Graphs and Graph Search, for Project 5

Binary Search

Tree Application Review

CSI 310: Lecture 21
Binary Search Tree Property: For the tree and each subtree \( T \), every number in the left subtree of \( T \) is < the number at the root, and every number in the right subtree of \( T \) is > the number at the root. Also, each node, not the question.

Is it \( u \) "? Only each \( u \) is stored in each node, not the question, "Is it \( u \) to \( u \)" and number is in a finite set, by using questions of the form: "Is it == to \( u \)"

Binary Search Tree: A decision tree for answering whether or not a given question (M/S consults this with taxonomy trees).

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

Other Name Space Trees: EC the Domain Name System of the Internet
directory names, plus a the name.

File Name Trees: Express a system to identify files using a sequence of search for "human".

Taxonomy Trees: SEE http://www.nclth.gov/ontology and expression (string, web document, program, etc.).

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

---

University at Albany Computer Science Dept.
For the tree and each subtree \( T \), the root contains the largest of the numbers in the heap property:

**Heap Ordered Tree** ("Heap") \( A \) tree (of numbers) with the heap property:

\( T \).
(2) Use the recursion idea to conceive of the solution.

Ideas (1) Use the recursion idea to conceive of the solution.

They are in a heap-ordered array-represented tree.

Given $N$ numbers stored arbitrarily in an array, how to rearrange them so

begins with the array representation.

stored in an element of a partially filled array. Study DSO section 10.2 which

nodes of a complete binary tree. The data in each node can then be

$N \leq \text{THE AS THE nodes instead of linked nodes.}$

$N = 0, \ldots, I$.

describe:

remove the top number (which is greatest) and "re-heapify" the tree. We then

We then viewed parts of DSO section 11.1 on Heaps, and described how to

search tree.

example of how a dictionary of states (or the USA) is implemented by a binary

We viewed Main's slide set 10b on the dictionary abstract data type and the

into it.

and described how to search and insert a new number (or other key, like a name)

We viewed Main/Summary's slides supporting section 10.5 on Binary Search Trees.

Binary Trees.

We viewed the Animal Guessing Game decision tree from DSO section 10.3 on

University at Albany Computer Science Dept.
Reformulate the solution to the problem using iteration instead of recursion.

University at Albany Computer Science Dept.
(Revise this in CS1403.)

sections, and I will do (some) more of it in lectures upon request. (At Albany, we

Search Trees will be covered in the final. So study those textbooks

Definitions, operations, and algorithms for both Heaps and Binary

Worst case.

use an amount of computer work proportional to the height of the tree, in the

Insertion into a heap,

and

Search and insertion in a binary search tree,

Main conclusion:
Some Applications of Trees:

- Express the structure of a search process to tell if a given key is in the tree or not.
- Express the structure of a decision process to find an answer.
- A Decision Tree (of yes-no questions for non-leaves and answers for leaves)

(3) A Search Tree (of numeric or other keys which can be compared with < and >)


def binary_search_tree(root, key):
    if root is None:
        return False
    elif root.key == key:
        return True
    elif key < root.key:
        return binary_search_tree(root.left, key)
    else:
        return binary_search_tree(root.right, key)


tree when you input these numbers, in any order.

Think of one explicit example: one binary search tree containing the keys 10, 20,
30, 40, 50, 60, 70 and 80. The demo program of L22 will build and print such a
taxonomy category, etc.

search, (D) a species, genus, family, group or other taxonomy category, etc.

state of knowledge you get from answers to questions or (C) progress in a
node represents something more abstract, like (A) a subexpression, (B) the
implicit, helpful way for people to understand the application. Each
object (D) variables) with a C++ data type (often a structure

In various applications, the trees occur in two ways:

University at Albany Computer Science Dept.
else return search(right subtree of T, k);
count >> No...`

{ count >> Yes...}

\( \text{if} \ A[i] \neq K \) ; \( ++i \)

for(int i=0; i < u (\exist (A[i] \neq K)) \text{ if (that's a precondition, so this code might fall if)}

\( \text{precondition: A[0..n-1] must be sorted.} \)

Sequential Search (not best):

array elements and output K's index. If K is in the array,
increasing order, and an input key K, tell whether or not K is in one of the

The sorted array search problem: Given an array A[0..n-1] of n keys sorted in
count >> "No..." >> end

else

{ bounded = mid + 1
 }

else

{ bounded = mid - 1

if A[bounded] > K

{ yes >> mid >> end

if A[bounded] == K

// rounds down

if bounded + bounded/2; M =

while bounded > bounded [ end ]

THEN K must be in A[bounded..bounded]

[ if K is in A[0..n-1]

if Loop Invariant:

INT mid;

then int bounded = n-1;

// in DSO 12.1

// Note the parenthesis.

