CS 5112
Algorithms for Applications

Prof Alex Conway

Why study algorithms?

Why study algorithms?

System Performance

Why study algorithms?

System Performance

Why care about performance?

What software properties are more important than performance?

What software properties are more important than performance?

- Compatibility
- Functionality
- Reliability
- Correctness
- Maintainability
- Robustness
- Clarity
- Debuggability
- Modularity
- Portability
- Testability
- Usability

What software properties are more important than performance?

- Compatibility
- Functionality
- Reliability
- Correctness
- Maintainability
- Robustness
- Clarity
- Debuggability
- Modularity
- Portability
- Testability
- Usability

If we're willing to sacrifice performance for these properties, why care about performance?

Choosies

Choosies (and the theme of these slides) are due to Prof. Charles Leisterson, MIT

Choosies

Bottle of water

Choosies (and the theme of these slides) are due to Prof. Charles Leisterson, MIT

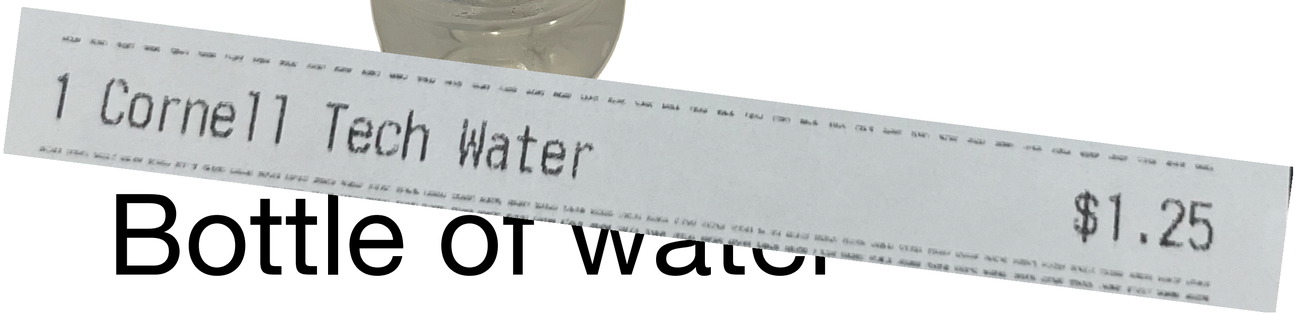
Choosies

Bottle of water

$10

Choosies (and the theme of these slides) are due to Prof. Charles Leisterson, MIT

Choosies



Bottle of water

$10

Choosies (and the theme of these slides) are due to Prof. Charles Leisterson, MIT

When might you choose the water?

When might you choose the water?

Gemini 2.5 Pro

GPT 5

Money can be exchanged for goods and services

We can “buy” all the other properties with performance

A little example

Search over a dataset:

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

Say if I make these choices,
users wait 30 seconds

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

Say if I make these choices, users wait 30 seconds

But python is easy to program, memory-safe

Exhaustive search is easy

Commodity hardware is cheap

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

I can stick with the basic algorithm, but it costs me

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

I can stick with the basic algorithm, but it costs me

C++ is harder to program in and debug

Custom hardware is expensive

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

If I choose a hash table,
latency goes down to <1s

A little example

Search over a dataset:

Algorithm:	Exhaustive	Tree-based	Hash table
Implementation:	Python	Java	C++
Hardware:	Commodity	Enterprise	Custom

If I choose a hash table, latency goes down to <1s

We use the performance we gain from a better algorithm to keep the other good properties

Back in the day…

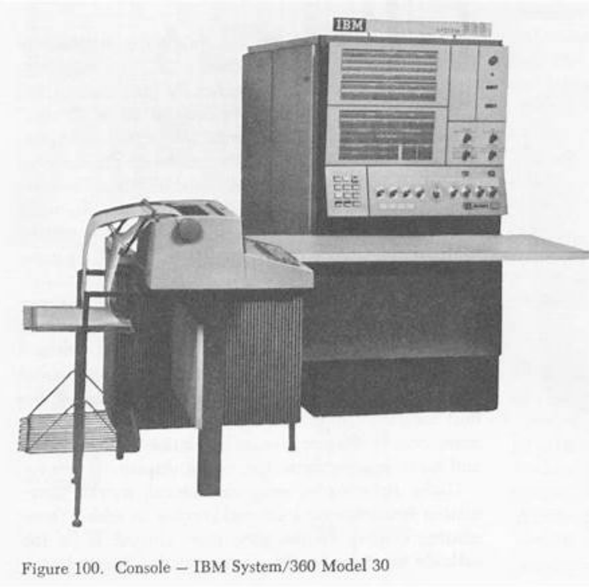


Figure 100. Console — IBM System/360 Model 30

1965: IBM System/360 Model 30

Clock Rate: ~0.034 MHz

Memory: 8 KB – 64 KB core memory

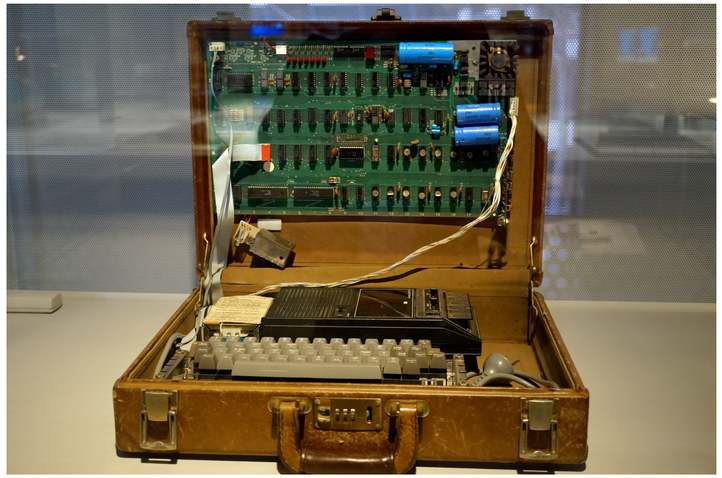
Cost: Around $133,000 (~$1.3M today)

1970: DEC PDP-11

Clock Rate: ~1 MHz

Memory: 8 KB – 124 KB

Cost: Around $20,000 (~$160,000 today)



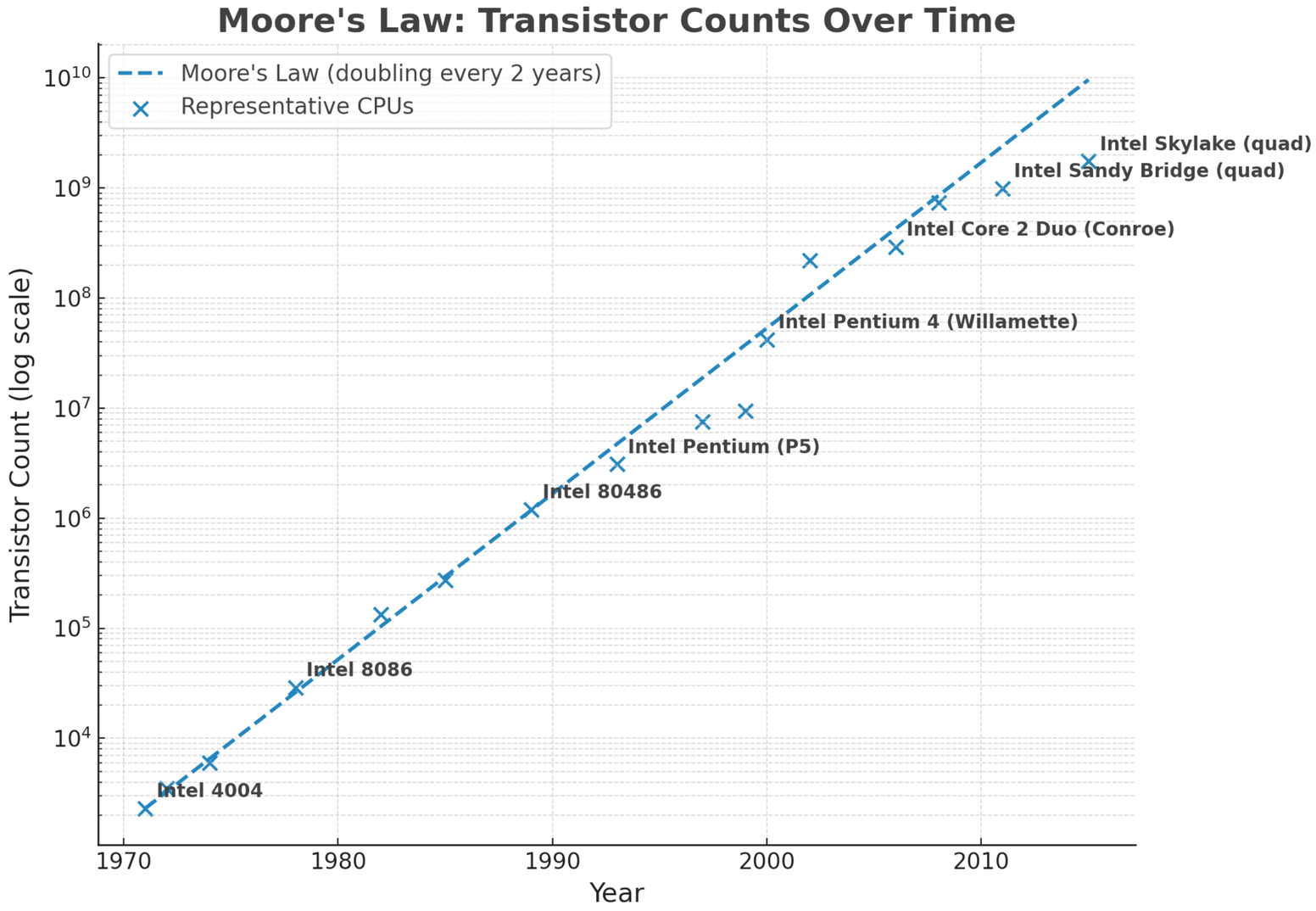
1976: Apple I

Clock Rate: 1.023 MHz

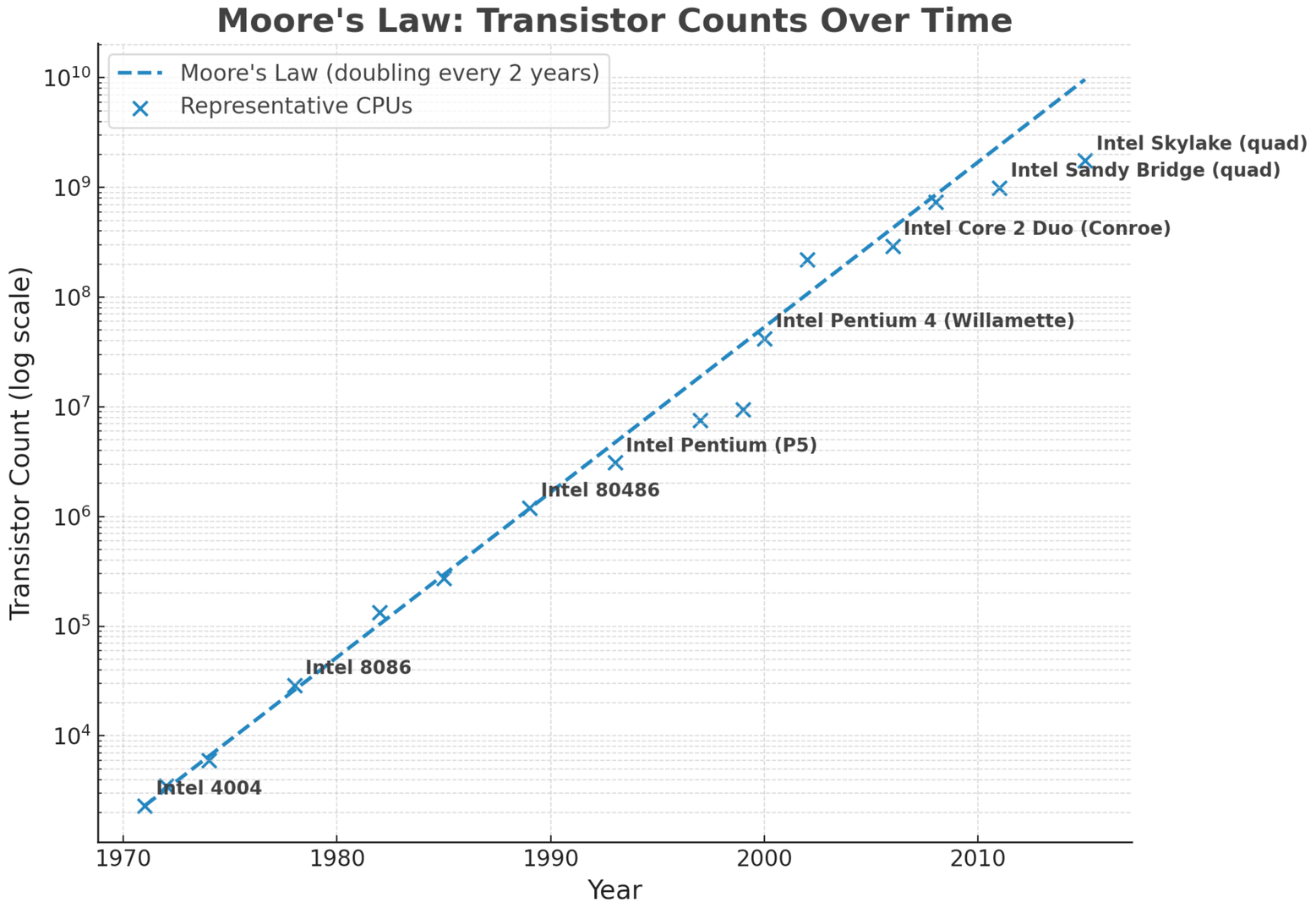
Memory: 4 KB standard RAM

Cost: $666.66 (~$3,500 today)

Early computers had limited hardware, so performance was critical!



Performance isn’t so important if hardware keeps getting faster



We’ve got performance to spend!

Performance isn’t so important if hardware keeps getting faster

SCIENCE & TECHNOLOGY | FEBRUARY 11, 2025

Is Moore's Law really dead?

Penn Engineering's Ben Lee and André DeHon discuss Moore's Law, the observation that the number of transistors on an integrated circuit will double every two years with minimal rise in cost, and reflect on the consequences and opportunities of its possible end.

The Death of Moore's Law: What it means and what might fill the gap going forward

Written By: Audrey Woods

Technology Quarterly | The relentless innovation machine

The end of Moore's law will not slow the pace of change

Semiconductors are likely to continue their transformational role

SCIENCE & TECHNOLOGY | FEBRUARY 11, 2025

Is Moore's Law really dead?

Penn Engineering's Ben Lee and André DeHon discuss Moore's Law, the observation that the number of transistors on an integrated circuit will double every two years with minimal rise in cost, and reflect on the consequences and opportunities of its possible end.

The Death of Moore's Law: What it means and what might fill the gap going forward

Written By: Audrey Woods

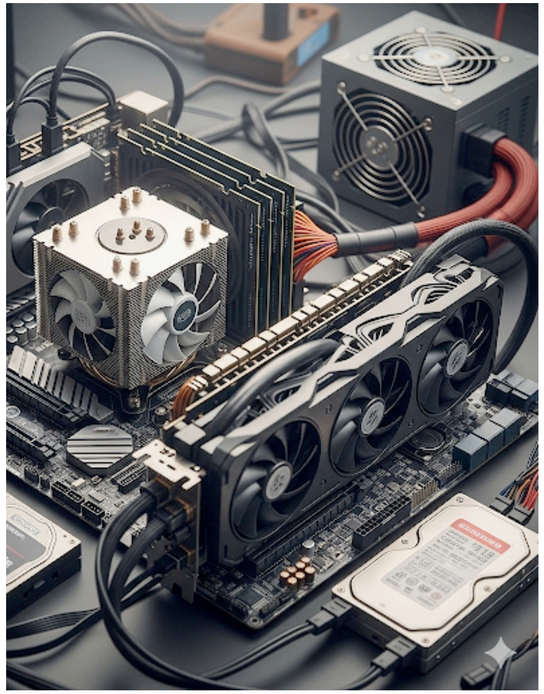
Technology Quarterly | The relentless innovation machine

The end of Moore's law will not slow the pace of change

Semiconductors are likely to continue their transformational role

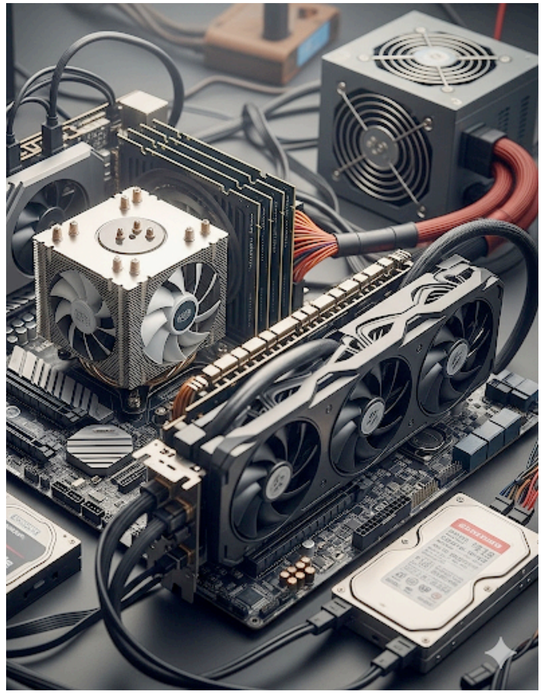
Performance matters again!

Ways to achieve performance

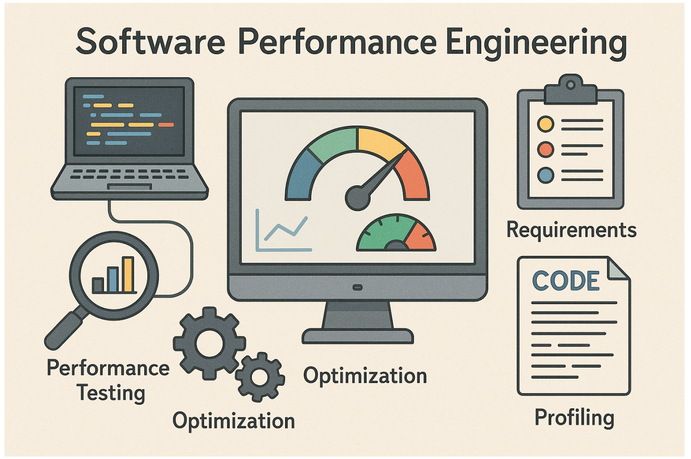


Hardware

Ways to achieve performance

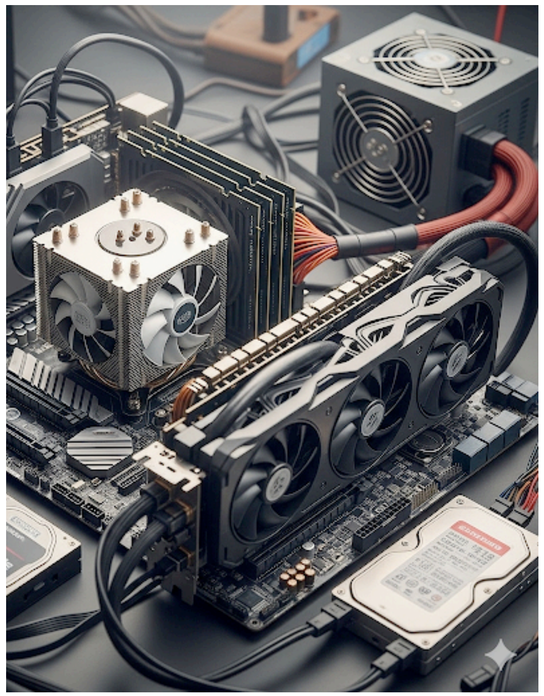


Hardware

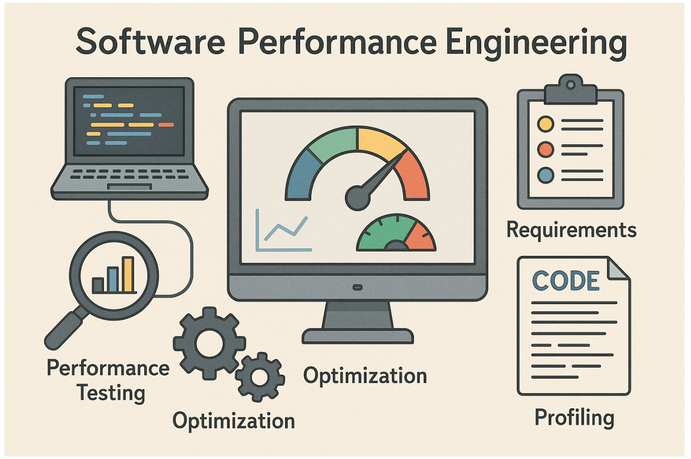


Performance Engineering

Ways to achieve performance



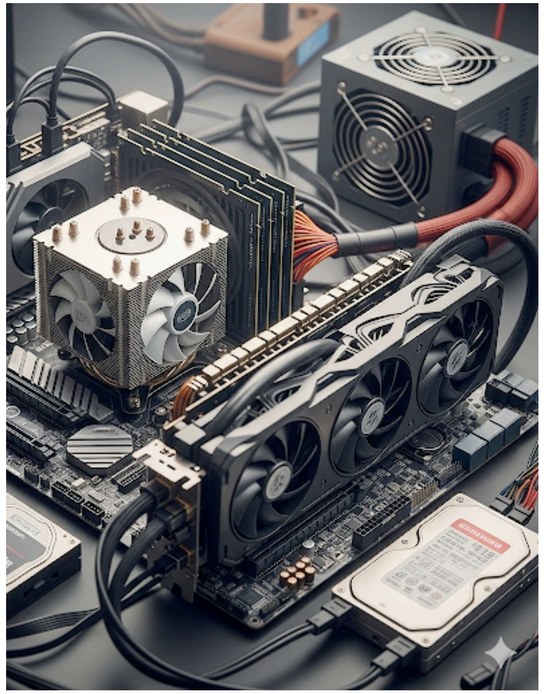
Hardware



Performance Engineering

See Prof. Leicesterson's MIT OpenCourseWare Course:
https://ocw.mit.edu/courses/6-172-performance-engineering-of-software-systems-fall-2018/

Ways to achieve performance

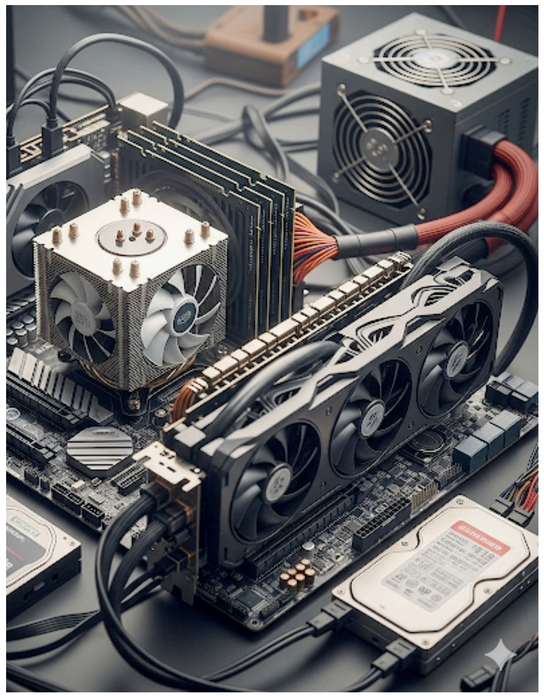


Hardware

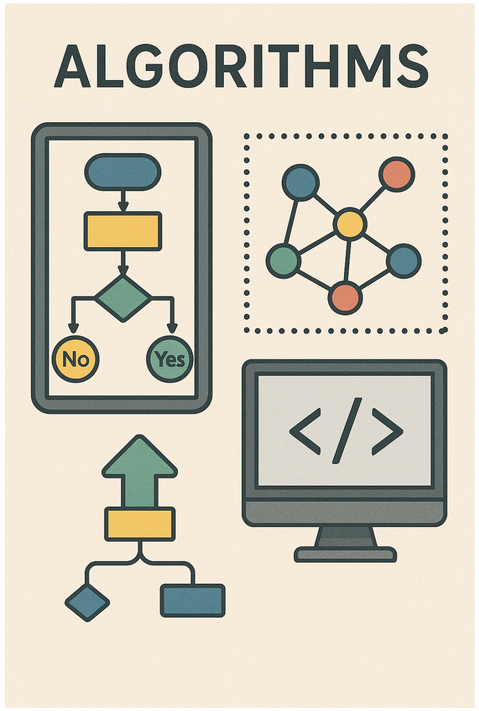
Performance Engineering

See Prof. Leicesterson's MIT OpenCourseWare Course:
https://ocw.mit.edu/courses/6-172-performance-engineering-of-software-systems-fall-2018/

Ways to achieve performance



Hardware



Algorithms

Performance Engineering

See Prof. Leicesterson's MIT OpenCourseWare Course:
https://ocw.mit.edu/courses/6-172-performance-engineering-of-software-systems-fall-2018/

I don’t need to know algorithms because reasons

I don’t need to know algorithms because ~~reasons~~ AI

JASON KOEBLER BUSINESS JUL 29, 2025 9:00 AM

Meta Is Going to Let Job Candidates Use AI During Coding Tests

Mark Zuckerberg has said vibecoding will be a major part of Meta's engineering work in the near future.

