Introduction to Real-Time Systems

ECE 397-1

Northwestern University

Department of Computer Science

Department of Electrical and Computer Engineering

Teachers:	Robert Dick	Peter Dinda
Office:	L477 Tech	338, 1890 Maple Ave.
Email:	dickrp@ece.northwestern.edu	pdinda@cs.northwestern.edu
Phone:	467–2298	467-7859
Webpage:	http://ziyang.ece.northwestern.edu/EXTERNAL/realtime	

Homework index

Goals for lecture

- Handle a few administrative details
- Form lab groups
- Broad overview of real-time systems
- Definitions that will come in handy later
- Example of real-time sensor network

Administrative tasks

- Backgrounds
- Question rule
- Office hours

Backgrounds

- Lab teams had best be balanced (low-level vs. high-level experience)
- Name
- Which are you better at?
  - Low-level ANSI-C/assembly experience
  - High-level object-oriented programming experience
- What's your major?

Question rule

- If something in lecture doesn't make sense, please ask
- You're paying a huge amount of money for this
- Letting something important from lecture slip by for want of a question is like burning handfulls of money

Core course goal

By the end of this course, we want you to
learn how to build real-time systems
and build a useful real-time sensor network.

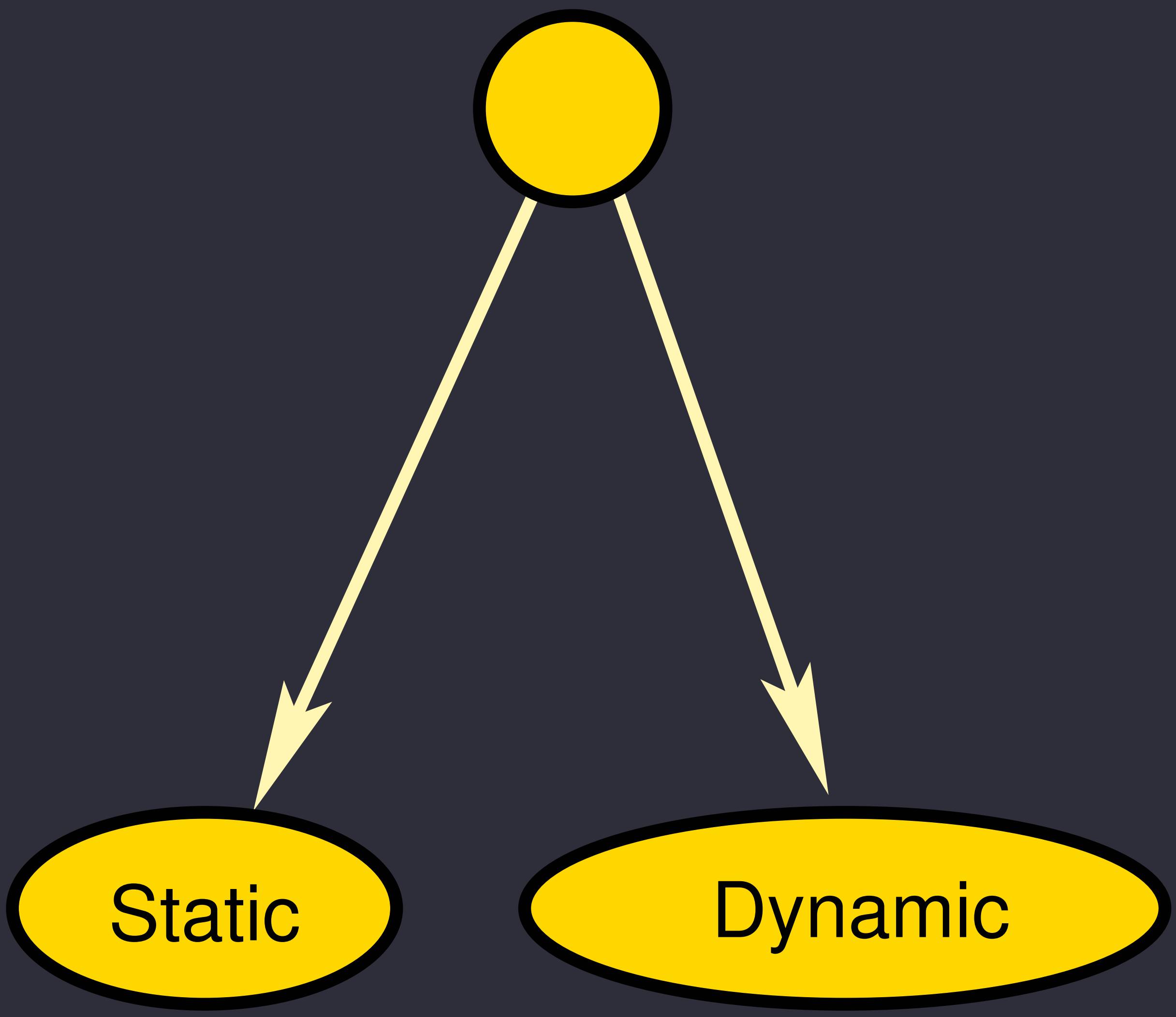
Office hours

- When shall I schedule my office hours?

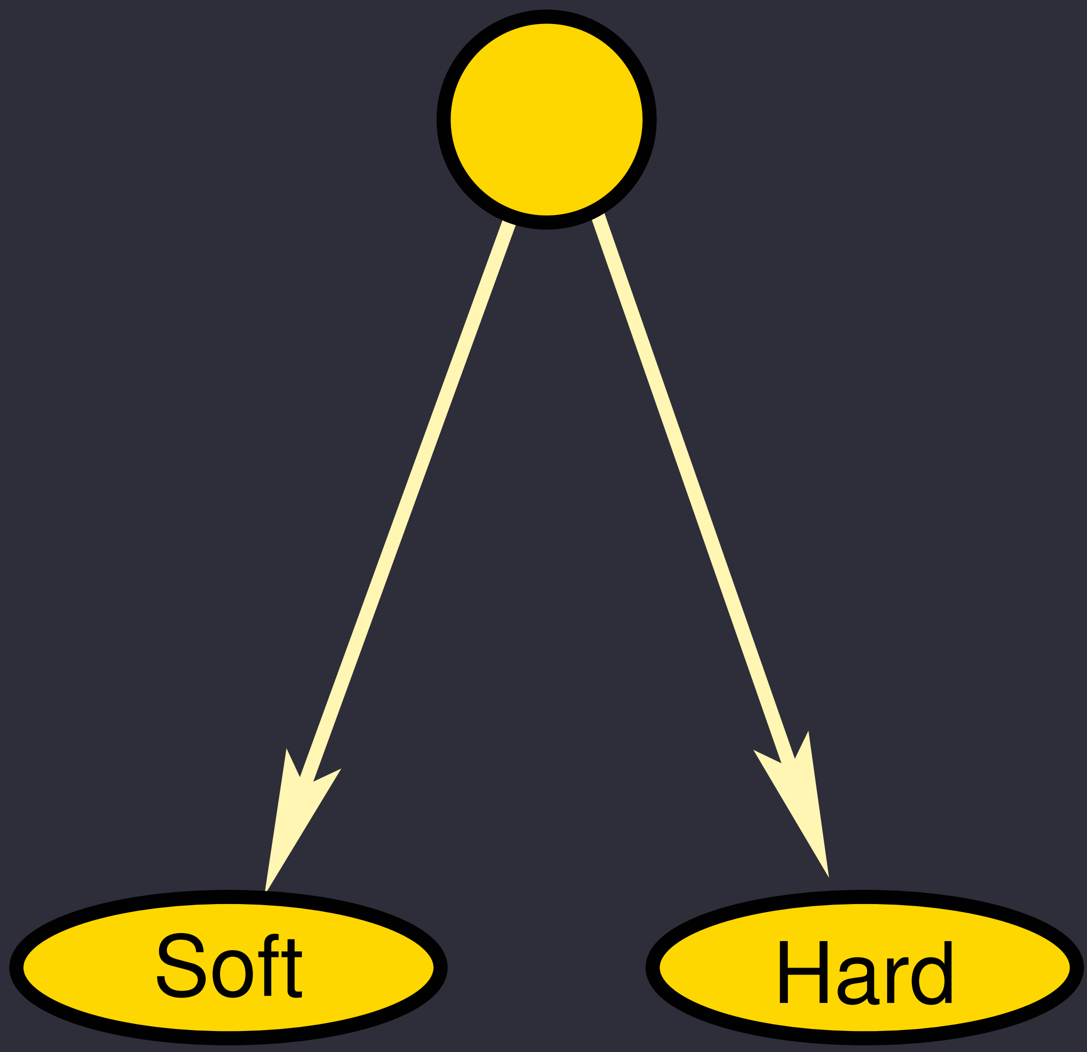
Today's topics

- Taxonomy of real-time systems
- Optimization and costs
- Definitions
- Optimization formulation
- Overview of primary areas of study within real-time systems

