Summary of Recursion/Induction.
Array implementation of binary trees and heapsort.
Glimpse at an NP-complete problem.
Topics
CSI 310: Lecture 27
General and Linked-List Mergesort (Merging of course)

13.2 QuickSort, mergesort in arrays

13.3 HeapSort

13.4 Std. sort funs.

14 (Inheritance (SKIPPED))

15.1 Graph Models

15.2 Graph Data Structs. Impl.

15.3 Graph Traversal

15.4 Path Finding

Topics for review guidelines: (Parenthetical) topics are "honors" / "optional"... same

(crib-sheet) will be allowed.

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

Final exam in TC-23: Thur, May 12, 1:00-3:00

O/A sessions: all lab sessions until and including Wed, 5/5

University at Albany Computer Science Dept.
10.1 Trees

10.2 Tree Nodes

10.4 Tree Traversal (in, pre, post order)

- A heap ordered tree
- A binary search tree

How to sort by building and using a binary heap search tree

11.1 Heaps

11.2 (B-trees)

11.3 (B-tree size/depth analysis)

12.1 Binary and Serial Search

12.2 (Open Address Hashing)

12.3 (Chained Hashing)

12.4 (Hashing Time Analysis)

13.1 Selection Sort
(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 App.: I/O Buffering

8.3 Queue Impl.

8.4 Priority Queues

Priority Queues in Discrete Event Simulation

9.1 Recursive Functions, Activation Records, Local/Automatic Variables

9.2 Fractals and Mazes

9.3 Reasoning About Recursion

Expression Trees
1.1 Specification, Design, etc.
1.2 Run Time Analyses
2 Classes, Separate CXX/H Files
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 Bag
5.4 Linked List Array
5.5 "Mix and Match" Reading
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?"

Does it easier to solve \#2 then by trying to solve \#1?

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

any.

1. Given a maze, find ONE simple path or shortest length, or report there isn't

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

3. Given a maze, find ONE simple path that visits ALL THE VERTEGES

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

(15.4) Path Finding (15.3) Graph Traversal

Pour Separate Problems:

Assorted closing remarks...

University at Albany Computer Science Dept.
The number of length \( N \) \( 0 \)-1 strings is \( 2^N \). For our \( N = 5 \) example, using this rule, the ones from this rule.

Now that there are (many) more solutions than solution paths. It is not an onto function; there are a function from length 5 (generally \( N \)) binary strings to \( 2^N \) corresponds to \( DDRDDDR \). String 0110 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 down (the following rule specifies how a length 5 string like this determines one solution: how the binary string 0101 01 01 10 10 down the left side. The following rule specifies steps.

ONLY. WE illustrated 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.

\( N = 5 \).

\( u = N - 1 \). The figure shows the graph with row. So, in terms of \( P \) of project 5, \( N \) \( = u \), 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 packaged 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.
rule demonstrates that there are over $32$ different solutions.
take a number of steps proportional to \( N^2 \) instead of getting this many 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.

is to put and retain a "mark" on each vertex as soon as we determine that it can be reached from the start vertex.

We sketched the operation of a "labelling" type search algorithm. The main idea algorithm from the project.

any and print "none" if there are none, done less work than the backtracking and search algorithms that can solve problem \( \#2 \), to find one solution path if and search algorithms that can solve problem \( \#2 \). See 15.3 and 15.4 detail graph traverse.

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

Here are the squares in the order they are inserted in the queue:

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

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

0, 5, 4, 3, 5, 5, 5, DONE!

Start
General and Linked-List Mergesort

(13.2) QuickSort, Mergesort in arrays

variables).

ADDITIONAL MEMORY REQUIRED, except for a few control, swapping, etc.

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

parent rounded down

$\frac{I}{2}$

right child

$2 + I \times 2$

left child

$I + I \times 2$

Non-root - position

$0 < I$

Root - position

0

(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.

Then, you are studying. Then, whenever it works in other cases, check that the recursive calls make sure it works for the base cases (first):

Golden rules for recursive programming:

1. If you can prove: (1) For all $u > 1$, if you assume $P(n)$ is true for every $n < l$, then $P(n)$ is true for all $u$.
   
   THEN the principle of mathematical induction.

   Principle of Mathematical Induction:

   Whenever it is run on a list of numbers $P$ is a mathematical statement about positive integer $n$ for example: "My mergesort function will work whenever it is run on a list of $n$ keys."

9.3 Reasoning About Recursion