Master Syllabus for TCSS 342
Version: September 2009

Catalog Description

Covers abstract data types, design and complexity analysis of data structures, and usage of generic data structure libraries in high-level programming languages. Includes sequential and linked lists, binary trees and balanced binary trees, B-trees, hash tables, and heaps. Prerequisite: TCSS 305 (may take concurrently); minimum of 2.0 in TCSS 321.

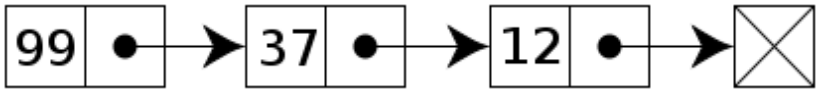
Content Description

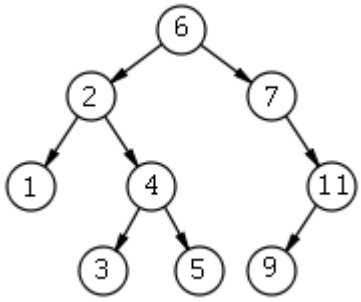
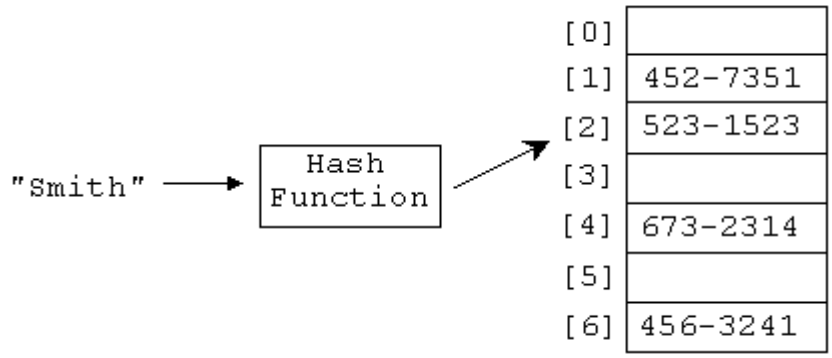
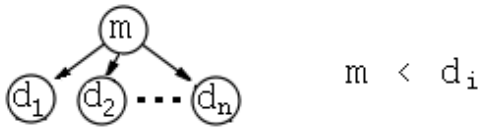
The TCSS 342 course serves multiple purposes in both the BS and BA degrees in CSS. It is a listed pre-requisite for both TCSS 343 and 360, as well as for a large number of CSS concentration courses (i.e. electives in the major). The broad goals of TCSS 342 are for students to understand the concepts of a data abstraction (i.e. *abstract data type*), the operations associated with each of a set of canonical data abstractions, the standard ways in which these abstractions are implemented using both the primitives and other abstractions in modern programming languages, how to use this canonical set as realized in the standard, generic libraries of modern programming languages, the analytical skill of worst- (big O) and average-case space/time complexity analysis, and the use of space/time complexity analysis in making choices between data abstractions and data structures for specific problems.

Student Learning Objectives

The specific learning objectives are as follows. At the completion of the course, students should be able to:

- *Define* what a data abstraction is.
- *Identify* the operations that characterize the following abstractions and structures. These will be called the *canonical set.*
  Data abstractions: **List, Stack, Queue, Priority Queue, Map/Dictionary**
  Data structures: **Array, Linked List, Tree, Binary Search Tree (BST), Balanced BST, B-tree, Heap, Hash Table**
- *Identify* the API of each abstraction or its closest correlates in the canonical set as realized in a standard generic library of a modern programming language, such as the Java Collections Framework.
- *Apply* pictorial representations to *explain* the implementations of the following data structures and their operations:

Data Structure	**Pictorial Representation**
Linked List, Array	Linked List, Array 

Data Structure	**Pictorial Representation**
BST, Balanced BST, B-tree	Rooted Tree 
Hash Table	Table 
Heap	Array, Partially Ordered Tree 

- *Adapt* and/or *extend* the generic code implementation of at least 3 data structures from the canonical set, at least 1 of which is recursively defined.
- *Instantiate* and *apply* the API's of at least 4 data abstractions (or their correlates) in the canonical set as provided in the standard generic library of a modern programming language.
- *Analyze* the worst- and average-case time and space complexity of the operations in common implementations of the data abstractions and structures in the canonical set.
- *Analyze* the worst-case time complexity of code that uses data abstractions in the canonical set.
- *Compare and contrast* time/space characteristics of different implementations of the same data abstraction.
- *Select* data abstractions, structures, and implementations that a developer would use in solving larger problems and *defend* the appropriateness of these choices.

Approved November 6, 2009