CSC 427: Data Structures and Algorithm Analysis
Fall 2007

Course goals

- To appreciate the role of algorithms in problem solving and software design; selecting among competing algorithms and justifying choices based on efficiency.
- To understand the specifications and implementations of standard data structures and be able to select appropriate structures in developing programs.
- To develop programs using different problem-solving approaches, and be able to recognize when a particular approach is most useful.
- To be able to design and implement a program to model a real-world system, and subsequently analyze its behavior.
- To recognize the importance of object-oriented techniques, and be able to utilize inheritance and polymorphism to build upon existing code.

Algorithm analysis

big-Oh notation

- formal definition, rate-of-growth analysis, asymptotic behavior
- underlying big-Oh classifications for data structure operations

searching & sorting

- sequential search vs. binary search
- insertion sort vs. selection sort vs. merge sort vs. heap sort
- specialized sorts: frequency counts, radix sort

recursion

- base case(s), recursive case(s), analysis via recurrence relation

algorithmic approaches

- *divide&conquer:* binary search, merge sort, tree algorithms, ...
- *greedy:* optimal optimal change (U.S.), job scheduling, Huffman codes, ...
- *backtracking:* N-queens, blob count, Sudoku, ...
- *dynamic:* optimal change, binomial coefficient, minimal edit distance ...

Data Structures

low-level data structures

- *linked lists:* singly-linked vs. doubly linked, header node(s)
- *trees:* recursive processing, binary search tree, balanced tree variants, heaps
- *hash table:* hash function, collisions, load factor, rehashing, probing vs. chaining
- *iterators:* Iterable interface, Iterator interface

lists

- *List interface:* get, set, add, remove, contains, indexOf, size, iterator, ...
- *implementations:* ArrayList implementation, LinkedList, SortedList

sets

- *Set interface:* add, remove, contains, clear, size, iterator, ...
- *implementations:* TreeSet, HashSet

maps

- *Map interface:* get, put, remove, containsKey, keySet, size, ...
- *implementations:* TreeMap, HashMap

priority queues

- *PriorityQueue class:* peek, add, remove, size, iterator, ...

Object-oriented programming

class design

- interacting classes, private data fields & public methods, generics, …

interfaces:

- implementing an interface, polymorphism

inheritance:

- extending a class, overriding methods, polymorphism

GUI design/building

HW1 (anagram checker): class design, String/List manipulation, GUI

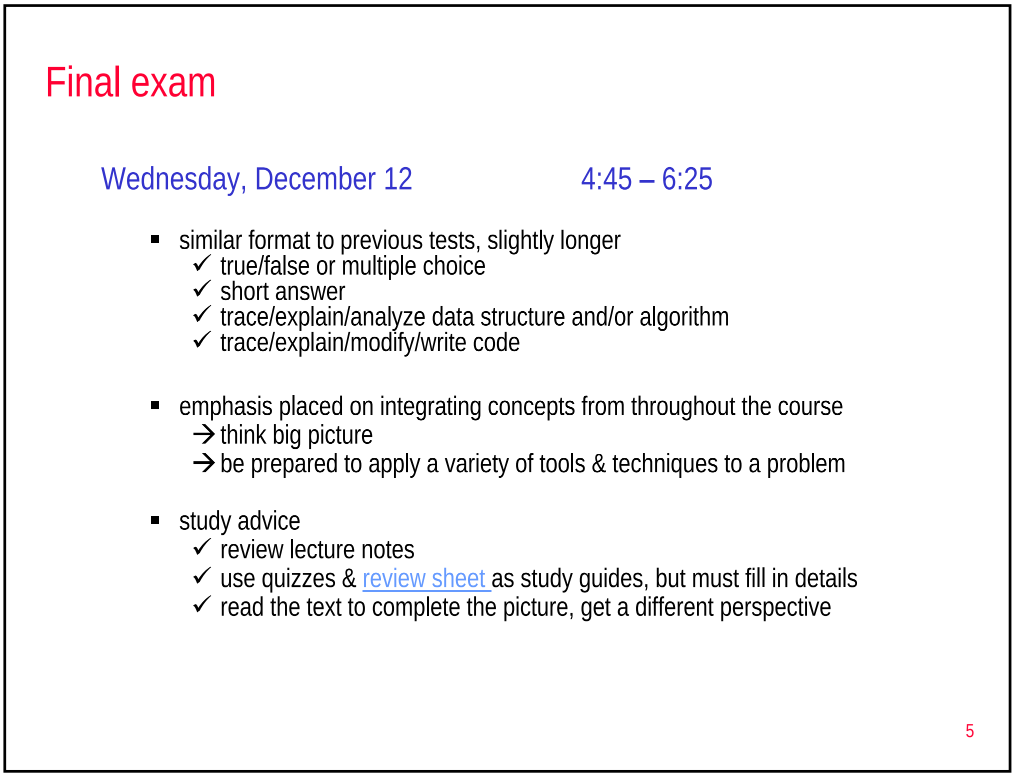
HW2 (radix sort): 2-D structure implementation, experimentation, big-Oh analysis

HW3 (file indexer): design with polymorphism, Set & Map, file I/O

HW4 (binary search trees): divide&conquer, linked structures, recursion, experimentation

HW5 (Sudoku): recursive backtracking, 2-D data structure, GUI

HW6 (minimal edit distance): dynamic programming, recursion, String manipulation



Final exam

Wednesday, December 12		4:45 – 6:25

- similar format to previous tests, slightly longer
  - ✓ true/false or multiple choice
  - ✓ short answer
  - ✓ trace/explain/analyze data structure and/or algorithm
  - ✓ trace/explain/modify/write code
- emphasis placed on integrating concepts from throughout the course
  - → think big picture
  - → be prepared to apply a variety of tools & techniques to a problem
- study advice
  - ✓ review lecture notes
  - ✓ use quizzes & review sheet as study guides, but must fill in details
  - ✓ read the text to complete the picture, get a different perspective