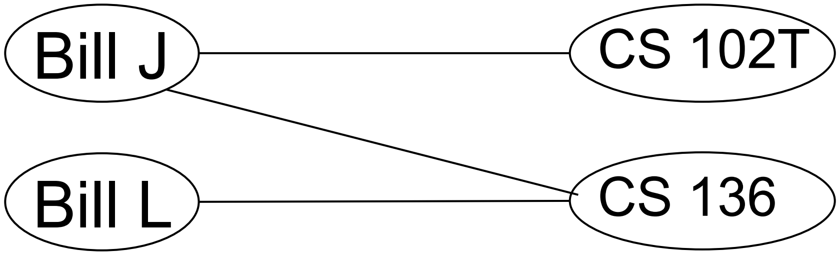
CSCI 136
Data Structures & Advanced Programming

Lecture 35

Fall 2017



Announcements

CS Holiday Party Today at 2:30 in 3rd Floor Common Room

- Final Class 😭
- Help Opportunities😀
  - TAs available this weekend
    - Sat. 3-5pm; Sun. 1-5pm
  - Review Session
    - Tuesday, Dec. 12, 1:30-2:30 pm in Physics 205
  - Office Hours
- Final Exam is Thursday, Dec. 14😬
  - 9:30-noon in Biology 112
  - Cumulative, but focused on second half of course
  - Sample exam and 2-page study sheet are on-line

Last Time

- Maps & Hashing Applications
    - “Advanced” data structures
        - Cuckoo hashing
        - Bloom Filters

Today

- Deduplication (one last hashing application)
- Course Wrap-up
  - Recap and answer any outstanding questions
- SCS Forms

Deduplication

- Imagine you are a cloud storage provider, and someone uploads the hit song `Shoot_pass_slam.mp3`
  - Millions of others will as well (Shaq Diesel went platinum…)
- Do you really want to store millions of copies of an identical file?
  - NO!* You would rather *deduplicate* extra copies
  - Map every song called `Shoot_pass_slam.mp3` to the same value?
    - Shoot_pass_slam.mp3 Shoot_pass_slam.mp3
  - The key shouldn't be the file *name*, but the file *data*

Data De-duplication Strategy

- Instead of mapping:

  `file_name` → `file_data`

- We map:

  `file_name` → `hash_of_contents`

- Then we have a separate Map that contains:

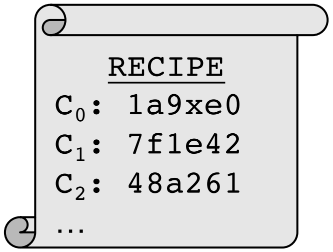
  `hash_of_contents` → `file_data`

- **Insight:** Many problems in computer science are solved by a layer of indirection!

Deduplication

- What if we aren’t storing music, but a file that is frequently modified?
  - We may not want to detect duplicates at the granularity of entire files – if even one byte changes, we store both copies
- Instead, break file into chunks and deduplicate chunks
  - Now we map:

`file_name → file_recipe`



RECIPE

C_0: 1a9xe0

C_1: 7f1e42

C_2: 48a261

…

- We only store one copy of each chunk!
- Use cases?
  - Labs where we give you starter files as a template
  - Keeping versions of your files as they evolve over time
    - Git version control system does this

Deduplication Problems

- How do we define a chunk?
  - Every n bytes, start a new chunk?
    - What if we "insert" into the middle? All data shifts right…
- What happens if chunks are really small?
  - Hashtable of fingerprints takes up as much space as data
- What if a really popular chunk gets lost/damaged?
- When do we create chunks and check for duplicates?
  - Before we write or after?
- Who saves money when deduplication saves space?

Wrapping Up

Why Data Structures?

Underlying Dictionary Structure	**put**	**get**	**space**
unsorted vector	O(n)	O(n)	O(n)
unsorted list	O(n)	O(n)	O(n)
sorted vector	O(n)	O(log n)	O(n)
balanced BST	O(log n)	O(log n)	O(n)
hash table	O(1)*	O(1)*	O(key range)

*On average---with good design---Don’t forget!

Data Structure Selection

- Choice of most appropriate structure depends on a number of factors
  - How much data?
    - Static (array) vs dynamic structure (vector/list)
  - Which operations will be performed most often?
    - Lots of searching? Use an ordered structure
      - If items are comparable!
    - Mostly traversing in arbitrary order? List
    - Process data in order you receive it? Stack/queue
  - Is worst case performance crucial? Average case?

Why Complexity Analysis?

- Provides *performance* guarantees
  - Captures effects of scaling on time and space requirements
- Independent of hardware or language
- Can guide appropriate data structure selection

Why Correctness Analysis?

- Provides *behavior* guarantees
- Independent of hardware or language
- Reduce wasted effort developing incorrect code
- A powerful debugging tool
    - Program incorrect: Try to prove it *is* correct and see where you get stuck
    - Frequently, such proofs are *inductive*

Why Java?

What makes it worth having to type (or read!)

```
Map<Airport,ComparableAssociation<Integer,
      Edge<Airport,Route>>> result = new
      Table<Airport,ComparableAssociation<Integer,
      Edge<Airport,Route>>>();
```

Why Java?

- Java provides many features to support
  - Data abstraction : Interfaces
  - Information hiding : public/protected/private
  - Modular design : classes
  - Code reuse : class extension; abstract classes
  - Type safety : types are known at compile-time
- As well as
  - Parallelism, security, platform independence, creation of large software systems, embeddability in browsers, ...

Why structure(5)?

- Provides a well-designed library of the most widely-used fundamental data structures
  - Focus on core aspects of implementation
    - Avoids interesting but distracting “fine-tuning” code for optimization, backwards compatibility, etc
  - Allows for easy transition to Java’s own Collection classes
  - Full access to the source code
    - Don’t like Duane’s HashMap---change it!

Want to Learn More?

- CS 237: Computer Organization
  - Learn about the many levels of abstraction from high-level language → assembly language → machine language → processor hardware
- CS 256: Algorithm Design and Analysis
  - We've only scratched the surface of what elegant algorithm and data structure design can accomplish. For a deeper dive, go here.
- Many CS electives require one of these two courses

Want to Learn More?

- CS 334: Principles of Programming Languages
    - There are many different types of programming languages: imperative, object-oriented, functional, list-based, logic, ... Why!? What is required to support languages of these kinds?
- CS Colloquium
    - Weekly (Fridays at 2:30pm) presentations from active researchers in CS from across the country

Thanks!

You’ve worked hard, asked great questions, and learned a lot!

Well done!

Any Questions?