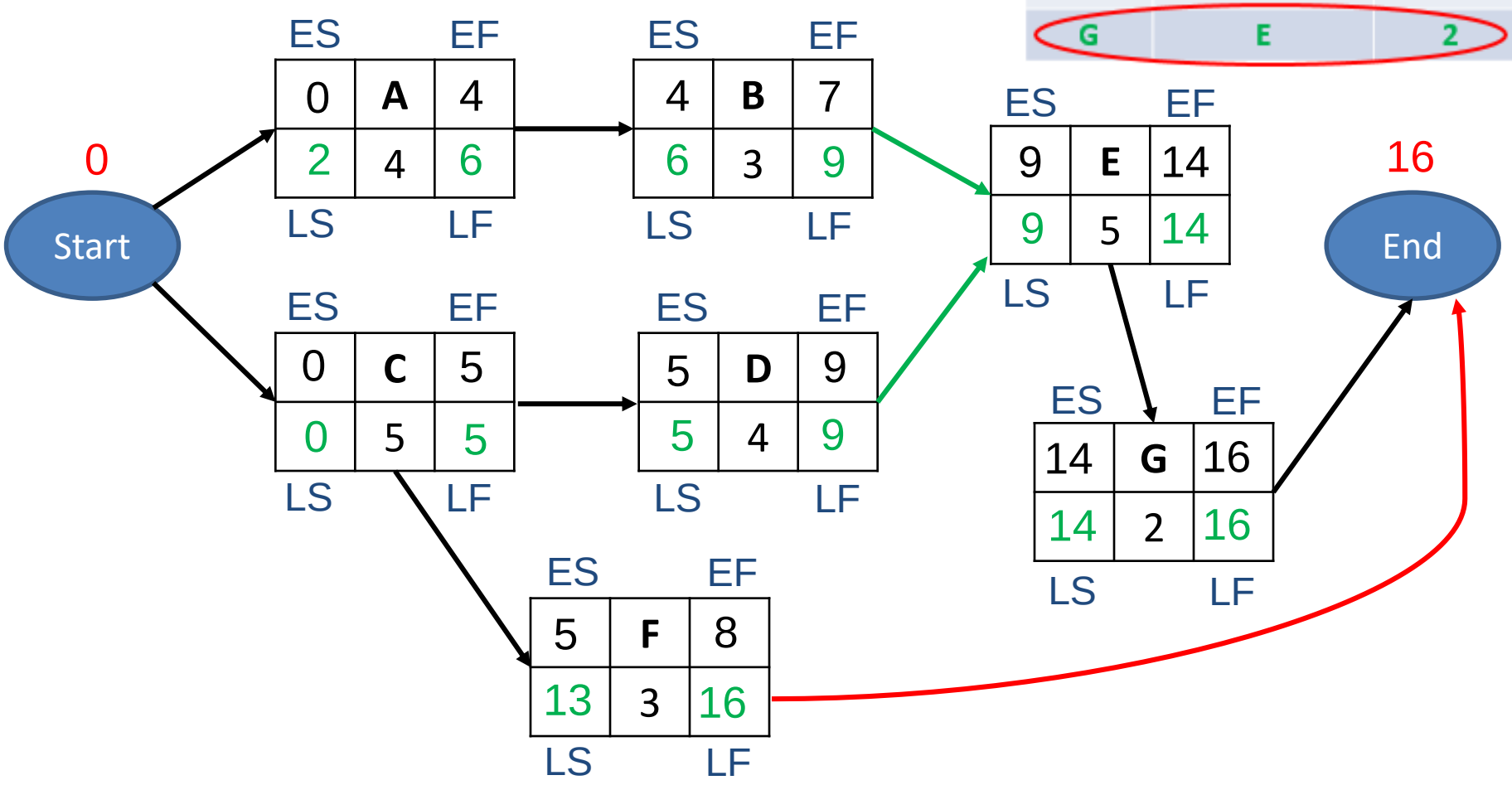


Late Finish of an activity = Late Start of successor activity
Late Finish of an activity = **Earlier** (least) Late Start time from **multiple** successor activities

| Task ID | Dependencies | Duration |
|---|---|---|
| A | | 4 |
| B | A | 3 |
| C | | 5 |
| D | C | 4 |
| E | B, D | 5 |
| F | C | 3 |
| G | E | 2 |

This network diagram has **three paths** from start to end.

- Start -> A -> B -> E -> G -> End {duration: 14 days}
- Start -> C -> D -> E -> G -> End {duration: 16 days}
- **Start -> C -> F -> End {duration: 8 days}**



# Agile Example – Tasks, Dependencies and Duration

**Sequential technique**

- Total time = 26 hours
- → 3.25 days [Ideal]
- → 4.6 days [Realistic]

**Critical Path Method**

- Total time = 16 hours
- → 2 days [Ideal]
- → 2.8 days [Realistic]

# Critical Path Method

- The **Critical Path Method** or **CPM** is a network analysis technique
  - concerned with planning and controlling of complex, but routine projects.
  - Simply, Critical path method is generally used for the projects whose time duration is known with certainty and also the amount of resources required for the completion of the project is assumed to be known.

# Critical Path Method

- *Critical path schedules* will...
  - Help you identify the activities that must be completed on time in order to complete the whole project on time.
  - Show you which tasks can be delayed and for how long without impacting the overall project schedule.
  - Calculate the minimum amount of time it will take to complete the project.
  - Tell you the earliest and latest dates each activity can start on in order to maintain the schedule.

# Critical Path Method

- The Critical Path Method has four key elements...
  - Critical Path Analysis
  - Float Determination
  - Early Start & Early Finish Calculation
  - Late Start & Late Finish Calculation

# Critical Path Method

- For each activity the following parameters need to be determined:
  - **Earliest start time (ES):** How early, the successor activity begins once the predecessor activity finishes.
  - **Earliest Finish Time (EF):** Earliest Start Time + duration of each activity.
  - **Latest Finish Time (LF):** The latest time within which the activity finishes without delaying the project.
  - **Latest Start Time (LS):** Latest Finish Time – Activity duration

# Critical Path Analysis

- The **critical path** is the sequence of activities with the longest duration. A delay in any of these activities will result in a delay for the whole project.
- **Float** is the amount of time an activity can slip before it causes your project to be delayed. Float is sometimes referred to as slack.