Intro to COMP2611

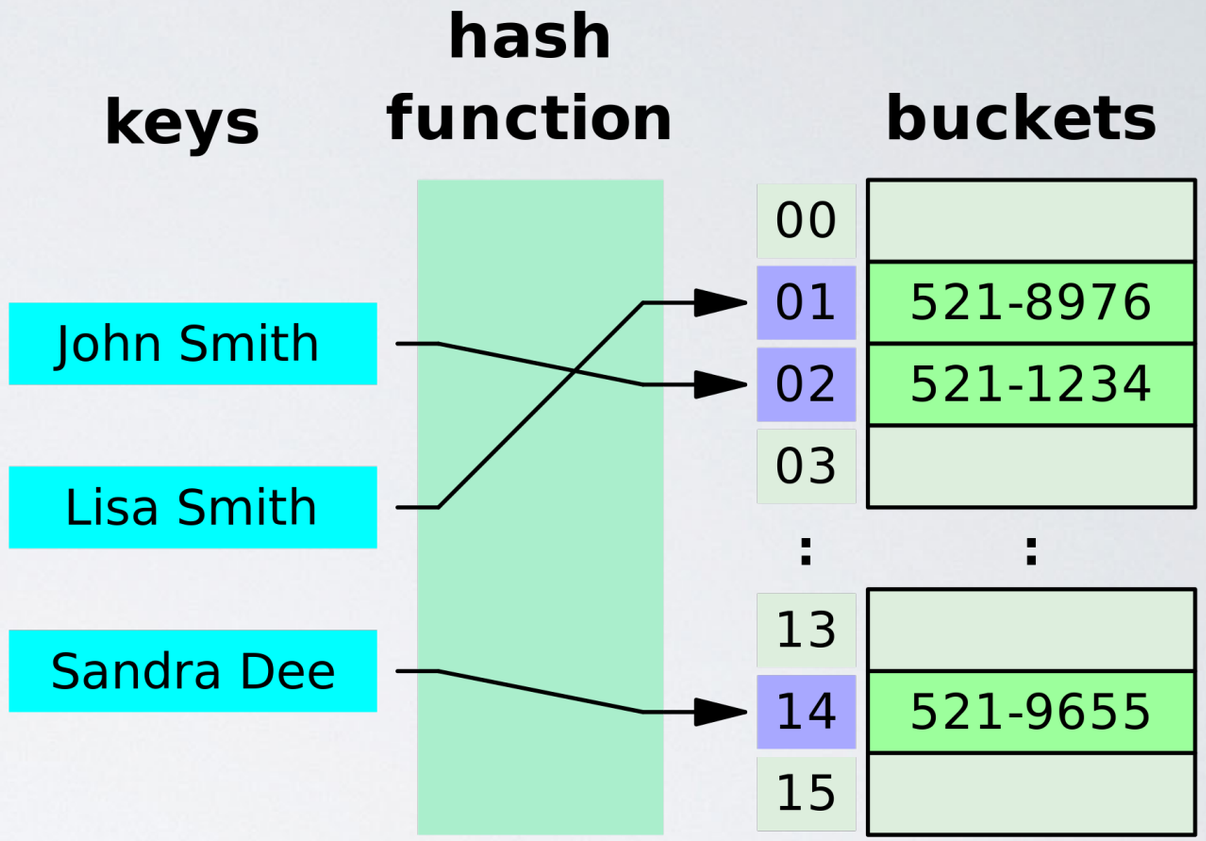
COMP2611: Data Structures
Semester I 2019/2020

What is 2611 about?

- Data Structures + Algorithms = Programmes
- Course is about fundamental data structures and the algorithms that operate and use them

Algorithms

"sequence of precise instructions for a given task"



Data Structures

"a **data structure** is a data organization, management, and storage format that enables efficient access and modification"

Why Care?

- Core of CS is built on DS and Algorithms
  - TBH, CS is actually really old even though computers are relatively new
- Important in practical engineering
  - running in seconds vs. age of the universe
  - filling up en entire hard-drive vs. a floppy
- Interesting and elegant ideas
- Important in everyday life