LAUREN GOODE THE BIG STORY AUG 21, 2025 6:00 AM

Why Did a $10 Billion Startup Let Me Vibe-Code for Them—and Why Did I Love It?

I spent two days at Notion and saw an industry in upheaval. I also shipped some actual code.

GAI and the software engineering industry

- 92% of US-based developers report using GAI in- or outside work [1]
- 51% of professional developers report using GAI daily [4]
- At Microsoft, maybe 20–30% of code is written by GAI [2]
- A quarter of recent startups have code bases where ≥ 95% of code written by GAI [3]
- But most developers (72%) are not vibe coding as of June 2025 [4]

1. https://github.blog/news-insights/research/survey-reveals-ais-impact-on-the-developer-experience/
2. https://www.cnbc.com/2025/04/29/satya-nadella-says-as-much-as-30percent-of-microsoft-code-is-written-by-ai.html
3. https://techcrunch.com/2025/03/06/a-quarter-of-startups-in-ycs-current-cohort-have-codebases-that-are-almost-entirely-ai-generated/
4. https://survey.stackoverflow.co/2025/ai

How’s it going?

- Early study finds developers are 56% faster using Copilot on "simple" task [1]
- Recent study finds developers are 19% slower using GAI-of-choice on "real-world" task despite self-predicting and estimating they will be faster [2]
- Junior developers (< 1 year) take up to 10% longer using GAI [3]
    - Need fundamentals: syntax, data structures, algorithms, design patterns

1. Peng et al. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv:2302.06590, 2023.
2. Becker et al. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer
3. Deniz et al. Unleashing Developer Productivity with Generative AI. McKinsey Digital report, 2023.

Why study algorithms?

Why study algorithms?

Algorithms are a critical component to system performance

Why study algorithms?

Algorithms are a critical component to system performance

To use AI effectively, need solid fundamentals

Who am I?

Alex Conway

Assistant Professor, Computer Science

Cornell Tech

Biography

My research is primarily focused on randomized data structures and their applications to memory and storage systems. My research projects often start with a new theoretical idea which can be leveraged build into a surprising new system, which in turn can be hardened for real-world use.

SplinterDB exemplifies this theory-systems-practice pipeline. It leverages a new theoretically optimal data structure, the Mapped B^ε-tree to build a general-purpose key-value store that has world-beating performance on modern systems. Research papers have appeared on the data structure (ICALP 2018), the core system (ATC 2020), and a data structural extension (SIGMOD 2023). SplinterDB is available as open-source on GitHub, and is deployed in VMware products, such as vSAN 8.0.

Mosaic Pages and Iceberg Hash Tables are related research projects which also build systems using new ideas in theory. Iceberg Hash Tables (SIGMOD 2023) use new ideas from the theory of load-balancing to construct the fastest space-efficient hash tables to date. Mosaic Pages (ASPLOS 2023, distinguished paper) is

Alex Conway

PhD Computer Science
Rutgers University

Senior Researcher
VMware Research Group

Assistant Professor
Cornell Tech

Alex Conway

My research

Theory of algorithms and data structures

Principled systems research

Commercial deployment

Course Slack Workspace

Join the class Slack!

Ask me course questions

Ask TAs about HW questions

TA office hours announced on slack

What to expect from this class

Develop a toolkit of algorithms and techniques

Learn how to apply them to a variety of different problems

Learn how to analyze them, modify them, understand them

In class:
Prove why they do
or do not work

Assignments:
Programming with
algorithms

What to expect from this class

Develop a toolkit of algorithms and techniques

Learn how to apply them to a variety of different problems

Learn how to analyze them, modify them, understand them

In class:
Prove why t
or do not

Assignments:
mming with
orithms

We’re going to do math

Background and Prereqs

Know how to program — Assignments will use python

Basic discrete math — Graphs, asymptotic notation, etc.

Mathematical reasoning

We’re going to do proofs in class, not on assignments

Nothing is strictly required, but it will be harder

Course Structure

90%

4 Programming Assignments — See tentative schedule

10%

Participation

Course Policies

Late Days. Homeworks are due on the due date by 11:59:59pm EST. You can use in total 3 late days throughout the semester. For every extra day beyond the 3 days limit, there will be a deduction of 10% in the respective assignment.

Students are allowed to collaborate on homeworks, but **each individual must submit their own solutions written up individually**. Homeworks should indicate who you collaborated with.

I expect students to attend lectures. 10% of final grades are reserved as participation; the default will be that everyone gets the full 10% but we reserve the right to start taking attendance or otherwise change how participation is alotted if lecture attendance/participation proves poor.

You may use external resources: Google, ChatGPT, Gemini, etc. Homeworks should indicate what resources you use.

I strongly recommend attempting the problems on your own.

Office Hours

TA Office Hours:

Announced on Slack

All HW-related questions

Prof. Conway:

Tuesday 2:00-2:45pm (before class)

Email/Slack on Monday (or I'll free up the time)

Questions?

Please ask questions during class

You’re not the only one confused

I make lots of mistakes

The Setting (background)

The Setting (background)

Start with a problem of size n

The Setting (background)

Start with a problem of size n

The Setting (background)

Start with a problem of size n

Sorting: array of size n

Scheduling: n jobs

Matching: n objects to match

The Setting (background)

Start with a problem of size n

Want to find a good algorithm
that solves the problem

Sorting: array of size n

Scheduling: n jobs

Matching: n objects to match

The Setting (background)

Start with a problem of size n

Want to find a good algorithm that solves the problem

Sorting: array of size n

Scheduling: n jobs

Matching: n objects to match

The Setting (background)

Start with a problem of size n

Want to find a good algorithm that solves the problem

Good can mean fewest time steps, least memory, etc.

Sorting: array of size n

Scheduling: n jobs

Matching: n objects to match

The Setting (background)

Start with a problem of size n

Want to find a good algorithm that solves the problem

Good can mean fewest time steps, least memory, etc.

Sorting: array of size n

Scheduling: n jobs

Matching: n objects to match

The Setting (background)

Start with a problem of size n

Want to find a good algorithm that solves the problem

Good can mean fewest time steps, least memory, etc.

Sorting: array of size n

Scheduling: n jobs

Matching: n objects to match

Any operation (+,-,*,/,<,>,=), any read or write to memory, any flow control (jump, if)

Asymptotic Notation (background)

Suppose an algorithm solves a problem of size n in T(n) time steps

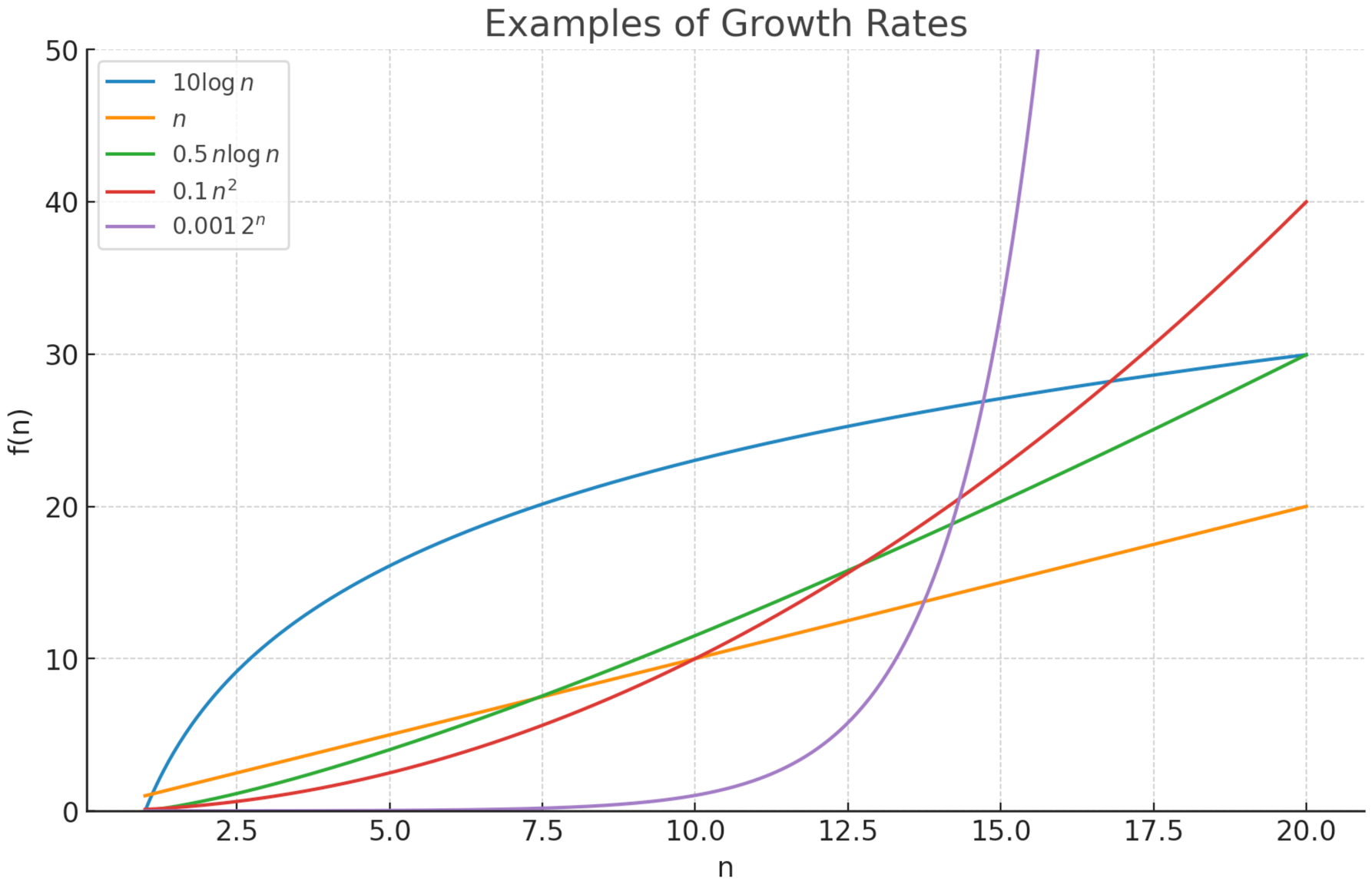
T(n) is some arbitrary function

$$T(n) = O(f(n)) \text{ if } \lim_{n \to \infty} \frac{T(n)}{f(n)} \leq C,$$

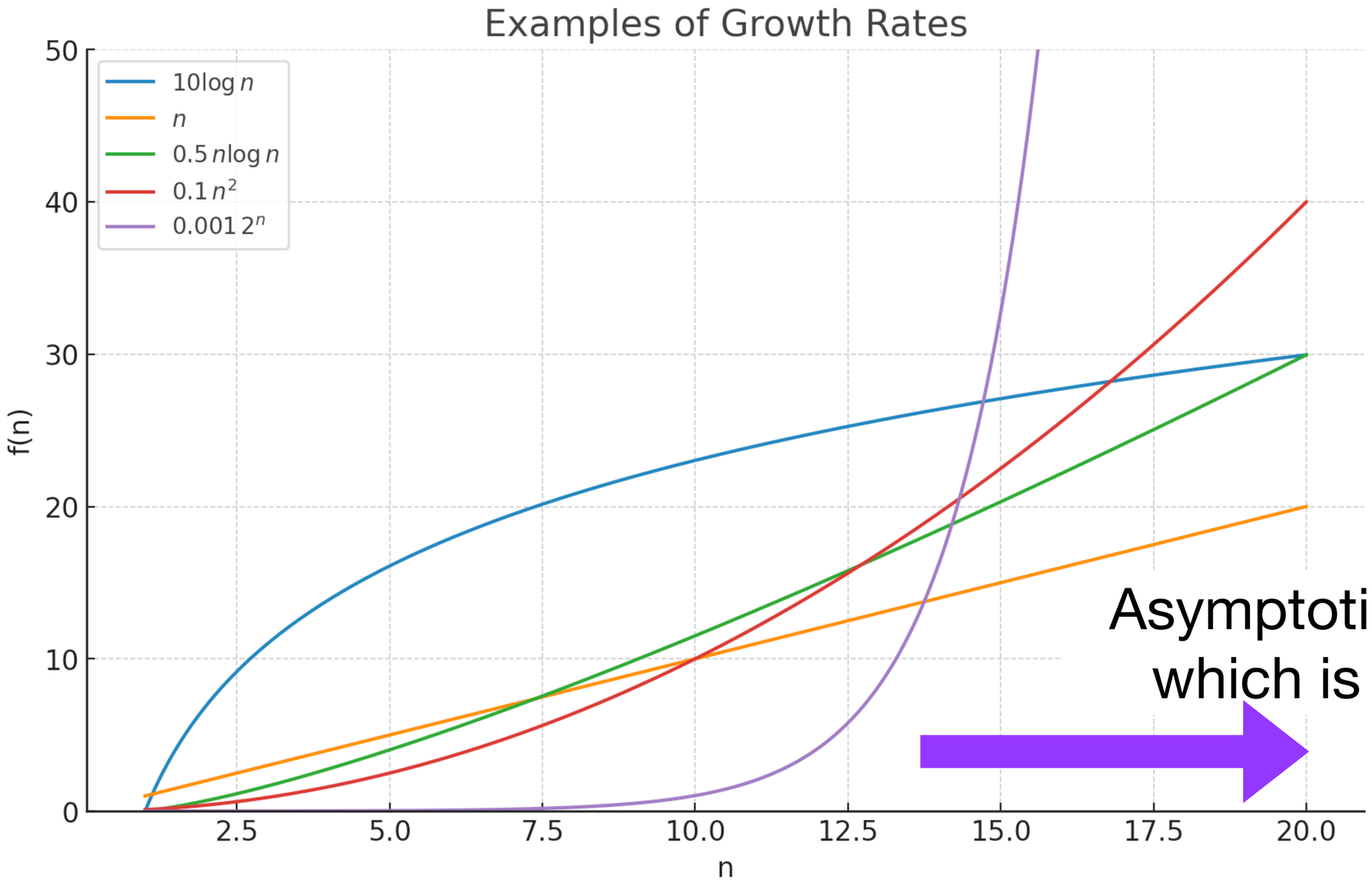
where C is a constant

Intuitively, this says that T(n) grows no faster than f(n)

Asymptotic Notation (background)

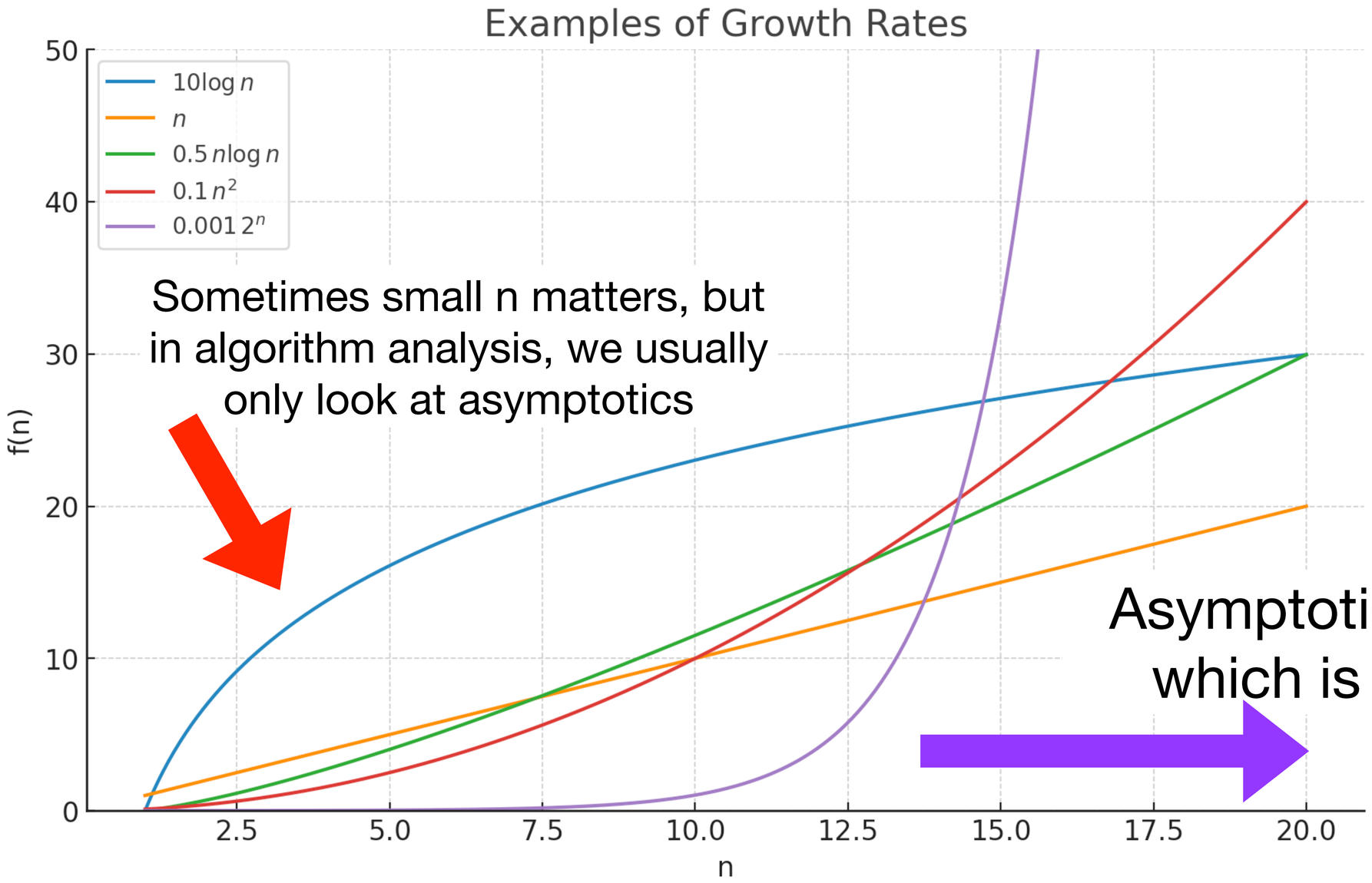


Asymptotic Notation (background)



Asymptotic notation tells you which is bigger for large n

Asymptotic Notation (background)



Sometimes small n matters, but in algorithm analysis, we usually only look at asymptotics

Asymptotic notation tells you which is bigger for large n

Greedy Algorithms

Greedy Algorithms

Greedy algorithms solve problems where
you want a maximal version of something

Greedy Algorithms

Greedy algorithms solve problems where you want a maximal version of something

Simple example:

Want to make change using as few coins as possible

25¢

10¢

5¢

1¢

Greedy Algorithms

Greedy algorithms solve problems where you want a maximal version of something

63¢

Simple example:

Want to make change using as few coins as possible

25¢

10¢

5¢

1¢

Greedy Algorithms

Greedy algorithms solve problems where you want a maximal version of something

Simple example:

Want to make change using as few coins as possible

25¢

10¢

5¢

1¢

63¢

13¢

Greedy Algorithms

Greedy algorithms solve problems where you want a maximal version of something

Simple example:

Want to make change using as few coins as possible

25¢

10¢

5¢

1¢

63¢

13¢

3¢

Greedy Algorithms

Greedy algorithms solve problems where you want a maximal version of something

Simple example:

Want to make change using as few coins as possible

25¢

10¢

5¢

1¢

63¢

13¢

3¢

Done!

Greedy Algorithms

Greedy algorithms solve problems where you want a maximal version of something

Simple example:

Want to make change using as few coins as possible

Greedy algorithm:

Choose the highest value coin less than the remaining amount

63¢

13¢

3¢

Done!

Interval Scheduling Problem

You’re a student. You want to graduate ASAP.

You want to take as many classes as possible.

