Information

- Exam: Wednesday May 8, 2-5pm in the same room we always meet.

- You can bring two hand-written (ok *hand-written* on tablets and then printed) sheets of papers (i.e. four pages).

- Cumulative with a bigger emphasis on topics we covered since midterm 2.

- Review midterm 1 and 2, quizzes, worksheets, and practice problems in each presentation. Use the practice questions in this presentation.
Java Basics

- Chapter 1.1 (Pages 8–35).
- Chapter 1.2 (Pages 64–77, 84–88, 96–99, 107).
- Quick overview of Java tutorials.
  - [https://docs.oracle.com/javase/tutorial/java/](https://docs.oracle.com/javase/tutorial/java/)
- In general, review the basics of OOP and of Java so that you are comfortable reading and writing code.
Analysis of Algorithms

- Chapter 1.4 (Pages 172-205).

- Experimental analysis including doubling hypothesis. Pick two pairs of the largest input sizes and check that the $T(n)/T(n/2)$ is consistently expressed as some power of 2.

- Mathematical analysis including reviewing (not memorizing) useful approximations of sums.
  - Use midterm review slides for practice.

- Order of growth classifications.

- Review of running time of operations on array lists, linked lists, stacks and queues.
ArrayLists

- Chapter 1.3 (Pages 136-137).
- code
- Java Oracle API [https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html)
- Amortized and worst-case time analysis.
Singly Linked Lists

- Chapter 1.3 (Pages 142-146).
- code
- Worst-case time analysis for standard operations.
Doubly Linked Lists

- Chapter 1.3 (Pages 126-157).
- code
- Java Oracle API.
  - https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
- Worst-case time analysis for standard operations.
Stacks and Queues

- Chapter 1.3 (Pages 142-146).
- code for alternative implementations
- Worst-case time analysis for standard operations based on the underlying implementation
Iterators and Comparators

- Comparable vs Comparator interface.
  - [https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html)
  - [https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html](https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html)

- Java Oracle Iterator and Iterable.
  - [https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html](https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html)
  - [https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html](https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html)

- code
Sorting

- Chapter 2 (Pages 244-296).
- Selection sort and Insertion sort.
- Mergesort.
- Quicksort.
  - We have seen Lomuto’s partition scheme
  - Know how to apply them, best and worst case running times, in-place or not, stability
Binary Trees, Binary Search, Heaps, and Priority Queues

- Chapter 2.4 (Pages 308-322)

- Definitions, basic properties of trees and traversals (pre-, in-, post-, level-order) for binary trees.

- Binary search implementation and complexity.

- Binary Heaps and operations.
  - Different implementations along with complexities.
Heapsort

- Chapter 2.4 (323-327).
- Know how to apply, running time analysis.
What you need to remember about sorting

<table>
<thead>
<tr>
<th>In place</th>
<th>Stable</th>
<th>Best</th>
<th>Average</th>
<th>Worst</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>X</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n$ exchanges</td>
</tr>
<tr>
<td>Insertion</td>
<td>X</td>
<td>X</td>
<td>$n$</td>
<td>$n^2$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>Merge</td>
<td>X</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>Guaranteed performance; stable</td>
</tr>
<tr>
<td>Quick</td>
<td>X</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n^2$</td>
<td>$n \log n$ probabilistic guarantee; fastest in practice</td>
</tr>
<tr>
<td>Heap</td>
<td>X</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n \log n$ guarantee; in place</td>
</tr>
</tbody>
</table>
Dictionaries and Binary Search Trees

- Chapter 3.1 (Pages 362-386).
- Different implementations along with complexities.
- Chapter 3.2 (Pages 396-414).
- Textbook code.
- Addition, Search, Hibbard's Deletion.
2-3 Search Trees

- Chapter 3.3 (Pages 424-291).
- Definitions, Search, Insertion, Construction.
- Performance.
Left-leaning red-black trees

- Chapter 3.3 (Pages 292-447).
- Definitions, Operations, Insertion.
- Performance.
Hash tables

- Chapter 3.3 (Pages 458-477).
- Hashing, separate chaining, open addressing.
## Summary for Dictionary operations

<table>
<thead>
<tr>
<th></th>
<th>Worst case</th>
<th></th>
<th></th>
<th>Average case</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Search</td>
<td>Insert</td>
<td>Delete</td>
<td>Search</td>
<td>Insert</td>
<td>Delete</td>
</tr>
<tr>
<td>BST</td>
<td>$n$</td>
<td>$n$</td>
<td>$n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>2-3 search tree</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Red-black BSTs</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Separate chaining</td>
<td>$n$</td>
<td>$n$</td>
<td>$n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Linear probing</td>
<td>$n$</td>
<td>$n$</td>
<td>$n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Undirected Graphs

