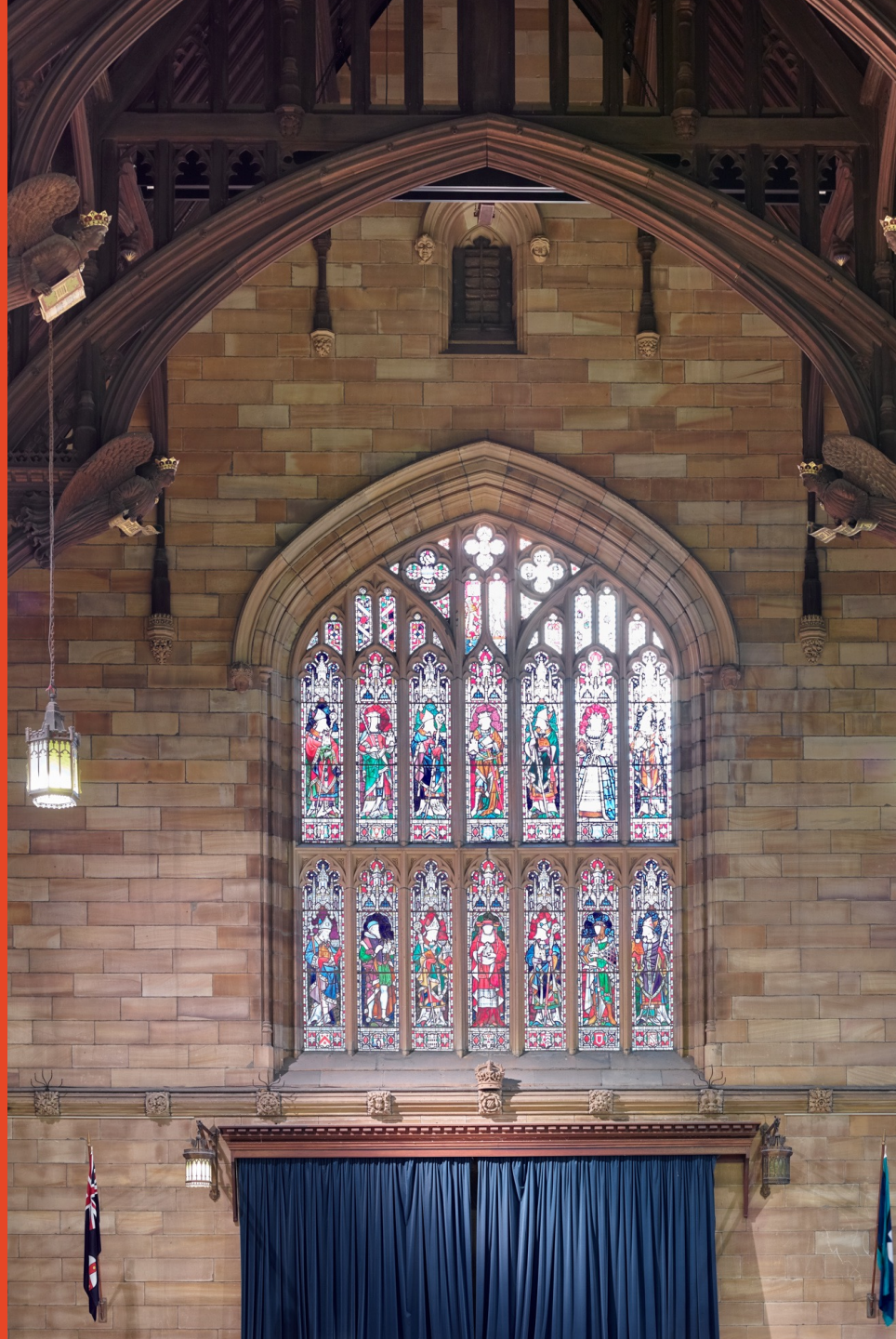
COMP2123
Data structures and Algorithms

Lecture 12: Empirical analysis

A/Prof Julian Mestre
School of Computer Science

Some content is taken from material provided by the textbook publisher Wiley.

THE UNIVERSITY OF SYDNEY



Empirical Evaluation

Worst-case analysis is the gold standard of Algorithm analysis:

- Elegant mathematical framework that drives the development of Algorithmic Theory
- It allows us to make definite predictions about how the running time of an algorithm scales with the size of the instance
- It allows us to make rough predictions about the range of instance sizes that are tractable

But sometimes worst-case analysis can lead us astray:

- Asymptotically best algorithm may hide large constants
- Worst-case behavior is not representatives of typical behavior

Empirical Evaluation

Instead of trying to bounding worst-case behavior of an algorithm, we want to evaluate the "typical performance" of an algorithm:

1. Select collection of instances
2. Run algorithm and collect data about the executions
3. Data analysis to find interesting patterns

Depending on the patterns found in 3, you may want to iterate through the process a couple of time to gather more evidence for a hypothesis or to refine a hypothesis.

Selecting Instances

1. If you have an application in mind for your algorithm, you should be able to gather representative instances.
   Your findings may not be widely applicable, but they will be most relevant for your application.
2. If no specific instances are available, use a related benchmark, e.g., Open Graph Benchmark.
   Your findings may be off for your applications, but they will be more robust.
3. If no benchmark is available, you can generate instances using randomization.
   Random instances have a lot of structure that typical instances don't.

Instance selection is key to reproducible experiments

Collect Data

1. Try to set up your environment so that the results are reproducible.
   If you are measuring running time, make sure there are program running in the background.
   Control the use of parallelism is available.

2. Run algorithm on your instances once and store results.
   Consider capping the running time of your experiments.
   Consider storing additional data beyond the result.

3. Document your environment: Computer specs, OS, compiler flags, etc.

Data Analysis

1. Identify general trends

2. Identify instance parameters (beyond size) that affect performance

3. Propose hypotheses

Coding Interview

- Do not ignore corner cases
- Pretend dictionaries take O(1) time unless otherwise stated
- Talk through you thought process
- Double check your work by coming up with test instances