pointers.
will try to provide material to help future learning of them all, and at least topic

Thank you: To students who responded to my request for topic suggestions. I

Balanced and Unbalanced Trees

More Mix-And-Match Topics

CSI 310: Lecture 27
12.3 Chained Hashing
12.4 Hashing Time Analysis
13.1 Selection Sort
13.2 QuickSort, mergeSort in arrays
13.3 HeapSort
13.4 Std. sort funs.
14 Inheritance
15.1 Graph Models
15.2 Graph Data Str. Impl.
15.3 Graph Traversal
15.4 Path Finding

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

Final exam as scheduled.

O/A Session: Mon 5/12, 11-98, 12:00-3:00PM
University at Albany Computer Science Dept.
8.4 Priority Queues

Priority Queues in Discrete Event Simulation

9.1 Recursive Functions, Activation Records, Automatic Variables

9.2 Fractals and Mazes

9.3 Reasoning About Recursion

Expression Trees

10.1 Trees

10.2 Tree Nodes

10.4 Tree Traversal

10.5 Binary Search Trees

11.1 Heaps

11.2 B-trees

11.3 Binary tree size/depth analysis

12.1 Binary and Serial Search

12.2 Open Address Hashing

University at Albany Computer Science Dept.
3 Containers
4 Pointers, Dynamic Arrays, C-strings
5.1 Linked Lists
5.2 More Linked Lists
5.3 Linked List Bag
5.4 Linked List Appl.
6.1 Template Functions
6.2 Template Classes
(6.6-6.3) STL and Iterators
7.1 Stacks
7.2 Balanced () and 2-stacks Algorithm
7.4 Evaluating Postfix (opt. precedence rules)
8.1 Queue Intro.
8.2 Queue Appl.: I/O Buffering
8.3 Queue Impl.

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

1.2 Run Time Analysis

2 Classes, Separate CXX HH Files
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? 

or reports there isn't any. (\#3 is the "Hamiltonian Path Problem")

1. Given a maze, find ONE simple path that visits ALL the vertices.

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

isn't any.

2. Given a maze, find ONE simple path from start to goal.

I. Given a maze, find/just count ALL simple paths from start to goal.

Three Separate Problems:

(15.3) Graph Traversal

(15.4) Path Finding

University at Albany Computer Science Dept.
The number of length \( N \) 0-1 strings is \( 2^N \). For our \( N = 5 \) example, we see that

one from this rule.

Our rule provides a \( 1 \) to \( 1 \) function from length \( 5 \) (generally \( N \)) binary strings to

\[ \text{DRDRDPRR} \]

So string 01010 corresponds to \text{DRDRDPRR}. String 01100 corresponds to

\[ \text{DDPRDDRR} \]

take right only steps to reach the goal.

\[ \text{DDPRDDRR} \]

left followed by one step down if the bit is \( 1 \). When we reach the bottom row,

When we are at each row, take one step down if the bit is \( 0 \), and take one step

\[ \text{DDPRDDRR} \]

When we are at each row, take one step down if the bit is \( 1 \). When we reach the bottom row,

\[ \text{DDPRDDRR} \]

the following rule specifies

\[ \text{DDPRDDRR} \]

how a length \( 5 \) string like this determines one solution:

\[ \text{DDPRDDRR} \]

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

\[ \text{DDPRDDRR} \]

some but not all of the solutions are formed by \( N \) right and \( N \) down steps.

\[ \text{DDPRDDRR} \]

ONLY. We illustrated the first such solution: 5 down steps followed by 5 right

\[ \text{DDPRDDRR} \]

The figure shows the graph with

\[ \text{DDPRDDRR} \]

\( N = 5 \).

\[ \text{DDPRDDRR} \]

\( u = N \) - 1. The figure shows the graph.

\[ \text{DDPRDDRR} \]

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.
(can be packed into the known universe, etc.)

way too big to be computed for mortals. (more than the number of protons that

However, the number of solutions, more than 2^{1000}, the computer must print is

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

there are over 32 different solutions.
take a number of steps proportional to $N^2$ instead of getting finding as many as $2^N$ paths. This way, the algorithm to find one path or determine that there is none, would be reached from the start vertex.

We sketched the operation of a "labelling" type search algorithm. The main idea of this is to put and retain a "mark" on each vertex as soon as we determine that it can be reached from the start vertex.

if there are none, doing less work than the backtracking and search algorithms that can solve problems #2, to find one solution path. Let us compare problems #1 and #2. Sec. 15.3 and 15.4 detail graph traverse.

University at Albany Computer Science Dept.
General and Linked-List Mergesort

13.2 Quicksort, mergesort in arrays

ARRAY REPRESENTATION OF COMPLETE BINARY TREES

13.3 Heapsort

13.4 Std. sort funs.

14 Inheritance(SKIPPED)

15.1 Graph Models

15.2 Graph Data Str. Impl.
Expression Trees
10.1 Trees
10.2 Tree Nodes
10.4 Tree Traversal
10.5 Binary Search Trees

Heaps
11.1 Heaps
(11.2) B-trees
11.3 Binary Tree Size/Depth Analyses

Binary and Serial Search
12.1 Binary and Serial Search
(12.2) Open Address Hashing
(12.3) Chained Hashing
(12.4) Hashing Time Analyses

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

* Then, if there are no problems that are SMALLER THAN the case you are studying, THEN when you study whether it works in other cases, check that the recursive calls work.

Make sure it works for the base cases (FIRST):

**Golden rules for recursive programming:**

THEN the PR of Induction Says: $P(1) (u) P$ is true for ALL $u > 1$.

If you can prove: (1) If $P$ is true for every $u$, $u > 1$, you can prove: (2) $P(2) (u) P$ AND (3) $P(1) (u) P$ is true.

PRinciple of Mathematical Induction:

```
mergeSort function will WORK whenever it is run on a list of n keys:
```

My $P(u) (u)$ is a mathematical statement about positive integer $u$, for example: "My

9.3 Reasoning About Recursion
5.3 Linked List Bag
5.4 Linked List Appl.

6.1 Template Functions
6.2 Template Classes
(6.6-6.3) (STL and iterators)

7.1 Stacks
7.2 Balanced () and 2-stacks Algorithm

7.4 Evaluating Postfix (opt. precedence rules)

8.1 Queue Intro.
8.2 Queue Appl: I/O Buffering
8.3 Queue Impl.
8.4 Priority Queues

Priority Queues in Discrete Event Simulation

9.1 Recursive Functions, Activation Records, Automatic Variables
9.2 Fractals and Mazes
1.1 Specialization, Design, Pre/Postconditions, etc.

search

1.2 Run Time Analysis (time proportional to $n$ for linear search, $\log n$ for binary

2 Classes, Separate CXX/Hi Files

3 Containers

4 Pointers, Dynamic Arrays, C-strings

5.1 Linked Lists

5.2 More Linked Lists