Precondition A[0..n-1] MUST BE SORTED

// Binary Search (in a sorted array):
THINK: What if \( u \) is a GIC, about 1 billion, which is \( 2^{30} \)?

1. \( u (u) + \log_2 \)

previous length. So, the number of comparison steps is reduced to half or less of its

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 \( A[0 \ldots 2] \).

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

TRY it with \( u = 8 \) on this array:
This binary search tree expresses the structure of the search done by the pseudo-code.

Cool, cover HASHING briefly (for 402, 403): See I2.2-3
Assigned Reading (among others): See I2.1
used by mortals to store.
The game state graphs of games like chess or Go are too big for any computer.

By the first vertex into the configuration of the second vertex called an edge of one legal move can change the configuration represented.

Game State Graph: Each node (also called vertex) represents some board position or configuration of a game. Two vertices are connected by an arc (also

...wite solution software...)

The solution is obtained by solving the equation (you can buy/donload or apply).

It's like applying match. Often a problem can be represented as an equation, and

problem on the corresponding graph.

Graphs in Problems Solving: Often a problem can be represented as

on the needs of an application.

Graph nodes may be linked in any pattern or lack of pattern-dependent.

... But in a graph, even this modulus of order [of a tree is between the nodes.]

A graph, like a tree, is a nonlinear data structure consisting of nodes and links.

Project 5.

We skip to Chapter 15 on Graphs, now! But only 15.1 and 15.3 for...
Problem: Can you get from the start position HTH to the goal position HTH?

(X, X, X) ← X, X, X and (X, X, X) ← X, X, X) as each other.

2. You may flip one of the end coins only if the other two coins are the same.

(Z, X, X) ← Z, X, X) (as you may flip the middle coin whenever you want to.

Rules:

Following:

Each move consists of turning over one of the 3 coins, according to these

Tail, and Head up in that order (denoted HTH).

The board is 3 coins placed in a row. At the start, the coins are turned Head, and a Move

Example of 2nd page of 15.1:
Let's draw the game state graph:

\[ \text{HHH} \rightarrow \text{HXH} \leftrightarrow \text{HX'H} \leftrightarrow \text{HTH} \]
There were two ways to get to the same position!
Graph tell us about how many moves it takes to win.

What does this

The same game state graph drawn differently, p. 698-699.
The computer sometimes computes from the data in MPSet the representation of the graph vertex, such as THE CURRENT SAME POSITION MPSet.

The data or value in a data structure, (i.e., object or variable) represents ONE IMPPLICIT:

(set<int>);

3. Edge Sets: Like edge lists, except a SET of edge lists, where the data type(such as STL

  vertex j,  

  The ith linked list contains all the numbers of the neighbors of number j. The ith linked list contains all the numbers of the neighbors of number j. Each node holds one vertex edge by A[1][j] = false. An edge from vertex i to vertex j is signified by A[1][j] = true, no such

  I. Adjacency Matrix: An n by n array A[n][n] is used.

  (n - 1).

  ...
adjacent or horizontally wrapped empty squares.

All (simple) paths from square $(h, f, x, f)$ to square $(h, s, x, s)$ with empty squares at $(h, f, x, f)$ and $(h, s, x, s)$ are empty.

Output: $n$ by $n$ lattice of squares, some “filled in” and the rest empty. Assume the maze.

Make the computer find all simple paths through a planar square lattice based.

