Summary of Recursion/Induction.
Array impl. of binary trees and Heapsort.
Glimpse at an NP-complete problem.

Topics

CSI 310: Lecture 27
General and Linked-List Mergesort

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 Str. Impl.

15.3 Graph Traversal

15.4 Path Finding

Topics for review/exam:
- Some (parenthesized) topics are "honors"/optional—some might be covered in low point value, simple but subtle final exam questions.

( crib-sheet) will be allowed.

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

Final exam in TC-2: Monday May 10 3:30-5:30

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

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

12.3 Chained Hashing

12.4 Hashing Time Analysis

13.1 Selection Sort

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

1.2 Run Time Analysis

2 Classes, Separate .CXX/ .H Files

3 Containers

4 Pointers, Dynamic Arrays, C-Strings

5 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 AppI.
If you can answer definitively, you win a Clay Institute of Mathematics Prize.

A million dollar question: "Is \#3 harder than \#2?" (More formally, "Does it is 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")

3. Given a maze, find ONE simple path that visits all the vertices.

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

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

Three Separate Problems:

(15.4) Path Finding (15.3) Graph Traversal

Assorted closing remarks...
The number of length \( N \) 0-1 strings is \( 2^N \). For our \( N = 5 \) example, we see that

one from this rule. It is not an onto function; there are (many) more solutions than the

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

So string 01101 corresponds to DDRDDDRRR. String 01101 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. Whenever we reach the bottom row,

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

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

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 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 Project 6, \( u = N \) steps needed to go from the top to the bottom

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.
$2^N$ paths.

Instead of getting nothing as many as $N$, take a number of steps proportional to $N^2$. 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 ideas:

- the algorithm from the project.
- any and print "none" if there are none, done.
- 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.
General and Linked-List Mergesort

(13.2) QuickSort, mergesort in arrays

variables.

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

Heapsort is an \( O \log n u \) array-in-place-only sorting algorithm!

\( \text{NO} \)

\[ \text{parent \ rounded\ down = } \frac{n}{2} \]

\[ \text{right\ child} = 2 + \text{parent} \]

\[ \text{left\ child} = 1 + \text{parent} \]

\[ 0 < I \text{ position - root} \]

\[ 0 \text{ root - position} \]

ARRAY IMPLEMENTATION OF COMPLETE BINARY TREES

13.3 Heapsort

University at Albany Computer Science Dept.
Understanding recursive definitions in computer science.

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

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

THEN the PR. of Induction says: \( p(u) \) is true for ALL \( u \) if you can prove: (1) \( p(1) \) is true. \( u > 1 \) if you assume \( p(f(u)) \) is true for every \( u \) such that \( 1 < u \).

IF you can prove: (1) \( p(1) \) is true.

The Principle of Mathematical Induction:

Whenver it is run on a list of \( n \) keys, merge sort function will WORK.

My reasoning about recursion is a mathematical statement about positive integer \( n \) for example: "My

9.3 Reasoning About Recursion