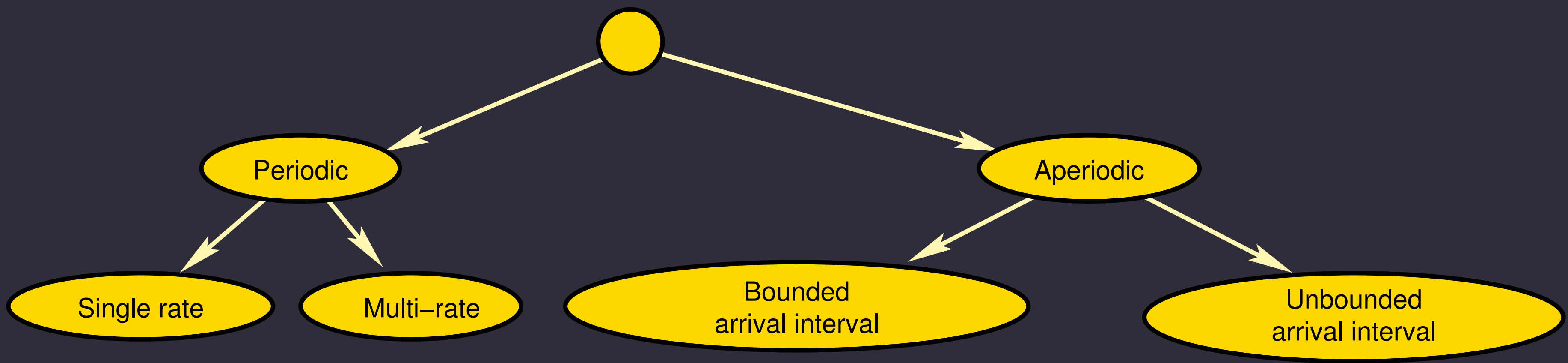
Taxonomy of real-time systems



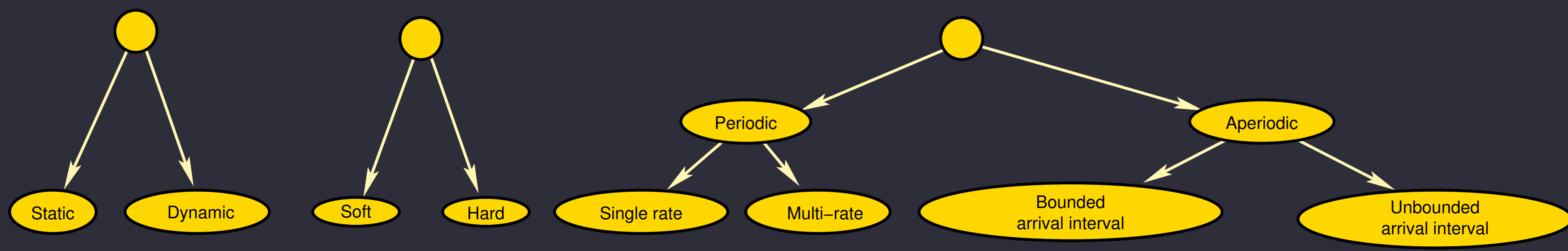
Taxonomy of real-time systems



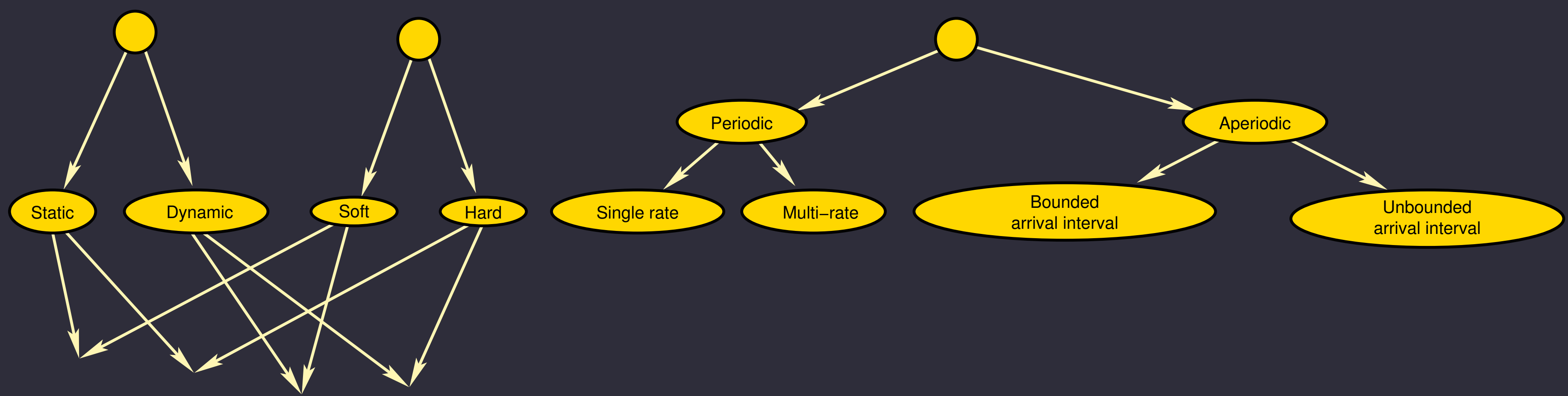
Taxonomy of real-time systems



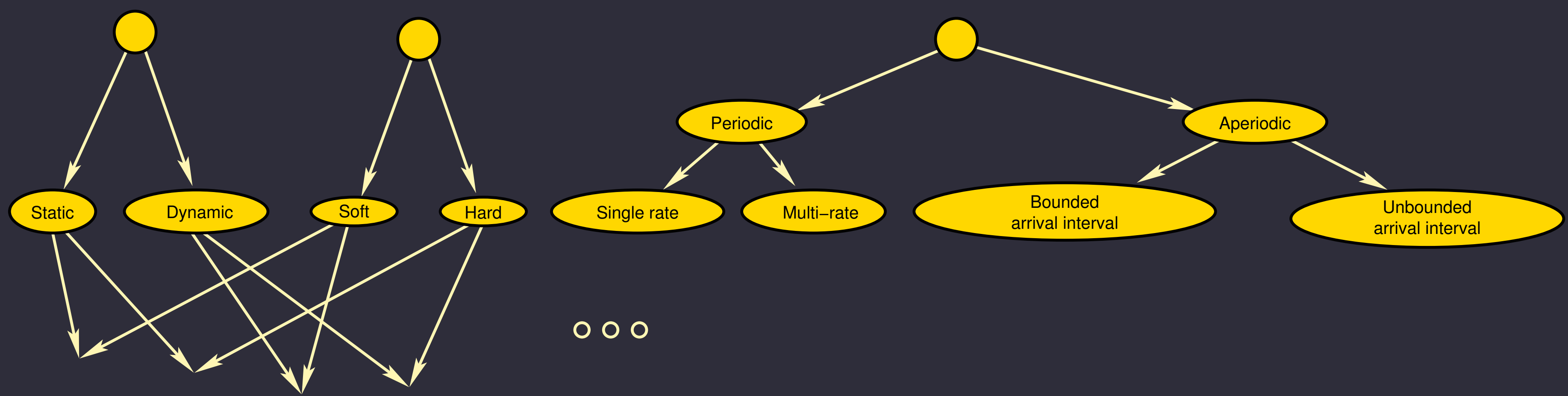
Taxonomy of real-time systems



Taxonomy of real-time systems



Taxonomy of real-time systems



Taxonomy: Static

- Task arrival times can be predicted.
- Static (compile-time) analysis possible.
- Allows good resource usage (low processor idle time proportions).
- Sometimes designers shoehorn dynamic problems into static formulations allowing a good solution to the wrong problem.

Taxonomy: Dynamic

- Task arrival times unpredictable.
- Static (compile-time) analysis possible only for simple cases.
- Even then, the portion of required processor utilization efficiency goes to 0.693.
- In many real systems, this is very difficult to apply in reality (more on this later).
- Use the right tools but don't over-simplify, e.g.,

  We assume, without loss of generality, that all tasks are independent.

  If you do this people will make jokes about you.

Taxonomy: Soft real-time

- More slack in implementation
- Timing may be suboptimal without being incorrect
- Problem formulation can be much more complicated than hard real-time
- Two common (and one uncommon) methods of dealing with non-trivial soft real-time system requirements
  - Set somewhat loose hard timing constraints
  - Informal design and testing
  - Formulate as optimization problem

Taxonomy: Hard real-time

- Difficult problem. Some timing constraints inflexible.
- Simplifies problem formulation.

Taxonomy: Periodic

- Each task (or group of tasks) executes repeatedly with a particular period.
- Allows some nice static analysis techniques to be used.
- Matches characteristics of many real problems...
- ... and has little or no relationship with many others that designers try to pretend are periodic.

Taxonomy: Periodic $\longrightarrow$ Single-rate

- One period in the system.
- Simple.
- Inflexible.
- This is how a *lot* of wireless sensor networks are implemented.

Taxonomy: Periodic $\longrightarrow$ Multirate

- Multiple periods.
- Can use notion of circular time to simplify static (compile-time) schedule analysis E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, pp. 699–719, July 1966.
- Co-prime periods leads to analysis problems.

Taxonomy: Periodic $\longrightarrow$ Other

- It is possible to have tasks with deadlines less than, equal to, or greater than their periods.
- Results in multi-phase, circular-time schedules with multiple concurrent task instances.
  - If you ever need to deal with one of these, see me (take my code). This class of scheduler is nasty to code.

Taxonomy: Aperiodic

- Also called sporadic, asynchronous, or reactive
- Implies dynamic
- Bounded arrival time interval permits resource reservation
- Unbounded arrival time interval impossible to deal with for any resource-constrained system

Definitions

- Task
- Processor
- Graph representations
- Deadline violation
- Cost functions

Definitions: Task

- Some operation that needs to be carried out
- Atomic completion: A task is all done or it isn't
- Non-atomic execution: A task may be interrupted and resumed

Definitions: Processor

- Processors execute tasks
- Distributed systems
  - Contain multiple processors
  - Inter-processor communication has impact on system performance
  - Communication is challenging to analyze
- One processor type: Homogeneous system
- Multiple processor types: Heterogeneous system

Task/processor relationship

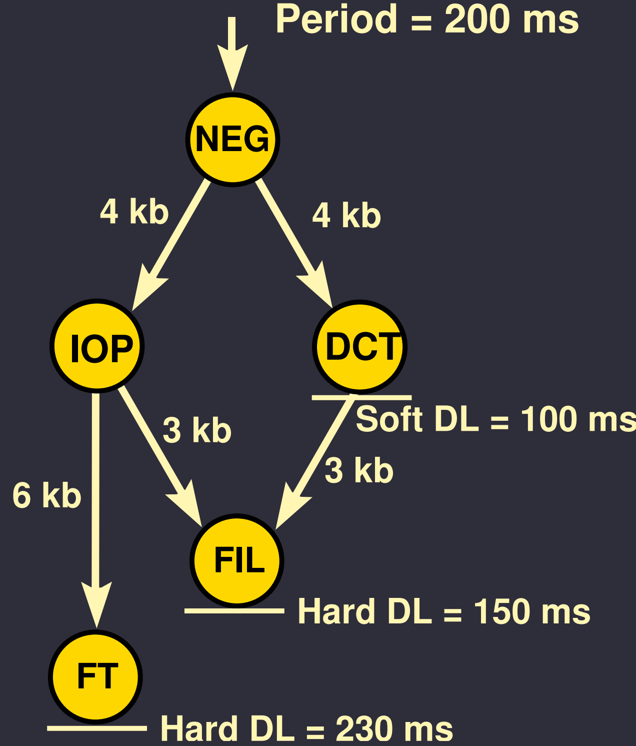
WC exec time (s)

	IBM PowerPC 405GP 266 MHz	IDT79RC32364 100 MHz	Imsys Cjip 40 MHz
Tooth	7.7E–6	···	
Road	330E–9	...	
FIR	4.1E–6	···	
Matrix	310E–3	···	

Relationship between tasks, processors, and costs

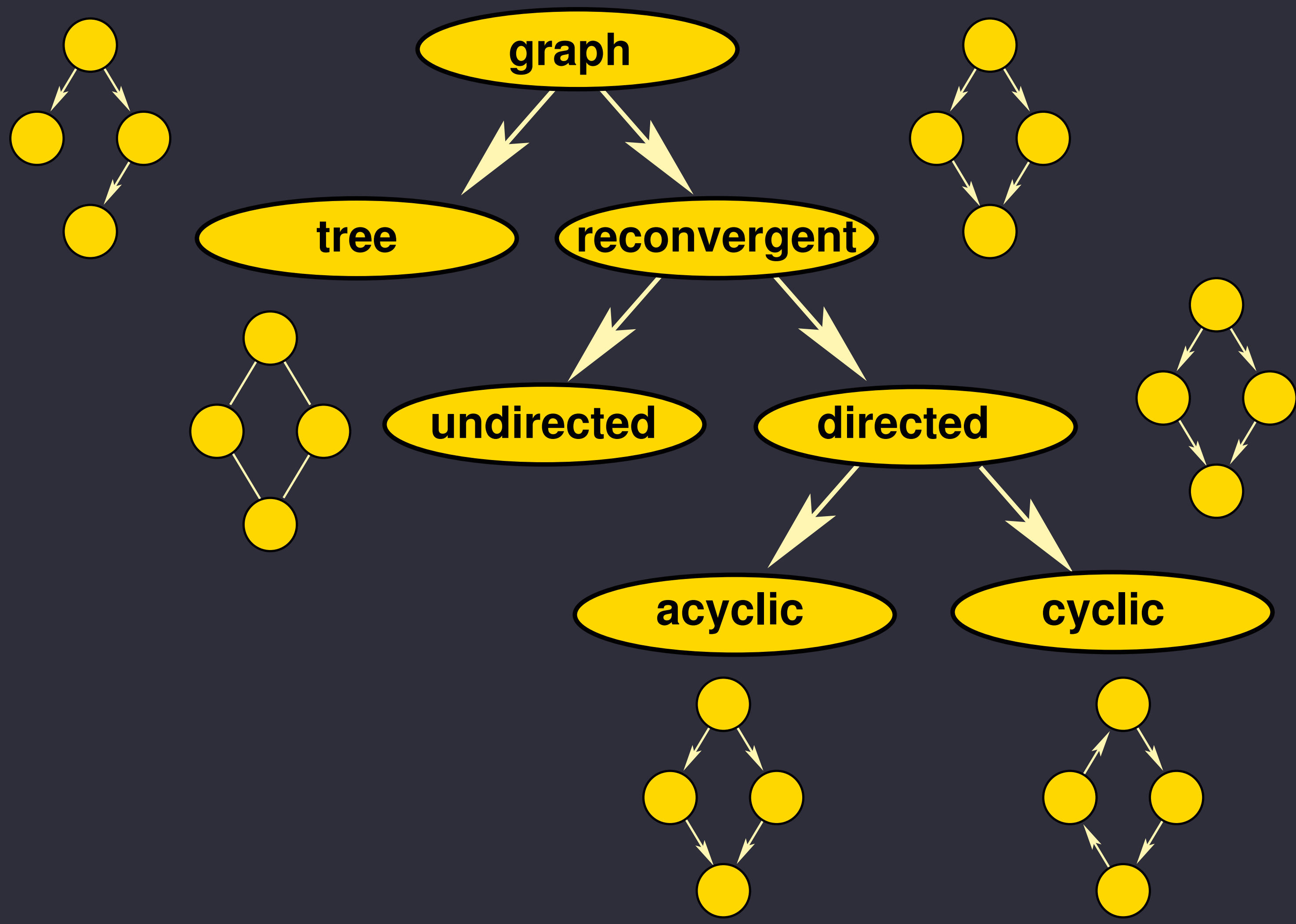
E.g., power consumption or worst-case execution time

Graph definitions



- Set of vertices (V)– usually operations
- Set of edges (E)– directed or undirected relationships on vertex pairs

