Seth Cram

CS121

Short Answer:

Describe all steps necessary to **remove** an arbitrary item from a **BST**.

Traverse through the tree until you find the item to delete, then

If no child- delete node.

If 1 child- move child to parent and delete the old parent node.

If 2 children- move leftmost child of the right subtree to the parent and delete old parent

node.

What are the differences between iteration and recursion?

Iteration is a loop, so it will continue doing the same thing until a specified condition is

Met.

Recursion is a function that calls itself, and will keep recursing because of the general

case until a specified base case that isn’t defined recursively is met.

Why are optimal trees useful for storing information? How do optimal trees store information?

All keys are equally likely so the frequency of access is equal for all nodes. So, they

provide the smallest search time. They are also used to tell if a token is a keyword or not

by some compilers. Whatever keywords show up more often, aka have higher usage, are placed closer to the root for quicker access.

Generally speaking, when are AVL trees better than BST’s?

AVL trees are better than BST’s when the BST is unbalanced because AVL trees are

always balanced. Being balanced, there’s at most only one level of height

difference. So search time is cut down because an AVL tree would be better

organized and have more nodes closer to the root.

When should a stack or queue be implemented using an array, as opposed to a linked list structure?

Only when you know the maximum size of the stack or queue because then you’d be

able to minimize the use of dynamic memory manipulation by using a static array.

Describe the steps needed to remove an item from a **min-Heap**.

Traverse through the tree until you find the item to delete, then

If it’s at the top entry- move last item into root, move old root down swapping with

the lowest priority child until the old root is a leaf, and then delete it.

If it has one child- move child to parent and delete old parent item.

If it has 2 children- move item down swapping with the

lowest priority child until the item is a leaf, and then delete it.

If it has no children- delete item.

Big O:

Unsorted array = O(n)

Sorted array = O(log(n))

AVL tree = O(log(n))

BST = O(log(n))

Singly linked list = O(n)

Hash Table = O(1)

Ops for inserting/deleting in heap = O(log(n))

Balanced tree = O(log(n))

Iteration is a loop, so it will continue doing the same operation until a specified condition is met.

Recursion is a function that calls itself (self-referential), and will keep recursing because of the general case until a specified base case that isn’t defined recursively is met. Recursion is

more likely to overflow since it has to recurse back on itself.