COMP 251

Algorithms & Data Structures

Professor: Michael Langer

Who am I ?

- Born & raised in Toronto
- (Computer Science) Education
  - high school late 70's — Fortran (cards)

http://homepage.cs.uiowa.edu/~jones/cards/collection/hp9320-2051big.gif

  - BSc (McGill) early '80s — Pascal, LISP (mainframe-terminals)
    Major in Math / Minor in CS
  - MSc in CS (U. Toronto) late '80s
  - PhD (McGill) early 90's
  } C / unix (workstation)

"Work" Experience in CS

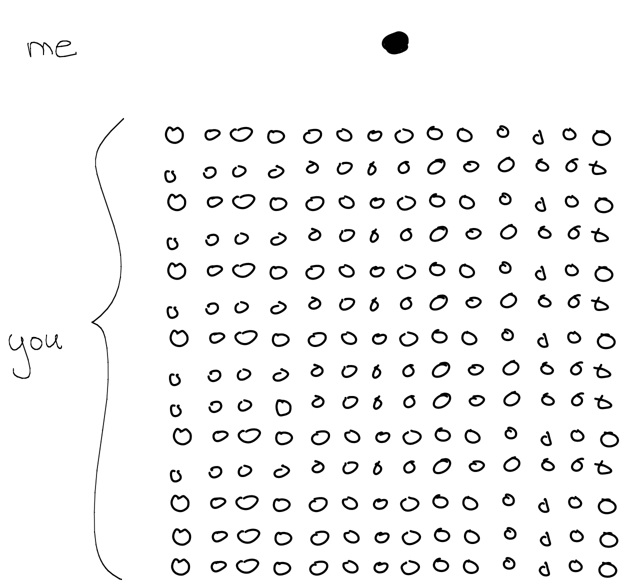
- post doc 1 - "basic research" in computer vision
  @ NECI Princeton NJ.
  mid 90's (birth of WWW, Java)
- post doc 2 - "basic research" in human vision
  @ Max Planck Institute, Tübingen Germany
  late 90's
- prof here since 2000 — Matlab, C
  Java (2009)
  Python (2013)

Research Interests

- Computer and human vision

- Applied Perception in Computer Graphics

(participants needed for lab experiments!



Who are you ?

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

  http://www.cim.mcgill.ca/~langer/251.html

  - official course outline, slides, exercises, etc.

Will lectures be recorded ?

Yes !

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, lets review some basics from COMP 250.

Tim Roughgarden's data structures video (motivation)

https://class.coursera.org/algo-004/lecture/61

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

ADT	Data Structure
list stack queue	array, linked list
priority queue	heap, sorted array, ...
map	hash table, ...
⋮	⋮

List (ADT)

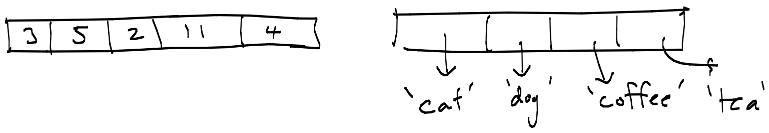
[3, 5, 2, 11, 4]

['cat', 'dog', 'coffee, 'tea']

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

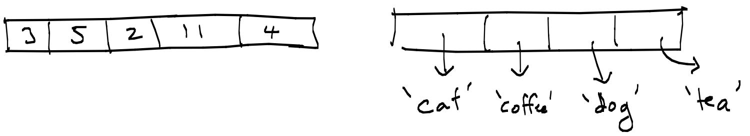
define what the user/client can do

Array (data structure)



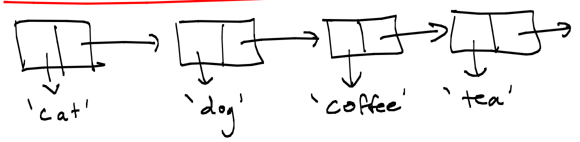
get (index)		O(1)
get (object)	e.g. returns index	O(n)
add (index, object)		O(n)
remove (index)		O(n)

Sorted Array (data structure)



get (index)		O(1)
get (object)	e.g. binary search	O(logn)
add (index, object)		O(n)
remove (index)		O(n)