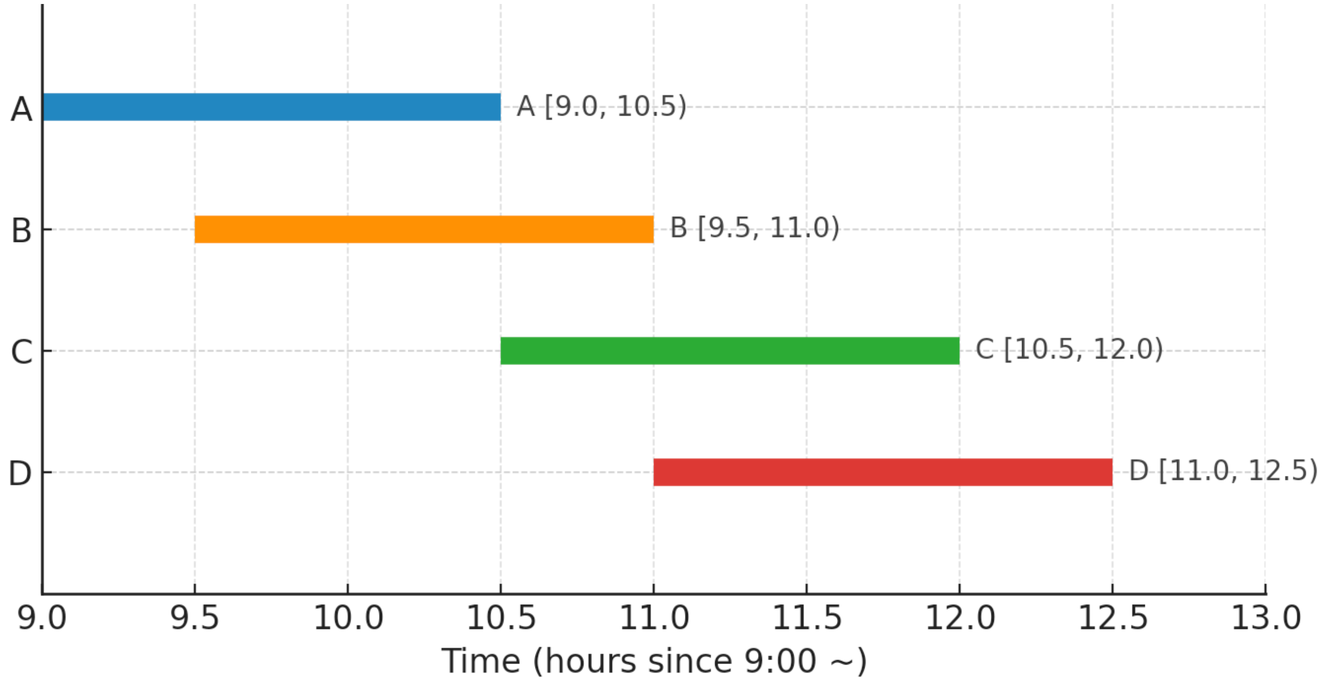
How do you choose which classes to take?

Interval Scheduling Problem

You’re a student. You want to graduate ASAP.

You want to take as many classes as possible.

How do you choose which classes to take?



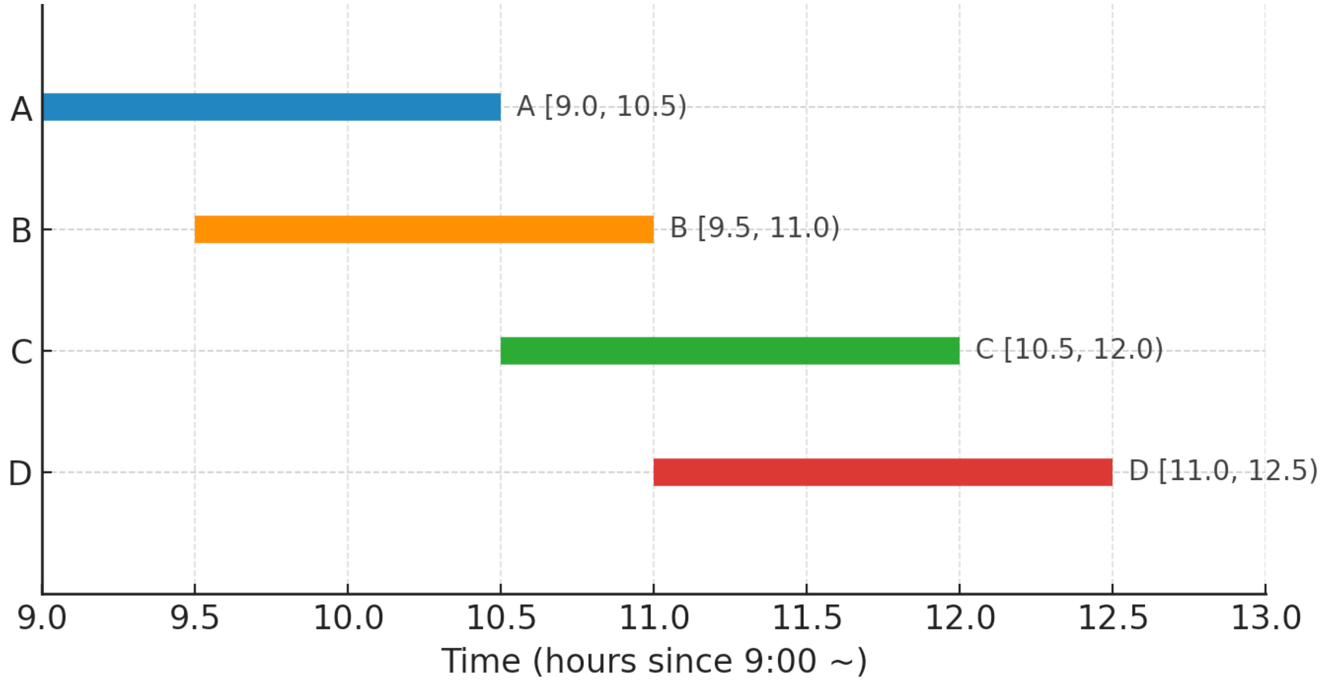
Interval Scheduling Problem

You're a student. You want to graduate ASAP.

You want to take as many classes as possible.

How do you choose which classes to take?

Problem: given a set of time intervals, find the largest non-overlapping set.



Interval Scheduling Problem

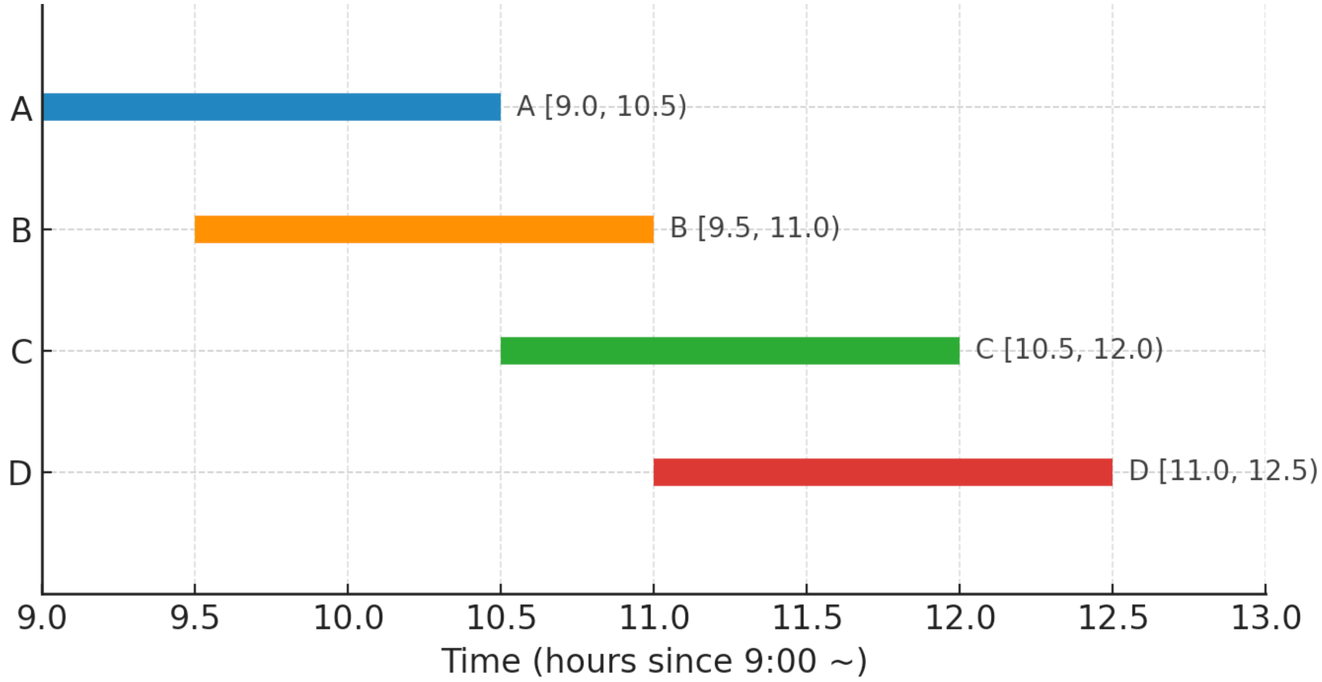
You’re a student. You want to graduate ASAP.

You want to take as many classes as possible.

How do you choose which classes to take?

Problem: given a set of time intervals, find the largest non-overlapping set.

How can we solve this problem?



Quick Aside

You're a student. You want to graduate ASAP.

You want to take as many classes as possible.

How do you choose which classes to take?

Problem: given a set of time intervals, find the largest non-overlapping set.

How can we solve this problem?

We could look at all possible combinations of intervals and see which is the largest non-overlapping one

Quick Aside

You're a student. You want to graduate ASAP.

You want to take as many classes as possible.

How do you choose which classes to take?

Problem: given a set of time intervals, find the largest non-overlapping set.

How can we solve this problem?

We could look at all possible combinations of intervals and see which is the largest non-overlapping one

2^n

Quick Aside

You're a student. You want to graduate ASAP.

You want to take as many classes as possible.

How do you choose which classes to take?

Problem: given a set of time intervals, find the largest non-overlapping set.

How can we solve this problem?

We could look at all possible combinations of intervals and see which is the largest non-overlapping one

2^n

At 266 intervals, this is ~ the number of atoms in the universe

Interval Scheduling Problem

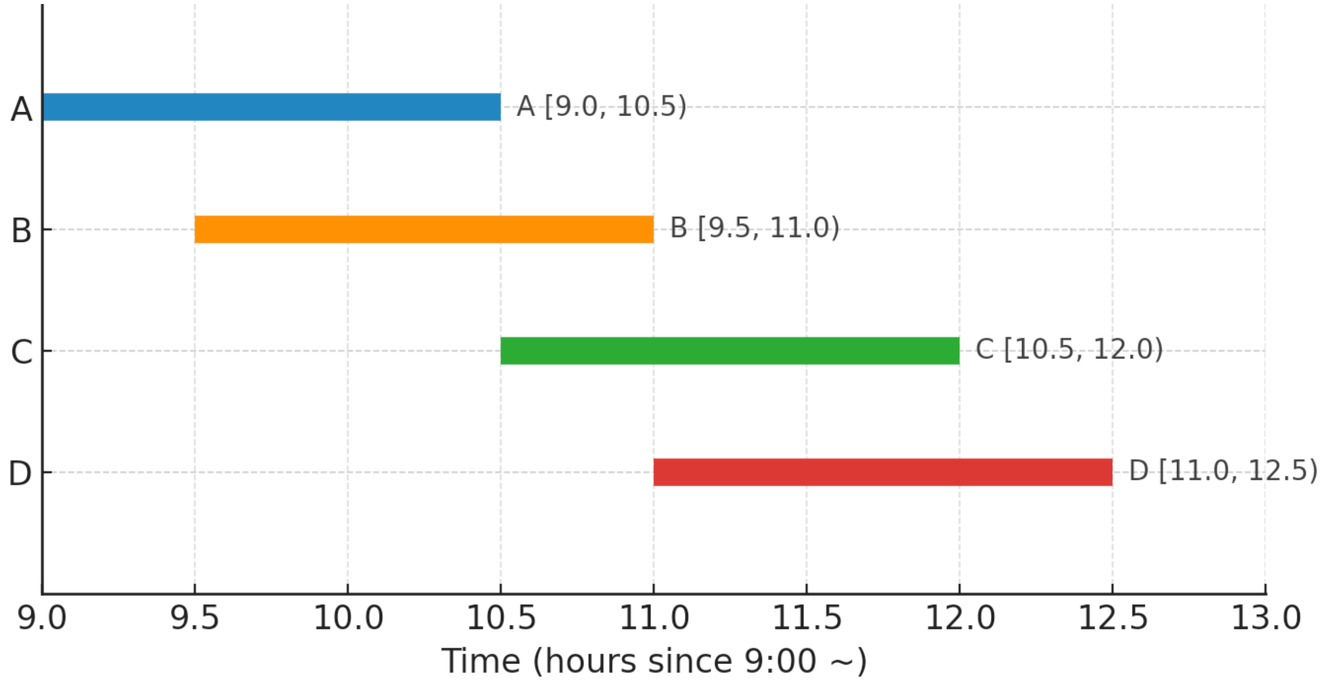
You're a student. You want to graduate ASAP.

You want to take as many classes as possible.

How do you choose which classes to take?

Problem: given a set of time intervals, find the largest non-overlapping set.

What are some ways to greedily pick classes?

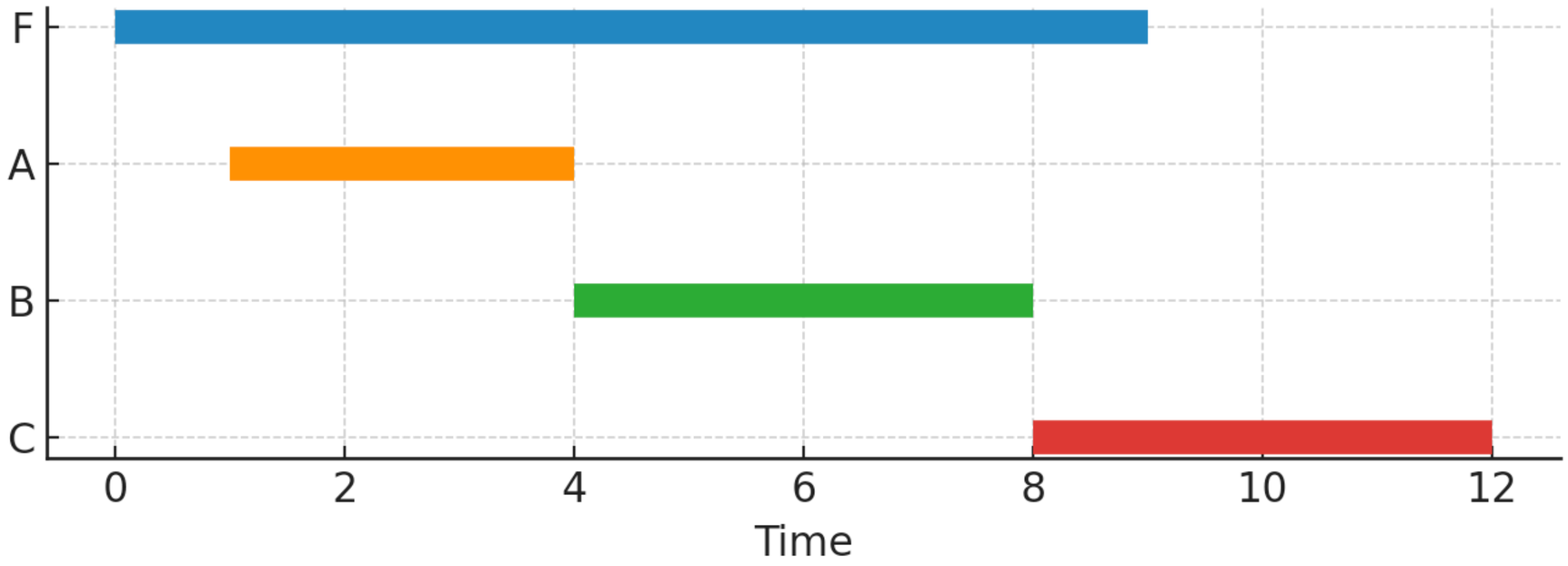


Interval Scheduling: FCFS

Greedily choose classes by earliest start time
(FCFS: first come first served)

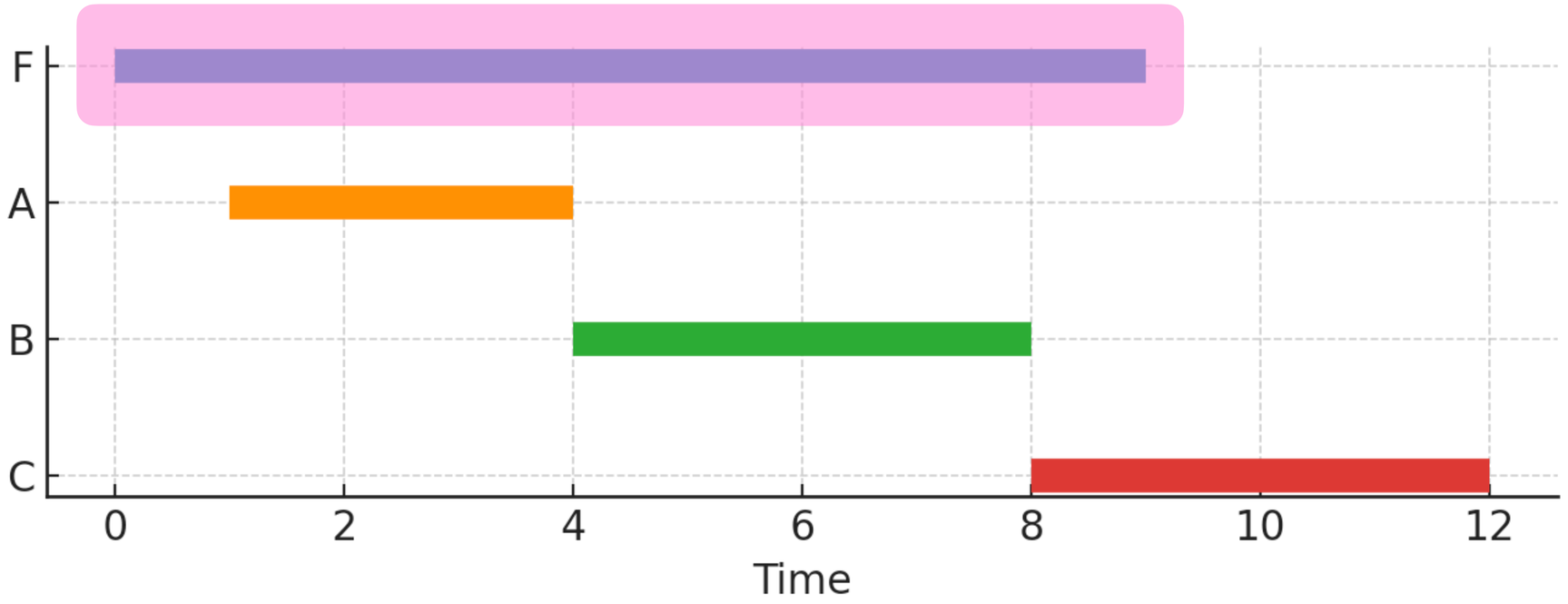
Interval Scheduling: FCFS

Greedily choose classes by earliest start time
(FCFS: first come first served)



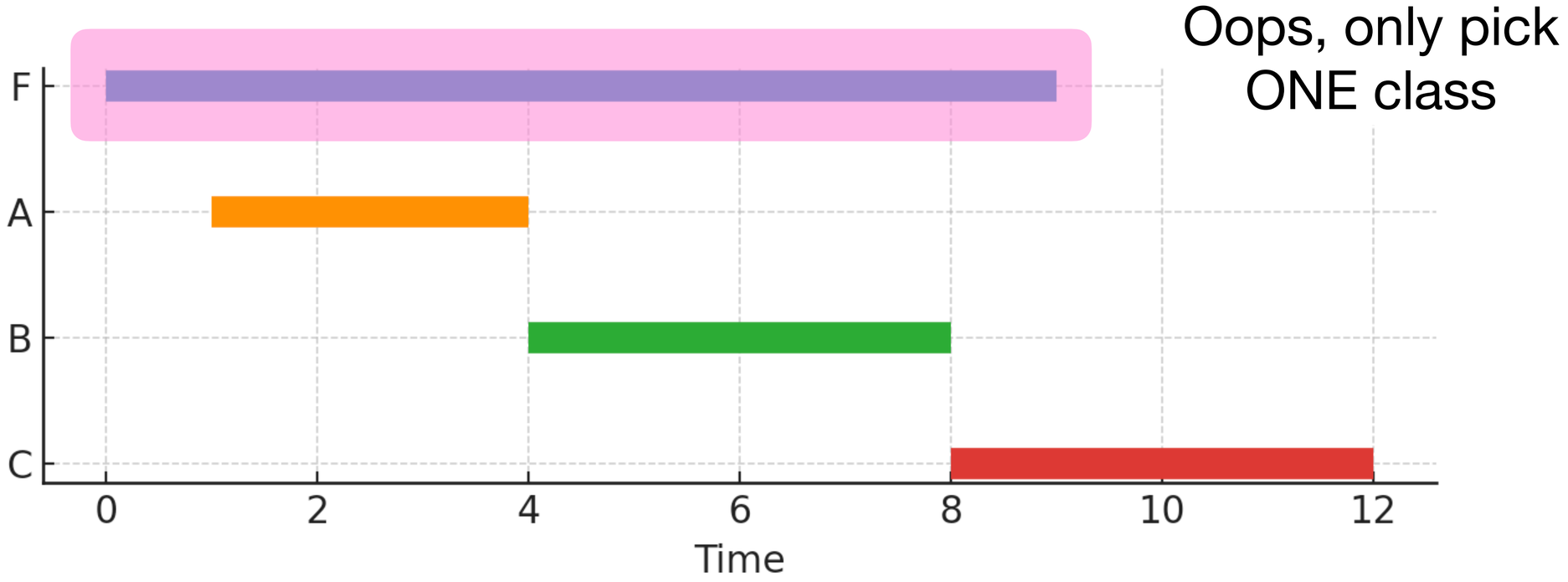
Interval Scheduling: FCFS

Greedily choose classes by earliest start time
(FCFS: first come first served)