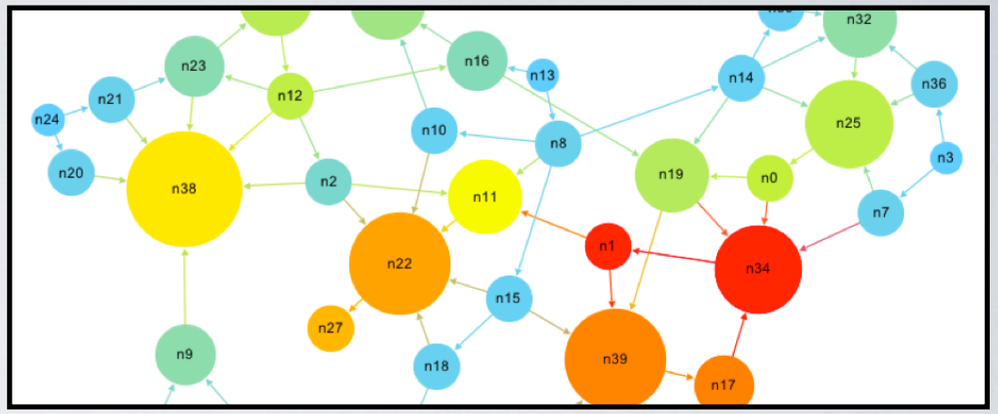
WEB SEARCH!

- Web search is fundamental to our daily web-driven labs
- Google is most well-known.
- But why?
  - Why are Google's results better?
  - What was Google's secret?
  - Google has a **better algorithm**!

Google

- Before Google
  - search engines ranked pages using keyword frequency
  - well-known and worked OK
- Larry Page & Sergey Brin (PhD students @ Stanford)
  - noticed that links were important too!
  - intuition that links conveyed information about importance
  - But what exactly? and how can you make use of it?
  - Lead them to design the PageRank algorithm

