CS 2100 - Recap of our semester

Announcements

- HW due now
- Practice final is at front of the room
- Review session: Monday during last class
- Final exam: Wednesday at 8am
Data Structures Covered

- stacks
- queues
- lists
- vectors
- sorting
- searching
- trees:
  - BST
  - AVL Trees (balanced binary trees)
  - Huffman trees
- heaps
- hashing
- graphs
Data Structures

Some data structures have limited functionality, but as a result are extremely efficient.

Ex.
- stack
- queue
"Full-featured" data structures

More versatile data structures have trade-offs:

- to get something faster, you sacrifice speed in another area.

Ex:

- list vs insert/remove
- vector vs access time
- AVL graphs: space vs access time
Randomized or Expected

Some work well in practice but have no theoretical guarantees.

Ex:

- hash functions
- quick sort
- amortized push-back vector
So which is best?

Ans: Depends.
In-order insertions: insert: 1, 2, 3, 4, ..., n
Reverse Order insertions: $n, n-1, n-2, \ldots, 1$
Random Inserts

Graph showing the performance of different data structures for random inserts.
Note: Hashing is fast.
The "randomness" of the hash function even hides order of elements.

Caution, though: They don't have all the extra functionality of others.