Misc. Post-CSI310 ideas.

Tree and Expression Definitions Compared.

Variations of tree traversals.

Heap-ordered trees.

Decision, Search and other trees, Array search, Binary array and tree search.
First, containment or "has-a" relationship...

A glimpse at inheritance...
cases and basically the same kind of thing, as the base class.

Unlike containment, inheritance is used when the subclass object are special.

data, methods or specific properties.

objects that share all the data and methods of the base, plus have additional

Given the base class, programmers can derive from it or create subclasses for

have some general usefulness.

The objects of one class, called the base class, have some data and methods that

Inheritance expresses the "is-a" relationship.
virtual void make-move( string move )
    { }
public:
    } } }

virtual void display-status( ) const = 0;
{ }
virtual void make-move( string move )
    { }
protected:
    { }

display-status( );
make-move( move );
string move; cin >> move;
... }

who play

enum who plays { // who wins
    human, who, computer
    public:
} class Game {

DSO (14.3) Example:

University at Albany Computer Science Dept.
```cpp
};

int most_recent_column;

int many_used[COLUMNS];

who_data[ROWS][COLUMNS];

private:

virtual void display_status() const;

{
    super::move();
}

// Code for making a connect 4 move goes here
```
virtual functions depending on the class of the object.

The same method calls made in base class method bodies call out to different
Inherited classes with virtual methods provide an example of polymorphism:

In Java, all methods are virtual.

Game calls their name, like make-move and display-status
Virtual functions FROM class Game connected will be called when code in class
Game class like class connected : Game.

To implement a specific game, the programmer derives from class Game a new
every Game should support.

Idea: The game class models a general game, methods reflect what operations
...;
...;
{

draw-rectangle(10.0, 10.0, 30.0, 40.0);

clear-canvas();
}

public: virtual void draw

...;
}

} class MyApplicationClass : Window

Programmers code:

{,
...
public: virtual void draw();

...;
}

} class Window

say

Object Oriented Graphics Application Programming Interfaces. The API defines

University at Albany Computer Science Dept.
(Graphics::Window)
draw() function that the programmer wrote. It codes the commands to draw the
constructed by the new call above. The function that is actually called in the
time, or to redraw it, the API library calls Window::draw() on the object
When the windowing system needs to display the application's window the first

MyApplicationClass* ptr = new MyApplicationClass();
Read Head First Design Patterns.
Read Head First Java.
Take CSI445 this summer.
For more:
the maze array represents a graph. (in 15.2 and 15.3) Adjacency array for representing a graph; difference from how

and binary search trees; pre, post, and in-order tree traversals. Representation of decision (DIOs, taxonomy) animal guessing
tree traversal (prefix, traversal of decision) 10 and lectures (some basic tree lore, tree applications, linked tree

Graph traversal codes. (6 (a little), 10.4, code in 15.3) Templates and function parameters in tree and

Breadth-First traversal of TREES.

15.3) Graph traversal, Depth-First and Breadth-First; Depth-First and

Data structures and algorithms for Project 5!

subject final exam questions.

honors”/”optional” some might be covered in low point value. Simple but

Topics for review Guidance: [Double Bracketed] topics are

(extra sheet will be allowed.

Final exam: Closed book/computer, like midterm, except one sheet of notes

Final exam in TC-19. Mon May 15 3:30-5:30(+15min)

A session; all recitation sections Wed and Fri.

0
1.2 Tree Nodes

1.4 Tree Traversal (in, pre, post order)
- A heap ordered tree.
- A binary search tree.

How to sort by building and using

1.5 Binary Search Trees

- Binary tree size/depth analysis
- B-trees
- Generalized binary search

1.3 Binary and Serial Search

13.1 13.2 Running time analysis of sorting algorithms.
- Selection Sort (Project 3)
- Mergesort in arrays (Project 3)

11.1 11.3 Heap sorting.
- What is a heap?
- What is a search tree?