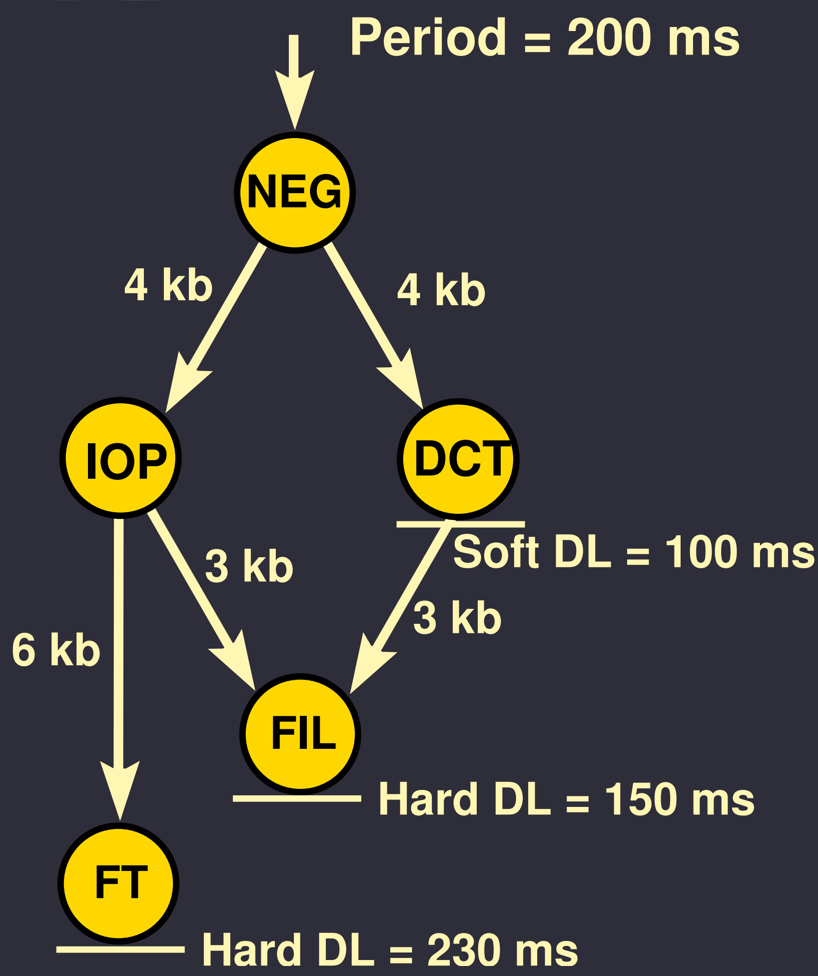
Example graph classifications



Some graph uses

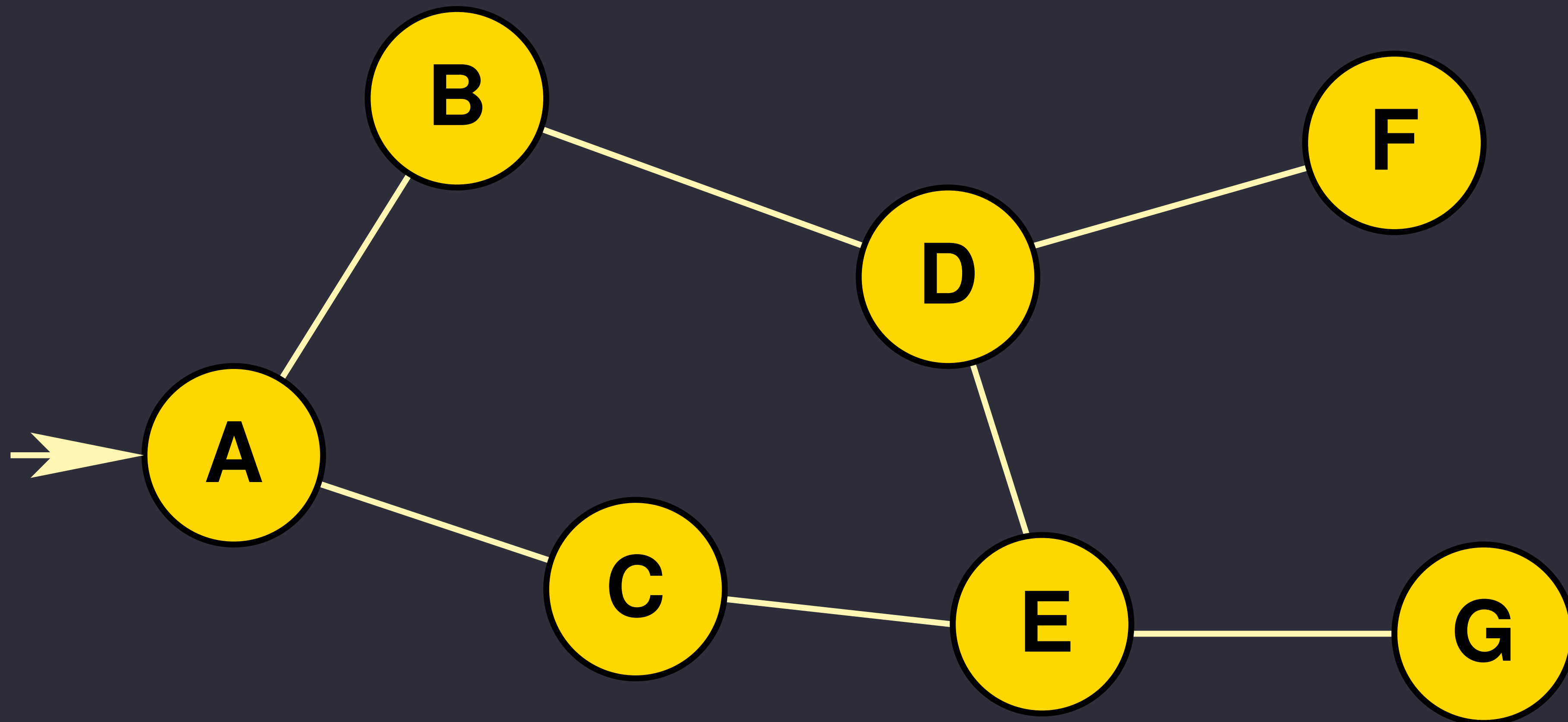
- Problem representations
- Timing constraint specification
- Resource binding
- And many more...

A few basic graph algorithms



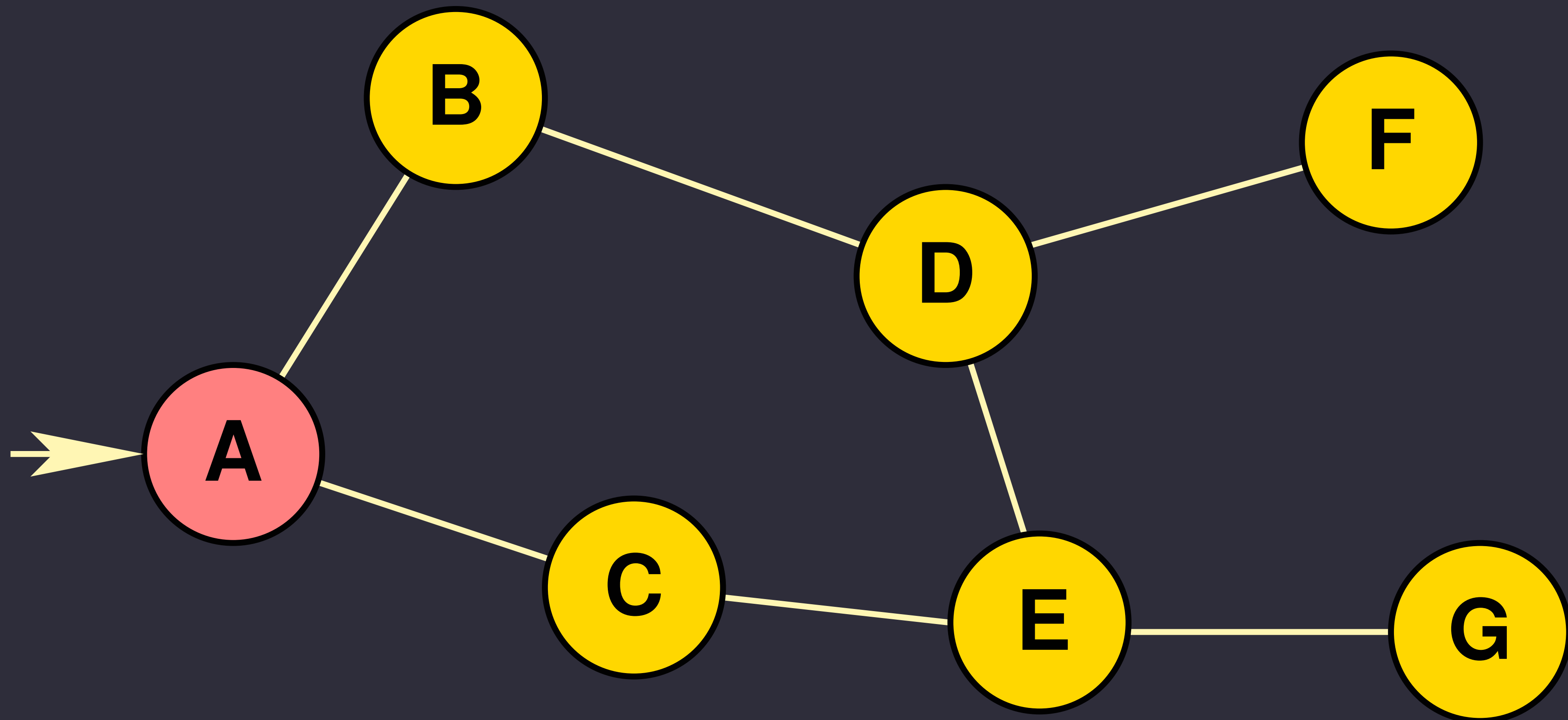
- Depth-first search (DFS)
- Breadth-first search (BFS)
- Topological sort
- Minimal spanning tree (MST)

