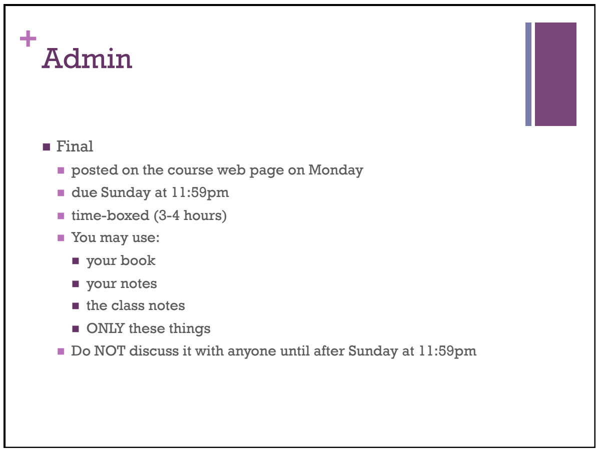
+

Review
CS302

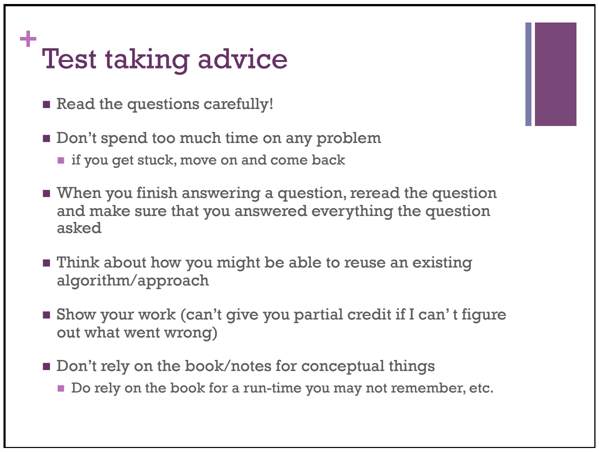
Spring 2013
David Kauchak



+

Admin

- Final
  - posted on the course web page on Monday
  - due Sunday at 11:59pm
  - time-boxed (3-4 hours)
  - You may use:
    - your book
    - your notes
    - the class notes
    - ONLY these things
  - Do NOT discuss it with anyone until after Sunday at 11:59pm



+

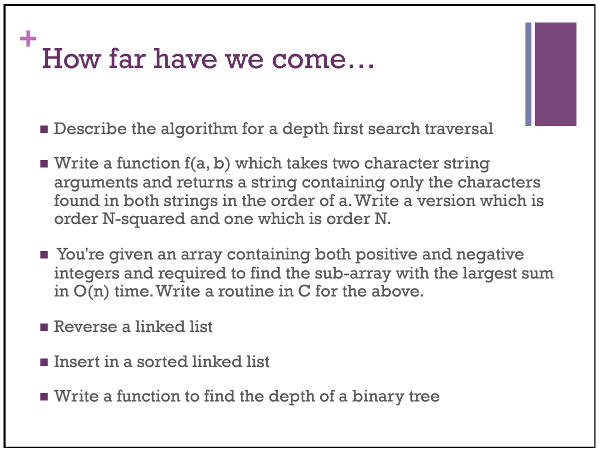
Test taking advice

- Read the questions carefully!
- Don't spend too much time on any problem
  - if you get stuck, move on and come back
- When you finish answering a question, reread the question and make sure that you answered everything the question asked
- Think about how you might be able to reuse an existing algorithm/approach
- Show your work (can't give you partial credit if I can' t figure out what went wrong)
- Don't rely on the book/notes for conceptual things
  - Do rely on the book for a run-time you may not remember, etc.