14.1 Inheritance
5.4 Linked List Appl.
5.5 "Mix and Match" Reading
6.1 Template Functions
6.2 Template Classes
6.6-6.8 [STL and Iterators]
7.1 Stacks
7.2 Balanced () and 2-stacks Algorithms
7.4 Evaluating Postfix [opt. precedence rules]
8.1 Queue Intro.
8.2 Queue Appl: I/O Buffering
9.1 Recapitive Functions, Activation Records, Local Extant/Automatic
9.2 Rectangles and Mazes
9.3 Reasoning About Recursion
10.1 Trees

University at Albany Computer Science Dept.
(1.1) Specification, Design, etc.

(1.2) Run Time Analysis

(2) Classes, Separate CXX/ H Files, Build Scripts, RCS

(3) Containers

(4) Pointers, Dynamic Arrays, C-Strings

Structure/Class types, some fields being pointers, others not; function members.

(5.1) Linked Lists

(5.2) More Linked Lists

(5.3) Linked List Bags
question. (M/S confuse this with taxonomy trees).

Binary Decision Trees: Each leaf is an answer, each non-leaf is a yes-no.

(It is like a telephone book).

Other Name Space Trees: EG, the Domain Name System of the Internet.

Directory names: plus a file name.

File Name Trees: Express a system to identify files using a sequence of

search for "human".

Taxonomy Trees: See http://www.ncbi.nlm.nih.gov/Taxonomy/ and

expression (string, web document, program, etc.).

Expression Trees: Express the structure of the computation expressed by an

Tree Examples/Applications
$O$ (height) operations: number can be moved to the root, with the heap remaining heap-ordered, using

Heap qualities: (1) The largest number is in the root. (2) The next-largest number in each and each subtree $L$, the root contains the largest of the numbers in the left subtree $L$, the root contains the largest of the numbers in the heap property: A heap is a tree (of numbers) with the heap property.

For the tree and each subtree $L$, a given key is in the tree or not. It can express the structure of a search process to tell if a given key is in the tree. Subtree of $L$ is the number in the left. The left subtree of $L$ is less than the number in the left subtree of $L$. Every number in every subtree $L$, every number in the left subtree $L$ is greater than. Every number is in a finite set, by using questions of the form: Is it $u$ or $\bar{u}$? and number is in a given set. A decision tree for answering whether or not a given number of data in a tree.

Binary search tree: Trees used for searching and sorting (different problems, different arrangements...
courses prove theoretical results about sorting algorithms.

Running on $N$ elements, (This tree has $N$ leaves; it is used in Graduate CS)

Combination of outcomes of comparisons possibly made by that algorithm

Sorting state tree of a sorting algorithm: One node for EVERY

That ends in a win, loss or draw.

According to the rules of the game, this tree has one leaf for every game-play

Game trees: One node for EVERY legal combination of moves by the player(s)

More conceptual kinds of trees:

University at Albany Computer Science Dept.
search, (D) a species, genus, family, group or other taxonomy category, etc.

In various applications, the trees occur in two ways:

Explicit, implemented by a data structure: Each node of the tree is an object (objects ARE variables), with a C++ data type (often a structure

Implicit, helpful way for people to understand the application: Each node represents something more abstract, like (A) a subexpression, (B) the state of knowledge you got from answers to questions, or (C) progress in a process.
{ else return search(right subtree of T, k) ?
    if (T has no right subtree) return false;
    else
    
    { else return search(left subtree of T, k) ?
        if (T has no left subtree) return false;
        (if (k > key(root of T))
            if (k is in the root of T) return true;
            search(binary subtree, key k)
    )

Think of one explicit example: one binary search tree containing the keys 10, 20,
count = "No..."

{ count = "Yes..." return

( if (A[i] == K) ++ )

for(int i=0; i < n; i++)

// that's a precondition, so this code snippet fail if
// precondition: A[0..n-1] must be sorted.