Depth-first search (DFS) – Pre-order for trees



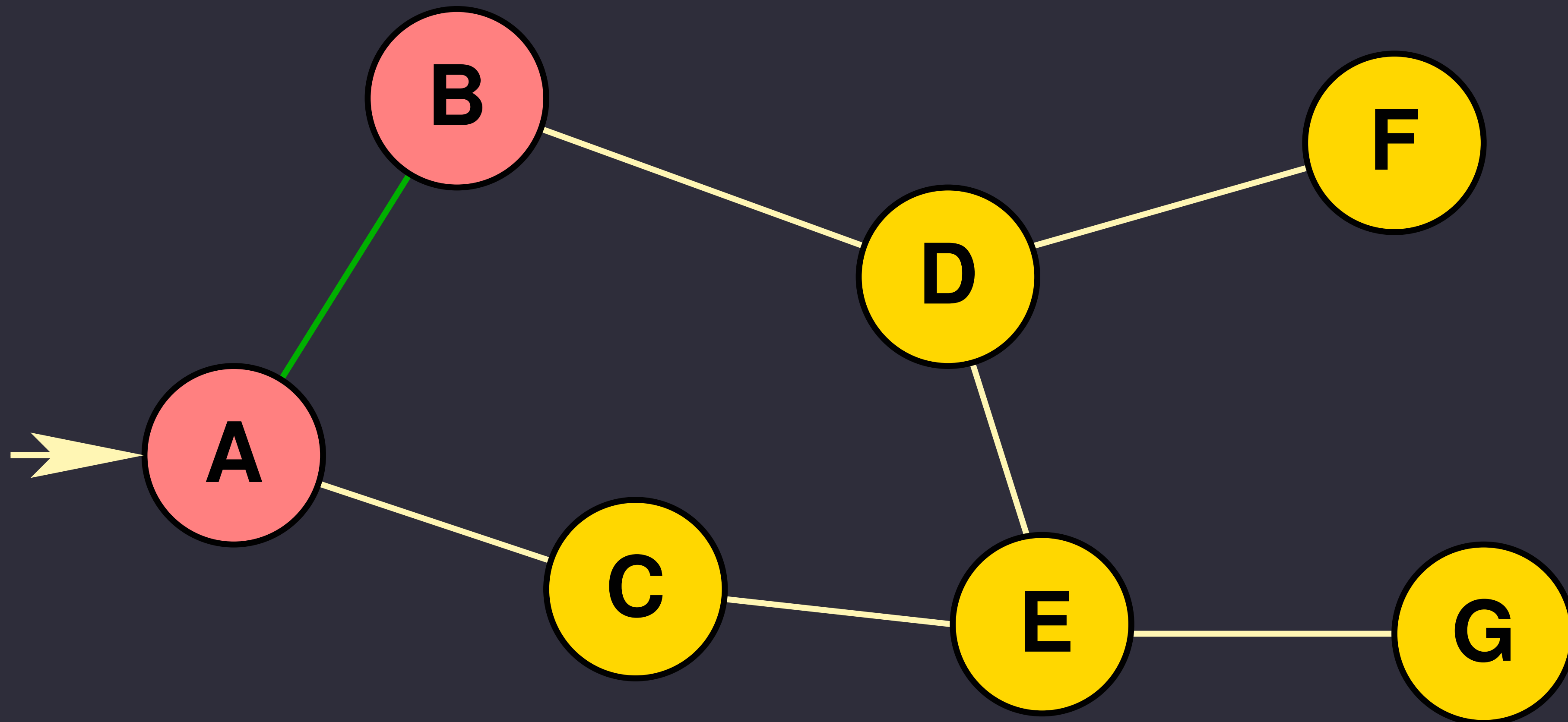
$$\mathcal{O}\left(|V| + |E|\right)$$

Depth-first search (DFS) – Pre-order for trees



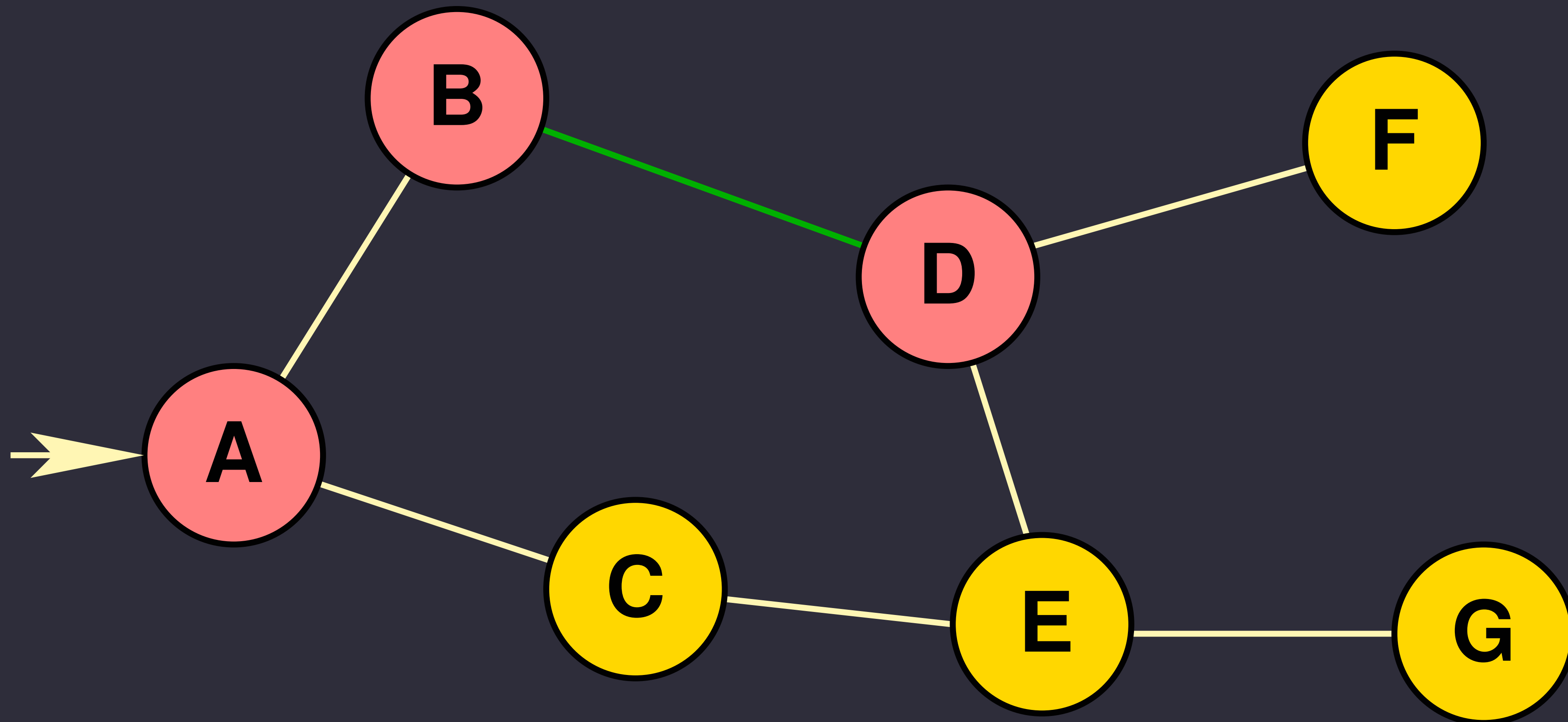
$$\mathcal{O}\left(|V| + |E|\right)$$

Depth-first search (DFS) – Pre-order for trees



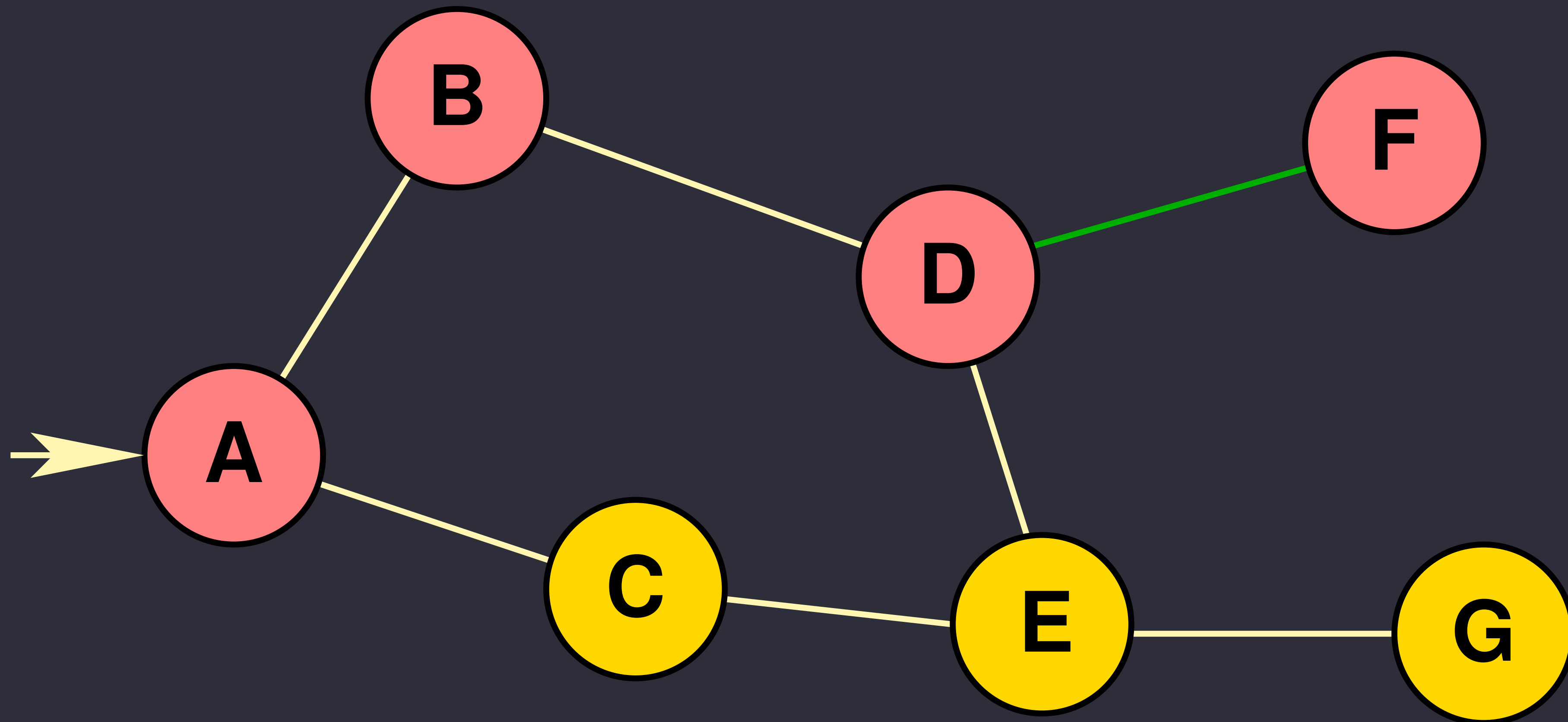
$$\mathcal{O}\left(|V| + |E|\right)$$

Depth-first search (DFS) – Pre-order for trees



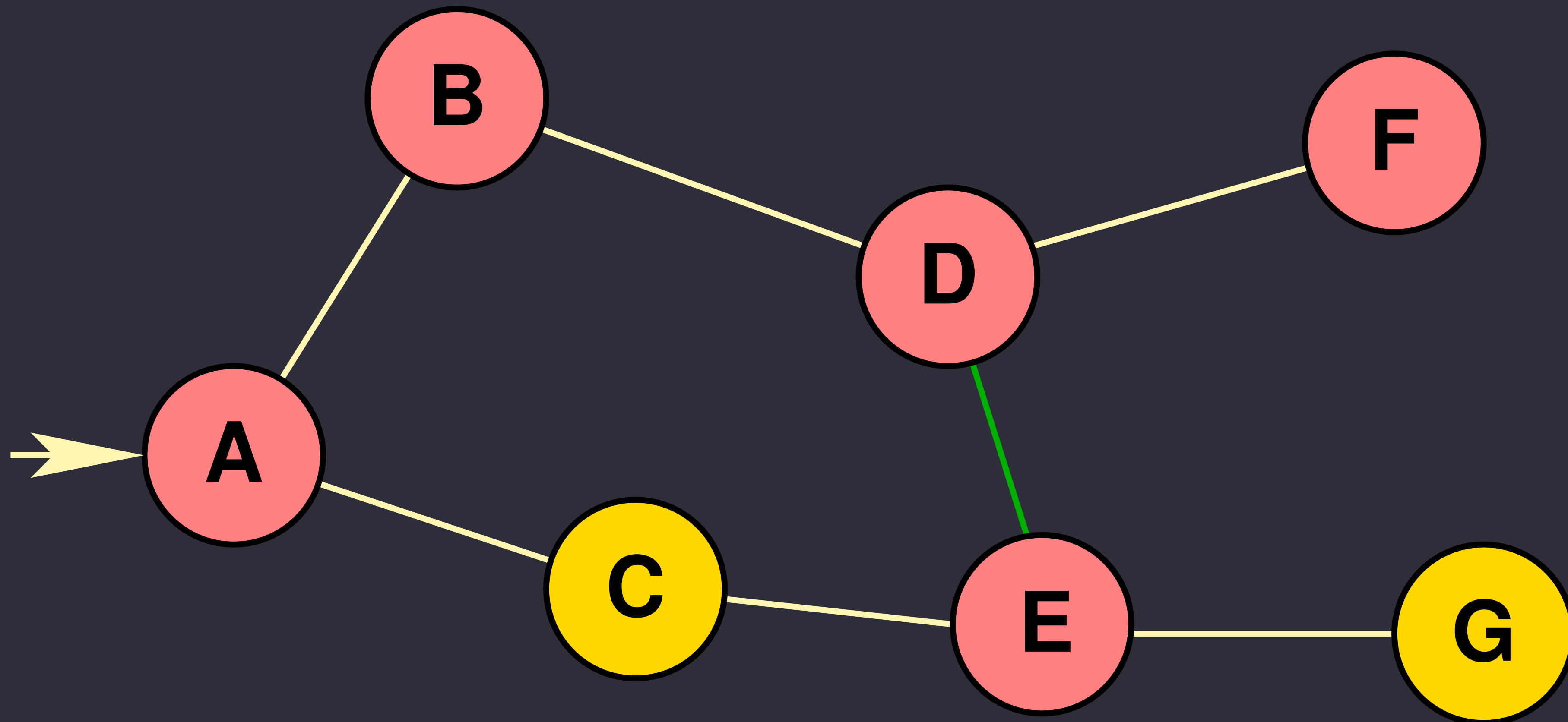
$$\mathcal{O}\left(|V| + |E|\right)$$

Depth-first search (DFS) – Pre-order for trees



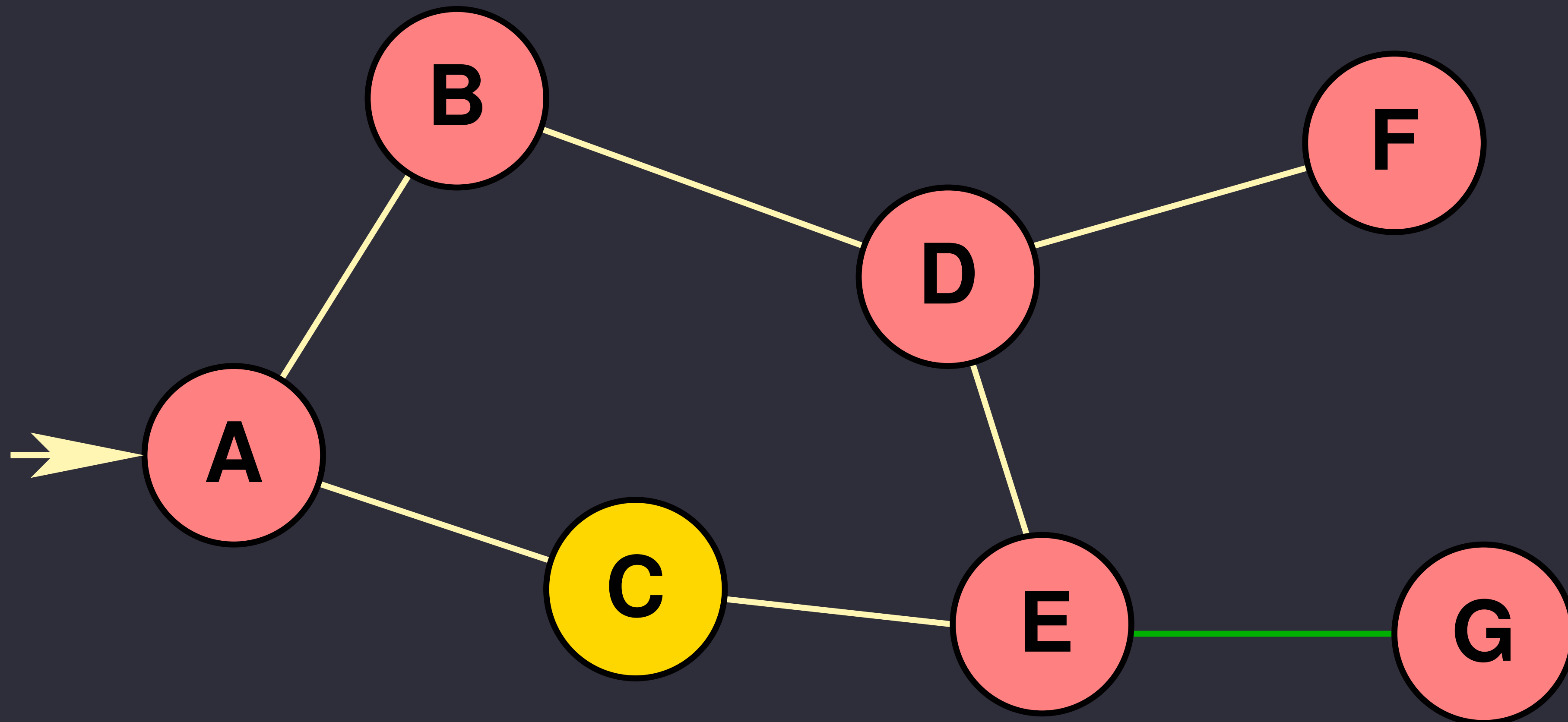
$$\mathcal{O}\left(|V| + |E|\right)$$

Depth-first search (DFS) – Pre-order for trees



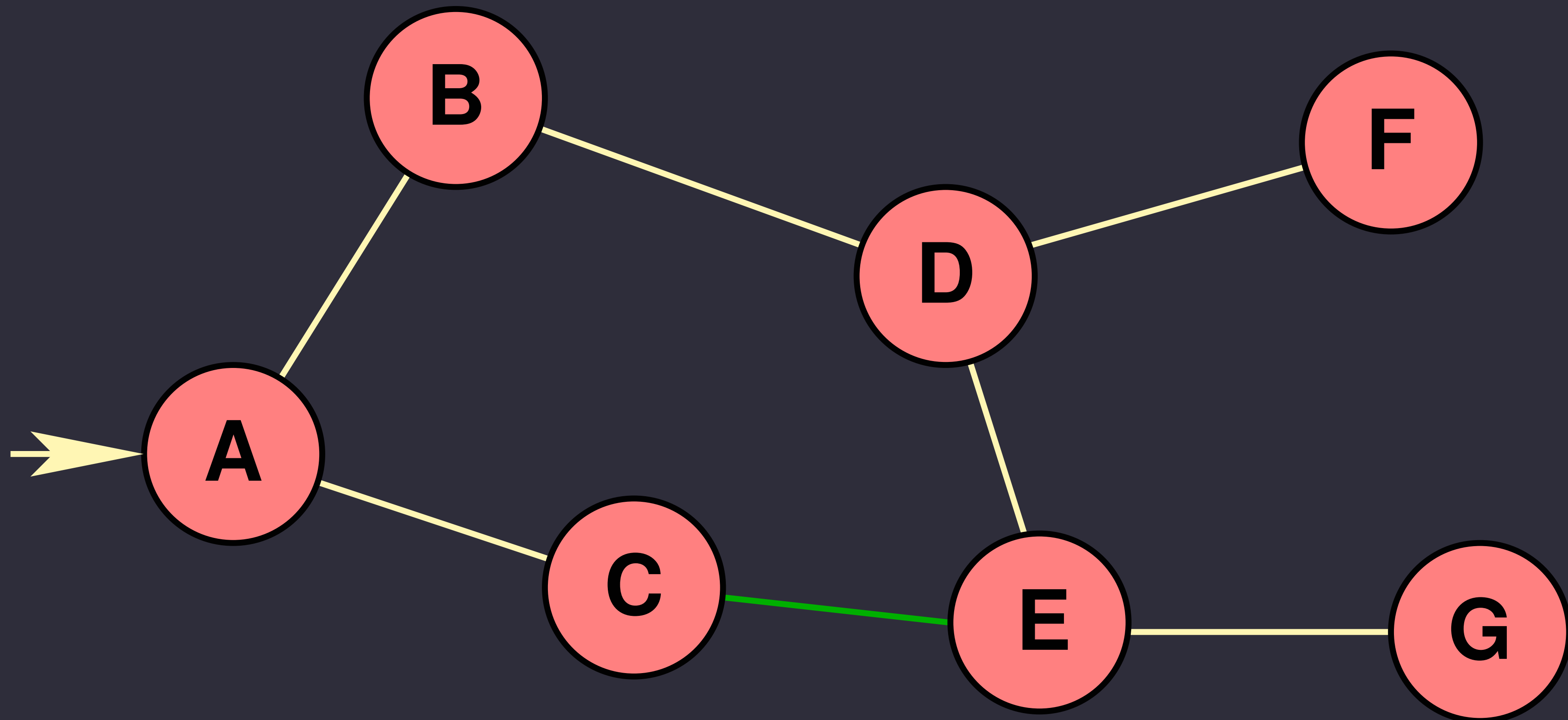
$$\mathcal{O}\left(|V|+|E|\right)$$

Depth-first search (DFS) – Pre-order for trees



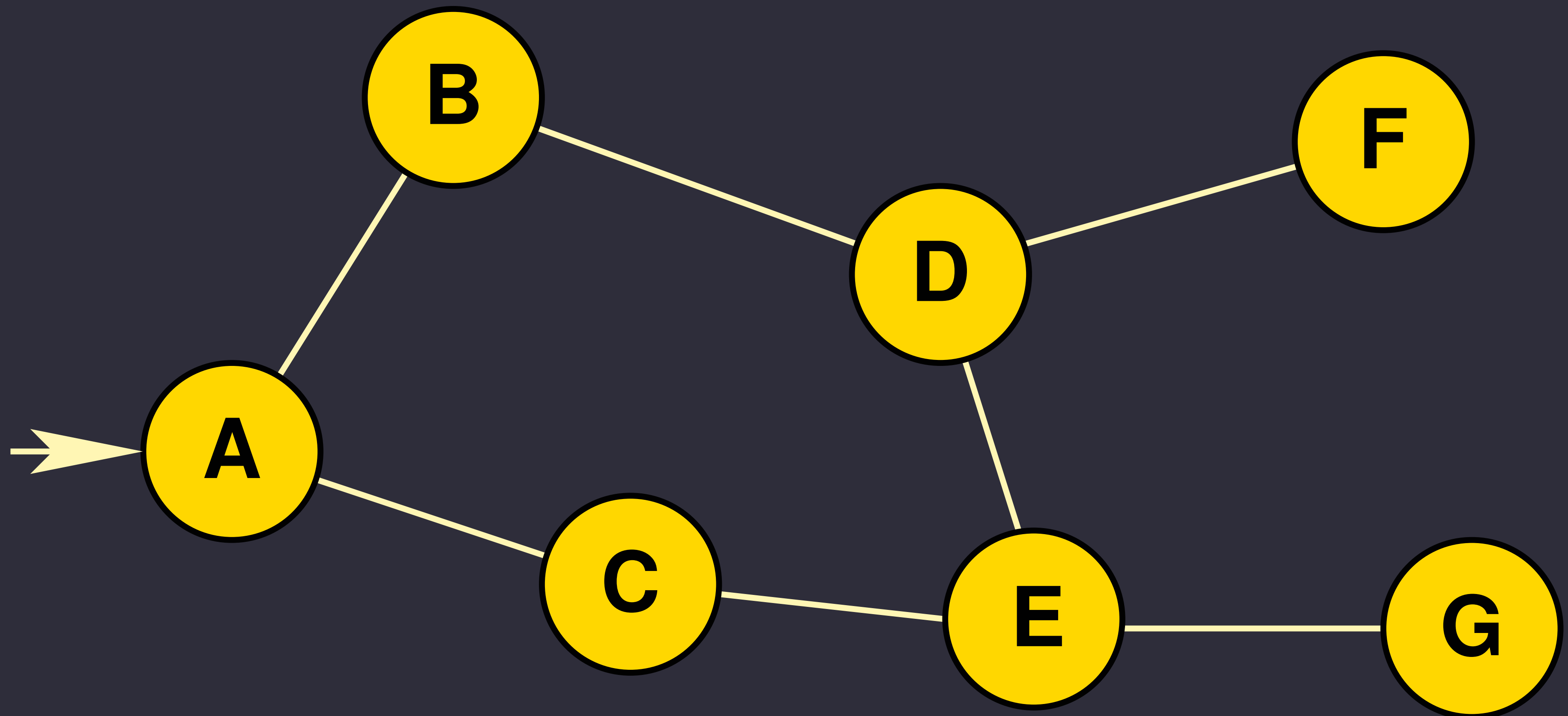
$$\mathcal{O}\left(|V| + |E|\right)$$

Depth-first search (DFS) – Pre-order for trees



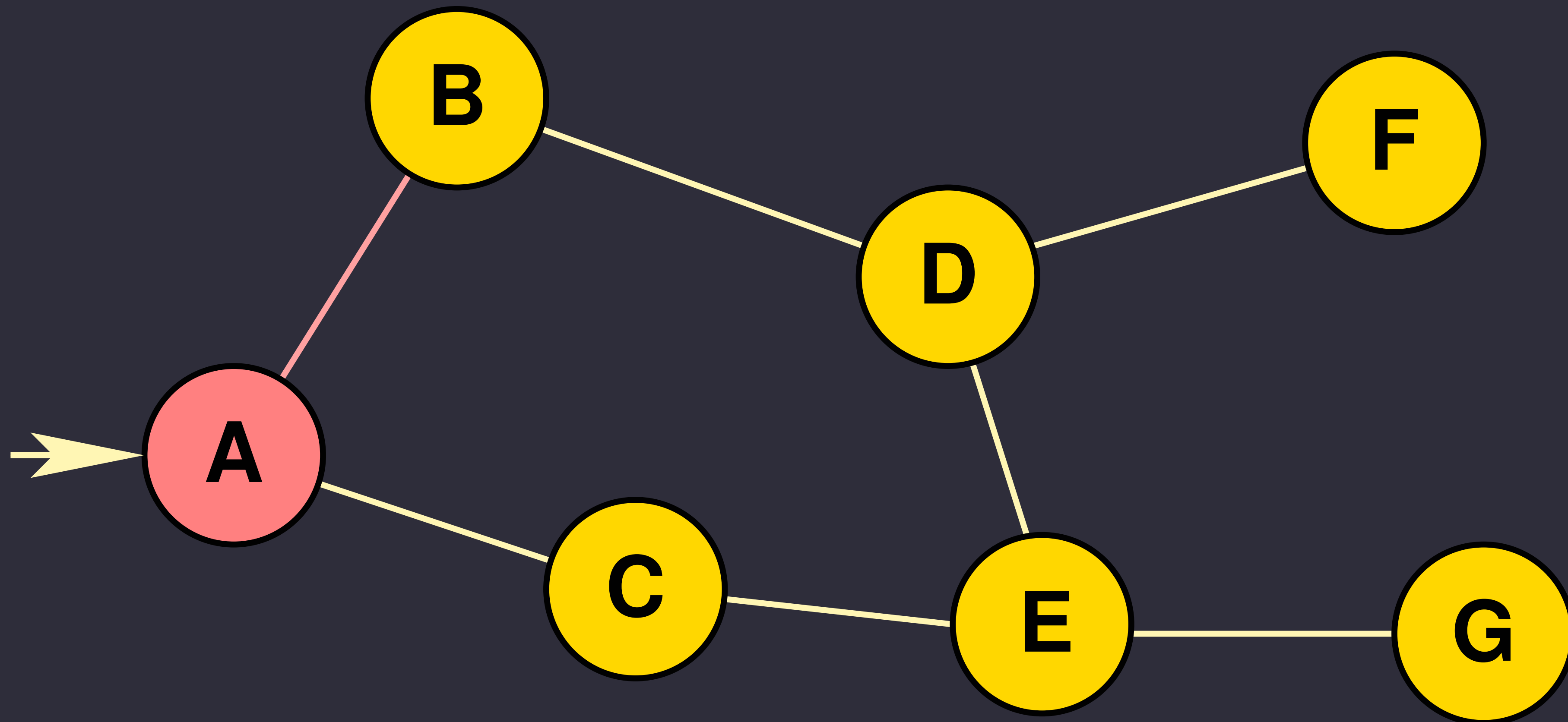
$$\mathcal{O}\left(|V| + |E|\right)$$

Breadth-first search (BFS) – Pre-order for trees



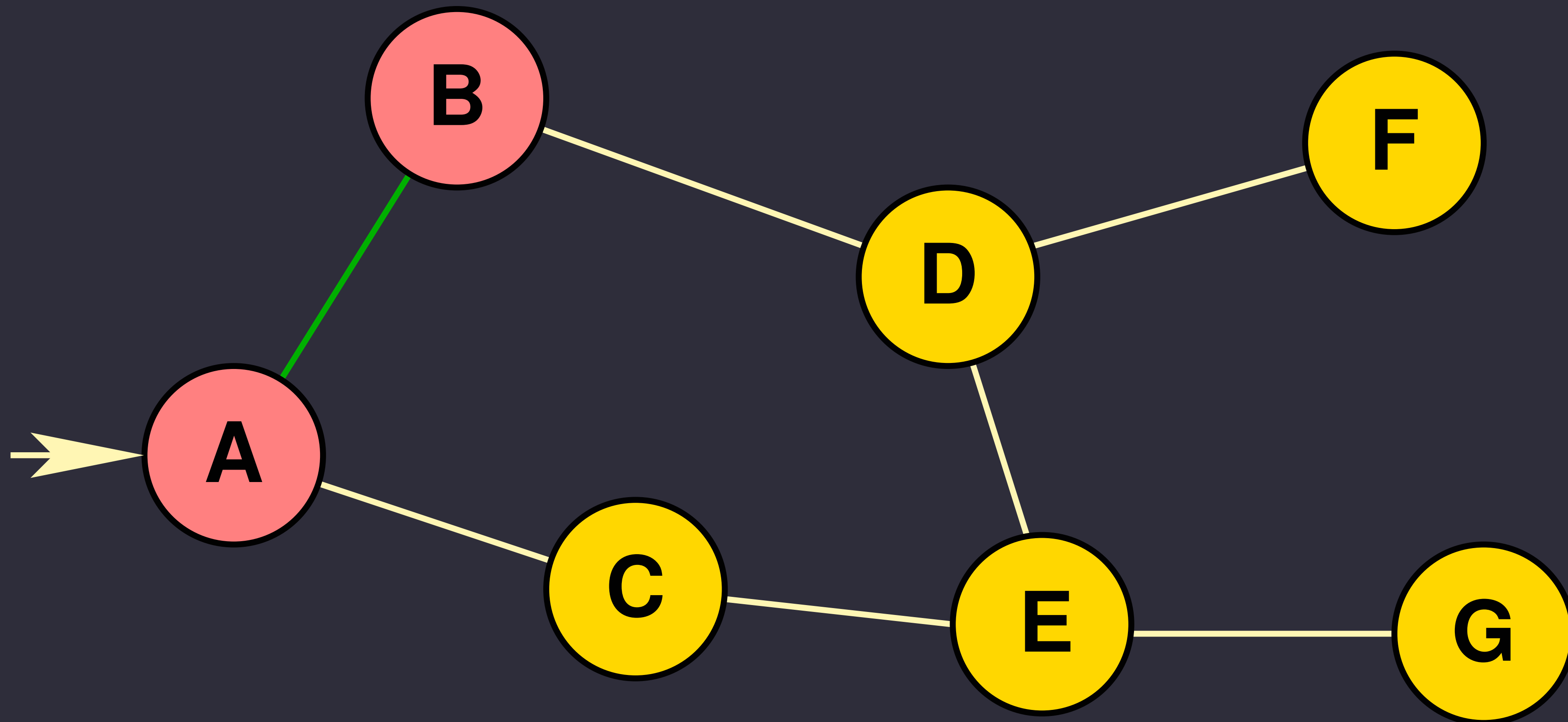
$$\mathcal{O}\left(|V|\right)$$

Breadth-first search (BFS) – Pre-order for trees



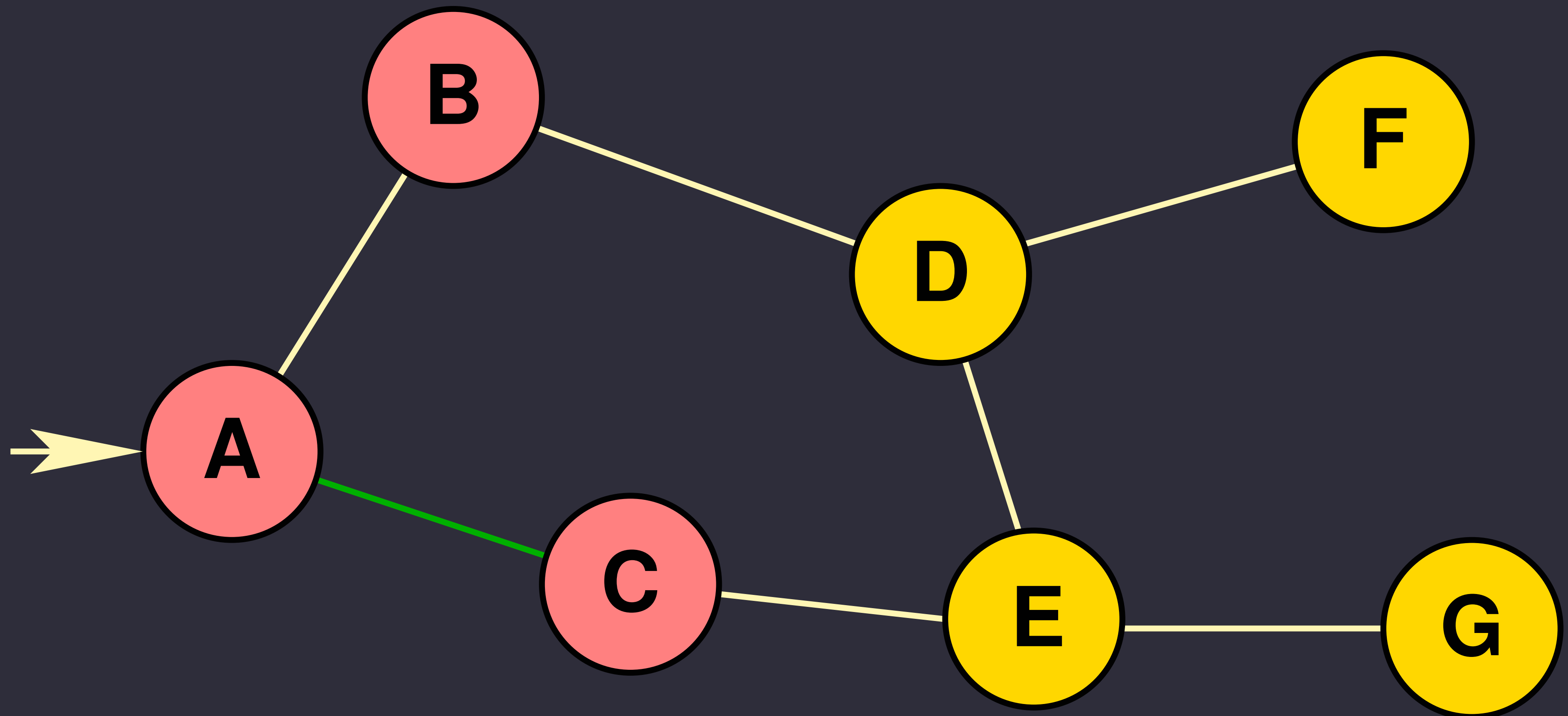