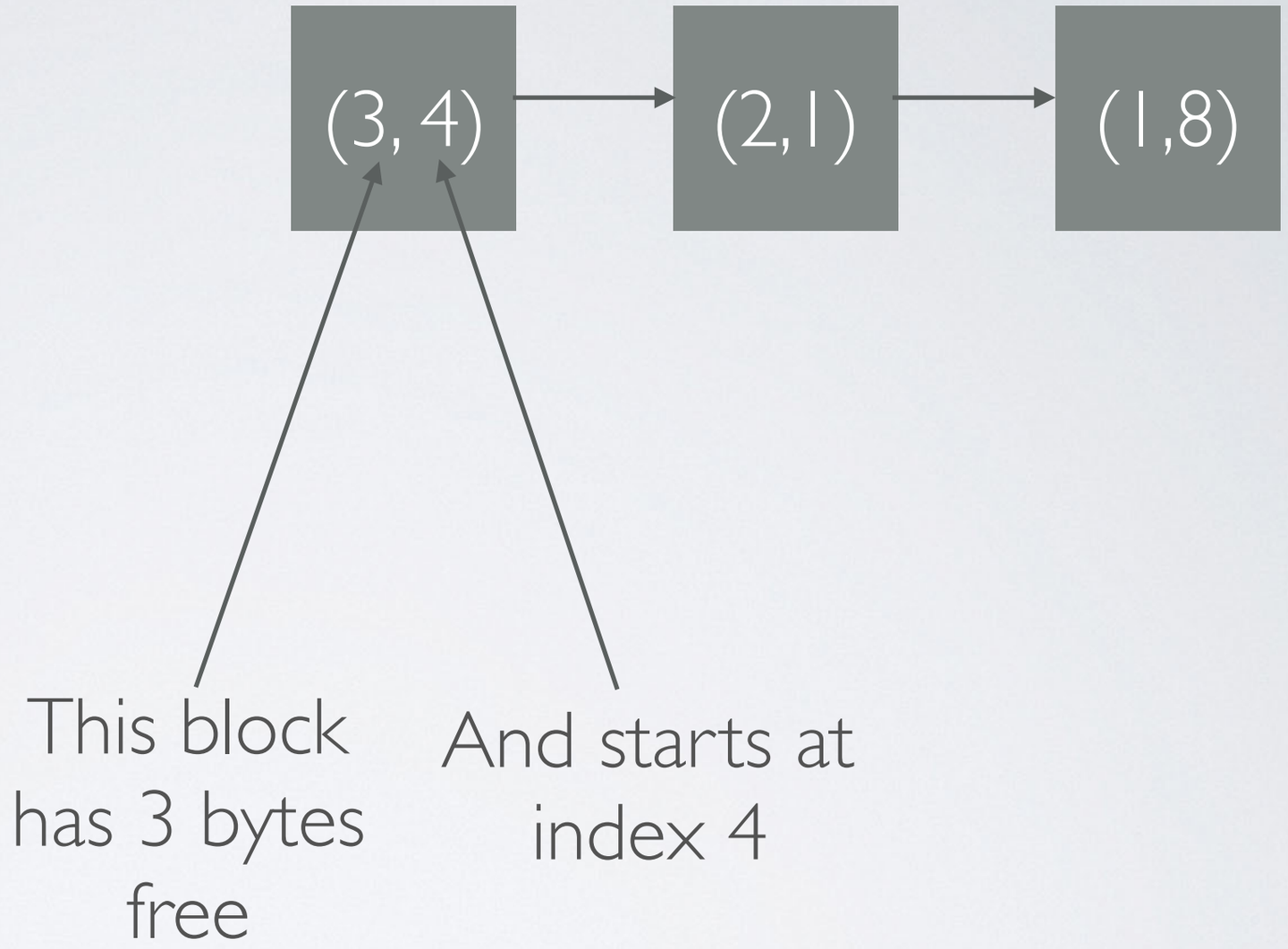
PageRank

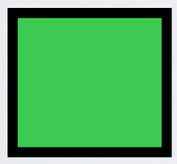


- How does PageRank work?
- Why does it work?
- How do you implement it efficiently?
  - Google indexes "hundreds of billions" of pages
    - answers and ranks in `0.5` seconds
    - processes `40,000` queries a second
    - `3.5` billion per day
- Using clever **algorithms** and **data structures**!
- You will cover this in **COMP 3610 :-)**

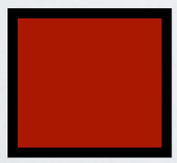
Memory Allocation by OS

- Sometimes need to allocate memory dynamically (e.g. malloc and calloc)
- OS stores free-chain
  - Link List of free blocks of memory
  - Stores memory size (in bytes) and starting location
  - Descending order by memory size





Free Byte



Occupied Byte

View Counts

#LuisFonsi #Despacito #Imposible

Luis Fonsi - Despacito ft. Daddy Yankee

6,393,605,918 views

34M 4.1M SHARE SAVE

View Counts

- Want to count unique views of a video (count-distinct problem)
- Naively, we can simply log IP address of every visitor. Let's see what happens
  - Every IPv4 address is 32 bits long
  - That means we need store at least $32 \times 6.4 \times 10^9$ bits ≈ 238 GB just to calculate the number of unique counts of Despacito. More than the video itself **:"(**
  - Would also need to scan 6.4 billion records each new view
  - How would you need to do this for real?

HyperLogLog

- Proposed by Flajolet *et al* in 2007 for count-distinct problem
- Probabilistic Data Structure that uses hash functions
- Can't get the correct answer, but can get very close
- HyperLogLog has amortised constant cost
- Uses about 2 KB of storage for 2% margin of error when number of elements $\geq 10^9$

Goals

- *Learn* fundamental algorithms and data structures
- Find and *design* new ones
- *Reason* about them
- *Use* them
- *Prepare* you for more CS
- Help you *hone* your critical thinking - help you think like a **scientist** and **engineer**

Lectures

- Cover various algorithms & data structures
  - How they work
  - Why they work
  - Analyze them
- Activities & discussions
- You are responsible for content in lecture (whether on slides or not)
- Remember you are **reading** for a degree

Textbooks

- Will pull from multiple sources
- Definitely useful
  - Dasgupta, Sanjoy, Christos H. Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2008. (freely available online)

1. Kalicharan, Noel. *Data Structures in C*. CreateSpace Independent Publishing Platform, 2008.
2. Sedgewick, Robert. *Algorithms in Java, Parts 1-5*, Portable Documents. Addison-Wesley Professional, 2002.
3. Kalicharan, Noel. *Data Structures in C*. CreateSpace Independent Publishing Platform, 2008.
4. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
5. Gayle McDowell. *Cracking the Coding Interview: 189 Questions and Solutions.* Career Cup. 2015.

Course Page

- Slides
- Notes
- Announcements
- Helpful readings
- Will host a mirror site for those of you without MyElearning access
  - https://inzamamrahaman.github.io/COMP2611-2019/

Slack

- Announcements
- Questions and answers
- Links to helpful material (blogs, Youtube videos)
- Download Slack app for mobile
- Fill out form on course site
  - Will email invite link to the Slack channel

Course Language

- C/C++ for all their merits can make simple things difficult; especially when pointers come into play
- Will use Python 3.7 as the main language for this course
  - Allows us to concentrate on understanding the algorithms rather than being bogged down by C/C++'s strictness
- Will use Anaconda distribution of Python and PyCharm Community Edition as the IDE
- Python is ***very simple***, so we expect you to learn most of the fundamental syntax on your own.
- Will provide **required** reading material for you to peruse in prep for labs next week
- However, you can use C/C++ if you want, but need to edit assignment config accordingly