// Sequential Search (not best)

array elements and output K's index, if K is in the array.

increasing order, and if input key K, tell whether or not K is in one of the

The sorted array search problem: Given an array A[ ] of n keys sorted in
count >> "\n\n  if \nBound = \nBound + 1; \n  \nelse \nBound = \nBound - 1; \n( [mid \nBound] \nK > \n\nBound \nBound \nBound \nBound, \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBound \nBoundary search in a sorted array: \n
Precondition [A[0..n-1] must be sorted. \n
Signed int Round = n-1; \n// in case 12.1. \nSigned int Round = 0; \n// Note the pitfalls. \n
University at Albany Computer Science Dept.
THINK: What if \( n \) is a \( G \), about \( 1 \) billion, which is \( 2^{30} \)?

previous length! So, the number of comparison steps is \( \geq \log_2(n + 1) \).

the subarray that remains to be searched is reduced to half or less of its

Why binary array search is efficient: After each \(< \) or \( > \) comparison operation,

If \( k > a[3] \), we restrict search to \( A[4] \ldots 7 \).

If \( k < a[3] \), we restrict search to \( A[0] \ldots 2 \).

\[
\begin{array}{cccccccc}
10 & 20 & 30 & 40 & 50 & 60 & 70 & 80 \\
\end{array}
\]

TRY it with \( n=8 \) on this array:

University at Albany Computer Science Dept.
This binary search tree expresses the structure of the search done by the pseudo-code. The diagram shows a binary search tree with keys 40, 60, 70, 80, 50, 30, and 10. Each node represents a comparison in the search algorithm.
Some kinds of traversals (whole-tree exploring) of trees:

1. Pre-order (EQUIVALENT to depth-first)

2. In-order

3. Post-order

4. Breadth-first
Breadth-first traversal of a tree
Depth-first traversal of a tree
Each arc expresses the structural relation between the root node and the
subtrees.

An expression is an identifier or constant,

or has a top level operator, excluding

or one operator and operands under

superiorly (and)

or more expressions as operands

(e) If it has an operator, it has one

or more expressions as operands

Any operator and operands under

overlapped!

(q) (p)

A non-empty tree has:

Zero or more rooted trees called

its subtrees, with no nodes or arcs in

its subtrees, with each other or the root,

common with each other or the root.

(e) One arc from this tree's root to

the root of each of the trees specified

under (q).

Each arc expresses the structural relation between the root node and the
subtrees.
used by the recursive evaluator.
The stack of activation records stores the subexpression values until they are

3. Solution to the problem of managing memory for the subexpression values:

order.

Recursive evaluation easily finds and executes all the operations in the right

2. The top level operation is in the root of the tree.

1. The expression tree directly reveals the order of operations.

Remember about expression trees:
root.

and keep moving upward to each node’s parent, you will eventually reach the
one. If you start at any node and move to the node’s parent (provided there is

3. Each node except for the root has exactly one parent; the root has no parent.

say that “is c’s parent” is that node’s children. If a node c is the child of another node d, then we

2. Each node may be [is] associated with [zero] or more different nodes, called

1. There is one special node, called the root.

empty, then it [the set of nodes] must follow these rules:

A tree is a finite set of “nodes” whose set might be empty (no nodes, which is

Main/Savitch’s definition:
{ } 

hold();  

cout << temp << " is the root. " << endl;  

while (temp is NOT the root )  

    temp = u;  

node temp;  

find-root(node u)  

The following algorithm, started at any node, always halts, eventually:

A more formal description of property A:
What about \#4?

Mathematics Prize:

If you can answer definitively, you win a Clay Institute of

A million dollar question: „Is \#3 harder than \#2?“ (More formally, „Does

Is it easier to solve \#2 than by trying to solve \#1?

How many large can the count of \#1 be?

any.