$$\mathcal{O}\left(|V|\right)$$

Breadth-first search (BFS) – Pre-order for trees



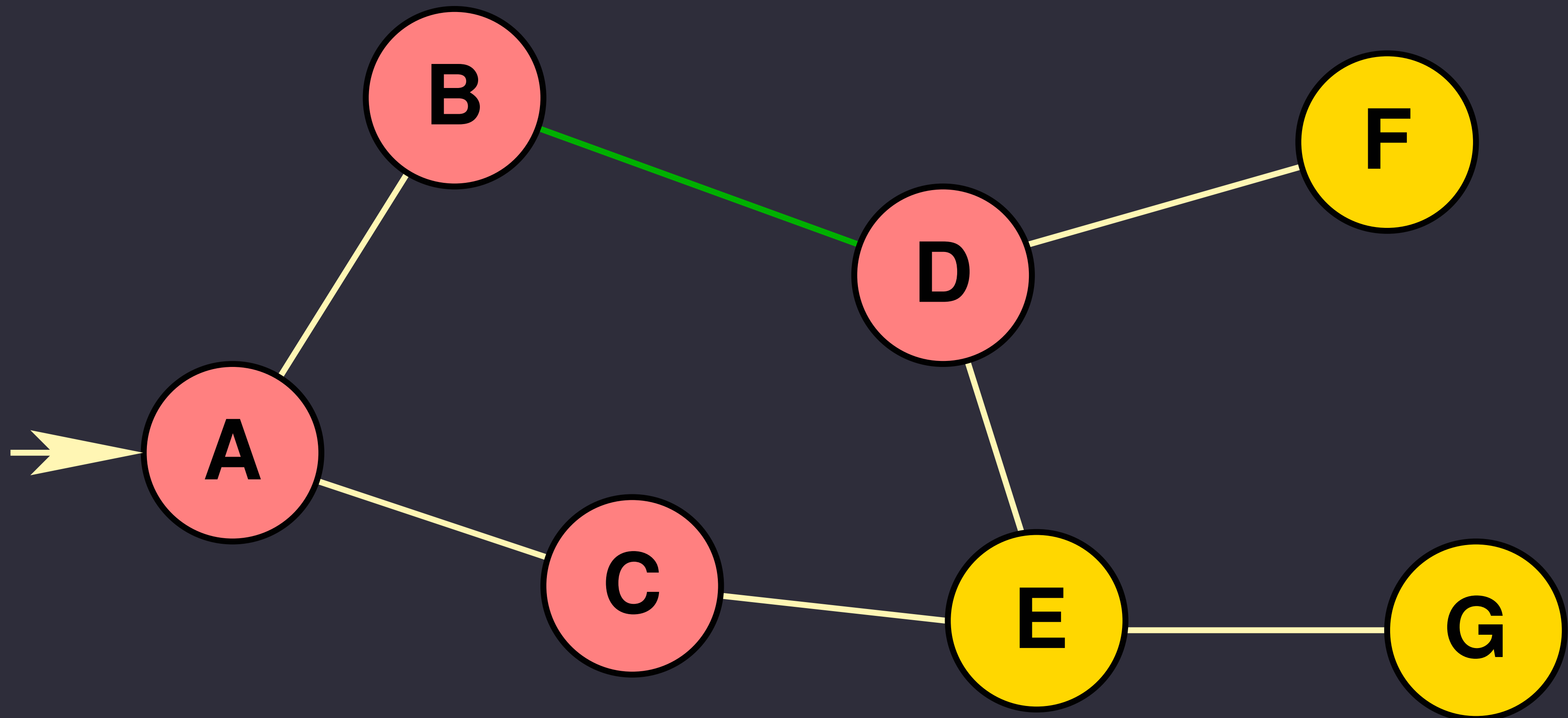
$$\mathcal{O}\left(|V|\right)$$

Breadth-first search (BFS) – Pre-order for trees



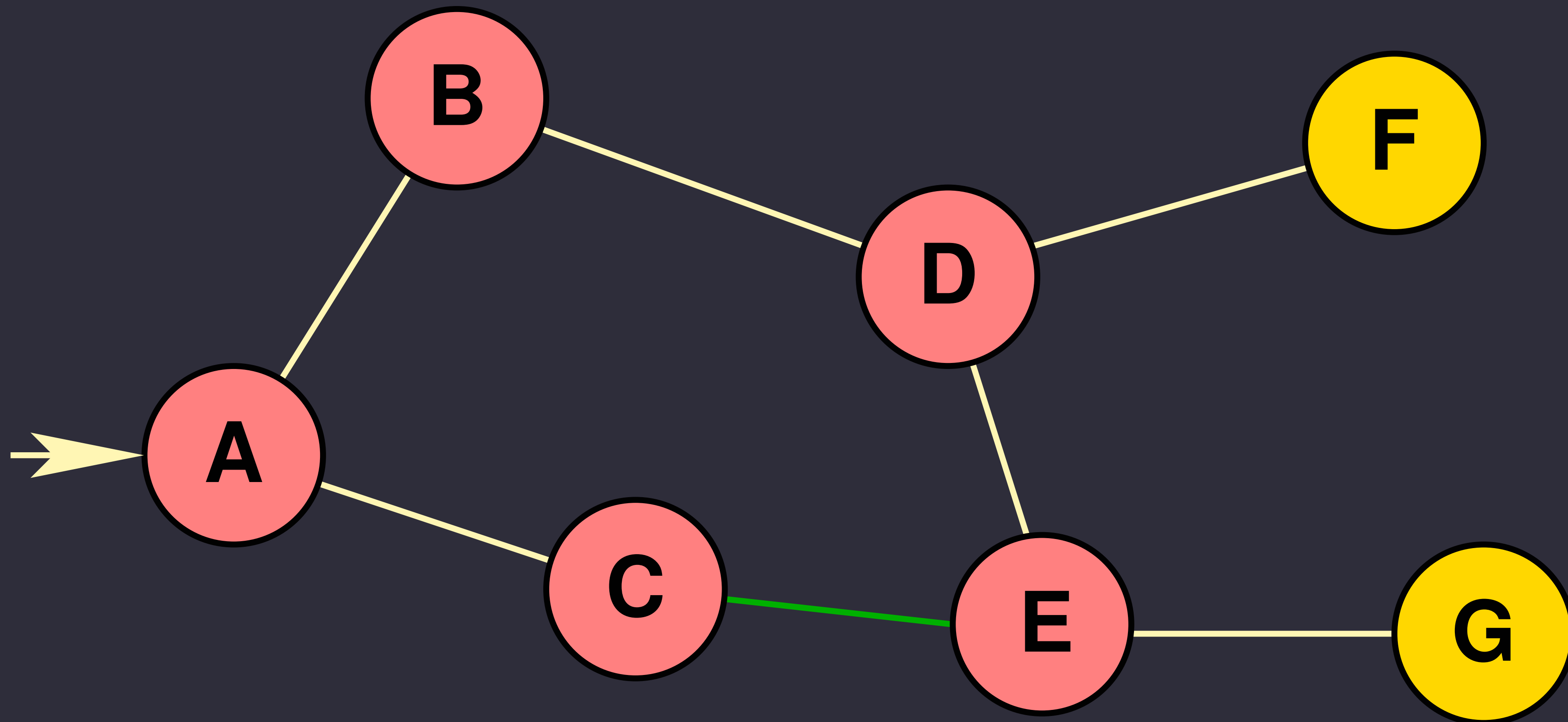
$$\mathcal{O}\left(|V|\right)$$

Breadth-first search (BFS) – Pre-order for trees



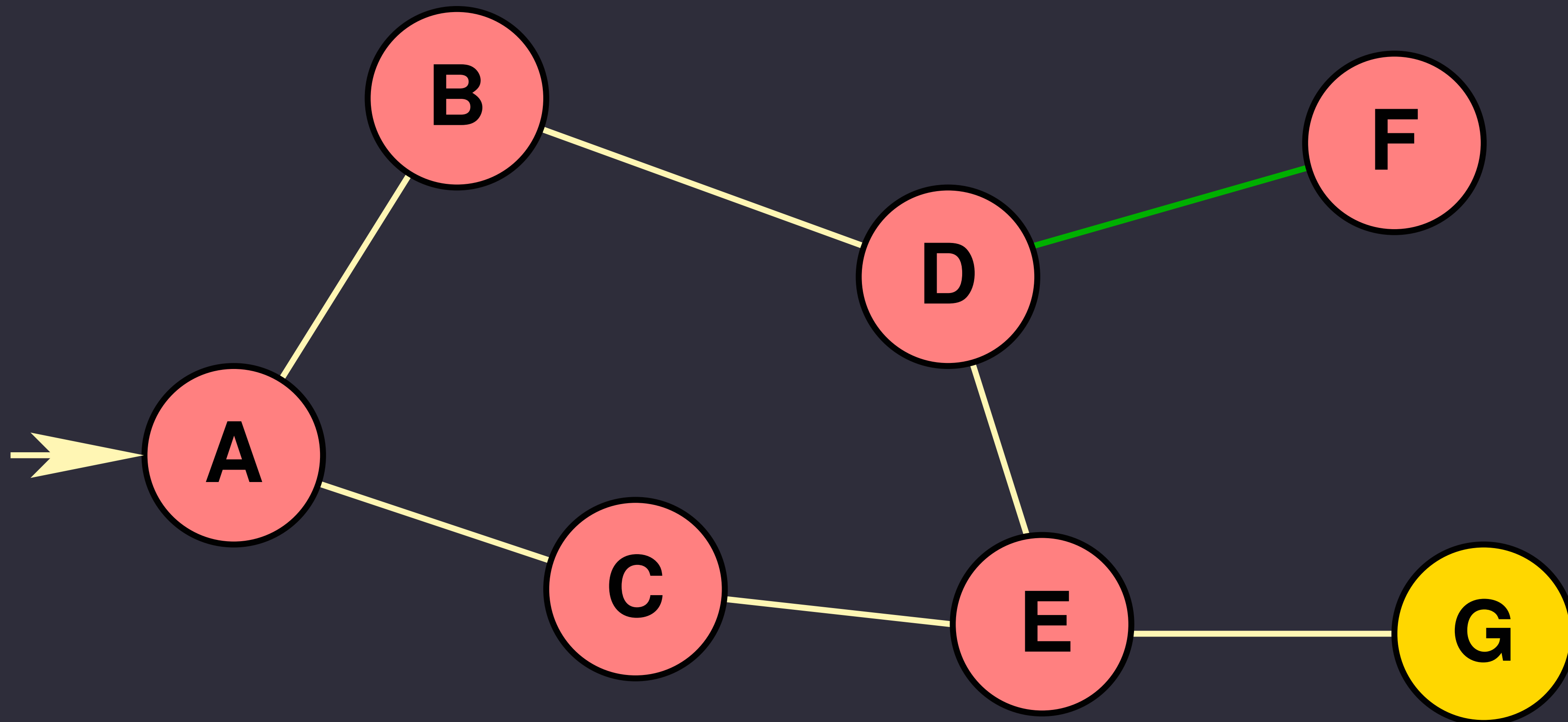
$$\mathcal{O}\left(|V|\right)$$

Breadth-first search (BFS) – Pre-order for trees



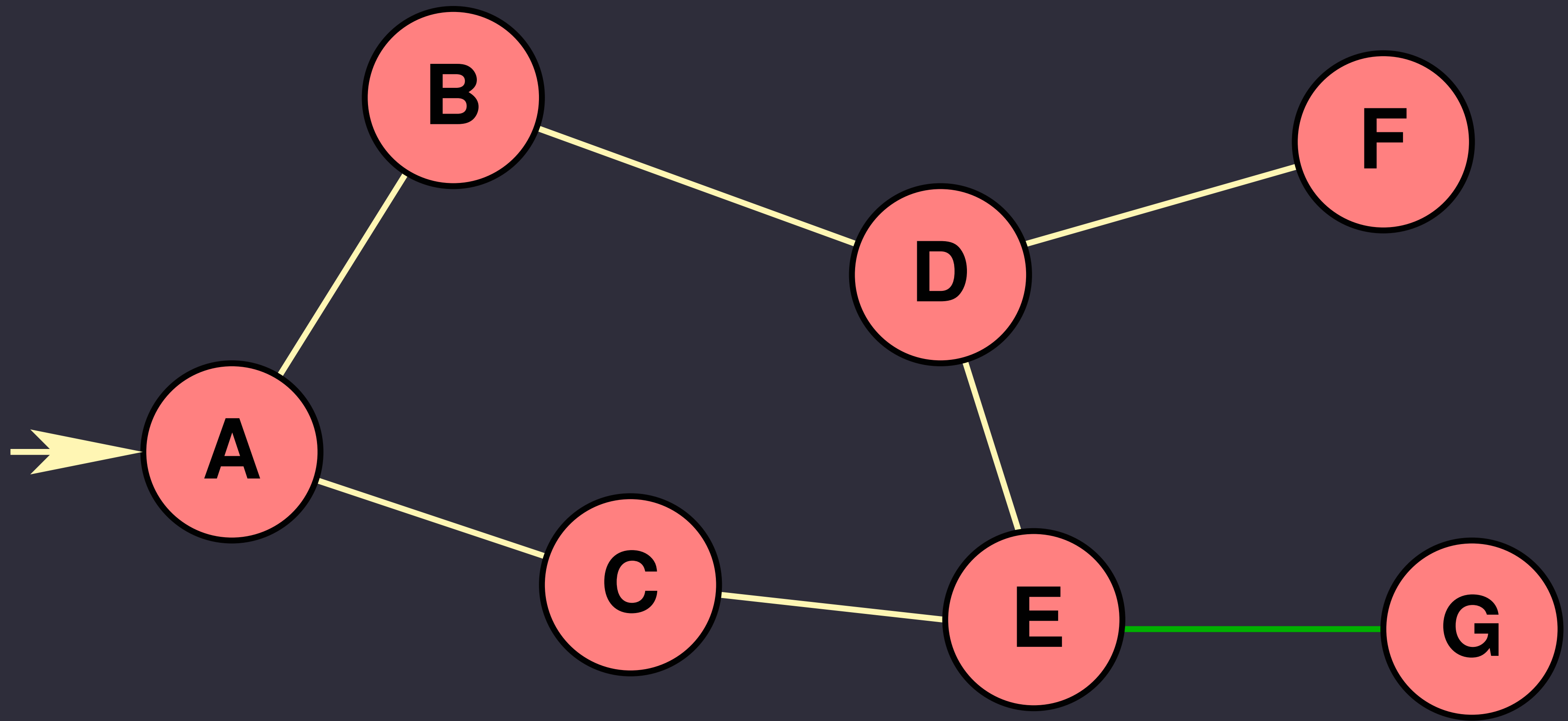
$$\mathcal{O}\left(|V|\right)$$

Breadth-first search (BFS) – Pre-order for trees



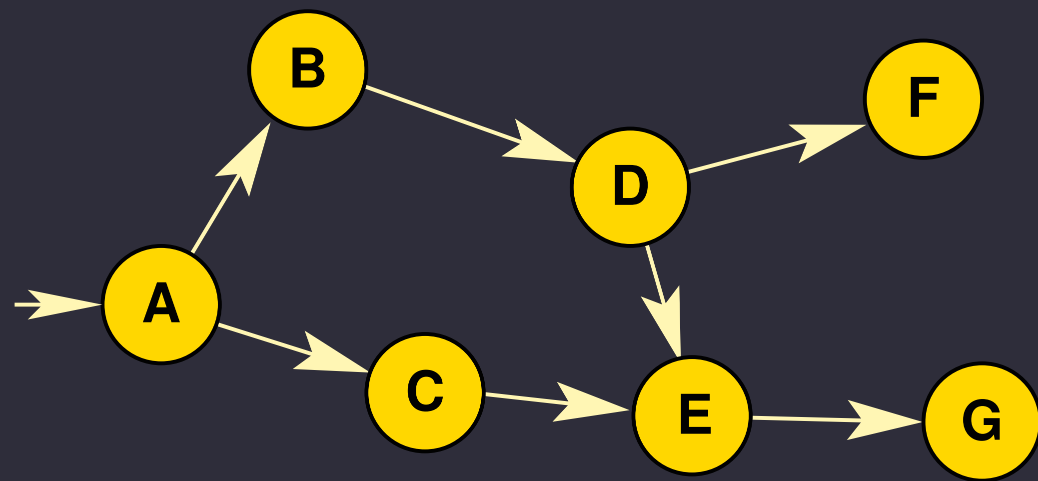
$$\mathcal{O}\left(|V|\right)$$

Breadth-first search (BFS) – Pre-order for trees



$$\mathcal{O}\left(|V|\right)$$

Topological sort

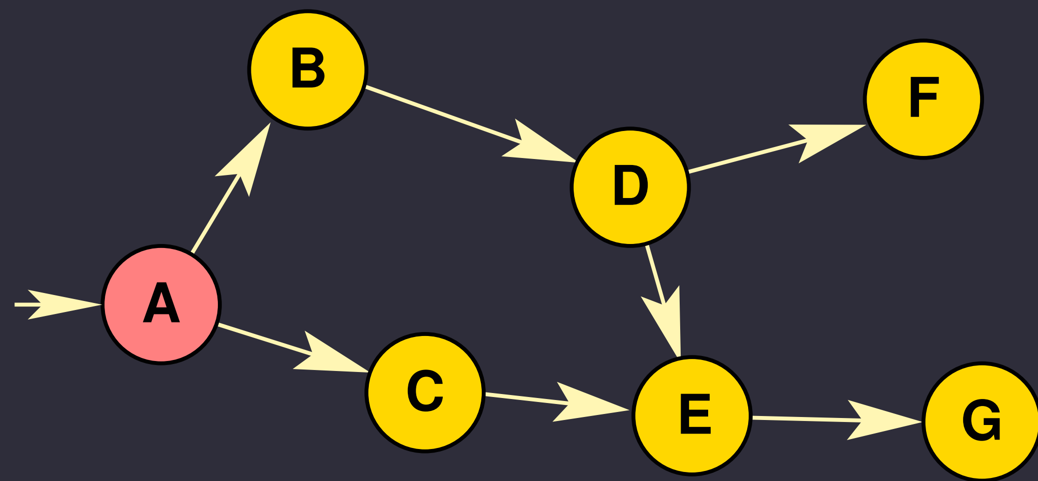


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

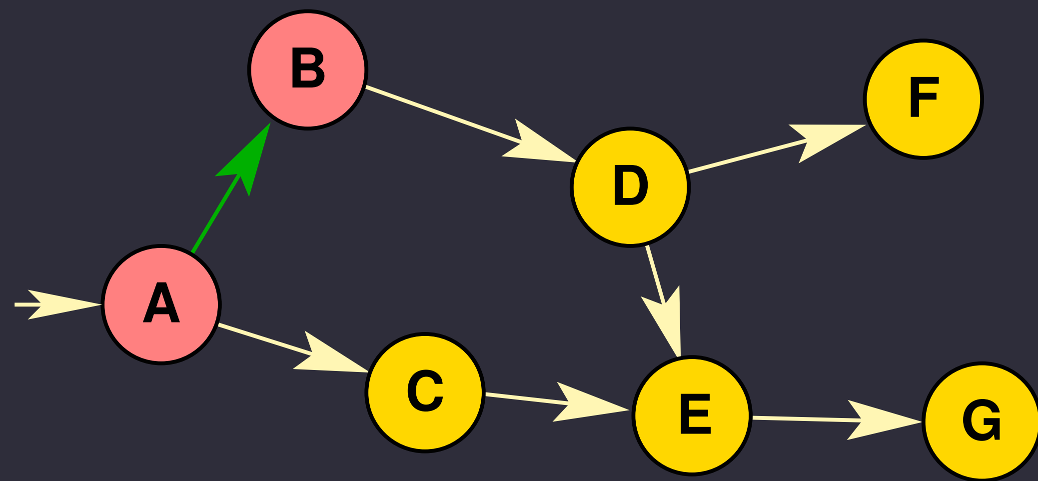


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

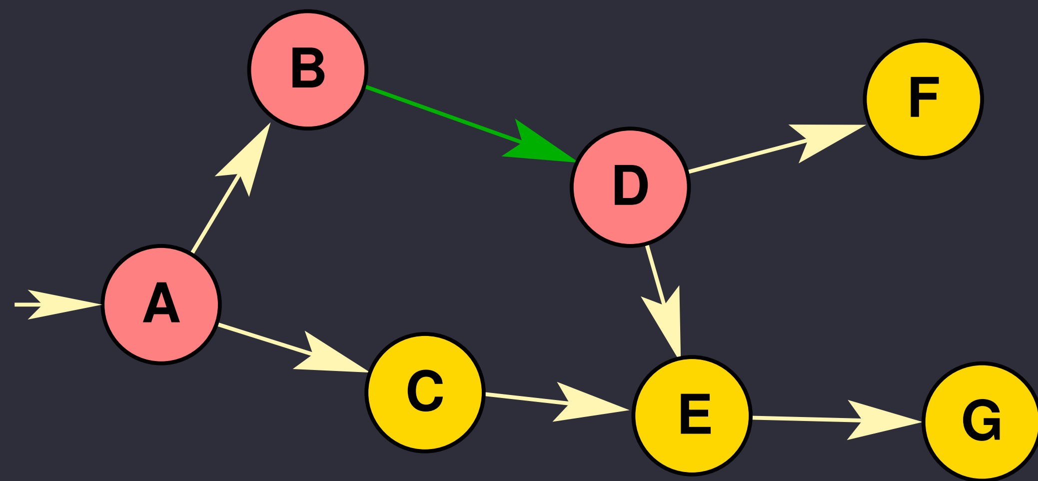


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

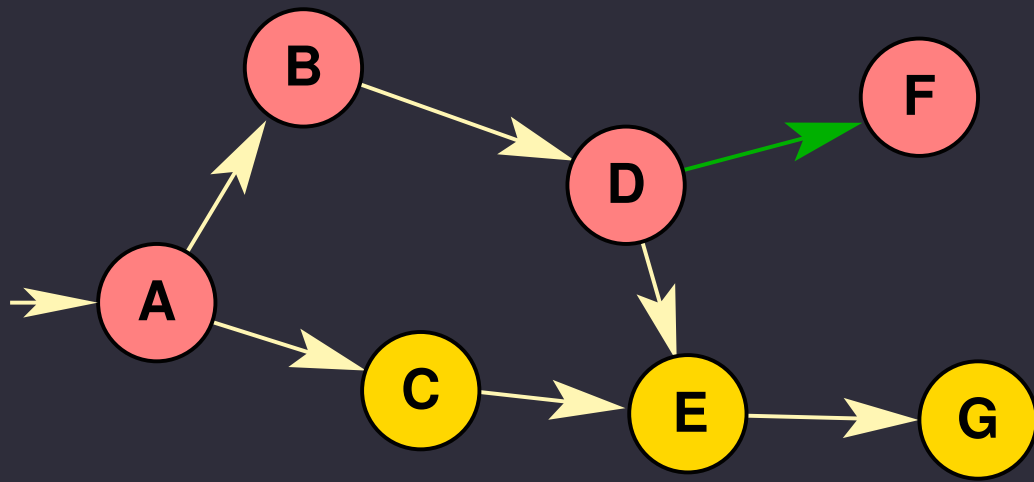


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

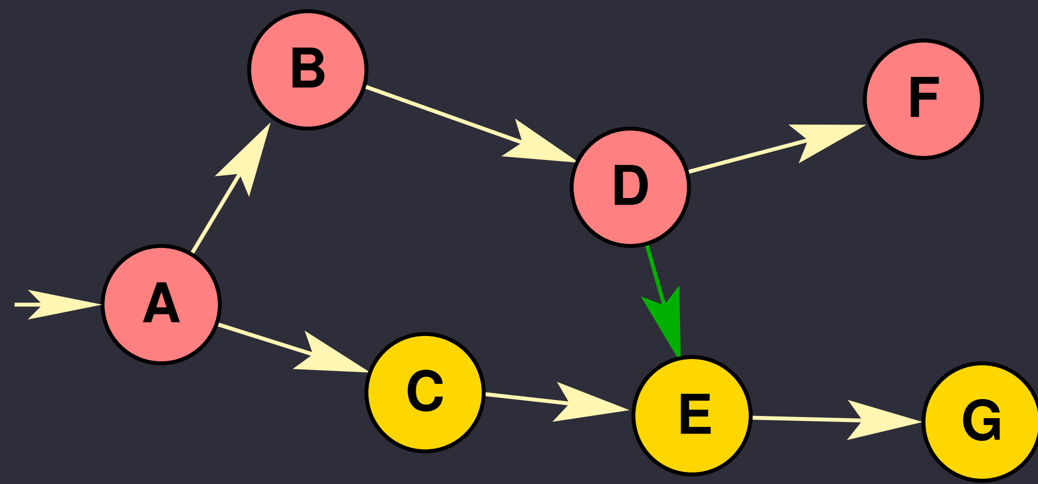


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

