Graphs and Graph Search, for Project 6

Binary Search

Tree Application Review

CSI 310: Lecture 23
tree or not.

(3) A Search Tree (of numeric or other keys which can be compared with > and =) expresses the structure of a search process to tell if a given key is in the tree.

(2) A Decision Tree (of yes-no questions for non-leaves and answers for leaves)

(1) An Expression Tree expresses the structure of an expression.

Some Applications of Trees:
else return search( right subtree of T', k );
    if ( T' has no right subtree ) return false;

else

    else return search( right subtree of T', k );
    if ( T' has no right subtree ) return false;

    if ( k > key( root of T') )
        if ( k is in the root of T ) return true;
    search( binary subtree T', key k )

Search (binary subtree T', key k )

Think 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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

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 T22 will build and print such a

Think of one explicit example: one binary search tree containing the keys 10, 20,
count >> \texttt{end};
{
  \textbf{count} >> i >> \texttt{end}; return;
}

// precondition: \texttt{A[1..n-1]} MUST BE SORTED.

// Sequential Search (not best):

array elements and output \texttt{K}'s index, if \texttt{K} is in the array.

increasing order, and an input \texttt{K}'s tell whether or not \texttt{K} is in one of the

The sorted array search problem: Given an array \texttt{A[]} of \texttt{u} keys sorted in
count >> "No..."

else

{ t = k - 1;
  bound = k - 1;
  if (k > A[mid])
    { if (count >> "yes" >> mid >> end; return;
        (if k == A[mid]
         (if (mid = (bound + bound) / 2);
         
         while (bound >= k)
          { when A[bound]...
            THEN k must be in A[bound].
            [if k is in A[0..n-1]
             // Loop Invariant:
             // int mid;
             // signed int bound = n-1; // on page 563.
             // signed int bound = 0; // note the prettiness
             // precondition A[0..n-1] must be sorted.
             Binray search (in a sorted array):
             University at Albany Computer Science Dept.
THINK: What if \( n \) is a GIC, about 1 billion, which is about 2.3? 

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

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

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

A sample binary search is shown on this array:

Try it with \( n = 8 \) on this array:
This binary search tree expresses the structure of the search done by the pseudo-code.
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 (when one legal move can change the configuration represented
also position of configuration of a game. Two vertices are connected by an arc (also

Game State Graph: Each node (also called vertex) represents some board,

write solution software.

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

“problem on the corresponding graph.”

a graph, and the solution to the problem in obtained by solving a
graphs in Problem Solving: Often, a problem can be represented as

on the needs of an application.

graphs. Graph nodes may be linked in any pattern—or lack of pattern—dependent
between the nodes. ... But in a graph, even this modicum of order [of a tree] is

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

716-726 for Project 6.

We skip to Chapter 15 on Graphs, now!! But only 695-702 and
Problem: Can you get from the start position HTH to the goal position THT?

\((XXA \leftarrow X A A \text{ and } AXX \leftarrow AX X)\)

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

\((ZAX \leftarrow Z AX)\)

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.

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

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.

Game Board and What is a Move

Example of p. 697-9:
Let's draw the game state graph:
Rule 2

HHT ↔ XHH

Rule 2

HHX ↔ HXH

XHH ↔ HHX

HHH

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 MyPos the representation of

\[
\text{THE current graph vertex, such as } \text{THE current graph position MyPos.}
\]

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

Implicit:

\[
\text{set<int> & set> (such as STL)}
\]

3. Edge Sets: Like edge lists, except a set abstract data type (such as STL

vertex i.

The ith linked list consists of all the numbers of the neighbors of

number. The ith linked list contains all the numbers of the neighbors of

2. Edge Lists: In linked list, each node pointer is used. Each node holds one vertex

edge by A[i][j] == false.

An edge from vertex i to vertex j is signified by A[i][j] == true, no such

1. Adjacency Matrix: An n by n array A[i][j] is used.

\[
\begin{array}{c}
\ldots
\end{array}
\]

Suppose the graph has n vertices. The vertices are numbered 0, 1, \ldots

Explicit (Suppose the graph has n vertices. The vertices are numbered 0, 1, \ldots

Graph Implementations (Sec. 15.2 in brief)

University at Albany Computer Science Dept.
Use adjacent empty hexagons only.

Output: All simple paths from hexagon \((0,0)\) to hexagon \((u'-I, u-I)\) that the hexagons at \((0,0)\) and \((u'-I, u-I)\) are empty. Assume input \(u\) by \(n\) lattice of hexagons, some "filled in" and the rest empty. Assume based maze.

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

Problem of Project 6:

University at Albany Computer Science Dept.
2-dim array: Represent filled cells with 1, empty cells with 0 in a C++ array view.

