CSE332: Data Abstractions

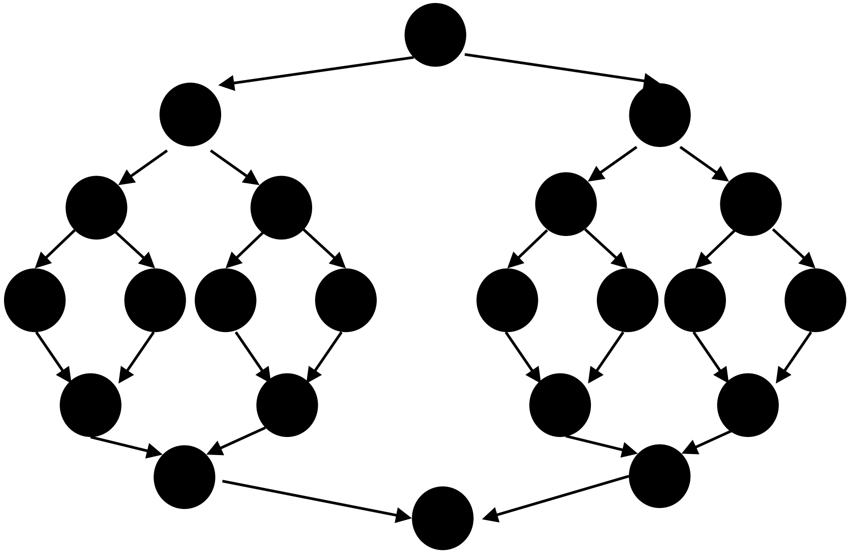
Lecture 28: Course Wrap-up

Tyler Robison

Summer 2010

Merging

So, we've spent the quarter exploring many different data structures; time to merge back together:



Some unexplored nodes

- Alternative data structures for balanced trees, priority queues
- Disjoint-set data structure (union-find)
- AVL deletion
- Max-flow / min-cut graph algorithms
- Huffman coding (compression)

What we've covered

- Really just a beginning
  - Many other priority queues: skew heap, leftist heap, binomial queue, …
  - Many other dictionaries: red/black tree, splay tree, …
    - Many variations on hash tables
    - Many variations on B-Trees
  - Many other sorts, graph algorithms, etc.
  - Just scratched the surface of concurrency
  - Run-time/recurrence-relation analysis go much deeper

What we've covered

- But we've covered the foundation
  - Many priority queues, but binary heaps are among the most common
  - Many dictionary data structures; among which balanced trees and hash tables are most important
  - Graph theory is an enormous area, but the basics will get you a long ways
  - Parallelism/concurrency issues covered are all that's needed for many situations

And now

- You should now be equipped to
  - Learn new data structures & ADTs
    - ‘Once you learn one programming language, others come much easier’
  - Understand/analyze run-times
  - Understand uses & trade-offs
    - In general make more informed use of them in programming
- Have had experience writing/debugging/testing data structures & parallel/concurrent software
- More experience with proofs
  - Maybe not up to proving P!=NP, but still
- Know a bunch of tools, and know how to pick the right tool for the job

Applications

- Hash tables: Everywhere
  - Seriously, everywhere
  - If you're interviewing for a programming job/internship, hash table questions are likely candidates
- Data Bases: B-Trees under the hood
- Graphs show up in CS again and again
  - Just very useful for modeling stuff:
    - Computer networks
    - Power grids
    - Road systems
    - Social networks
    - Knowledge/concept maps