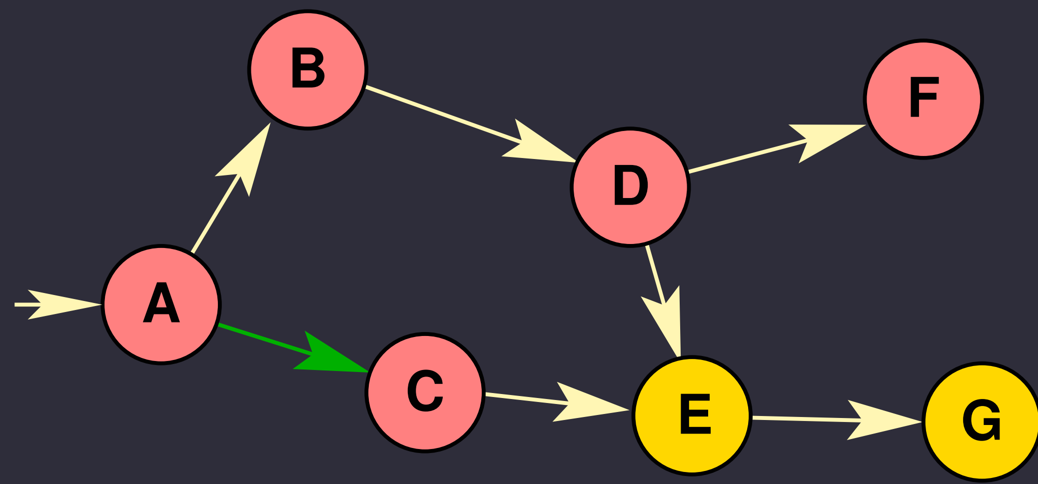


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

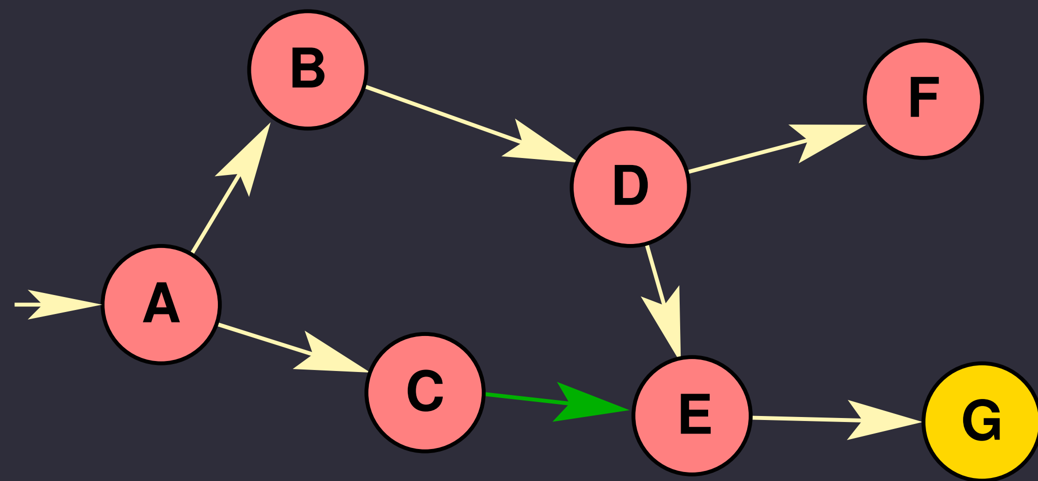


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

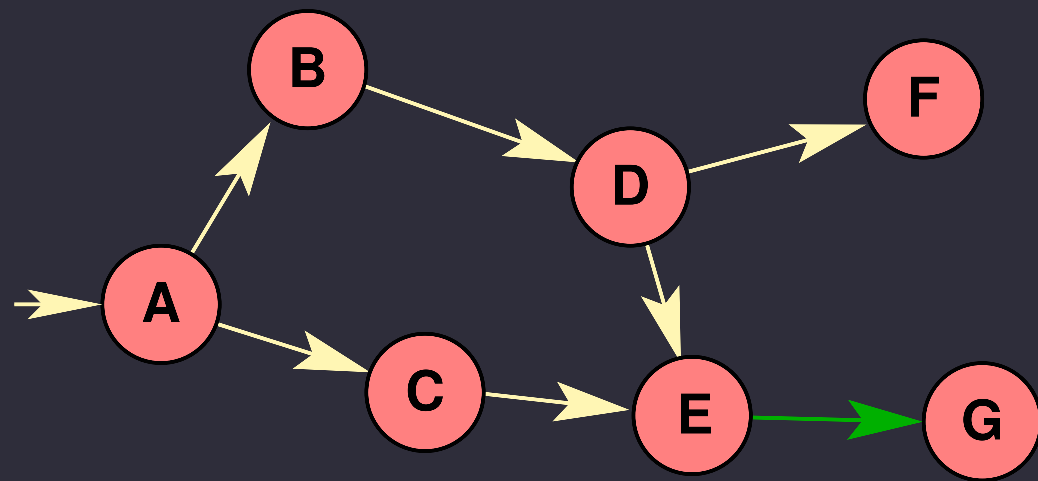


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Topological sort

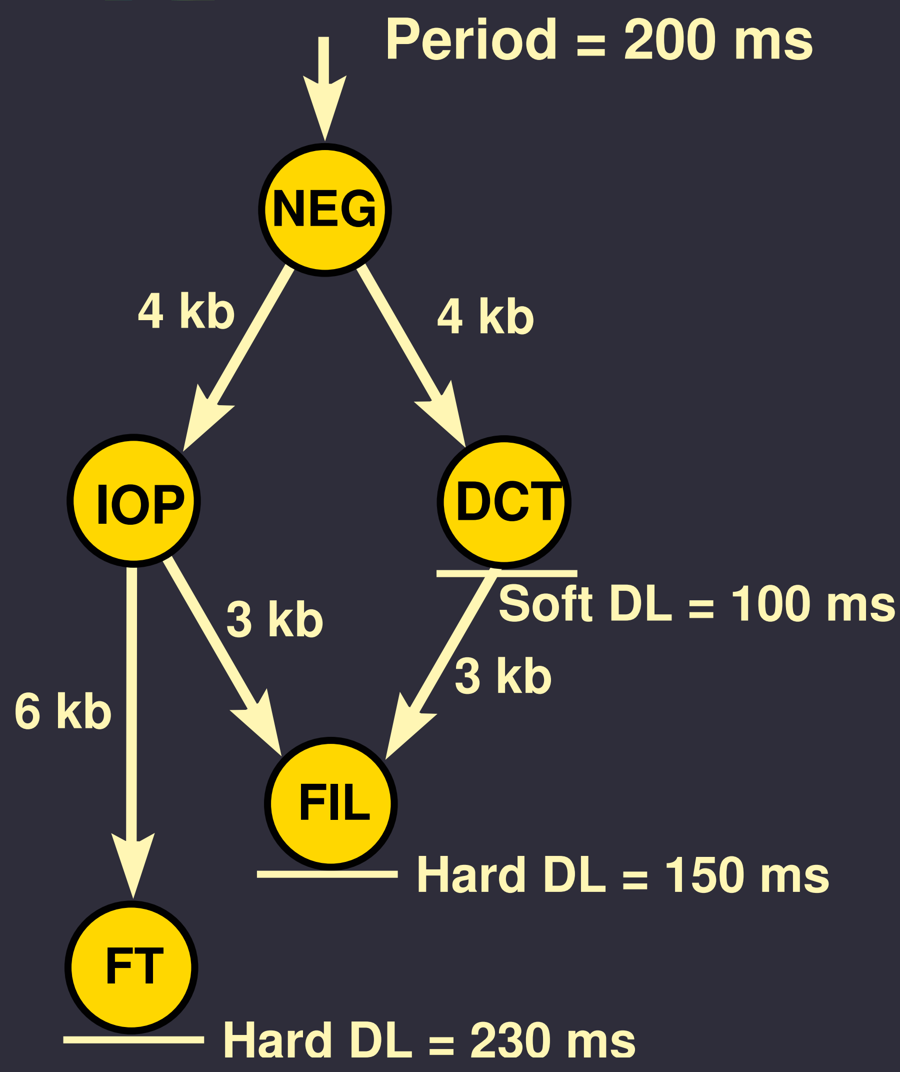


Static timing analysis of data-dependent real-time systems

- Earliest finish time (EFT)
- Earliest start time (EST)
- Latest finish time (LFT)
- Latest start time (LST)

$$\mathcal{O}\left(|V| + |E|\right)$$

Definition: Deadline violation



Cost functions

- Mapping of real-time system design problem solution instance to cost value
- I.e., allows price, or hard deadline violation, of a particular multi-processor implementation to be determined

Back to real-time problem taxonomy: Jagged edges

- Some things dramatically complicate real-time scheduling
- These are horrific, especially when combined
  - Data dependencies
  - Unpredictability
  - Distributed systems
- These are irksome
  - Heterogeneous processors
  - Preemption

Central areas of real-time study

- Allocation, assignment and **scheduling**
- Operating systems and **scheduling**
