



Course Review for Midterm II

Cpt S 223

Fall 2011



Midterm Exam 2

- When: Friday (11/18) 1:10 -2pm
- Where: in class

- Closed book, closed notes
- Comprehensive
 - With primary focus on the topics covered after Midterm I that includes: **B-trees, priority queues, hashing**.
 - Questions could also indirectly include any of the topics covered before Midterm exam 1

- Material for preparation:
 - Lecture slides & in-class notes
 - Homeworks & program assignments
 - Weiss book



Priority Queues

- Binary heap, Binomial heaps
- Need to know:
 - Binary heap
 - Structure property:
 - complete binary tree except the last level (filled breadth-first left to right)
 - Heap order property
 - Min heap : each node's value is less than or equal to its children
 - Binomial heap
 - Structure property:
 - A forest of binomial trees (similar to binary representation)
 - Heap order property:
 - Min heap: within each binomial tree, same heap order like in binary heap



Implementation: Need to know...

- Binary heap
 - How to convert tree representation to array and vice versa
 - When to use `percolateUp` and `percolateDown`
 - Computation complexities of all basic heap operations on binary heap
 - Insert, `buildheap`, `deleteMin`
- Binomial heap
 - Array of pointers to each binomial tree
 - $\log n$ binomial tree pointers
 - Know relationships between binomial *heap*, binomial *trees* and their properties
 - Computation complexities of all basic heap operations on binary heap
 - Insert, `deleteMin`, `merge`



Run-times for each heap operation

	Insert	DeleteMin	Merge
Binary heap	$O(\lg n)$ worst-case, $O(1)$ amortized using buildheap	$O(\lg n)$	$O(n)$
Binomial Heap	$O(\lg n)$ worst-case $O(1)$ amortized (or equivalently, $O(m)$ for m successive calls to insert)	$O(\lg n)$	$O(\lg n)$



Other heap operations

- deleteMax
- decreaseKey
- increaseKey
- remove

Need to know all the complexities and be ready to write pseudocodes for the above

Also be prepared to think of writing basic application code that use heap functionalities



Hashing

- Hash functions
 - purpose: string to integer, integer to integer
- Choice of a “good” hash functions
 - Reduce chance of collision
 - Relatively smaller key value
 - Does not need huge hash table size
- Hash Tables use hash functions
- Hash table size should be a prime
- Load factor
 - Measure to tell how crowded a hash table is



Hashing

- Know algorithms & analysis for the following
 - Collision resolution by chaining
 - Collision resolution by open-addressing
 - Linear probing, quadratic probing
 - Double hashing
 - Rehashing
- Hash table operations:
 - Insert
 - Find
 - Delete
 - Be prepared to write pseudocodes and work out examples for all these operations under different collision resolution implementations
 - Know how to measure performance –
 - # failed probes for probing techniques
 - # comparisons along the linked list for chaining
- Think of applications where hash tables could work out better than other data structures discussed in class.

GOOD LUCK!



Some General Tips

- Work out examples of basic operations
- For algorithm design questions, the default expectation is that you come up with the best possible (by performance) solution, unless otherwise explicitly stated.
- Show steps to convey your thought process
 - Think from the perspective of the grader when writing solutions
- Questions mostly objective, but be prepared to see one or two subjective ones too
- Write less and to the point
- Don't leave questions unanswered
 - If unable to solve, at least write down your approach idea(s). I try to give partial credits wherever possible.