Maze Image

Array View: Represent

View
(Induced subgraph)

Example of an
containing 0's.
Vertices: Array positions
of this Maze:

Image of a Different
Maze

Graph Representation
Here is the rule for telling which array positions are neighbors (form an edge):

\[(i-1,j+1), (i,j), (i-1,j), (i,j+1), (i+1,j-1), (i+1,j)\]
\[(i, j + 1), (j, i + 1), (i - j, i), (j - i, j)\]

For one of the \( q = 0, 1, 2, 3, 4, \) or 5.

\[
\text{dy} [\text{dy}] y = \lambda + \text{Rule} [y] \times x \quad \text{and} (3)
\]

\[
= [\lambda', x] 0 = 0 \quad \text{and} (2)
\]

\[
u \quad \text{is adjacent to (1)}
\]

\[
\{\{0,0\}, \{0,1\}, \{1,0\}\} =
\]

\[
\text{Table based coding of our graph adjacency rule:}
\]
Problem 6:

Input: $n$ by $n$ lattice of hexagons; some "filled in" and the rest empty. Assume the hexagons at $(0,0)$ and $(n-1,n)$ are empty.

Output: All (simple) paths from hexagon $(0,0)$ to hexagon $(n-1,n)$ that use adjacent empty hexagons only.

University at Albany Computer Science Dept.
Sample output (first, reprint input maze, indented):

```
0 0 0 1 1
0 1 0 0 0
0 0 1 1 0
0 0 0 1 0
1 1 1 0 0
```

Sample input:

```
0 0 0 1 1
0 1 0 0 0
0 0 1 1 0
0 0 0 1 0
1 1 1 0 0
```
Number of solutions?  
Where, of course,  

A few more...  

<table>
<thead>
<tr>
<th>0 0 1 1</th>
<th>0 0 1 1</th>
<th>0 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>0 0 1 1</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td></td>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

In the solution path printed by a instead of a 0:

Then, print each solution, preceded by a blank line, with each empty hexagon
Let's name it `printSolutions`(...).

recursive.

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

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

Goal hexagon (n, n - 1, u - 1, I)

Print each path that ends at the

4. What should the key procedure do? Print each path that ends at the

with a parameter that indicates the first hexagon (0, 0).

3. Every journey begins with the first step. The key procedure will be called

print the paths.

2. Print the maze; to verify it was formed correctly and to help develop code to

empty hexagons of the maze.

DONE)

1. Input the maze size `n` and then `n` 0's or 1's to represent the filled and

Solution Strategy
So, let's try making the path found so far be a parameter to
path found so far. This data varies with where we are in doing the work.

8. So, pathparameters (...).

the (i + I)st queen may be placed in row (i + I).

This condition has a flavor from the n-Queens Problem: After some queens
have been tentatively placed in rows 1, 2, ... i, those queens restrict where
hexagons twice. Otherwise, there might be an infinite number of solutions.

Each solution path must be simple, which means it must not use the same
a hexagon outside the $n$ by $n$ maze.

or path found so far, nor does it use a

Each such continuation path does not use any hexagon in the

tar and continue to the goal hexagon $h(n-1,n-1)$ are printed.

path found so far.

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

most currently found hexagon in the path found so far. This

precondition: print all paths terminating with the given path so

9. Direct algorithm pseudo-code for
{ } else { done
{return; }
*/
for (loop finished)
{
printf( "C:\\nNext = this hexagon:\nNext->data = this hexagon;"

) ( NOT in the path so far
AND is in the maze, AND is empty,
according to Rule 4)
if( the hexagon adjacent to plant->data
) } }
for (b=0; b<6; b++)
next->link = plant;
pathlistnode *next = new pathlistnode;
int q; // AUTOMATIC VARIABLE!
else
{ print the path found so far; return; }
( if( plant->data is the goal hexagon
)
10. else 
printf( "C:\\nNext = this hexagon:"

) }
can access the (private) the 2-dim array and the size variable.

...empty hexagon. These must be member functions of class Maze so they

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 hexagon) is in a linked list. (B) Test if
deragon helper methods to help implement its operations: For example, (A)
design the pseudo-code in the outline of the printSolution() method to

Analyze the pseudo-code in the outline of the printSolution() method to

found so far.

pathListnode to implement the linked list of hexagons that is the path

The Maze class uses some kind of linked list node structure or class type

include printSolution()

they may be accessed by any methods that need them. Such methods
so

The 2-dim Maze array and size variable are private data members so

reading it, printing it, printing paths, and printing all the solution paths.

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

Design Ideas

27
4. Write a skeleton implementation of Maze.cc (which must include Maze.h).

You have in your mind:

and postconditions are consistent with each other and with the intentions

3. Check that the function declarations, data member declarations and pre-

and postconditions to do.

2. Start writing the file Maze.h, and try to write pre- and postconditions to

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.

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

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

Implementation and Test