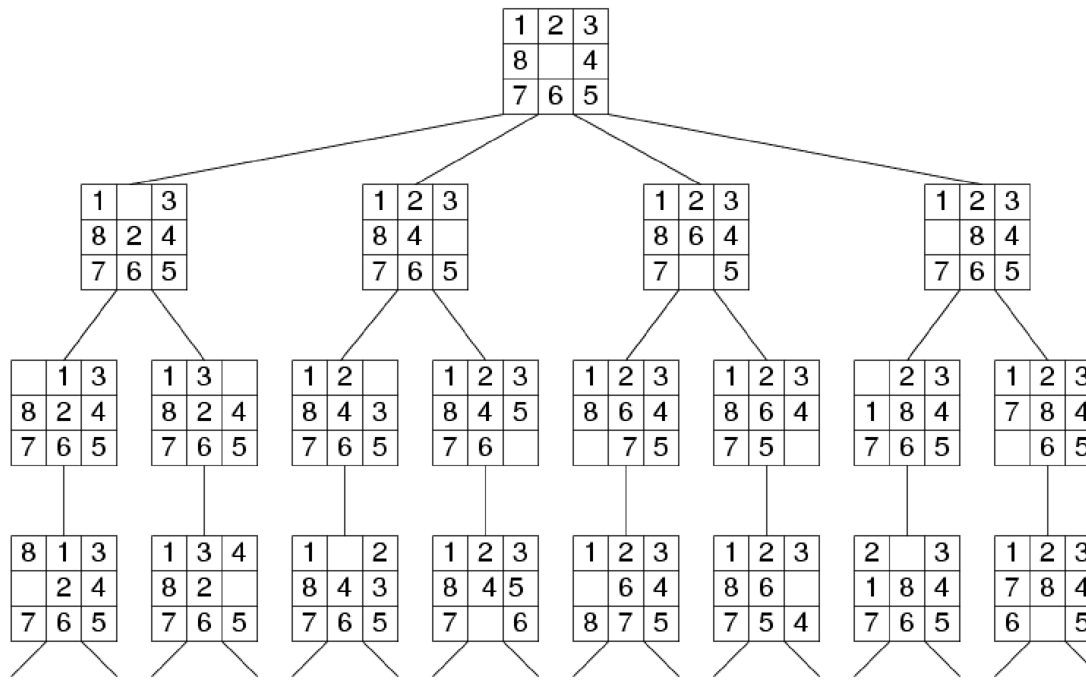
Applications

- Parallelism & Concurrency
  - Increasingly important
    - Many more cores is likely the future of computing hardware
    - Programming for many cores is going to be important
  - Speed, and thus parallelism, hugely important in many areas
    - Games (Xbox 360: 3 cores)
    - Servers
    - Scientific/mathematical simulations
    - Many others; anything concerned with speed
  - Concurrency problems pop up even in some simple Java applications
    - Ex: Handling GUI events
- Big Oh analysis: ubiquitous in CS