Interval Scheduling: FCFS

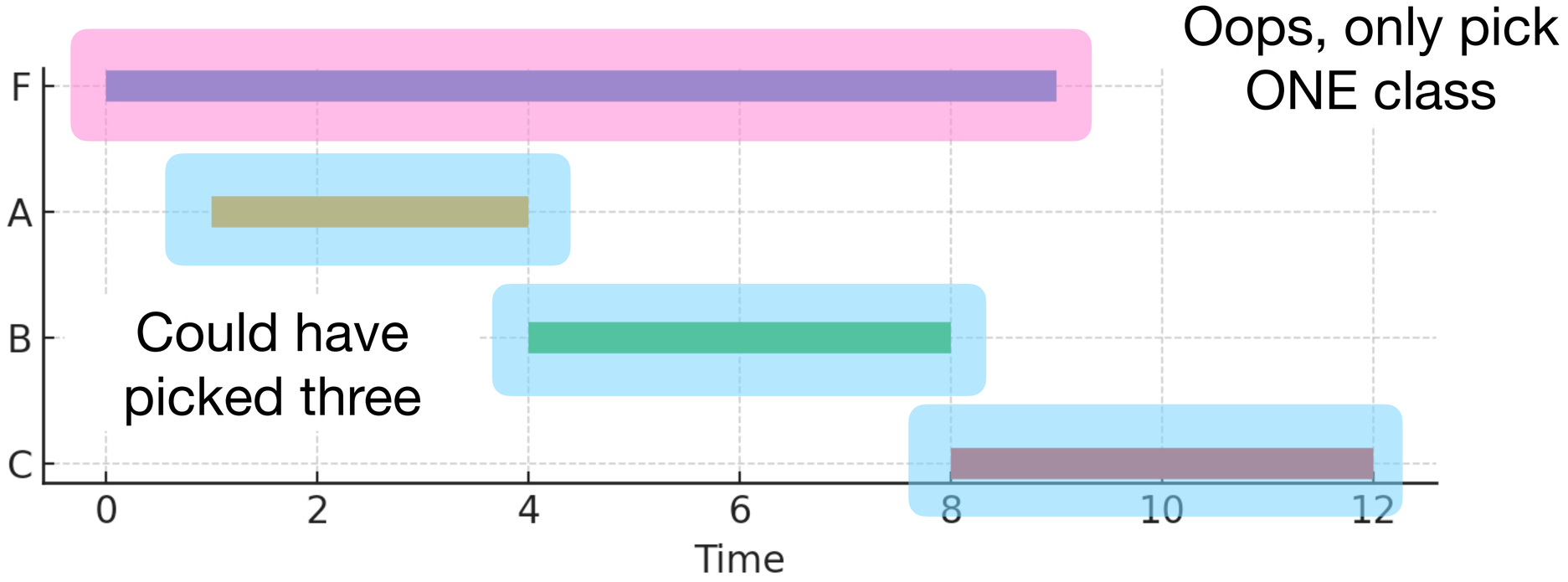
Greedily choose classes by earliest start time
(FCFS: first come first served)



Oops, only pick ONE class

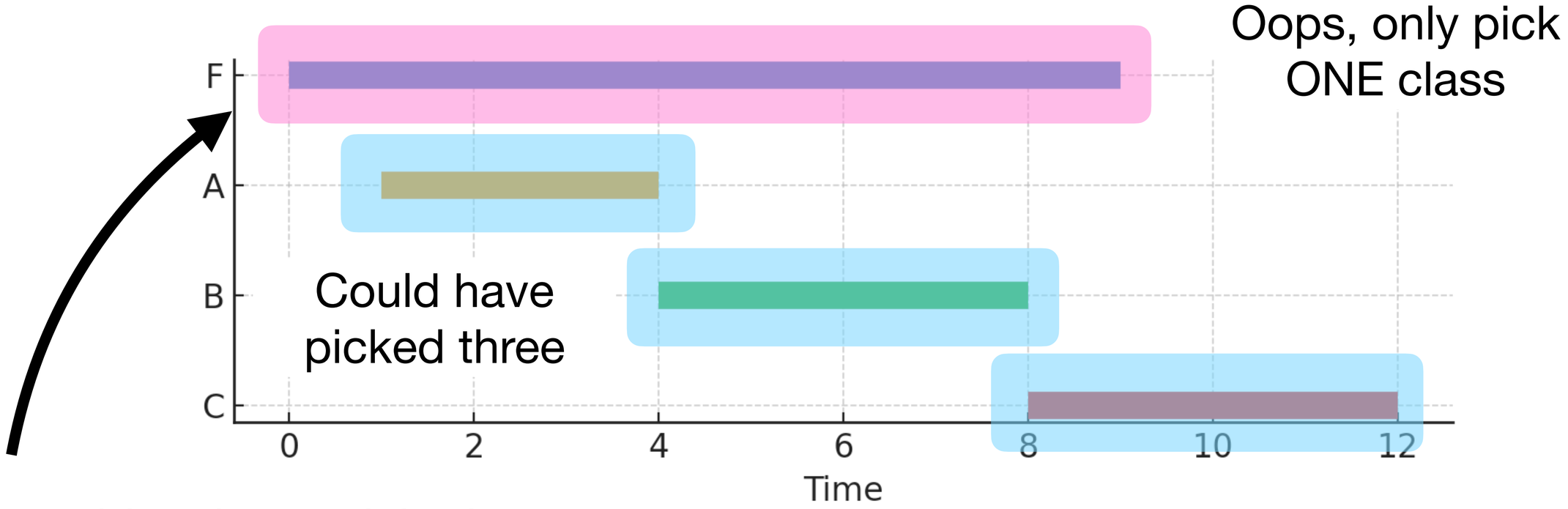
Interval Scheduling: FCFS

Greedily choose classes by earliest start time
(FCFS: first come first served)



Interval Scheduling: FCFS

Greedily choose classes by earliest start time
(FCFS: first come first served)



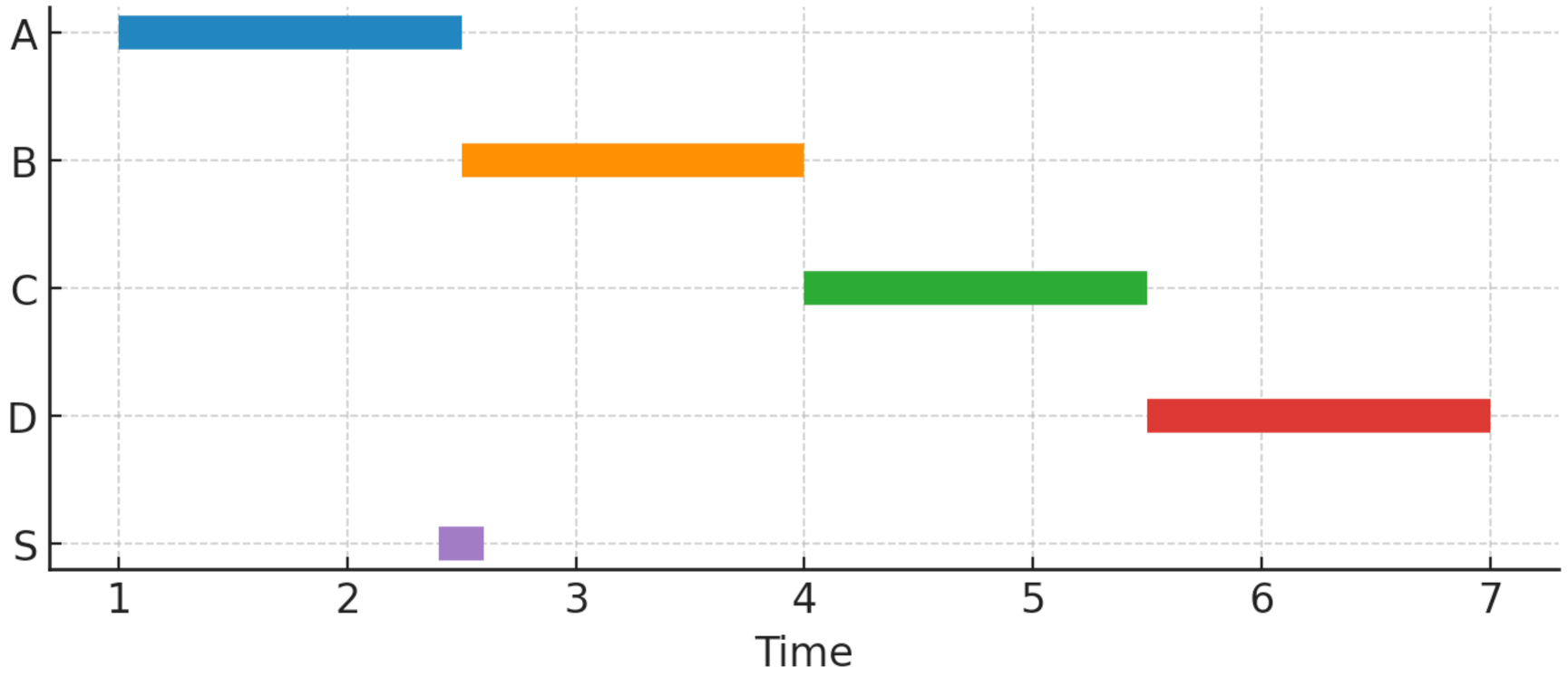
The problem is we picked this super-long class

Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration

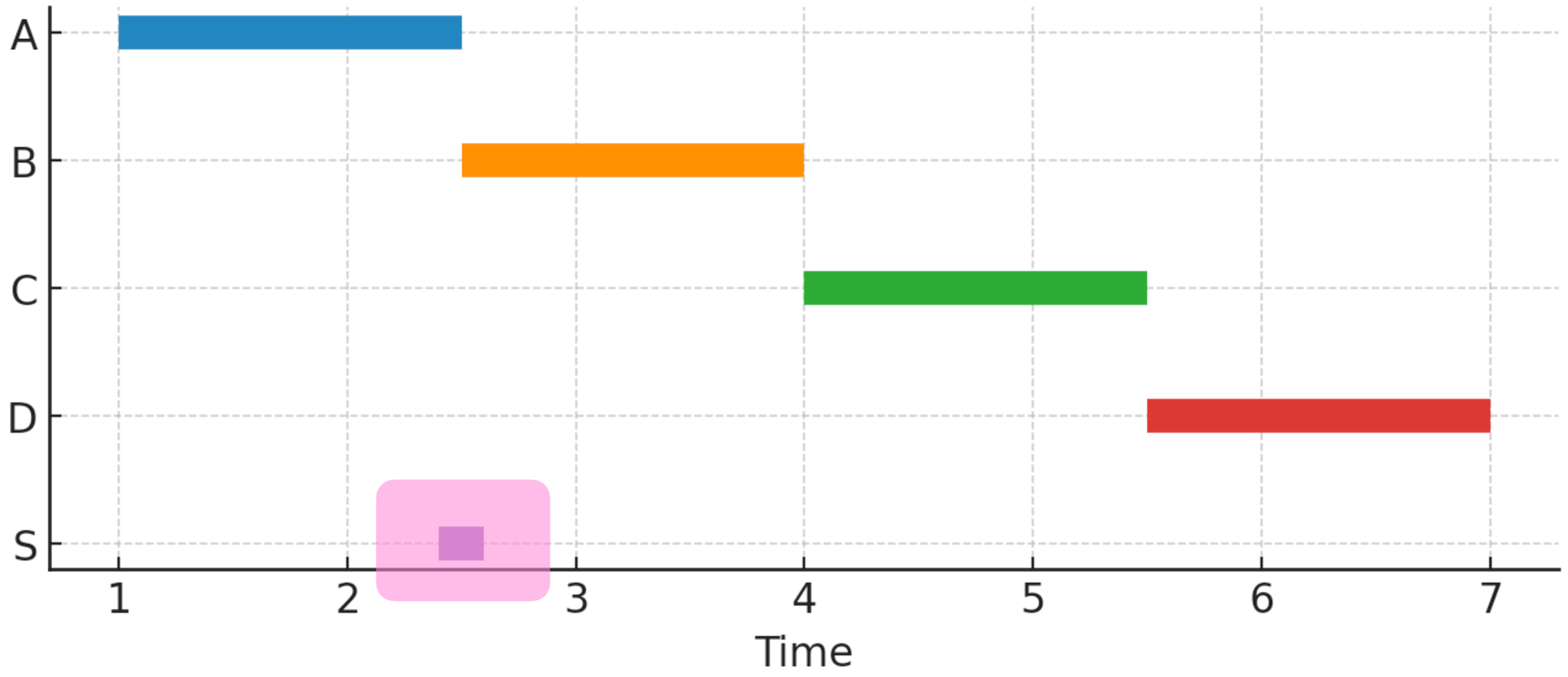
Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration



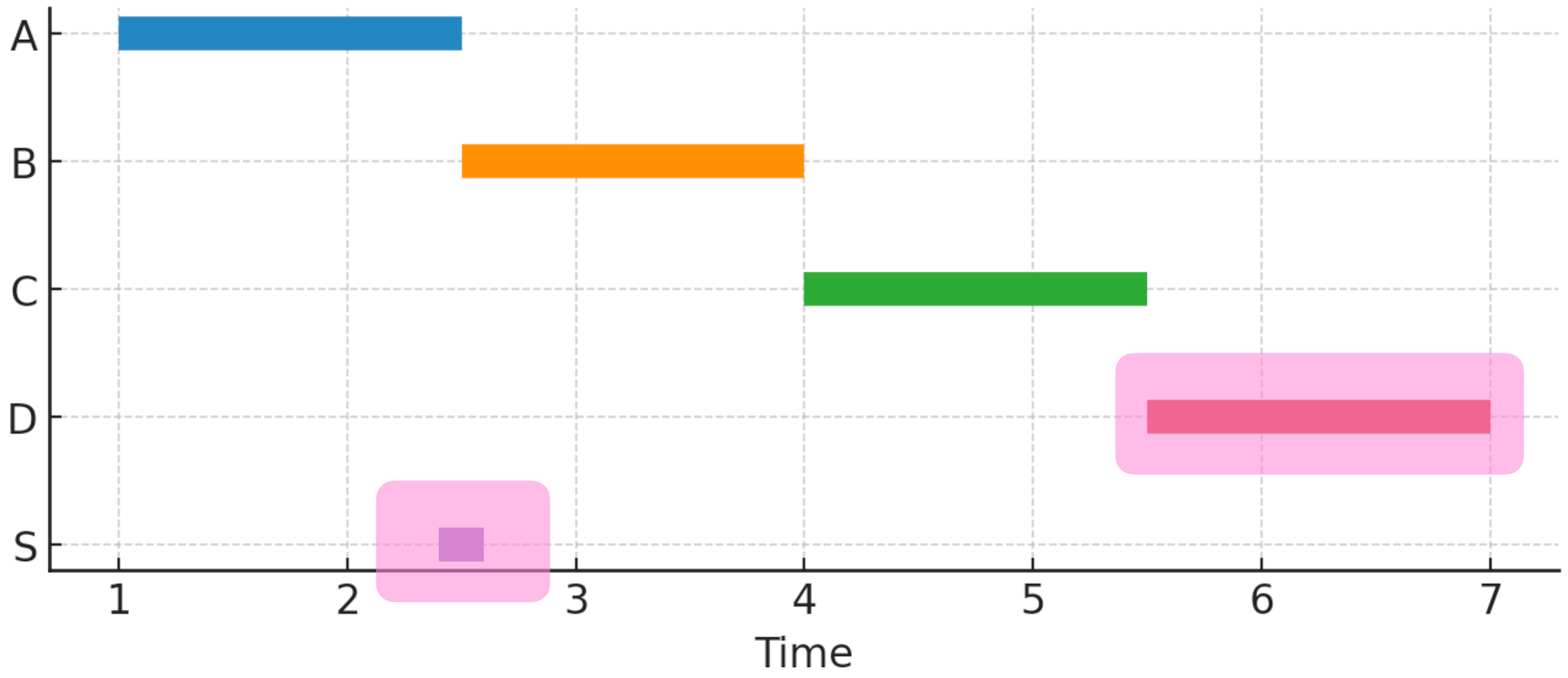
Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration



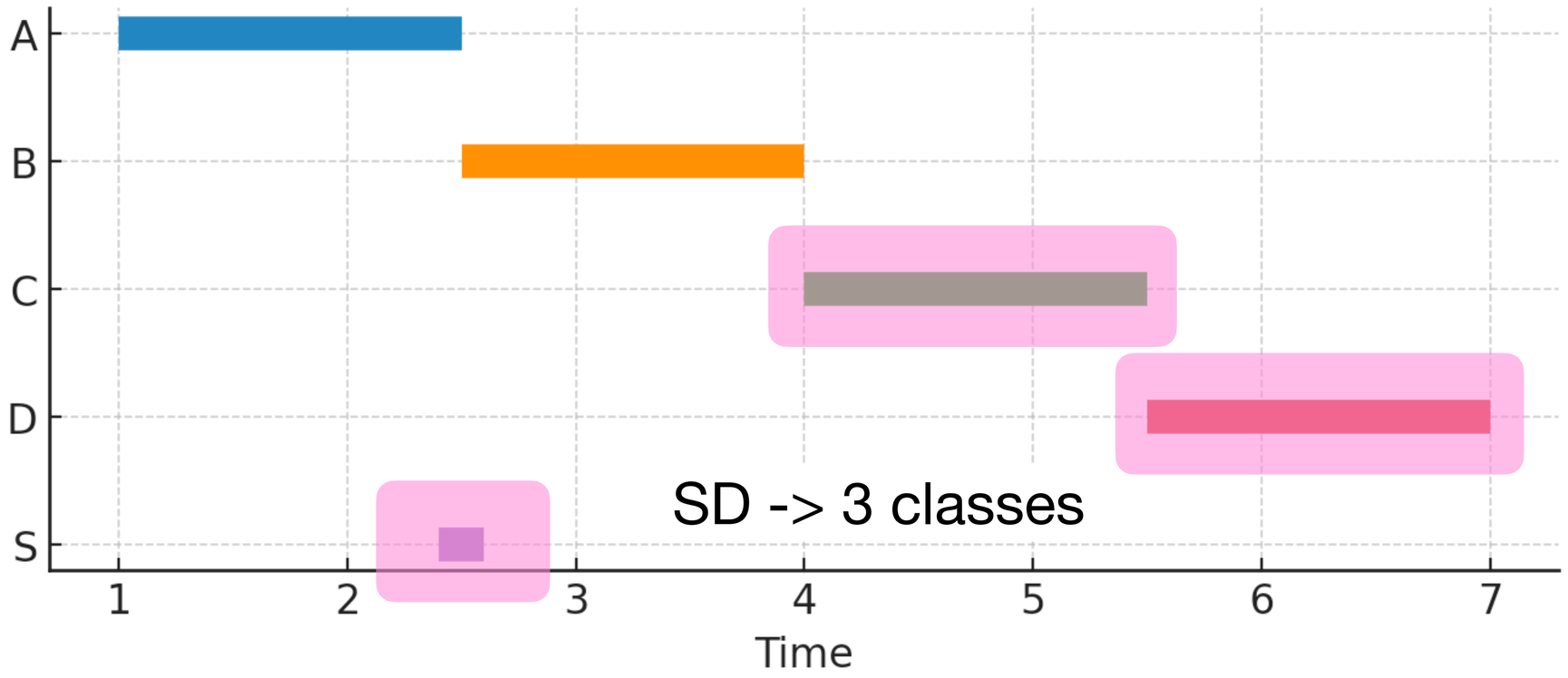
Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration



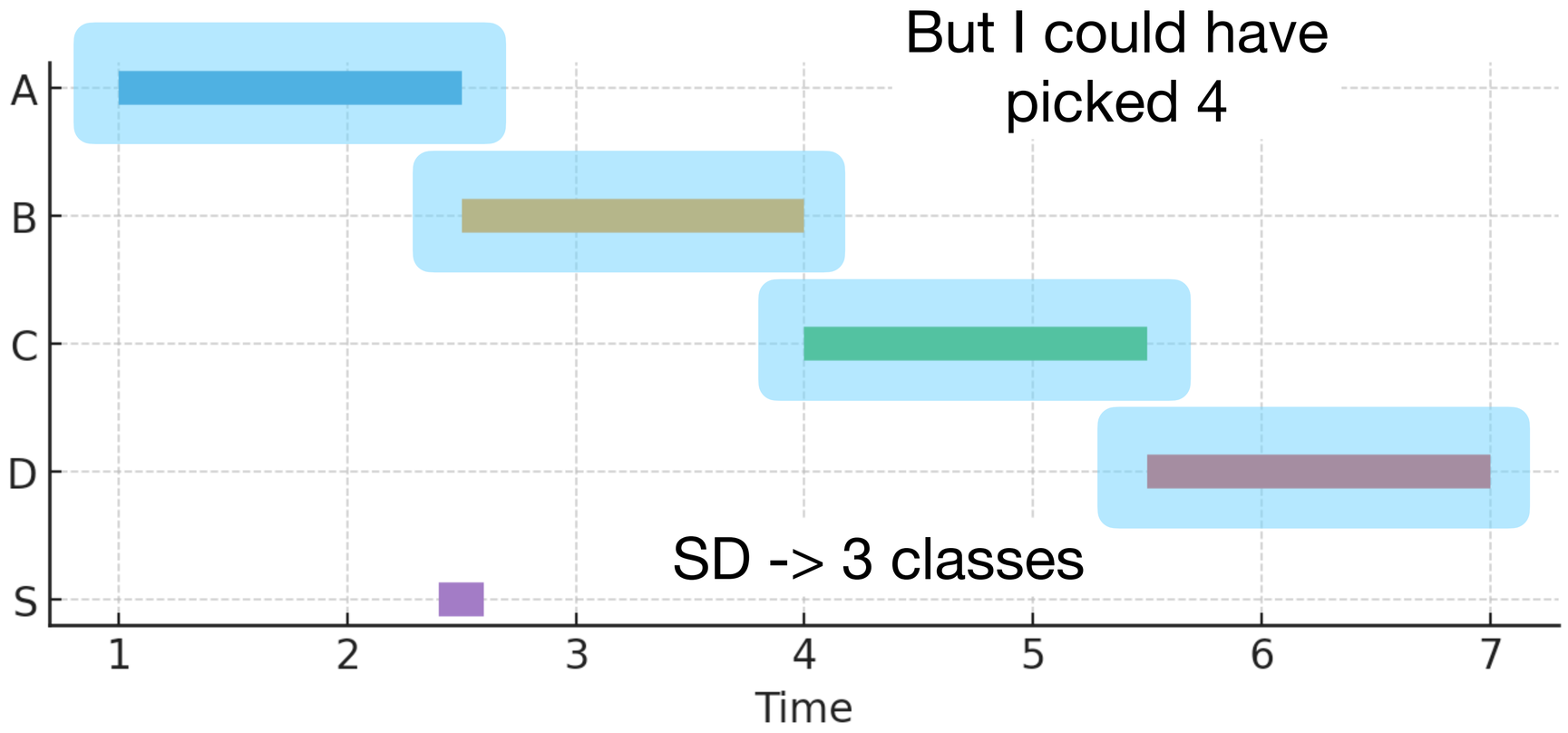
Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration



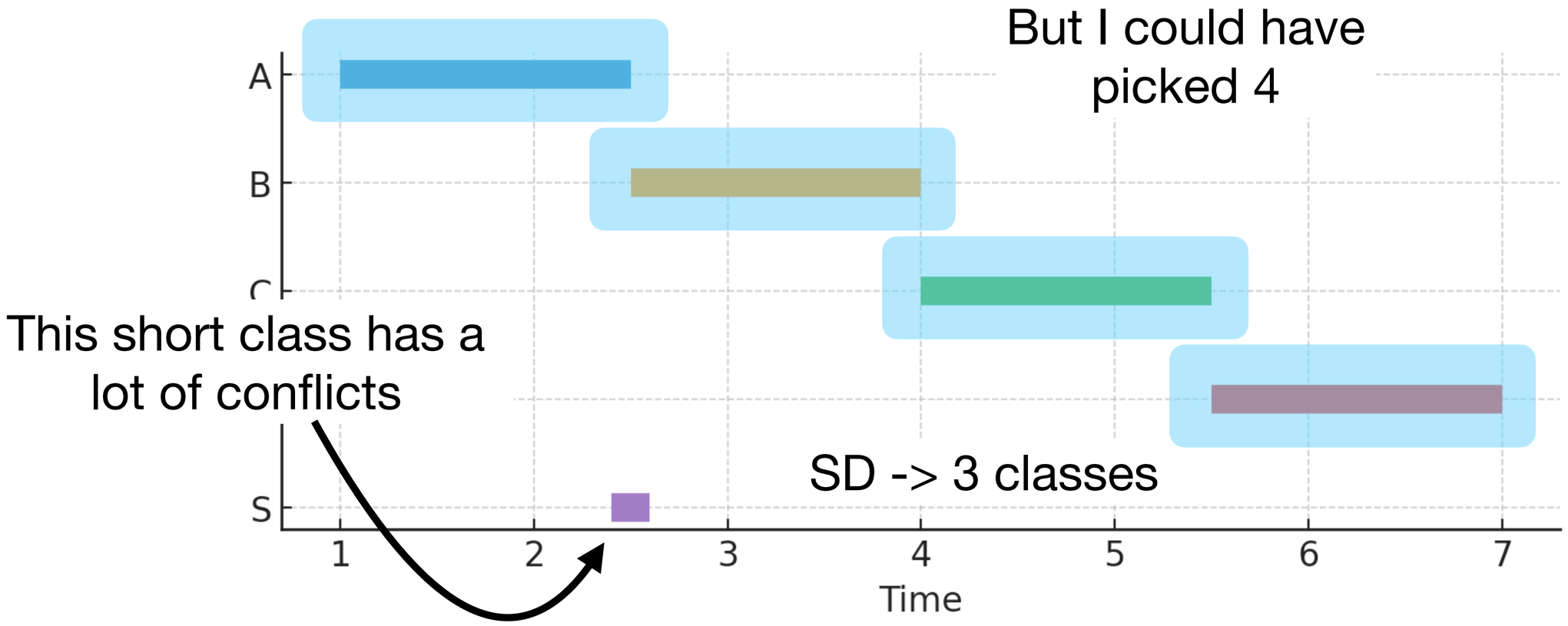
Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration



Interval Scheduling: Shortest Duration

Greedily choose classes by shortest duration

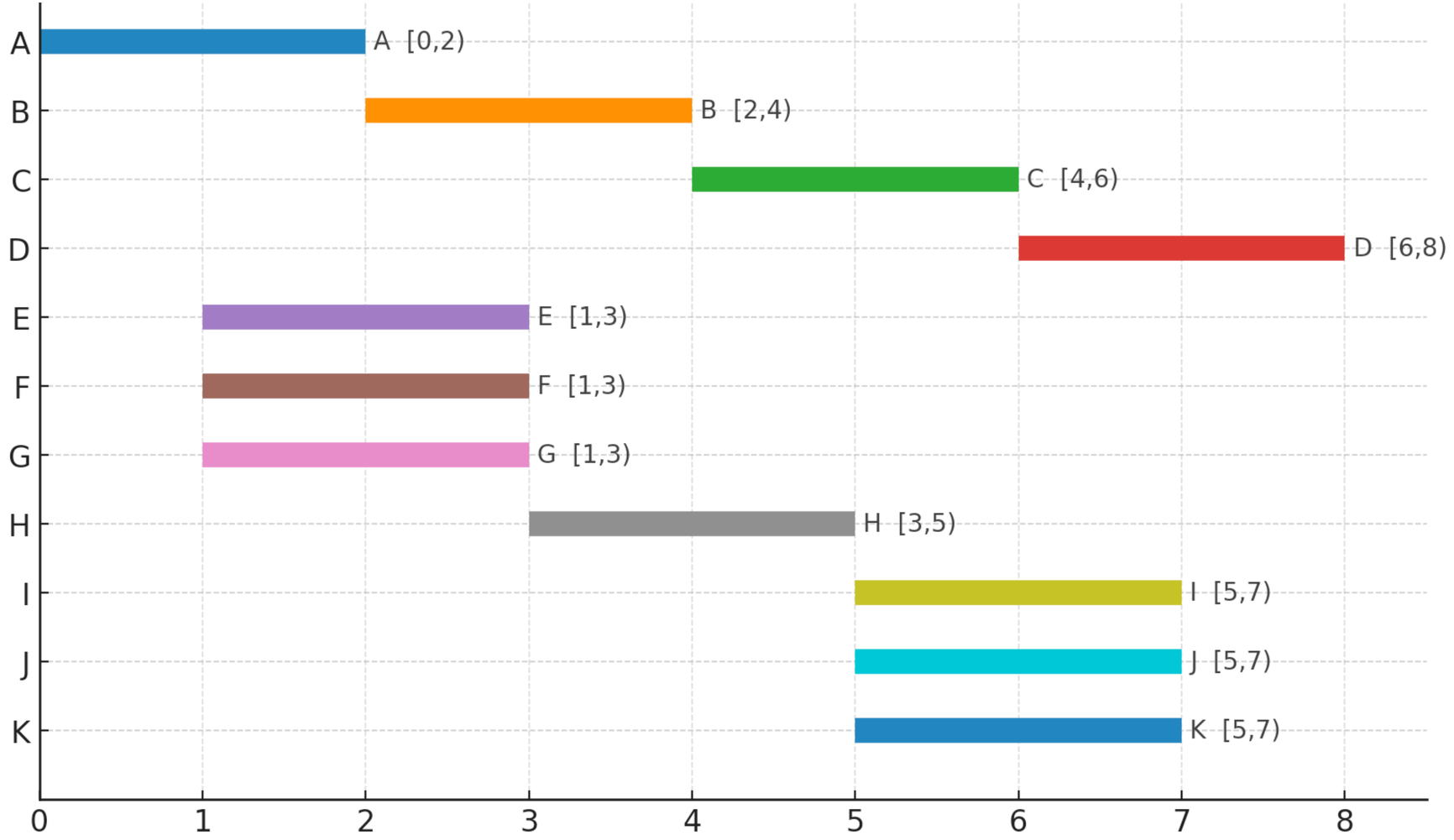


Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts

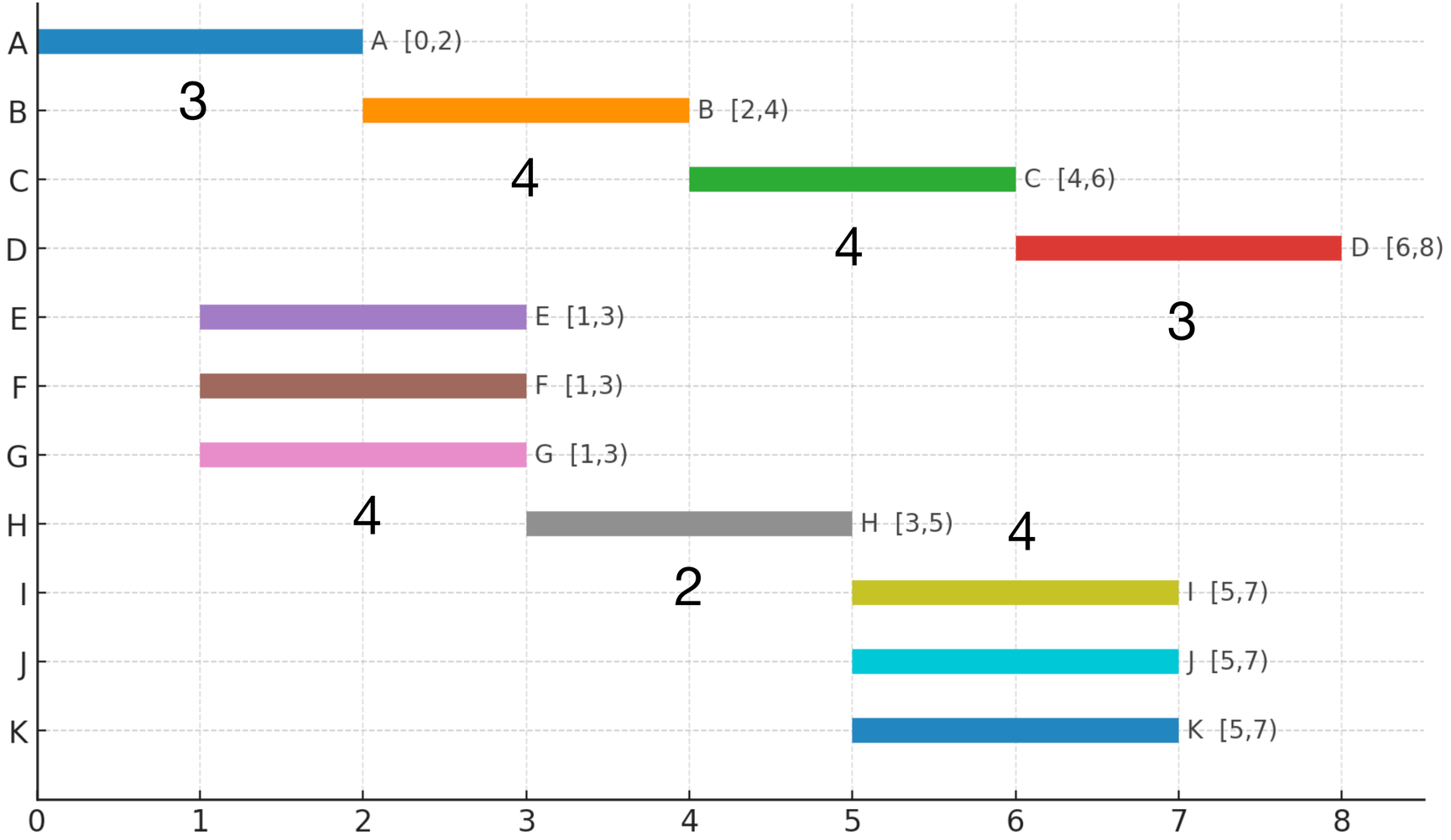
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



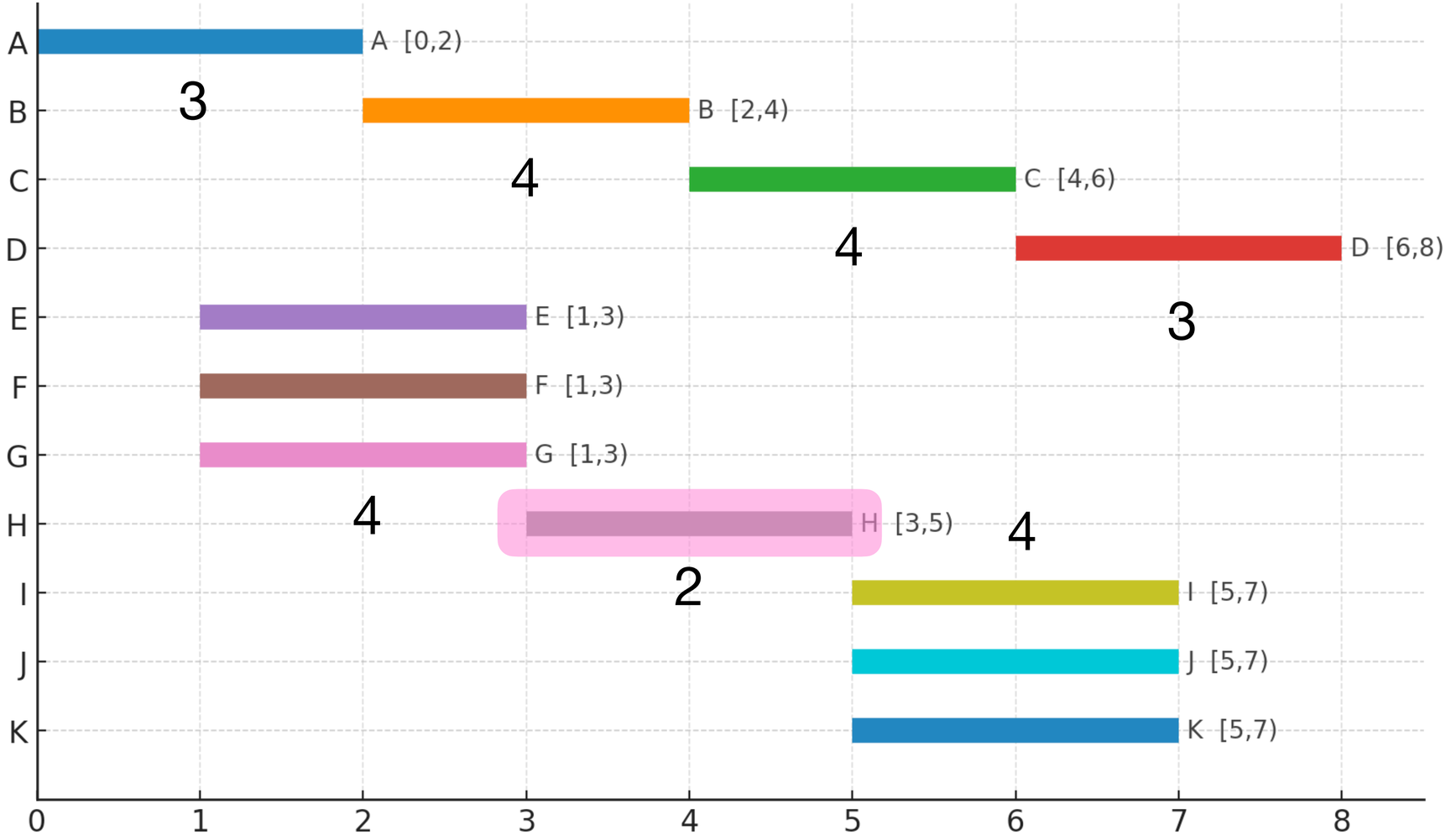
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



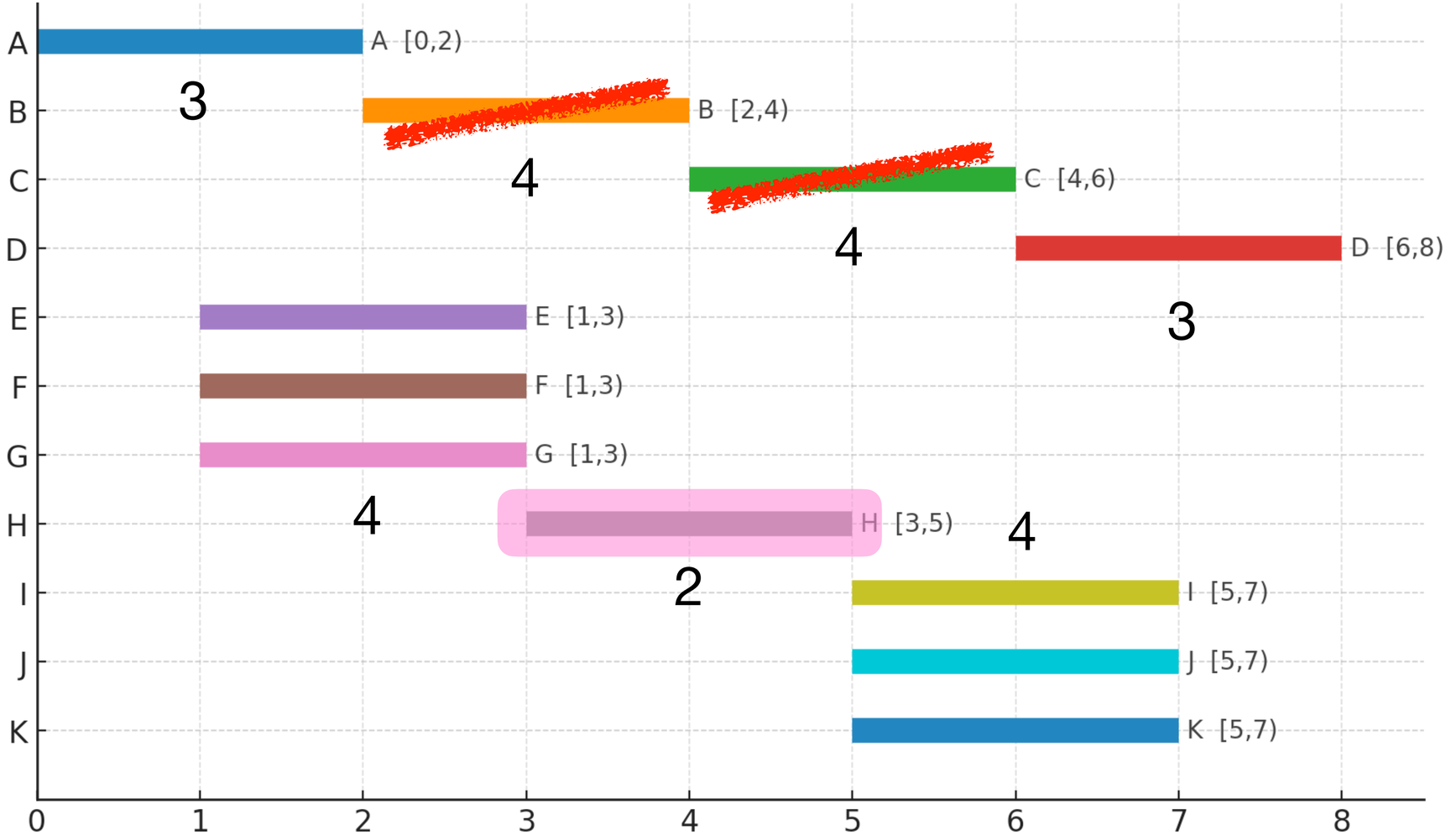
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



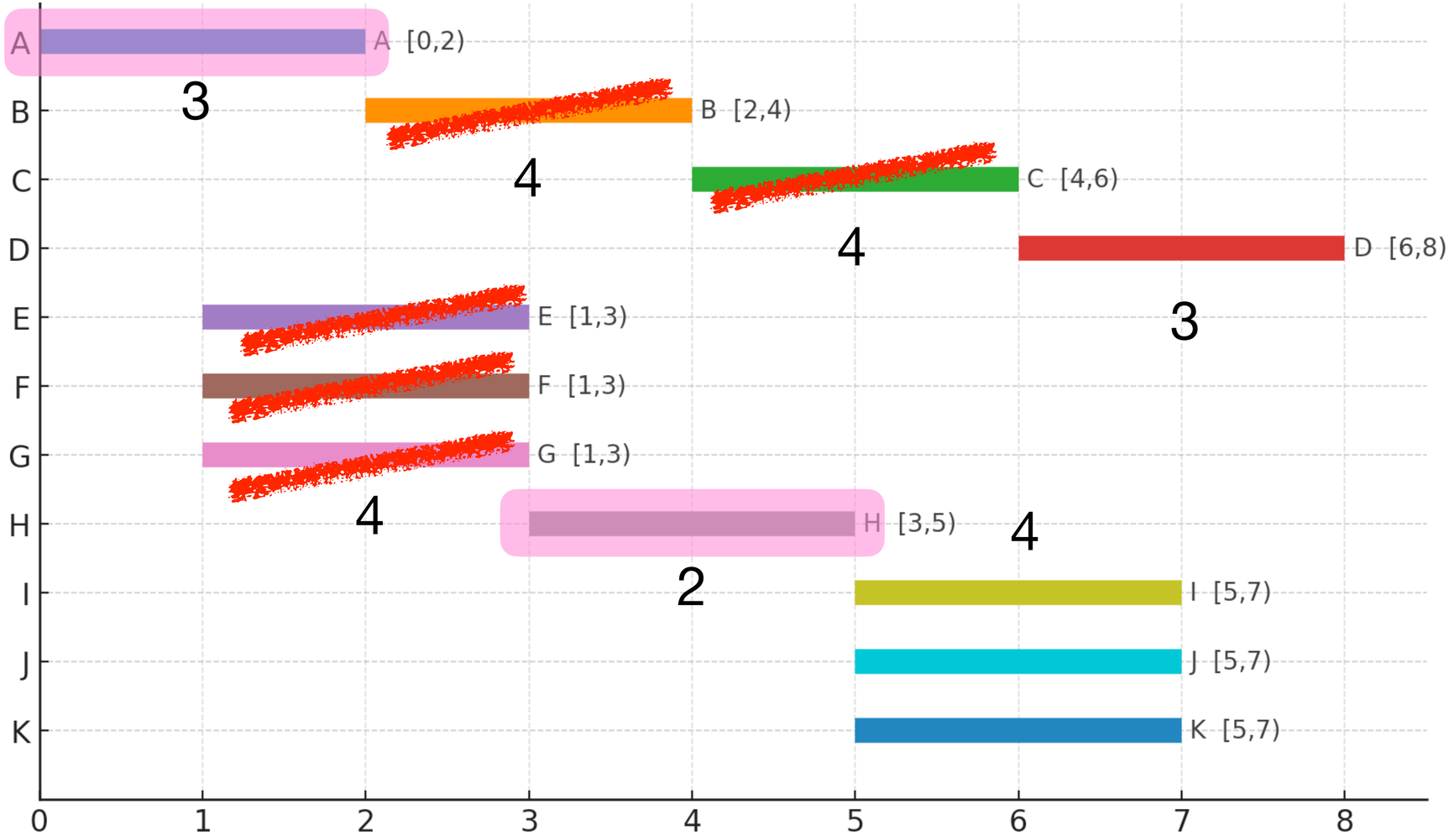
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