+

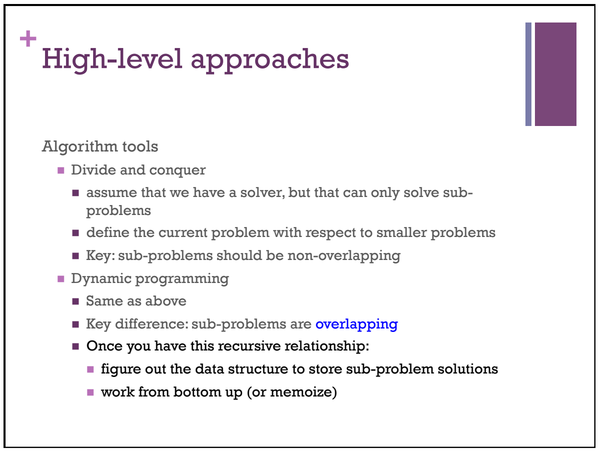
Where we've been

- 21 assignments
  - 67 problems!
- 23 classes
  - Number of slides?
    - 1537!!!
  - Hung out for: 27 hours



How far have we come…

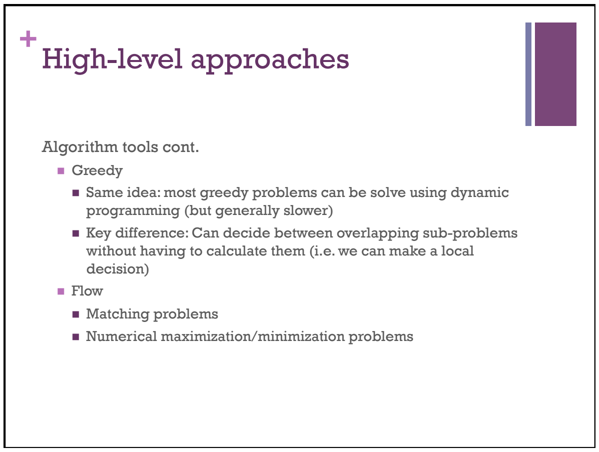
- Describe the algorithm for a depth first search traversal
- Write a function f(a, b) which takes two character string arguments and returns a string containing only the characters found in both strings in the order of a. Write a version which is order N-squared and one which is order N.
- You're given an array containing both positive and negative integers and required to find the sub-array with the largest sum in O(n) time. Write a routine in C for the above.
- Reverse a linked list
- Insert in a sorted linked list
- Write a function to find the depth of a binary tree



High-level approaches

Algorithm tools

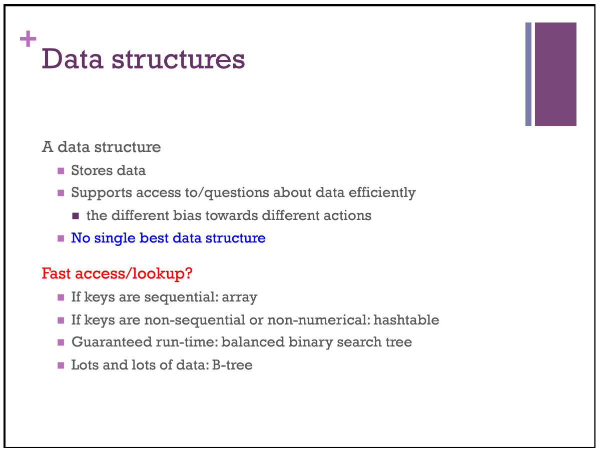
- Divide and conquer
  - assume that we have a solver, but that can only solve sub-problems
  - define the current problem with respect to smaller problems
  - Key: sub-problems should be non-overlapping
- Dynamic programming
  - Same as above
  - Key difference: sub-problems are overlapping
  - Once you have this recursive relationship:
    - figure out the data structure to store sub-problem solutions
    - work from bottom up (or memoize)



High-level approaches

Algorithm tools cont.

- Greedy
  - Same idea: most greedy problems can be solve using dynamic programming (but generally slower)
  - Key difference: Can decide between overlapping sub-problems without having to calculate them (i.e. we can make a local decision)
- Flow
  - Matching problems
  - Numerical maximization/minimization problems



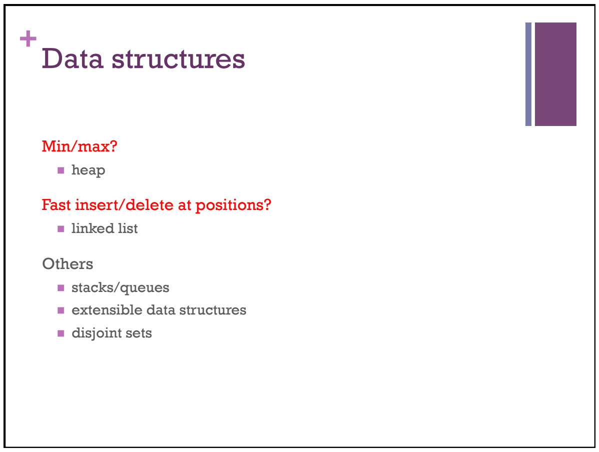
Data structures

A data structure

- Stores data
- Supports access to/questions about data efficiently
  - the different bias towards different actions
- No single best data structure

Fast access/lookup?