- Chapter 4.1 (Pages 515-556).
- Definitions, representations, APIs.
- BFS.
- DFS.
- Textbook code.
Directed Graphs

- Chapter 4.2 (Pages 566-594).
- Definitions, representations, APIs.
- BFS.
- DFS.
- Textbook code.
Shortest Paths

- Chapter 4.4 (Pages 638-676).
- Dijkstra’s algorithm.
- https://visualgo.net/en/sssp
Minimum Spanning Trees

- Chapter 4.3 (Pages 604-629).
- Know how to apply Kruskal's and Prim's algorithms.
  - [https://algs4.cs.princeton.edu/43mst/](https://algs4.cs.princeton.edu/43mst/)
  - [https://visualgo.net/en/mst](https://visualgo.net/en/mst)
Problem 1: Balanced Binary Search Trees

- a. Starting with an empty 2-3 search tree, what is the resulting 2-3 search tree after you insert the keys 2, 7, 17, 1, 90, 3, 36, 47?

- b. Starting with an empty LLRB search tree, what is the resulting LLRB search tree after you insert the keys 15, 6, 23, 4, 7, 5, 50, 71?
Problem 2: Hashtables

Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size $m = 11$ using the hash function $h(k) = k \mod m$. For each of the following questions, fill in the following hash table:

- a. using open addressing and linear probing with $h(k, i) = (h(k) + i) \mod m$.
- b. using open addressing and quadratic probing with $h(k, i) = (h(k) + i^2) \mod m$.
- c. using external chaining.
Problem 3: Traversing Graphs

Show the adjacency matrix and adjacency list representations of the undirected graph above.

Run a. recursive and b. BFS starting at vertex A assuming adjacent vertices are returned in lexicographic order. Fill in the table below:

<table>
<thead>
<tr>
<th>v</th>
<th>marked</th>
<th>distTo</th>
<th>edgeTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 4: Shortest Paths

- Run Dijkstra’s algorithm on this graph, starting at vertex a. Fill in the resulting distTo[] and edgeTo[] arrays below. In the edgeTo[] column, please indicate the last edge in the shortest path from a to every other vertex and mark the shortest path tree.

<table>
<thead>
<tr>
<th>v</th>
<th>distTo</th>
<th>edgeTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Solution to Problem 1a: Balanced Binary Search Trees

- Starting with an empty 2-3 search tree, what is the resulting 2-3 search tree after you insert the keys 2, 7, 17, 1, 90, 3, 36, 47?
Solution to Problem 1b: Balanced Binary Search Trees

Starting with an empty LLRB search tree, what is the resulting LLRB search tree after you insert the keys 15, 6, 23, 4, 7, 5, 50, 71?
Solution to Problem 2a: Hashtables

- Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size \( m = 11 \) using the hash function \( h(k) = k \mod m \). For each of the following questions, fill in the following hash table using open addressing and linear probing with \( h(k, i) = (h(k) + i) \mod m \).
Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size $m = 11$ using the hash function $h(k) = k \% m$. For each of the following questions, fill in the following hash table using open addressing and linear probing with $h(k, i) = (h(k) + i^2) \% m$.
Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size $m = 11$ using the hash function $h(k) = k \% m$. For each of the following questions, fill in the following hash table using external chaining.
Solution to Problem 3a: Traversing Graphs

- Show the adjacency matrix and adjacency list representations of the undirected graph above.

- Run a recursive DFS assuming adjacent vertices are returned in lexicographic order.

- Order of visit: A, B, D, E, C

<table>
<thead>
<tr>
<th>v</th>
<th>marked</th>
<th>edgeTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>T</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>T</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>T</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>T</td>
<td>B</td>
</tr>
</tbody>
</table>
Solution to Problem 3b: Traversing Graphs

- Show the adjacency matrix and adjacency list representations of the undirected graph above.

- Run b. BFS assuming adjacent vertices are returned in lexicographic order.

- Order of visit: A, B, C, E, D

<table>
<thead>
<tr>
<th>v</th>
<th>marked</th>
<th>distTo</th>
<th>edgeTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>T</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>T</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>T</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>T</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>T</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>
Solution to Problem 4: Shortest Paths

- Run Dijkstra’s algorithm on this graph, starting at vertex a. Fill in the resulting distTo[] and edgeTo[] arrays below. In the edgeTo[] column, please indicate the last edge in the shortest path from a to every other vertex and mark the shortest path tree.

<table>
<thead>
<tr>
<th>v</th>
<th>distTo</th>
<th>edgeTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>b</td>
<td>15</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>25</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>35</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>25</td>
<td>b</td>
</tr>
<tr>
<td>f</td>
<td>35</td>
<td>e</td>
</tr>
<tr>
<td>g</td>
<td>30</td>
<td>e</td>
</tr>
<tr>
<td>h</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>i</td>
<td>35</td>
<td>h</td>
</tr>
</tbody>
</table>