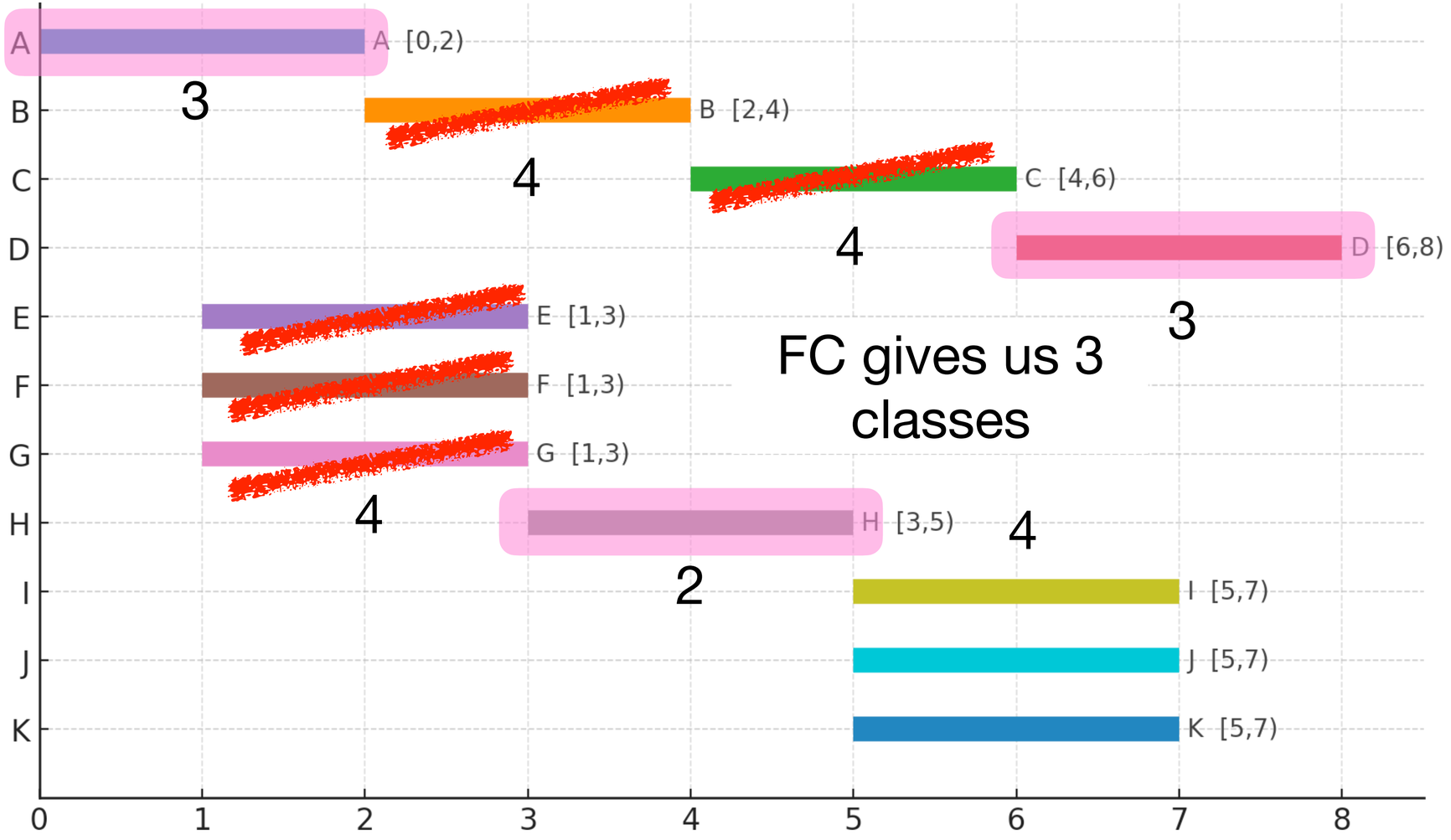
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



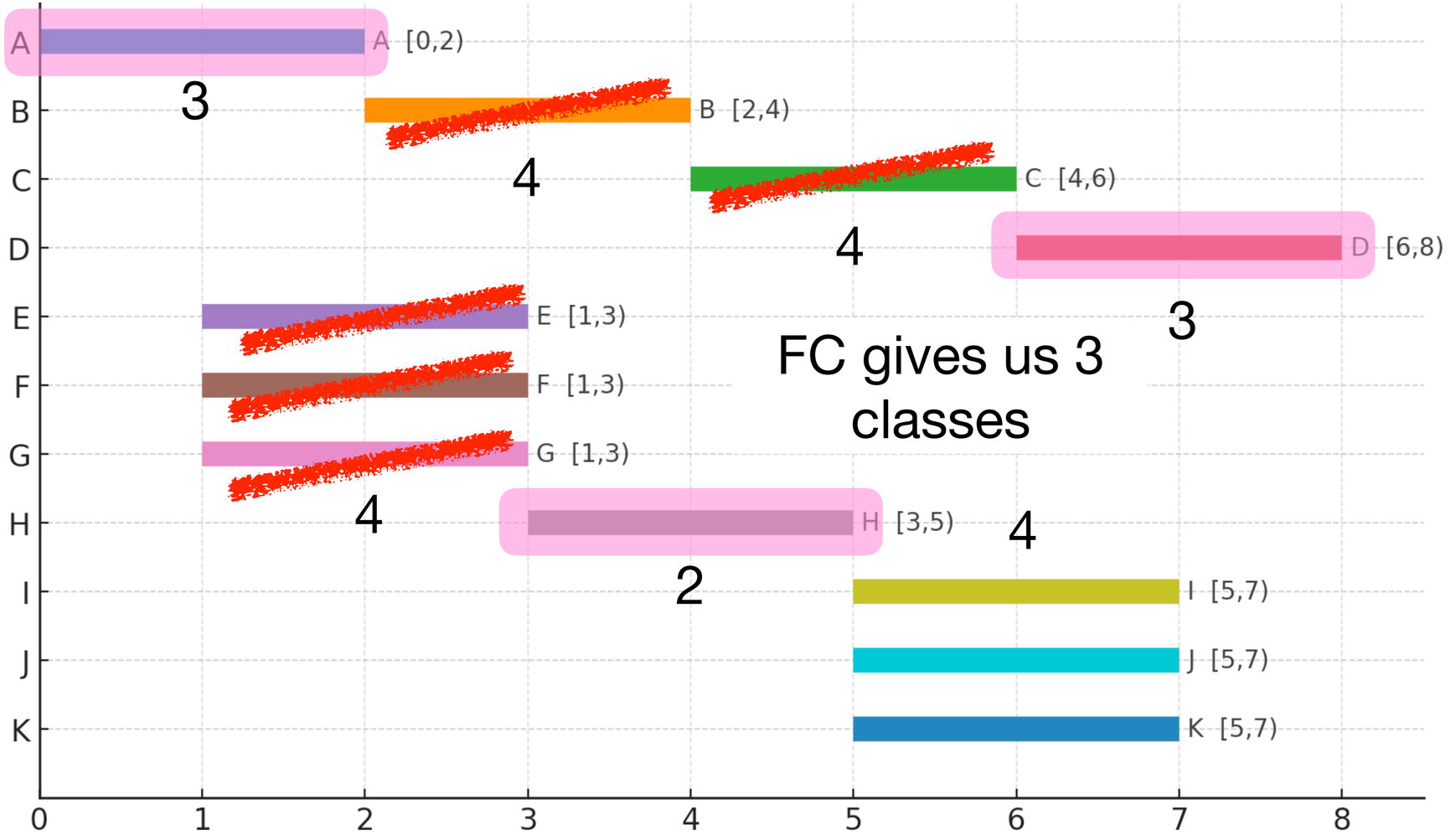
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



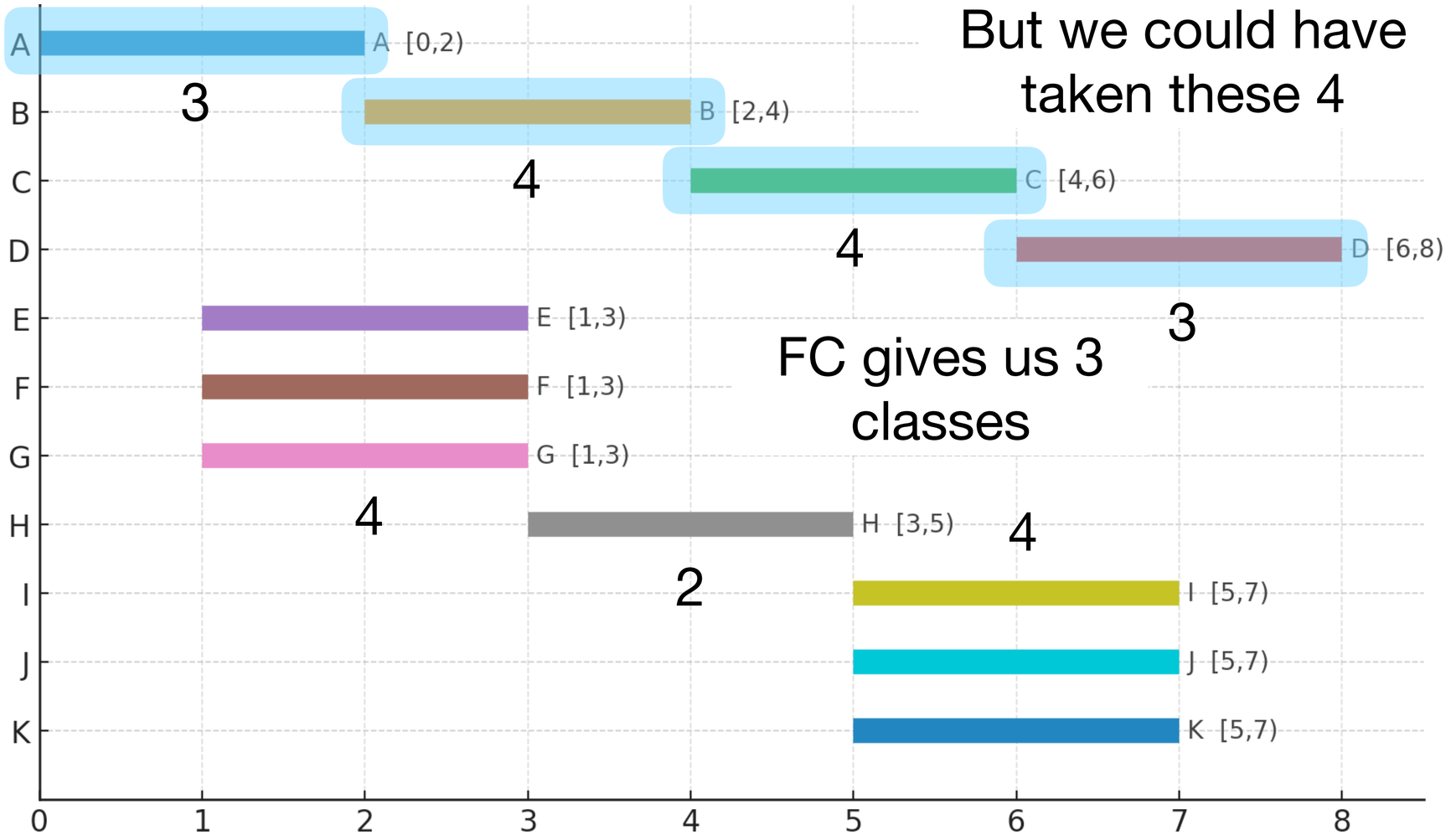
Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts



Interval Scheduling: Fewest Conflicts

Greedily choose classes by fewest conflicts

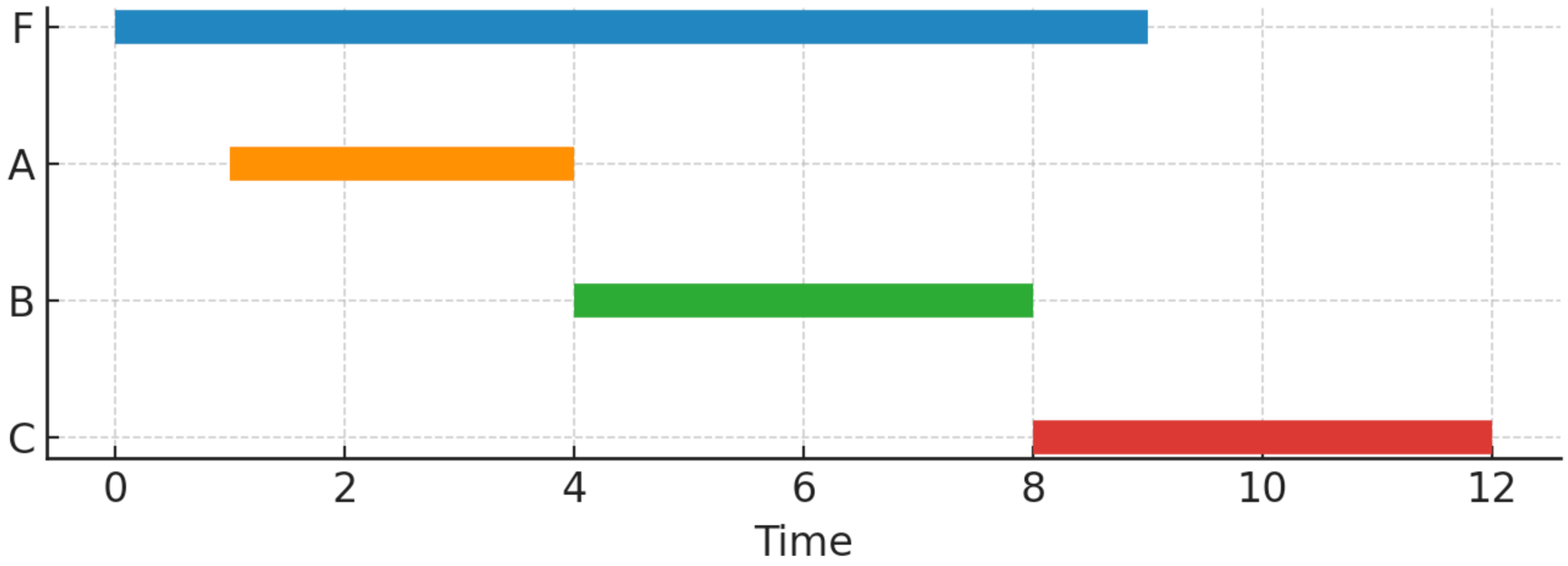


Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first

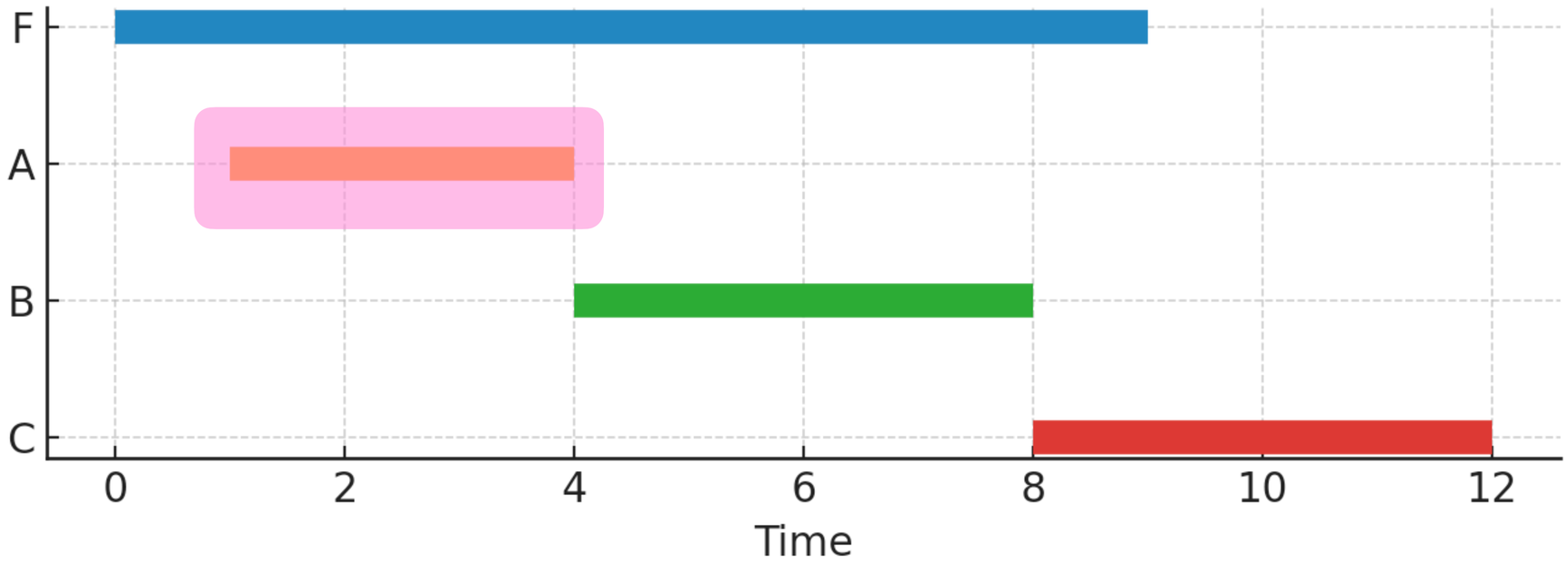
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



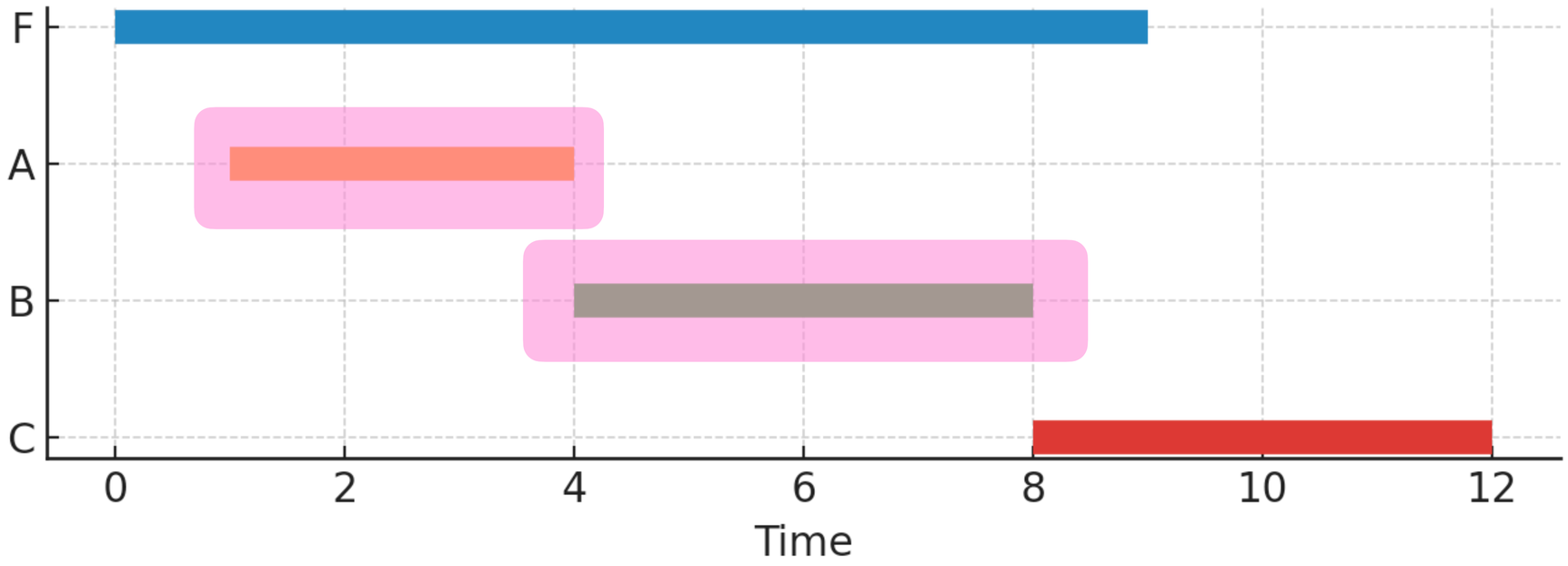
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



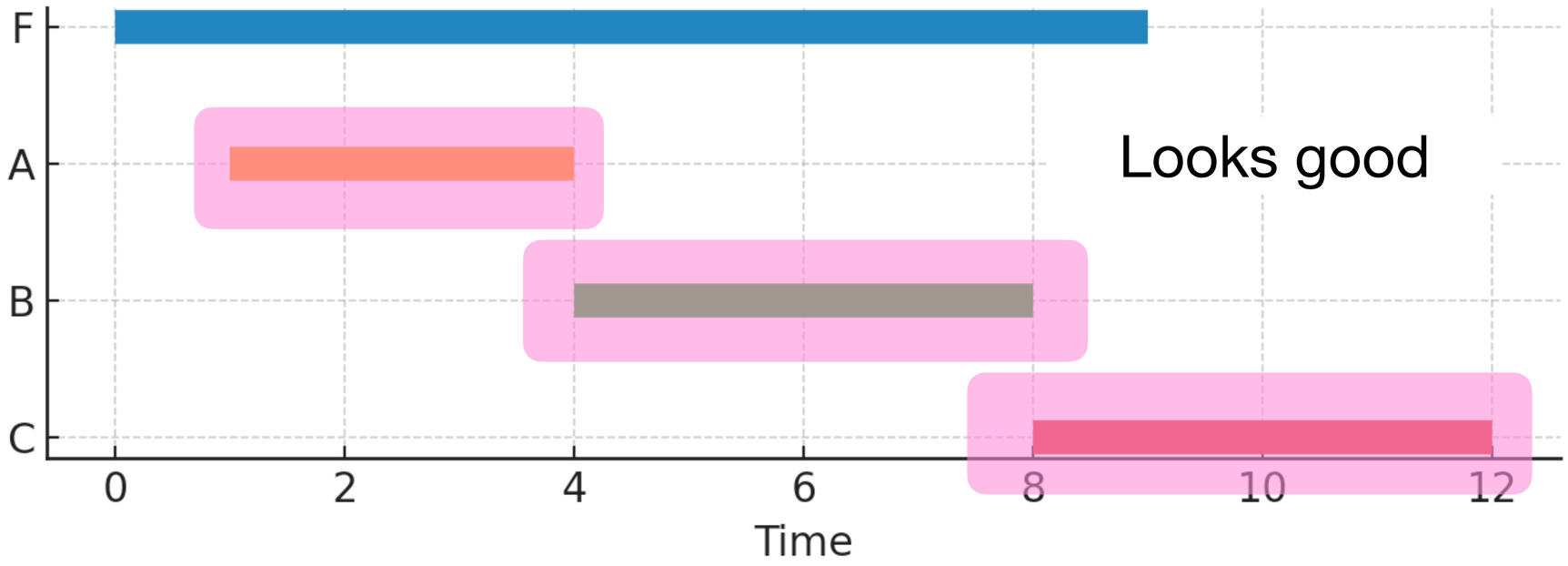
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