4. Given a maze, find ONE simple path of shortest length, or report there isn’t

(An Hamiltonian path problem"

3. Given a maze, and ONE simple path THAT VISITS ALL THE VERTEXES,

2. Given a maze, and ONE simple path from start to goal, or report that there

1. Given a maze, and list/count ALL simple paths from start to goal.

Pour Separate Problems:

(15.4) Path Finding (15.3) Graph Traversal

Assorted closing remarks...

University at Albany Computer Science Dept.
The number of length $N$-1 strings is $2^N$. For our $N=5$ example, using this rule provides a 1-1 function from length 5 (generally $N$) binary strings to strings 01100 corresponds to DDRDRDR.

So string 01010 corresponds to DDRDRDR. String 01100 corresponds to take right only steps to reach the goal.

When we reach the bottom row, left followed by one step down if the bit is 1, when we reach the bottom row, take one step down if the bit is 0, and take one step when we are at each row, take one step right, when we are at each row, take one step down. The following rule specifies how a length 5 string like this determines one solution:

We write the binary string 01010 down the left side. The right side of the following rule specifies steps.

ONLY WE illustrate the first such solution: 5 down steps followed by 5 right.

Some but not all of the solutions are formed by $N$ right and $N$ down steps.

$N = 5$.

In the figure shows the graph with row. So, in terms of Project 5, $N = 1$. The figure shows the graph with row.

Let $N$ be the number of down steps needed to go from the top to the bottom. The start and goal vertices are the upper left and lower right ones.

University at Albany Computer Science Dept.
can be packed into the known universe, etc.

However, the number of solutions, more than 2,100, the computer must print is way too big to be computed for mortal, the number of protons that

Consider \( N = 1000 \). The description of the maze can fit on a floppy disk.

Rule demonstrates that there are over 32 different solutions.
\[ 2^N \text{ paths.} \]

...take a number of steps proportional to \( N \) instead of getting nothing as many as

This way, the algorithm to find one path or determine that there is none, would

be reached from the start vertex.

is to put and retain a "mark" on each vertex as soon as we determine that it can

We sketched the operation of a "labeling" type search algorithm. The main idea

algorithm from the project.

any and print "none" if there are none, doing less work than the backtracking

and search algorithms that can solve problems #2, to find one solution path if

Let us compare problems #1 and #2. See 15.3 and 15.4 detail graph traverse
Depth-first Labeling Search
Breadth-first Labelling Search

0, 0, 1, 1, 0, 0, 3, 1, 4, 3, 0, 3, 2, 5, 0

here are the squares in the order they are inserted in the queue.
General and Linked-List Mergesort

(13.2) Quicksort, Mergesort in arrays

terables).

ADDITIONAL MEMORY NEEDED: except for a few control, swapping, etc.

Heapsort is an \( O(\log n) \) array-in-place-only sorting algorithm! (NO

parent \( \left\lfloor \frac{I}{2} \right\rfloor \) rounded down •

right child \( 2 + I \) •

left child \( 1 + I \) •

\( 0 < I \) Non-root - position \( 0 \)

Root - position

(ARRAY IMPLEMENTATION OF COMPLETE BINARY TREES)

13.3 Heapsort
understanding recursive definitions in computer science.

These rules apply to reading and writing inductive proofs in mathematics, and

BELIEVE (assume by induction) the recursive calls will work.

When you study whether it works in other cases, check that the recursive calls

\( \text{FIRST} \)

Golden rules for recursive programming:

1. The Principle of Induction Says: If \( u \) is true for ALL \( u \), then the Principle of Induction Says: \( p(1) \) is true.

2. You can prove \( p(u) \) AND \( p(u') \) if you assume \( p(u) \) is true for every \( u > u' \).

3. If you can prove: (1) \( p'(u, n) \) for all \( u < n \), then you can prove: \( p'(u, n) \) for all \( u < n \).

Principle of Mathematical Induction:

Whenever it is run on a list of keys, the mergeSort function will WORK.

My reasoning about recursion:

9.3 Reasoning About Recursion