Who am I?

- Born & raised in Toronto
- Computer Science Education
  - high school: late 70's
  - B.Sc. (McGill) early '80s
    Major in Math/Minor in CS
  - M.Sc. in CS (U.Toronto) late '80s
  - Ph.D. (McGill) early '90s
  - Fortran cards
  - Pascal, LISP
    mainframe terminals
  - C/unix workstation

"Work" Experience in CS

- post doc 1 - "basic research" in computer vision
  - NECI Princeton NJ
  - mid 90's (birth of WWW, Java)
- post doc 2 - "basic research" in human vision
  - Max Planck Institute, Tübingen Germany
  - late 90's
- prof here since 2000
  - Matlab, C
  - Java (2009)
  - Python (2013)

Research Interests

- Computer and human vision
- Applied Perception in Computer Graphics

Who are you?

- U0
- U1 new: 15
- U1 returning: 30
- U2: 85
- U3: 60
- visiting: 5

Total: 200
Who are you?

- B. Sc.: 90
- B. Eng.: 25
- B. Soft. Eng.: 30
- B. Arts: 35
- B. Arts & Sci.: 5
- B. Com.: 10
- Other: 5

Course Resources

- My Courses (private)
  - Discussion boards
  - Submit assignments
  - Grades

- Course web page (public)
  - Official course outline, slides, exercises, etc.

How much time for average 251 student to get a B? I assume you are working for 40 hours a week and you are taking five courses:

- 8 work hours per week per course
- *13 weeks
- 104 hours total, which breaks down to:

  - 40 hours of scheduled lecture time (3 hours per week)
  - 40 hours of review/exercises, including studying for midterm exams (3 hours per week)
  - 25 hours for 4 assignments (‘amortized’ 2 hours per week)

To get an A, you need more....

COMP 251 versus COMP 252 (Honours)

COMP 251 has 200 students. COMP 252 has about 20 students, roughly.

COMP 252 covers roughly the same material as COMP 251 but covers it more quickly and in more depth. COMP 252 is typically taught by a prof who does research in the area of data structures and algorithms. (Luc Devroye is teaching it this semester, for example.)

Many COMP 252 students go on to graduate school (MSc and/or PhD in CS).

COMP 251 assignments will be almost entirely programming (Java). COMP 252 assignments typically do not require any programming.

First few lectures: Data Structures

- Balanced search trees
- Hash tables
- (binary) heaps
  - ...

To warm you up, let’s review some basics from COMP 250.
Q: What is the difference between an ADT (abstract data type) and a data structure (concrete data type)?
A: Think Java interface vs. class.
ADT only specifies what the client/user sees.
Data structure tells you about the implementation (what’s “under the hood”).

Examples

<table>
<thead>
<tr>
<th>ADT</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>array, linked list</td>
</tr>
<tr>
<td>stack</td>
<td></td>
</tr>
<tr>
<td>queue</td>
<td>heap, sorted array, ....</td>
</tr>
<tr>
<td>priority queue</td>
<td></td>
</tr>
<tr>
<td>map</td>
<td>hash table, ....</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

List (ADT)

- [3, 5, 2, 11, 4]
- ['cat', 'dog', 'coffee', 'tea']

- get (index)
- add (index, object)
- remove (index)

Sorted Array (data structure)

- get (index) $O(1)$
- get (object) e.g., return index $O(n)$
- add (index, object) $O(n)$
- remove (index) $O(n)$
**Linked List (data structure)**

- `cat` ➔ `dog` ➔ `coffee` ➔ `tea`

  - get (index)
  - get (object) \( \Theta(n) \)
  - add (index, object)
  - remove (index)

*Good for stacks & queues*

**ADT**

- **Stack**
  - `push object`
  - `pop ()`

- **Queue**
  - `enqueue object`
  - `dequeue ()`

- **Priority Queue**
  - `add (key, object)`
  - `remove Min ()`

*array* 

*linked list* 

*ordered array* 

*heap* 

*binary search tree*

---

Sometimes the boundary between ADT and data structure is unclear and arbitrary.

In Comp 251, we will use many partially implemented data structures — often just enough details to discuss \( O() \).

**Binary Search Tree (next lecture)**

- **Binary Search Tree**
  - one key at each node
  - two nodes cannot have same key
  - for any node, keys in left subtree < key at that node < keys in right subtree

**BST:** you should review how to ...

- find the minimum key
- find the maximum key
- add a key
- delete a key
- traverse BST to visit keys in order
Map

Keys

Values

1. \{ (key, value) \}
2. For each key, there is one value (but two keys might map to same value)

Map ADT

- get(key)
- put(key, value)
- remove(key)
- containsKey(key)
- containsValue(value)
- ...

see Java Map interface

Map data structures

- linked list

\[
\begin{array}{c}
\text{put (key, value)} \quad \Theta(1) \\
\text{get (key)} \quad \Theta(n)
\end{array}
\]

Map data structures

- binary search tree

\[
\begin{array}{c}
\text{put (key, value)} \quad \Theta(\log n) \quad \text{if balanced} \\
\text{get (key)} \quad \Theta(\log n)
\end{array}
\]

Map data structures

- hash table

\[
\begin{array}{c}
\text{put (key, value)} \quad \Theta(1) \quad \text{but we cannot get keys in order}
\end{array}
\]

http://whatismyipaddress.com/
Resources for this lecture

- my COMP 250 lectures
  - lists (4-6), BST (18-21), hashing (31-32)
- Sedgewick Coursera Algorithms 'symbol tables' (maps)
- Roughgarden Coursera Algorithms
  - weeks 5, 6

Next lecture

- I will introduce balanced search trees
  - (AVL trees, 2-3 trees, but not red-black trees...)
- prepare by reviewing BST's from COMP 250.