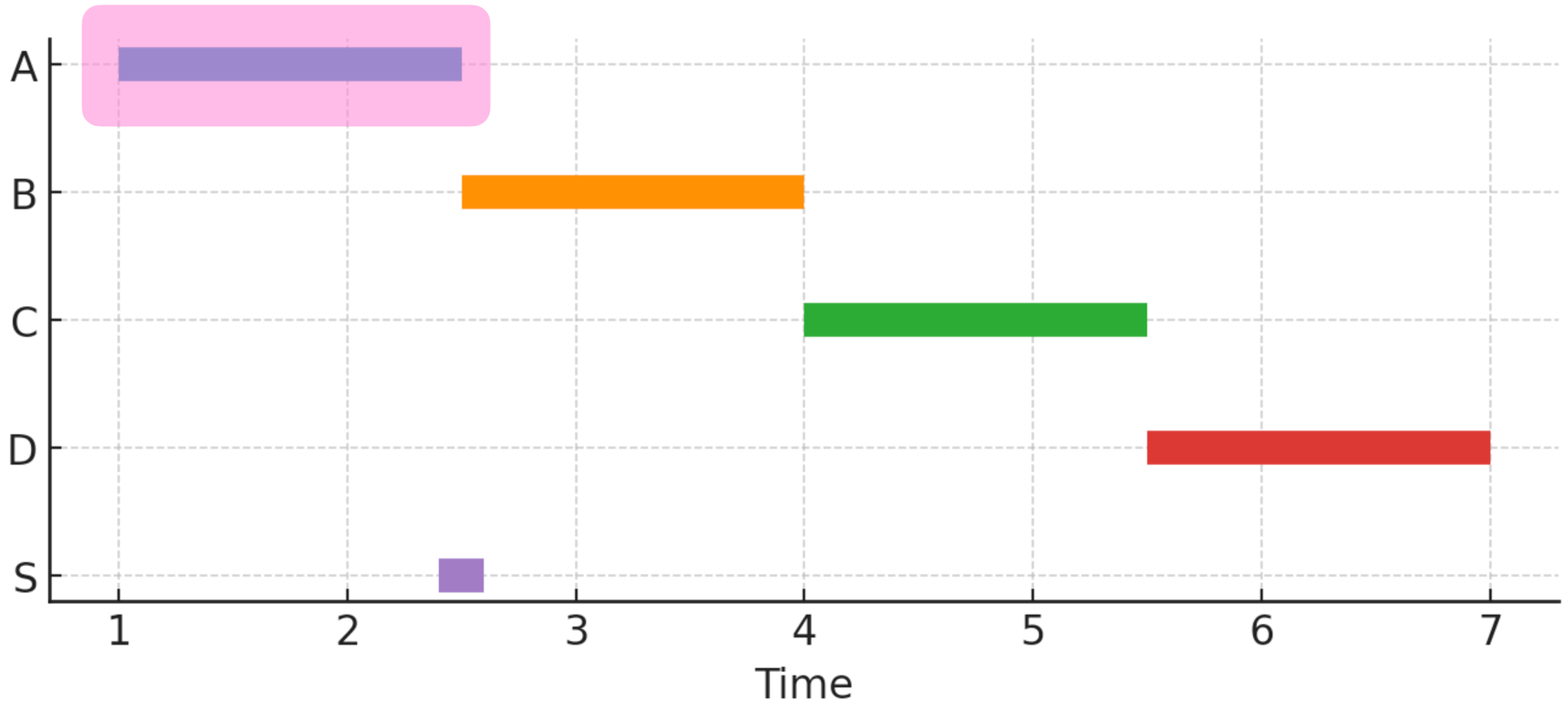
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



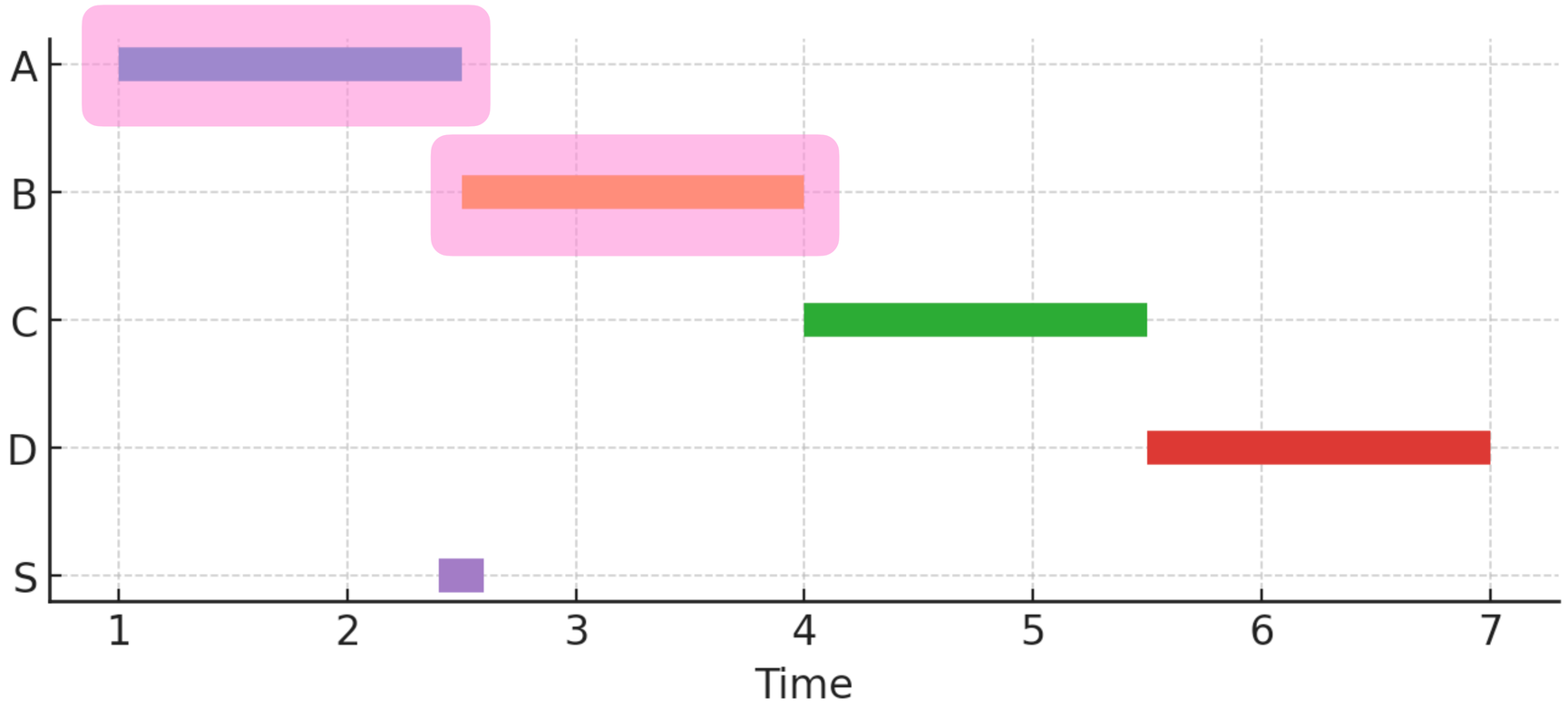
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



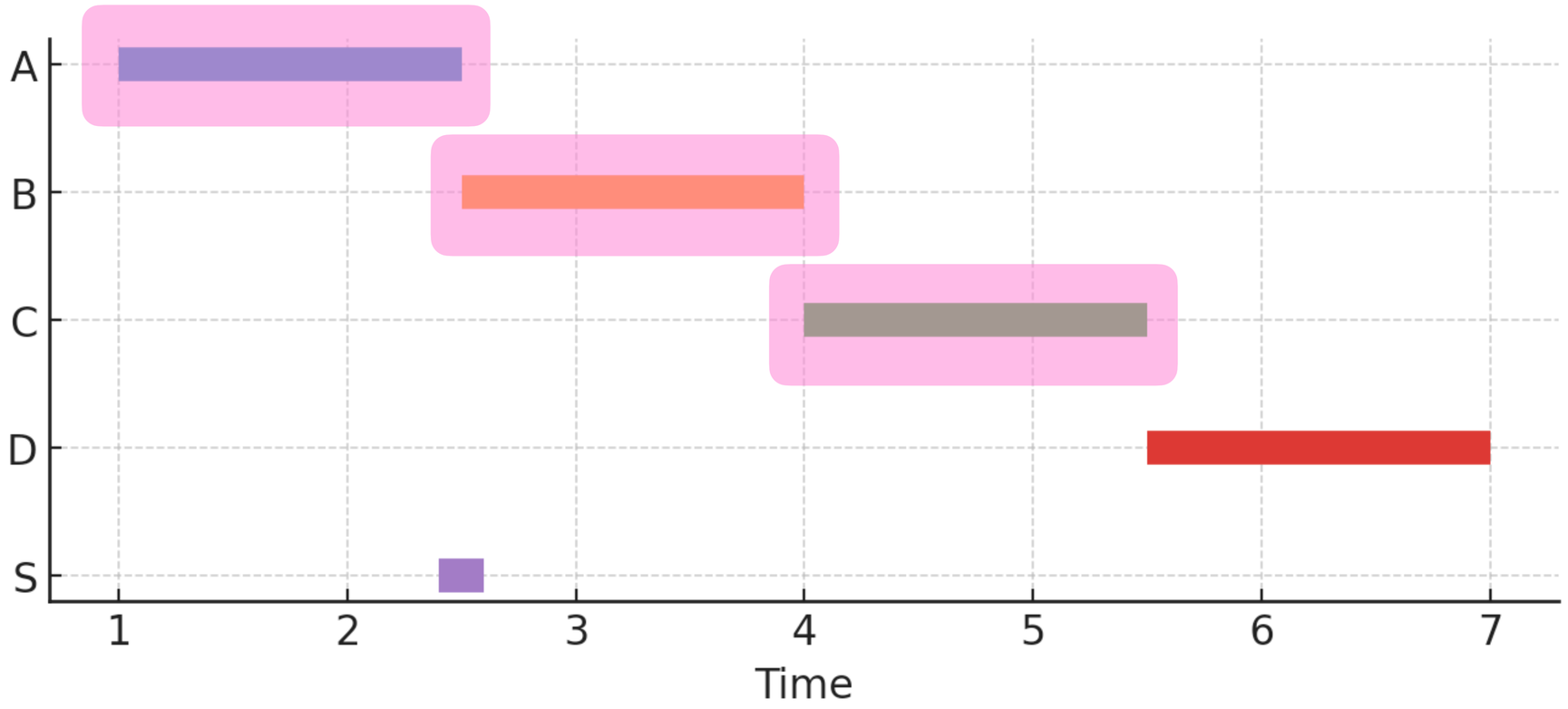
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



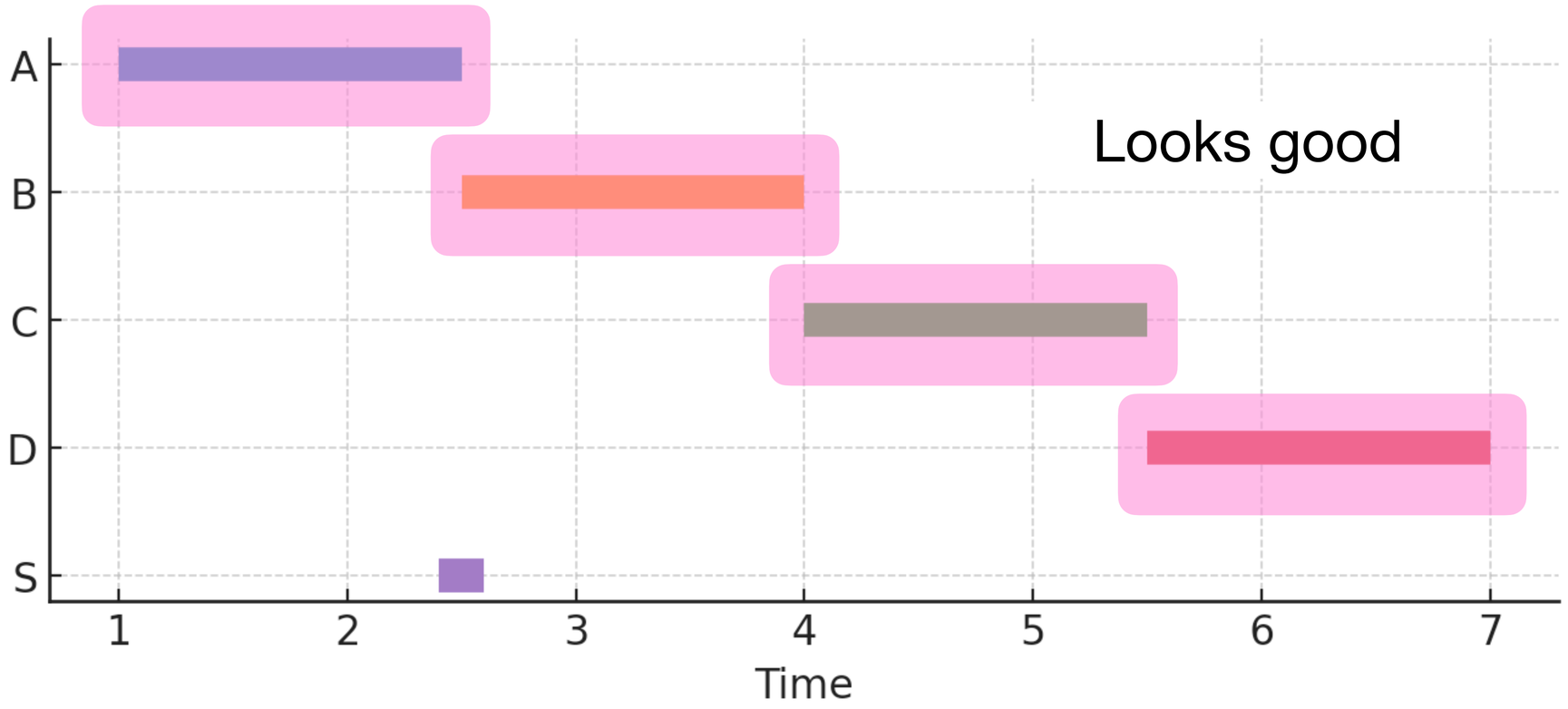
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



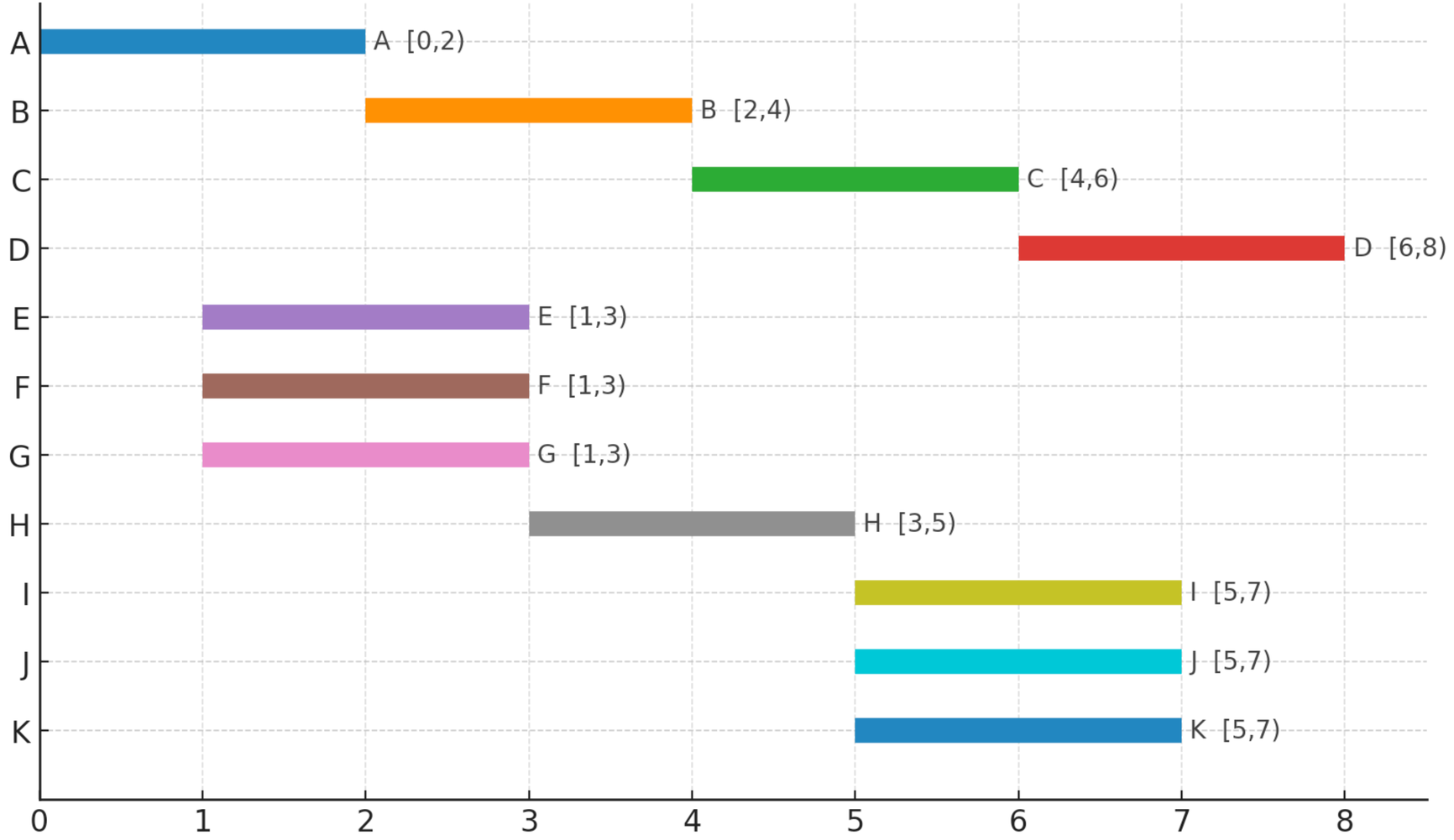
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



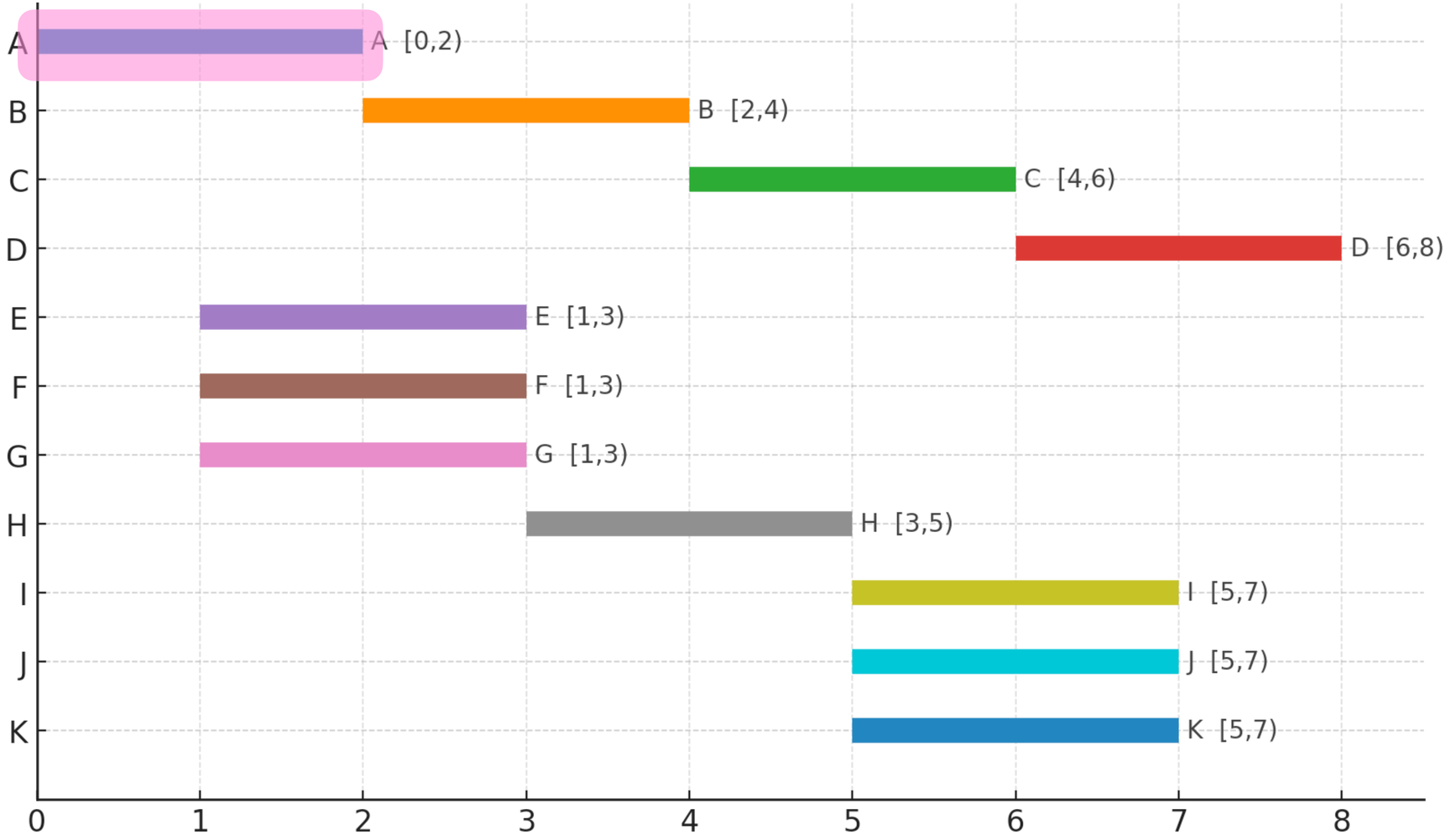
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