- Now some specific examples in AI; trees & graphs

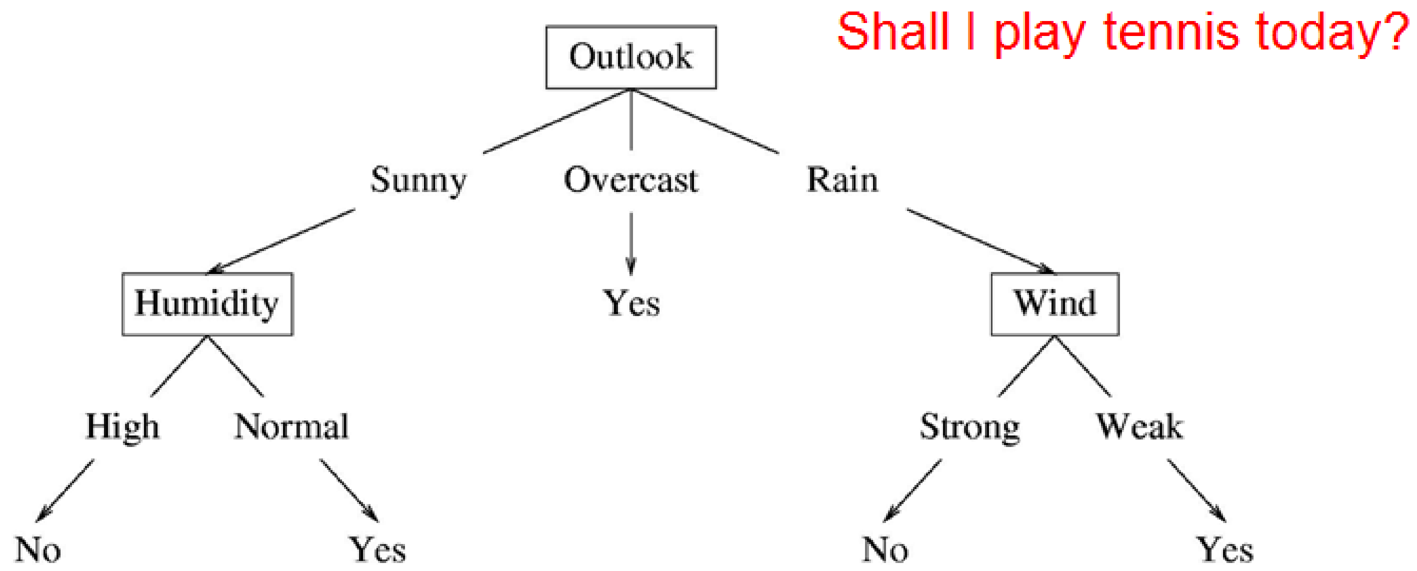
Trees & Traversals

- Problem space as tree
- Want to find optimal solution
- BFS & iterative deepening search both work well
- Better technique called A* search:
    - Instead of ‘closest’ or ‘furthest’, choose lowest cost=g( )+h( )
    - g( ) is cost so far
    - h( ) is expected distance to goal



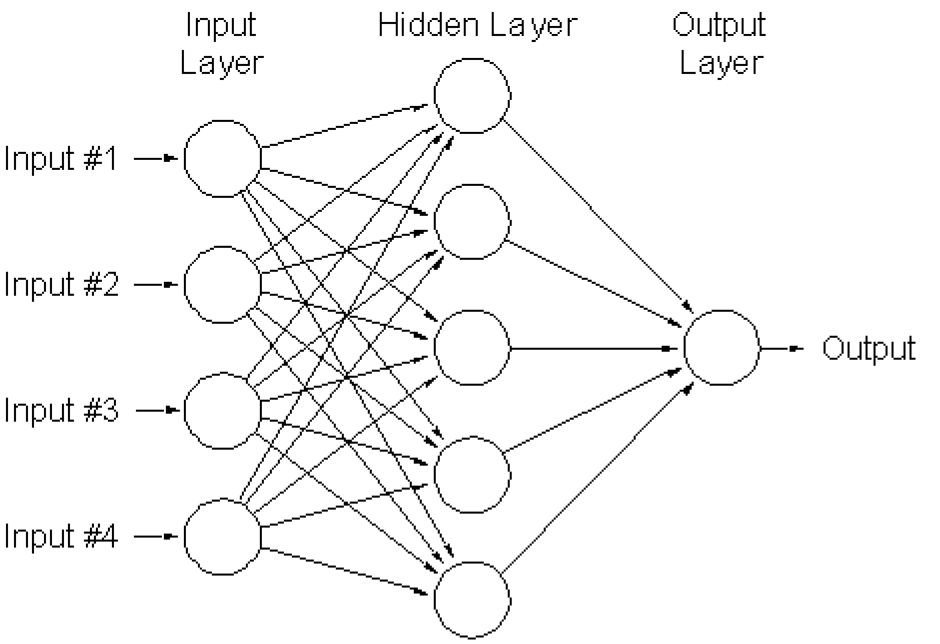
Decision Trees

- Basis for simple decision-making agents
- Algorithms to create optimal decision tree:
  - Take set of labeled data ('Sunny,Normal Humidity,Strong Wind: Yes')
  - Uses 'information gain' to decide what attribute to ask about next
- Of course, real decision trees likely to be much larger
  - Ex: Face detection features



Neural Networks

- Usually DAG of 'neurons'
- Edges represent how information propagates from input nodes (observations) to output nodes (decision)
- Uses include OCR:
    - Conceptually have each pixel as a binary input
    - Each output represents a character: 'Is this image a 9?'



Thanks!

- Extra office hours