A successful student will be able to:

• Describe and implement a variety of advanced data structures {hash tables, graphs, priority queues, quad trees, balanced search trees}.

• Analyze the space and time complexity of the algorithms studied in the course.
  • ArrayList Vs {HashMap, HashSet, PriorityQueue }
  • Balanced Search Trees { AVL, RedBLack, BTree }

• Identify different solutions for a given problem; analyze advantages and disadvantages to different solutions.
  {Dijkstra Vs A*, many indexes: Hash Vs Tree }

• Demonstrate an understanding of external memory and external search and sorting algorithms. {Serialization, RAF, BTree, hashing}

• Demonstrate an understanding of simple Entity-Relationship models for databases. {Normalization, Normal Forms}
• Demonstrate an understanding of basic principles of software design and development and be able to apply these principles in course programming projects. { UML class diagram, Generic types }

• Construct programs of significant length that require modularization in order to be understood.

• Use and understand basic object oriented programming concepts (e.g., classes, inheritance, interfaces, exception handling and JAVA JCFs). { JCF <generic>, SQL }

• Generate source code that is easy to read, well organized, well commented.

• Employ debugging techniques to assist in problem solving { visualization w/ SimulationFramework }
Data Structures

- Reviewed 182 ADTs: ArrayList, stack, queue, binary tree
- Discussed and compared Balanced Search Trees (BST): AVL, red-black, B trees: properties, insertion (search), deletion single pass Vs multi-pass algorithms
- Discussed Priority Queues
- Discussed hashing: hash functions, collision resolution properties, insertion (search), deletion (physical Vs. logical) compared w/ BSTs
- Discussed and implemented Graphs: properties, adjacent nodes dfs, bfs, spanning trees, path finding algorithms.
- Relational databases – normalization (design – dependencies) SQL data definition / manipulation languages, JDBC
- External memory – conceptual organization, searching Serialization, hashing w/ Java random access file technique
Programming: design, implementation

- **P1** Java / Swing API and Simulation Framework  
  JCF ArrayList, Swing event handling, algorithm simulation

- **P2** Graphs: path finding, adjacency nodes, using JCF classes:  
  HashMap, HashSet $O(1)$ // constant  
  PriorityQueue, retrieve $O(1)$, insert $O(\log n)$  
  ArrayList insert, contains() & most operations $O(n)$,  
  document design and implementation

- **P3** Database: create, use sqlite database, extend / modify existing program (P2)  
  document design and implementation

- **UML class diagrams, state transition diagrams, documentation generations (javadoc)**

- **Small groups – communication, teamwork, peer-learning, planning**
## Taxonomy

<table>
<thead>
<tr>
<th>compute</th>
<th>search</th>
</tr>
</thead>
<tbody>
<tr>
<td>primary</td>
<td>primary</td>
</tr>
<tr>
<td>external</td>
<td>external</td>
</tr>
<tr>
<td>hashing</td>
<td>arrayList &lt;E&gt;</td>
</tr>
<tr>
<td>HashMap &lt;K, V&gt;</td>
<td>PriorityQueue &lt;E&gt;</td>
</tr>
<tr>
<td>hashing</td>
<td>graph</td>
</tr>
<tr>
<td>SQL</td>
<td>AVL</td>
</tr>
<tr>
<td>hashing</td>
<td>B tree</td>
</tr>
<tr>
<td>HashMap &lt;K, V&gt;</td>
<td>SQL</td>
</tr>
<tr>
<td>SQL</td>
<td></td>
</tr>
</tbody>
</table>

### APIs

- JCF, Swing (Simulation Framework)
- SQL / JDBC
Can you?

Describe and contrast graphs with trees
  perform bfs, dfs, generate a minimum spanning tree
  contrast and perform Dijkstra and A* path finding for a graph

Describe the properties of an AVL tree, Red Black tree, and a BTree
  perform insertion and deletions on tree, stating cases

Discuss the use of Hashing Vs Tree access/update on a data set, on an integrated dataset.

Describe and perform database normalization (perform to 3NF, 4NF)

Given a set of relations answer questions about the data using SQL queries.
Whats next?

More experience with JCF:  HashSet\(<E>\), LinkedHashMap\(<K, V>\)
TreeSet\(<E>\), TreeMap\(<K, V>\)

gain more experience with generic ADTs, use of wildcards

Additional ADTs  Heaps: binomial  Trees: Tries, Splay

Experience with other Collection class libraries:

C++  STL  C#  .NET

Algorithms / Data Structures  Comp 256/L, Comp 310,  Comp 496Alg or Math 482

Databases  Comp 440,  Comp 598ADB

Web Services  Database + SQL query + Web UI
Comp 484 (web) , Comp 541 (data mining),  Comp 585 (GUI)

Software development  Comp 380  Comp 490 +  Comp 491,
Comp 585 (GUI), Comp 586 OOP software design, others ...
Any Questions?

Problem Solving

Think about a simpler problem first.

Design and analyze 2 solns, implement best or merge ideas.

Planning

Estimate effort = Design + iterative (implementation + testing + validation)

Hack problem solving

There was no course / instructor evaluation. If you have suggestions for improvement please email me after the semester, or anonymously drop off in CS office for me.