- If keys are sequential: array
- If keys are non-sequential or non-numerical: hashtable
- Guaranteed run-time: balanced binary search tree
- Lots and lots of data: B-tree



+ Data structures

Min/max?

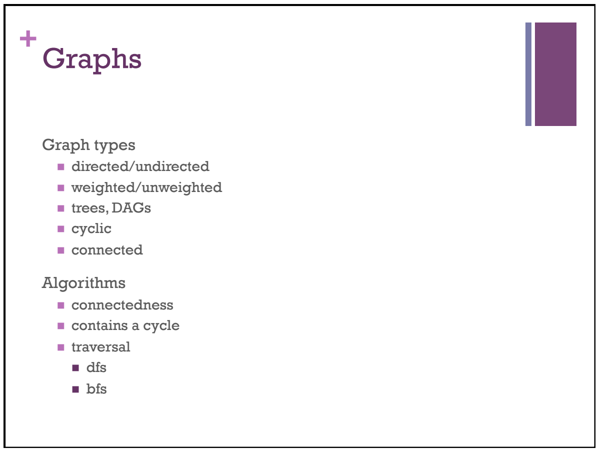
- heap

Fast insert/delete at positions?

- linked list

Others

- stacks/queues
- extensible data structures
- disjoint sets



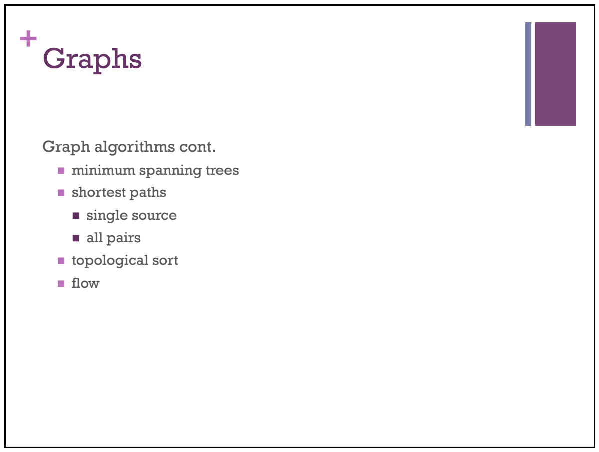
+ Graphs

Graph types

- directed/undirected
- weighted/unweighted
- trees, DAGs
- cyclic
- connected

Algorithms

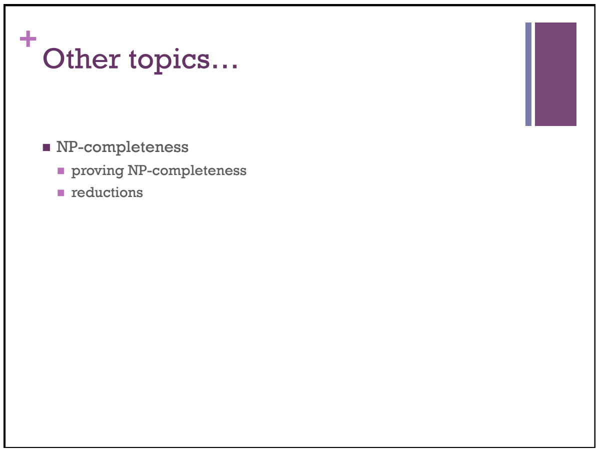
- connectedness
- contains a cycle
- traversal
  - dfs
  - bfs



+ Graphs

Graph algorithms cont.

- minimum spanning trees
- shortest paths
  - single source
  - all pairs
- topological sort
- flow



+ Other topics…

- NP-completeness
  - proving NP-completeness
  - reductions