Outline

Last time:

- Tree Terminology
- Tree Traversal
- $N$-ary Tree

Today:

- Binary Search Tree (BST)
- Binary Space Partition (BSP) Tree
- Heap and Priority Queue
Binary Search Tree (BST)

What is the difference between a binary tree and a binary search tree?

Definition of BST:

- all values in $T_l$ is $<$ value on root
- all values in $T_r$ is $>$ value on root
- where ‘$<$’ and ‘$>$’ can be user defined

Search algorithm, search for key starting from $T.root$:

- if $T.root$ is null, not found
- if key == $T.root.value$ found
- if key > $T.root.value$ search $T.T_r$
- if key < $T.root.value$ search $T.T_l$
BST Search Time Complexity

Average-case search time:

<table>
<thead>
<tr>
<th></th>
<th>successful</th>
<th>unsuccessful</th>
</tr>
</thead>
<tbody>
<tr>
<td>linked list:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hash table:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BST:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Expressed in terms of depth (or height),
successful search on BST takes $O( )$
unsuccessful search takes $O( )$

Worst-case successful search time on BST: $O( )$
Worst-case unsuccessful search time on BST: $O( )$
BST Insertion

Insertion of new element:

- starting from root, find “appropriate” place for new
- insert new
- example: build a BST consisting of k, b, m, f, v, z, o, p, a
- will the “appropriate” place always be a leaf node?

Given a BST, where are the smallest and largest elements?

How do you remove an element from a BST?
Remove z, b, m from the above BST
BST Removal

After removal of a node, tree must remain a BST

Removal of an element \texttt{elt}:

- search for \texttt{elt}
- if \texttt{elt} is a leaf node, just remove it
- what if \texttt{elt} is non-leaf?

  o swap with either the smallest element of \(T_r\)
    or the largest element of \(T_l\)

  o if \texttt{elt} is now leaf, remove it

  o if non-leaf, it should have only one child,
    replace parent’s pointer to this node with
    pointer to child and remove the node
Problem: Collision Detection

Blue wants to go to object $e$. Which objects are in the way?
Binary Space Partitioning (BSP) Tree

What is a BSP?

A spatial data structure used to help organize geometry in $n$-dimensional space

Roughly: a BST that sorts objects in space by their coordinates

Used to speed up queries about whether geometric objects overlap or about the spatial ordering of geometric objects

Example uses:

- computer graphics: whether an object is behind another, to determine display
- robot motion planning: whether there is obstruction in a path
- computer games: collision detection
- in general: occlusion culling and collision detection

Sugih Jamin (jamin@eecs.umich.edu)
BSP Partitions Space

Blue wants to go to object \( e \). Which objects are in the way?
Creating a BSP Tree

Algorithm:

- use a plane to divide space in 2
- objects to the left of plane are put in $T_l$
- objects to the right of plane are put in $T_r$
- repeat for $T_l$ and $T_r$

Types of BSP:

- polygon-aligned: pick existing objects (polygons) in space as dividing planes
- axis-aligned: dividing planes placed along $x$ and $y$ axes
BSP Creation

Sugih Jamin (jamin@eecs.umich.edu)
BSP Found Colliding Object

Blue wants to go to object e. Which objects are in the way?

Next check for collision against items ’c’ and ’m’

Sugih Jamin (jamin@eecs.umich.edu)
BSP and Occlusion Culling

What can Blue see?
MinHeap

MinHeap: an *ordered* tree with the following properties:

- every subtree of $T$ is a heap
- the key in the root of $T$ is $\leq$ the key in every subtree of $T$

Binary heap: must be a **complete** binary tree (see Preiss Fig. 11.4, note that it is **not** a representation of the ternary tree in Fig. 11.3)

A complete binary tree can be efficiently stored as an array
Binary MinHeap

findmin(): time complexity $O( )$

enqueue(): must maintain binary heap properties (Fig. 7.6 GTM)
  • insert new item as the rightmost leaf of the complete tree
  • percolate up: swap new with parent as long as (new’s key < parent’s key)

time complexity: $O( )$

dequuememin(): (Fig. 7.7 GTM)
  • remove root
  • move rightmost item (elt) of lowest level to root
  • percolate down: swap elt with children with the smallest key until elt’s key is smaller than children’s or elt is a leaf node.
  • how to find the rightmost item?

time complexity: $O( )$
Priority Queue

Priority Queue: a list of items with priorities

Examples:

- shortest job first print or cpu queue
- simulation of events (in games, for example)
- scheduling in general

Only need the following operations:

- enqueue()
- findmin() (or findmax())
- dequeueemin() (or dequeueemax())

Implementation: as a sorted linked list, enqueue() takes $O(\ )$

What data structure would be good to implement a priority queue?
Discrete Event Simulations

Most common operations in discrete event simulations (for example, video games):

- dequeue() \( \mathcal{O}(\log N) \) for all kinds of heap
- enqueue() \( \Theta(\log N) \) for all kinds of heap except FibHeap \( \Theta(1) \)
- “hold”, dequeue() followed immediately by enqueue() \( \mathcal{O}(\log N) \) for all kinds of heap

Sugih Jamin (jamin@eecs.umich.edu)