Data Structures Summary

Data Structures Summary

- By this point we’ve now examined four different ways to store sets of data:
    - *Arrays*
    - *Linked lists*
    - *Hash tables*
    - *Tries*

- There are even some variations on these (trees and heaps, quite similar to tries, stacks and queues quite similar to arrays or linked lists, etc.) but this will generally cover most of what we’re looking at in C.

Data Structures Summary

- How do all of these data structures measure up? How do you know which to choose for your situation?

- Usually it’s a matter of weighing the pros against the cons. Let’s consider some of the important metrics for each.

Data Structures Summary

- Arrays
    - Insertion is bad – lots of shifting to fit an element in the middle
    - Deletion is bad – lots of shifting after removing an element
    - Lookup is great – random access, constant time
    - Relatively easy to sort
    - Relatively small size-wise
    - Stuck with a fixed size, no flexibility

Data Structures Summary

- Linked lists
  - Insertion is easy – just tack onto the front
  - Deletion is easy – once you find the element
  - Lookup is bad – have to rely on linear search
  - Relatively difficult to sort – unless you're willing to compromise on super-fast insertion and instead sort as you construct
  - Relatively small size-wise (not as small as arrays)

Data Structures Summary

- Hash tables
    - Insertion is a two-step process – hash, then add
    - Deletion is easy – once you find the element
    - Lookup is on average better than with linked lists because you have the benefit of a real-world constant factor
    - Not an ideal data structure if sorting is the goal – just use an array
    - Can run the gamut of size

Data Structures Summary

- Tries
  - Insertion is complex – a lot of dynamic memory allocation, but gets easier as you go
  - Deletion is easy – just free a node
  - Lookup is fast – not quite as fast as an array, but almost
  - Already sorted – sorts as you build in almost all situations
  - Rapidly becomes huge, even with very little data present, not great if space is at a premium