Problem of Project 5:

University at Albany Computer Science Dept.
You can prepare the input in a grid or not, your choice! Note what the first 5 numbers are for.

 cin >> "... for input makes the program

 cin >> A[i][j];

 cin >> .." for input makes the program

 cin >> A[i][j];

 for (int t=0; t<n; t++)
 for (int u=0; u<n; ++u)
     cin >> u << sX << sY << dx << dy;

 cin >> n << sX << sY << dx << dy; // data members
     int n, sX, sY, dx, dy;  // maze class
     int A[24][24];  // maze class

 int A[n][n];  // Maze class

 partially filled:

 0 in a C++ 2-dim array

 Squares with 1, empties with

 Array View: Represent  Filled

 Maze Image View

 University at Albany Computer Science Dept.
Graph Representation

Example of an induced subgraph containing 0's.

Vertices: Array positions of this Maze: Maze

Image of a Different
Here is the rule for telling which array positions are neighbors:

$$\begin{align*}
(i, j) &\quad (i-1, j) \\
(\lfloor i/4 \rfloor, i) &\quad (\lfloor i/4 \rfloor, i+1) \\
(\lfloor i/4 \rfloor, j) &\quad (\lfloor i/4 \rfloor, j-1) \\
end{align*}$$

edge

$$(\lfloor i/4 \rfloor, j)$$
(\textcolor{red}{\mathcal{L}}, \textcolor{red}{\tau} + \textcolor{red}{\mathcal{L}}) \\
\textcolor{red}{\text{(N mod (\textcolor{red}{\tau} + \textcolor{red}{\mathcal{L}})) \rightarrow (\textcolor{red}{\mathcal{L}}, \textcolor{red}{\tau}) \rightarrow (N \text{ mod (\textcolor{red}{\tau} - \textcolor{red}{\mathcal{L}})) , \textcolor{red}{\tau})}} \\
\textcolor{red}{(\textcolor{red}{\mathcal{L}}, \textcolor{red}{\tau} - \textcolor{red}{\mathcal{L}})} \\
\text{(n \text{ where } equiv (n,p,q) = true \text{ means } p = q \mod n)}
either operand is negative, the behavior will be machine-dependent... (Harbison)

the behavior of the remainder operation is coupled to that of integer division... If

The FACTS: It is always true the \((a/b) \times b + a \% b\) is equal to \(a\) if \(b\) is not 0, so

```cpp
{ return x;
  if (0 <= x)
    { u + x = x
      assert(u + x = x
    if (0 > x)
      N % x = x
      assert(N % x = x
    }
} int mod (int n, int x)
```

So, roll your own: you can copy this into your project...

The C++ "remainder" operator \(\%\) sometimes behaves this way:

**negative integer \(\%\) is NEGATIVE**
{
    return mod(p, n) == modn(p, n);
}

{ // of equivalence classes
    return concatenated representations
    // compare and compute
}

(bool equivalent(int n, int p, int q) {
    // Two ways to code "mod N equivalence" testing, based on two mathematical
    return (n % (p - d)) == 0;
    //
    // Test if difference is divisible by n.

    //
    // characteristics:
}
Here is the rule for telling which array positions are neighbors: (i,j) form an edge. The array idea is hexagonal. Let's do a hexagonal maze.

\[
\begin{align*}
(\zeta, \tau+1) & \quad (\zeta, \tau-1) \\
(\tau+1, \zeta) & \quad (\tau, \zeta) \\
(\tau, \tau-1) & \quad (\tau, \tau+1)
\end{align*}
\]
(i, i + 1)

(i + i, i - 1)

(i + 1, i)

(i, i - 1)

(i - 1, i)

(i - 1, i + 1)

(i - 1, i + 1)

(i, i + 1)

(i, i - 1)

(i + 1, i - 1)

(i + 1, i + 1)

(i - 1, i + 1)

(i - 1, i + 1)

(i - 1, i + 1)

(i - 1, i + 1)

(i - 1, i + 1)

(i - 1, i + 1)
adjacent empty squares.

The input is a lattice of squares, some filled in, and the rest empty. Assume the squares at $(i,f,x,f)$ and $(h,s,x,s)$ are empty. Output: All simple paths from square $(i',1')$ to square $(0,0)$. Find all simple paths through a planar square lattice based.

Problem of Project 5:

University at Albany Computer Science Dept.
Sample output (first, repeated input):

```
0 0 0 1
1 1 0 1
0 0 0 0
1 1 0 0
4 3 0 0
```

Sample input:

```
0 0 0 1
1 1 0 1
0 0 0 0
1 1 0 0
4 3 0 0
```
Number of solutions = 3

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Then, print each solution, proceeded by a blank line, with each empty square in the solution path printed by a capital letter instead of a 0.
Let's name it PrintPostitions (\ldots).