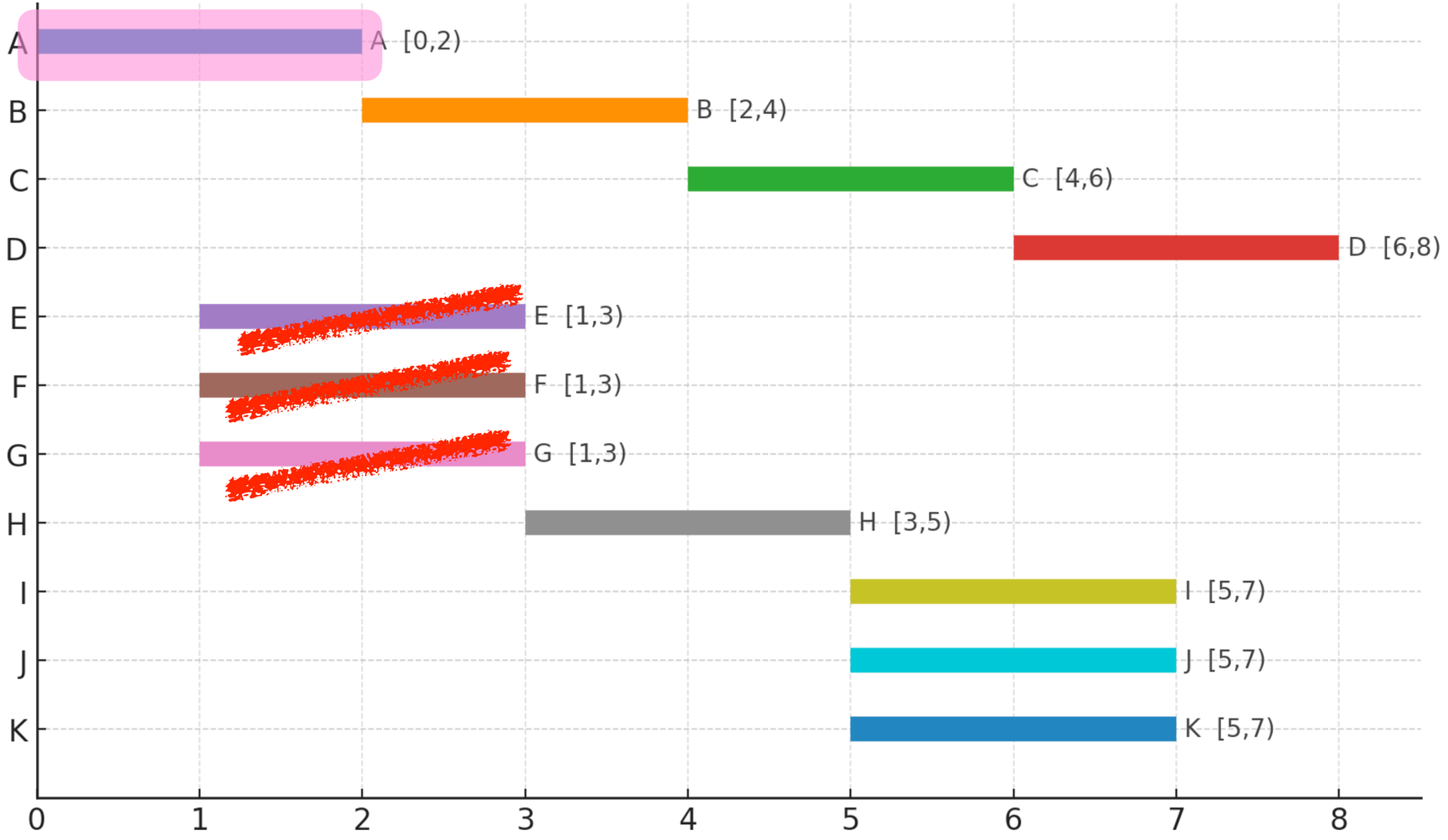
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



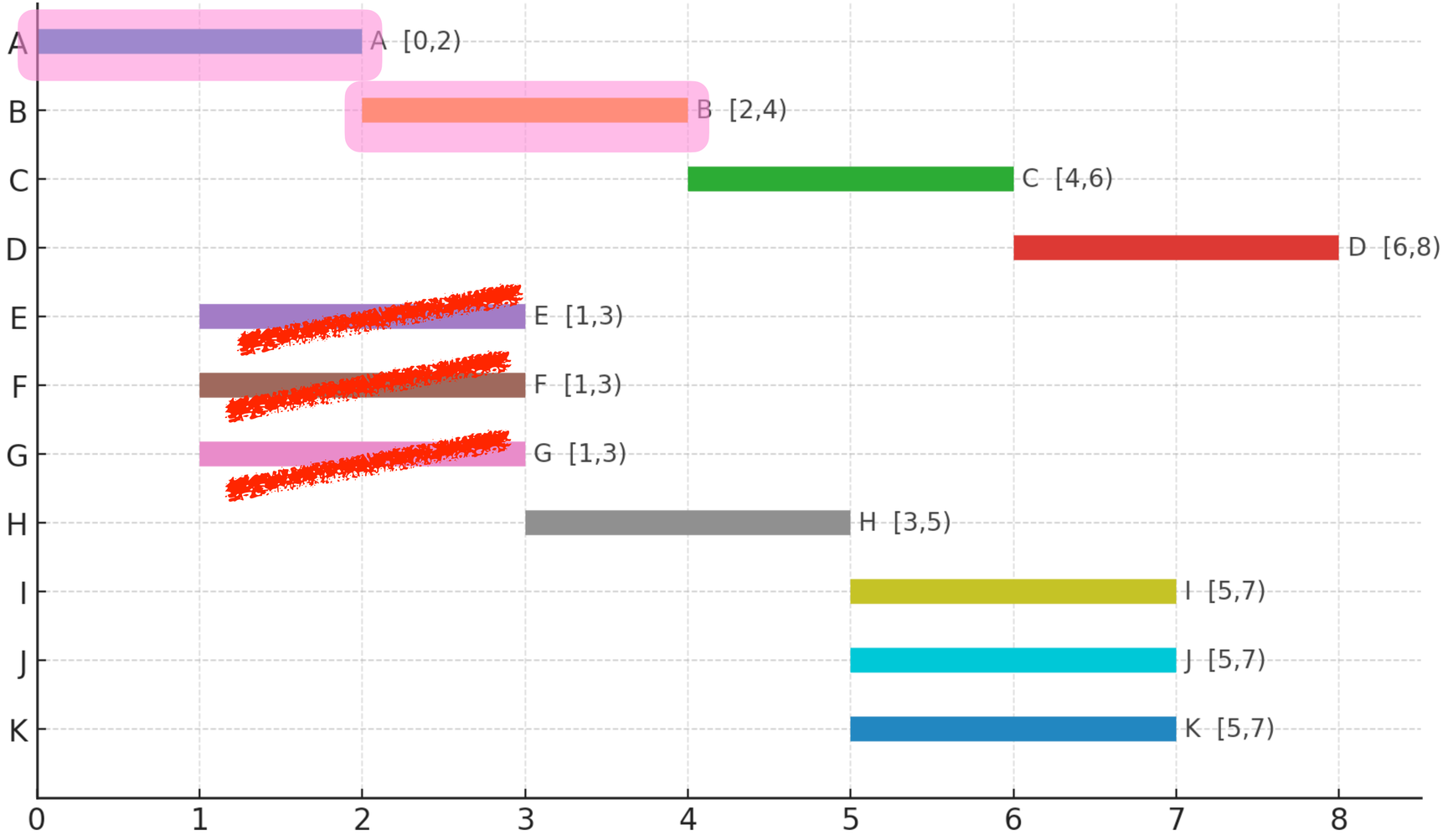
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



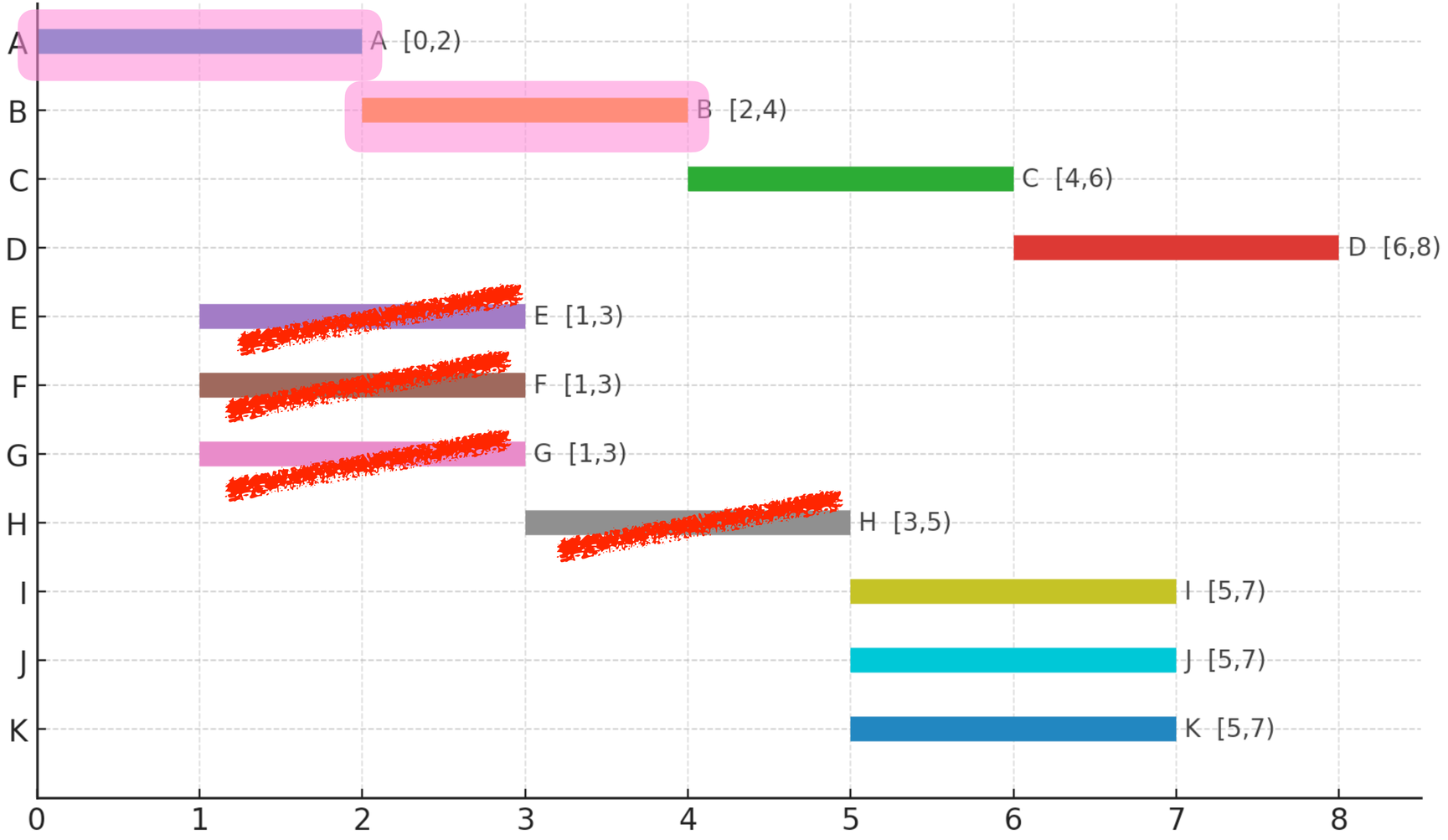
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



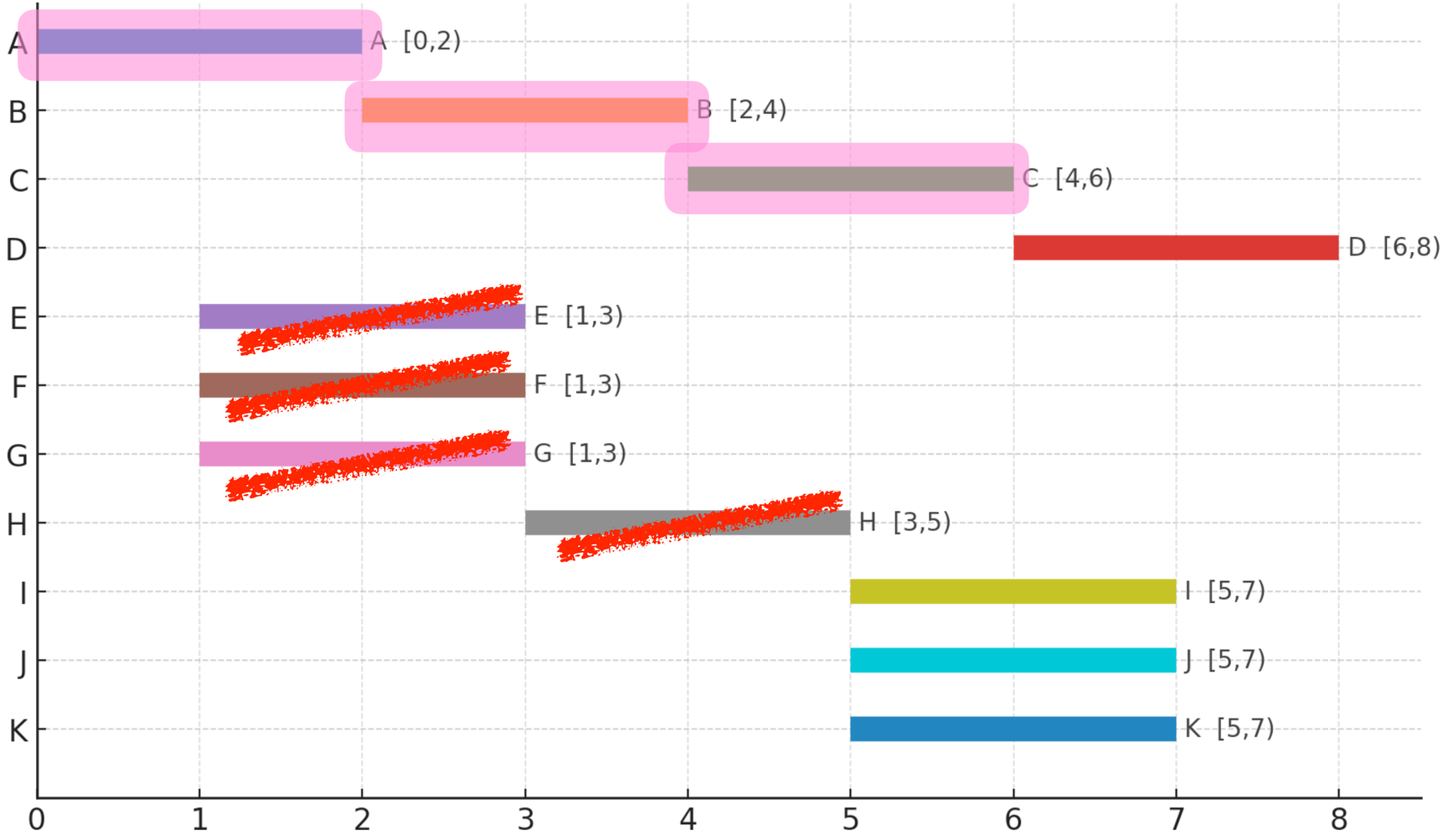
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



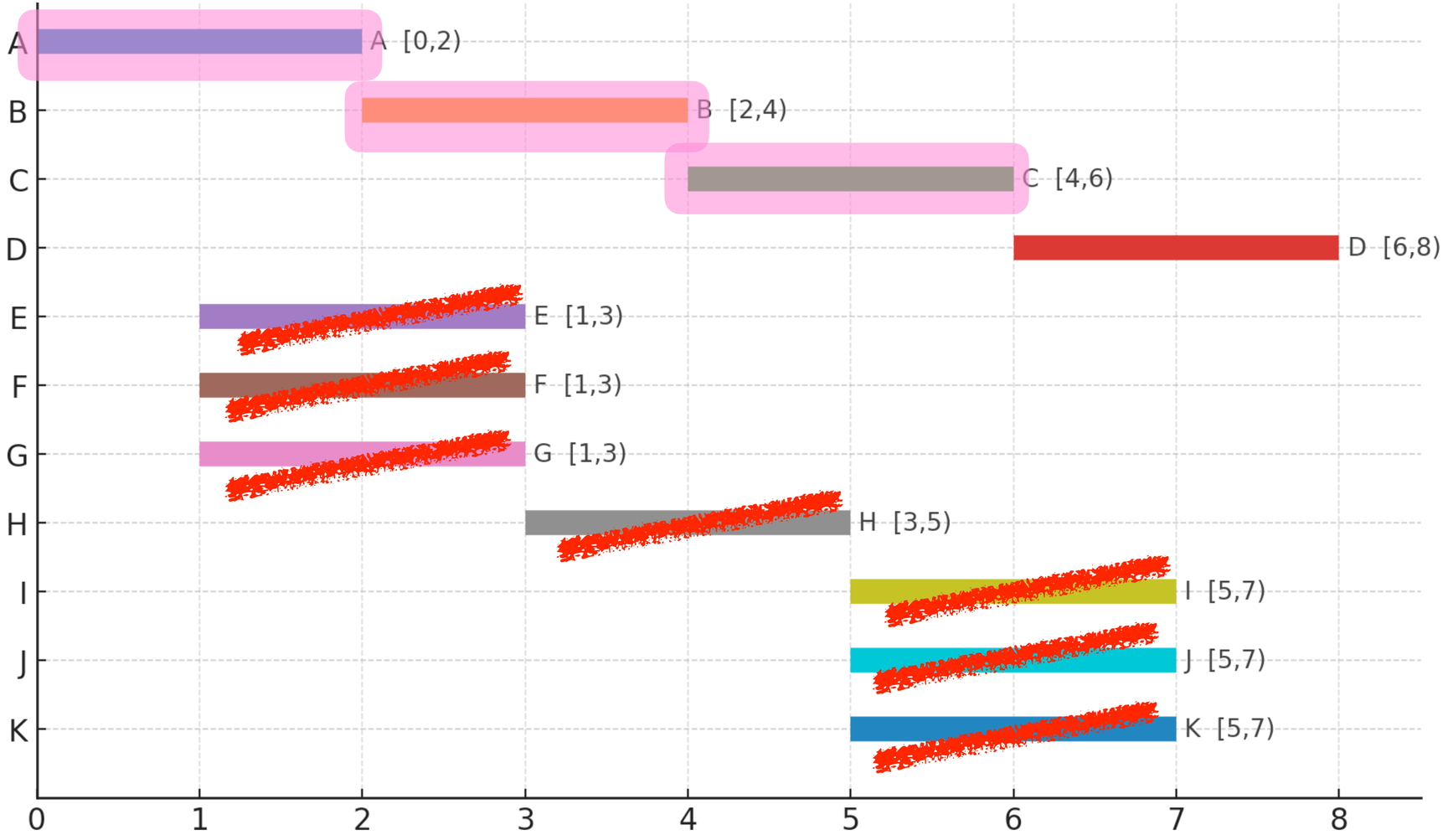
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



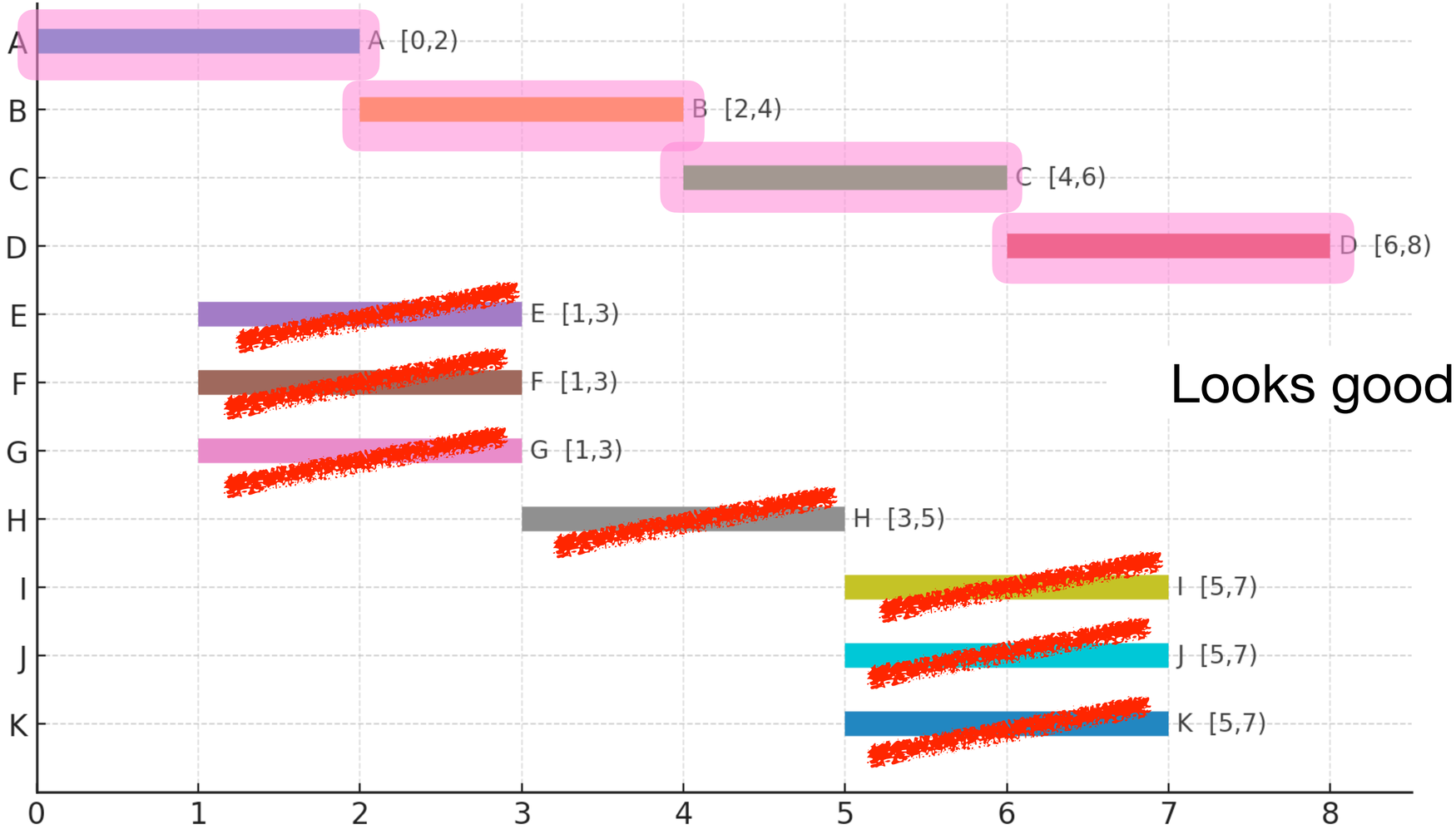
Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



Interval Scheduling: Earliest Finish

Greedily choose classes by which finishes first



Looks good