Linked List (data structure)



get(index)
get(object)
add(index, object)
remove(index)
} O(n)

good for stacks & queues {
removeFirst()
addFirst(object)
addLast(object)
} O(1)

ADT

Stack
- push(object)
- pop()

Queue
- enqueue(object)
- dequeue()

} • array • linked list

Priority Queue
- add(key, object)
- removeMin()

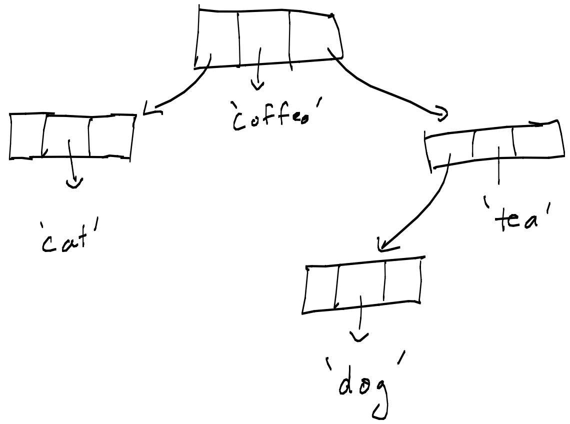
} • ordered array • heap • binary search tree

data structure

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)



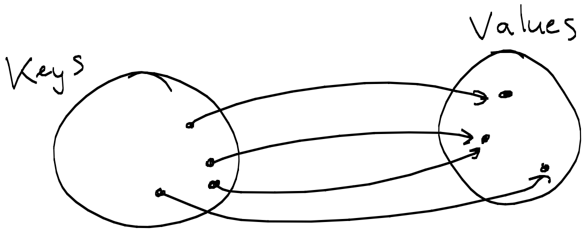
Binary Search Tree

- 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



- { (key, value) }
- For each key, there is one value (but two keys might map to Same value)

Map

Keys may be:

- social insurance (security) number
- name
- IP address

http://whatismyipaddress.com/

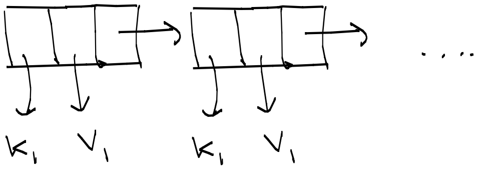
Map ADT

- get (key)
- put (key, value)
- remove (key)
- contains Key (key)
- contains Value (value)
- ....

see Java Map interface

Map data structures

- linked list

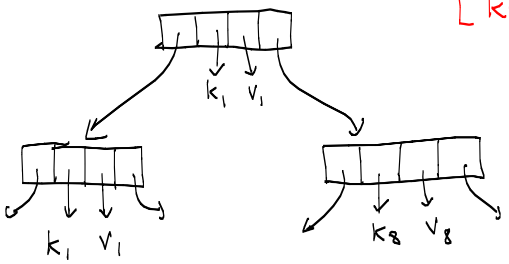


put (key, value) O(1)
get (key) O(n)

Map data structures

- binary search tree

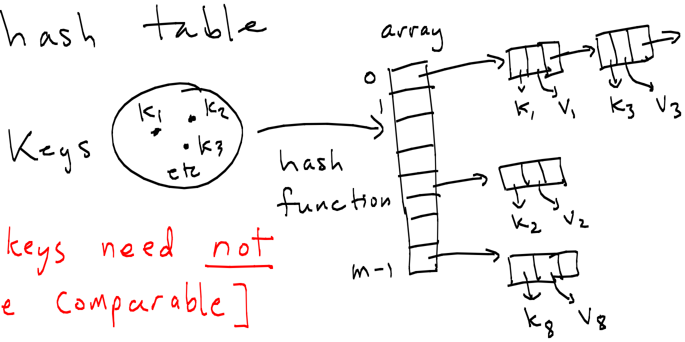
[keys must be comparable]



put (key, value) O(log n) }
get (key) O(log n) }
if balanced

Map data structures

- hash table



[keys need <u>not</u> be comparable]

put (key, value) O(1) }
get (key) O(1) }
but we cannot get keys in order ☹

Resources for this lecture

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

Next lecture

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