- Distributed systems and **scheduling**
- **Scheduling is at the core or real-time systems study**

Allocation, assignment and scheduling

How does one best

- Analyze problem instance specifications
  - E.g., worst-case task execution time
- Select (and build) hardware components
- Select and produce software
- Decide which processor will be used for each task
- Determine the time(s) at which all tasks will execute

Allocation, assignment and scheduling

- In order to efficiently and (when possible) optimally minimize
  - Price, power consumption, soft deadline violations
- Under hard timing constraints
- Providing guarantees whenever possible
- For all the different classes of real-time problem classes

This is what I did for a Ph.D.

Operating systems and scheduling

How does one best design operating systems to

- Support sufficient detail in workload specification to allow good control, e.g., over scheduling, without increasing design error rate
- Design operating system schedulers to support real-time constraints?
- Support predictable costs for task and OS service execution

Distributed systems and scheduling

How does one best dynamically control

- The assignment of tasks to processing nodes...
- ... and their schedules

for systems in which computation nodes may be separated by vast distances such that

- Task deadline violations are bounded (when possible)...
- ... and minimized when no bounds are possible

This is part of what Professor Dinda did for a Ph.D.

The value of formality: Optimization and costs

- The design of a real-time system is fundamentally a cost optimization problem
- Minimize costs under constraints while meeting functionality requirements
  - Slight abuse of notation here, functionality requirements are actually just constraints
- Why view problem in this manner?
- Without having a concrete definition of the problem
  - How is one to know if an answer is correct?
  - More subtly, how is one to know if an answer is optimal?

Optimization

Thinking of a design problem in terms of optimization gives design team members objective criterion by which to evaluate the impact of a design change on quality.

- Still need to do a lot of hacking
- Know whether its taking you in a good direction

Summary

- Real-time systems taxonomy and overview
- Definitions
- Importance of problem formulation

Reading assignment (for next class)

- J. W. S. Liu, *Real-Time Systems*. Prentice-Hall, Englewood Cliffs, NJ, 2000
- Chapter 2
- Start on Chapter 3