Review for the Final Exam

Final Exam

- **Final exam is on Feb, 3rd at 13:30**
- Cumulative, but will emphasize more on the second half of the semester
- Same format as before: short answer, true/false + Programming sections.

Summary of what we have learned

- **Fundamental Storage Data Structures**
  - Arrays
  - Linked Lists
  - Trees
- **High-Level, Abstract Data Types**
  - List (sorted, unsorted), Stack, Queue
  - BST (Binary Search Tree), AVL
  - Priority Queue, Heap
  - Hash Table
  - Graph

Comparison

Data structure	Add	Search	remove
Unsorted array	O(1)	O(N)	O(N)
Sorted array	O(N)	O(log N)	O(N)
BST (balanced/worst)	O(log N)/O(N)	O(log N)/O(N)	O(log N)/O(N)
Heap	O(log N)	-	O(log N)
Hash table	O(1)	O(1)	O(1)

Summary of what we have learned

- **Fundamental Algorithms**
  - Recursion
  - Sorting (simple sorting,merge sort, quick sort, heap sort)
  - Tree traversal
  - Graph Search (DFS, BFS), shortest path
- **C++ specific knowledge**
  - Classes, Templates, Exceptions, Overloading
  - Constructors, destructors
  - C++ std provides implementations of many data structutes we've learned

How to Prepare

- Study lecture slides and text book
  - Study lecture slides first. If anything is unclear, refer to the textbook
  - Make sure you understand all the questions in the slides
- Study the practice Final Exam
- Study online exercise questions
- Study lab materials
- Study programming projects

Second half of the semester

- **Heap and Priority queue**
  - What is priority queue? How is it different from a standard queue?
  - What is a heap? What are its properties? How is it different from a BST? How is a heap stored?
  - How to perform add and remove in a heap?
  - Understand bubbleUp and bubbleDown, and be able to write down code for them.
  - What are the costs of add and remove with a heap? How does this compare to using sorted array?

Second half of the semester

- **Hash Table**
  - What is a hash function? What is a hash table?
  - What are the advantages / disadvantages of hash table compared to other data structures?
  - How to compute the hash function?
  - What is collision? How to handle collision?
  - Open addressing: linear probing, quadratic probing, double hashing
  - Separate chaining
  - What is load factor? Why is it important?
  - Understand the analysis of successful and unsuccessful search in hash tables.

Second half of the semester

- **Graphs**
  - Graph terminologies
  - Directed / Undirected, Connected / Unconnected,
  - Weighted / Unweighted
  - How is a graph typically stored?
  - Graph traversal
    - Depth-First Search (DFS): implementation
    - Breadth-First Search (BFS): implementation
    - Shortest-distance in a weighted graph (Dijkstra's algorithm)

Second half of the semester

- **Simple sorting algorithms: $O(N^2)$**
  - Selection sort
  - Insertion sort
  - Bubble sort
- **Advanced sorting algorithms: O(N log N)**
  - Merge sort and merge() operation
  - Quick sort and partition() operation
  - Heap sort and heapify() operation
- Their best, worst and average case running times

Comparison

Sort	Best case	Average case	Worst case
Selection sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Insertion sort	$O(N)$	$O(N^2)$	$O(N^2)$
Bubble sort	$O(N)$	$O(N^2)$	$O(N^2)$
Merge sort	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
Quick sort	$O(N \log N)$	$O(N \log N)$	$O(N^2)$
Heap sort	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$

Sample Questions

- Find the second largest element in an unsorted array.
- Implement binary search in a sorted array.
- Search / Insertion in a BST
- Find the predecessor / successor of any node in a BST
- Implement bubbleUp / bubbleDown
- Check if a binary tree is a heap or not; is a BST or not.
- Print out vertices in a graph in DFS and BFS order
- Implement Bubble / Selection / Insertion sort
- Implement merge(), partition(), heapify() methods
- Find the shortest path from s to t in a graph
- Find a cycle in a graph
- Find all connected components in a graph