6. Apply the recursion/divide and conquer pattern: Make the key procedure recursive.

5. What data does the key procedure need? It needs access to the whole maze, (hf, fx).

4. What should the key procedure do? Print each path that ends at the goal square (sx, sy).

3. Every journey begins with the first step. The key procedure will be called correctly and to help develop code to print the paths, 2. Print n, start and goal squares and the maze; to verify it was formed squares of the maze. (DONE)

1. Input the maze size n, row and column coordinates of the start and destination squares, and then n? 0's or I's to represent the filled and empty spaces. Solution Strategy
7. Each solution path must be simple, which means it must not use the same square twice. (Otherwise, there might be an infinite number of solutions). This condition has a flavor from the $n$-Queens problem (see a future lecture): After some queens have been tentatively placed in rows 1, 2, ..., $i$, THOSE queens restrict where the $(i + 1)$st queen may be placed in row $(i + 1)$.

8. So, PrintSolutions(...) MUST be able to tell which square are in the path found SO FAR. This data varies with where we are in doing the work. So, let’s try making the PATH FOUND SO FAR be a parameter to PrintSolutions(...).
a square outside the $\$y$ by $\$y$ maze.

or, if the current square is not the goal square, use a

path to the next square. If the goal square is reached,

each path found so far is printed.

Far and continuing to the goal square $技(fy)作$ are printed.

All paths beginning with the given path so far:

path found so far.

node starts a (linked) list holding all the squares in the

most recently found square in the path found so far. This

leads theプリント条件: path to the address of a node holding the

プリント状態条件: path & node *plast + staples of the

9. Direct algorithm pseudo-code for
{ } else clause done
{ } /*for loop finishes*/
{ } /*for loop finishes*/

printSolutions (pnext) ;
pnext-1->next = this square ;

) // END IF NOT in the path so far

AND NOT in the path so far

AND is empty, AND is empty,
ACCORDING TO RULE [4] IF (the square one step to pnext-1->data...

for (b=0 ; b<n ; b++)

pnext->link = plast ;
pnext = new PathListNode ;

int g ; AUTOOMATIC VARIABLE

}
else

{ } PRINT THE PATH FOUND SO FAR ; RETURN ;

( IF (pnext) ; PRINTSOLUTIONS (PATHLISTNODE *PLAST) ;

10. 2D PRINTSOLUTIONS (PATHLISTNODE *PLAST) ;
Let's get back to main ideas now.

Find and fix it yourself.

Whoops... There's a memory leak!
can access the (private) the 2-dim array and the size variables. Theses must be member functions of class maze so they
empty square. Theses are member functions of class maze which is an index pair is within the 0...n-1, 0...n-1 range it corresponds to an
Test if a given index pair (denoting a square) is in a linked list. (B) Test if
design helper methods to help implement its operations. For example, (A)
designed in the following way. (A)
Analyize the pseudo-code in the outline of the printSolution() method to
so far.
pathNode to implement the linked list of squares that is the path found
The Maze class uses some kind of linked list node struct or class type
printSolution()
method may be accessed by any methods that need them. Such methods include
private data members so they
2. The 2-dim A array and size variable are private data members
reading it, printing it, printing paths, and printing all the solution paths.

1. Design a class named Maze to model one maze, and have methods for

Design Ideas

University at Albany Computer Science Dept.
detests.

4. Write a skeleton implementation the `Maze.cxx` which must include `maze.h`.

You have in your mind.

3. Check that the function declarations, data member declarations and pre-conditions and postconditions are consistent with each other and with the intentions.

2. Start writing the file `Maze.h`, and try to write pre- and postconditions to document what you want each function member to do.

1. Invent (or use our given) names for the data and function members.

Complete the Design of the `Maze` class.
those that read in the maze data and print it out.

3. Finish coding, testing and debugging the skeleton functions, beginning with

warning, errors and linking errors go away.

2. You might want to write test drivers for some of the helper functions.

I. Code the main module; edit, compile and try to link it until all compilation

Implementation and Test