Auxiliary Resources

- Try to get unstuck on your own first. That being said….
- Several auxiliary resources available
- Programming help desk
  - Should start running in a few weeks.
  - Check department for schedule
- My office hours
  - Tuesdays 11AM-1PM
  - Tuesdays 3PM-5PM
  - Wednesdays 3PM-5PM
- Questions about assignments or project:
  - Post on Slack (question might be helpful for everyone)

Assignments (15%)

- 3 Assignments
- Python/C/C++ code, proofs, analysis, …
- Marked based on
  - Test cases (total out of 10 marks per part)
  - Code quality - readability, good comments, proper naming conventions, modularity, etc…. (out of 5 marks iff you get at least 5 marks from the test cases)
  - Written component
- Assignments that break directory outline, file, input, and output specifications are regarded as **not submitted**
  - Will provide skeleton structure for you to edit
- Will provide directory structure

Assignments (15%)

- A bit of an experiment
  - Two phase marking (all but assignment 4)
    - Draft phase - mark based on test cases alone, give preliminary marks after a few days. Give week for improvements
      - Final phase - final mark on test cases, code quality, and written (typed) component
- No class-wide extensions barring extreme conditions
- Each student has a reserve of 3 grace days for the semester for assignments (only applies to final phase of marking)
- Will check code using MOSS to detect plagiarism

Assignment 4 (10%)

- Implement Data Structure from a list that we didn't cover in class
- Research said data structure. Write about its functionality, theoretical efficiency, applications, and popularity
- Design and execute experiments to measure performance of said structure vis-a-vis other approaches on realistic (generated) data. Can also comment on engineering effort vs benefit
  - Code and implementation (5%)
  - Final Report (5%)

Assignment 4 (10%)

- Final Report should be written using ACM conference template (both Word and **LaTeX** versions available)
- Report graded on analysis, methodology, **grammar and spelling.**
- In Groups no larger than 4. We might allow you to work solo if you want, but please see us in advance
- Can use language other than Python or C/C++, but check us first
- Due Date: 18th November 2019

CW Exams (20%)

- 1st Course Work
  - Tentative Examinable topics: Asymptoptic Analysis, Hashing, Matricies, Binary Trees
  - Tentative Date: 9th October
- 2nd Course Work
  - Tentative Examinable topics: Binary Trees, Sorting, Heaps, Graphs
  - Tentative Date: 13th November
- BTW, no code only pseudocode - same for finals

Participation (5%)

- Given based on degree of participation and involvement during in class discussions
- As well as on the Slack
- Also for participating and attending the labs

Content of COMP2611

- **Analysis of algorithms:** Basics of empirical and theoretical analysis, big-O
- **Hash tables:** hash functions, hash tables as dictionaries and sets, (some) collision resolution techniques and their performance characteristics
- **Matricies:** storage of general sparse matrices (LIL, COO, CSR, CSC) and special sparse matrices (triangular matricies, diagonal matrices)
- **Binary Trees:** trees as hierarchal representations and their traversal, binary trees and their traversal, binary search trees and their properties, deficiencies of bsts and their improvements
- **Heaps:** heaps as special binary trees, representing heaps in arrays, basic operations on binary heaps
- **Sorting:** formal definition of sorting, stable sorts vs unstable, comparison-based vs non-comparison based, mergesort, heapsort, quick sort, bucket sort
- **Graph algorithms:** graph definitions, types of graphs, graphs as generalizations of trees, basic graph problems and solutions (single-source shortest path, all pairs shortest path, mining spanning tree, topological sorting)

Collaboration

- Encouraged to discuss assignments **:-)**, but
  - **Write up assignments by yourself**
  - **Code by yourself**
  - **No sharing of code or pseudocode**
- **No collaboration between groups on Project**

- Signed Plagiarism declaration must be signed and sent in with assignment
- We will use code similarity tester
- Random live audits
  - might ask you "what would happen if we did X to your code?"

Email Policy

- Unless matter is private post on the Slack
  - Your question might help the entire class
  - Will get a faster response
  - Feel free to chime into other students' questions
- Emails should follow proper netiquette:
  - https://bowvalleycollege.libguides.com/c.php?g=10214&p=52001

References

- Slide #2
  - A statue of Muhammad ibn Musa al-Khwarizmi; a persian scholar from the 9th century
  - "Algorithms" is derived from "Algoritmi" which is the Latin translation of his name
  - Worked in mathematics, astronomy and geometry
  - Founded the field of Algebra
- Slide #3
  - Data Structures image from Wikipedia
- Outline inspired by Brown's CS16