ASSORTED MINUTIAE

• Exam review
  • Wednesday evening (Canvas announcement)

• Regrade requests for HW2 by end of day Monday
TODAY’S LECTURE

• Finish AVL Trees
  • Proof
• Memory analysis
  • Framework and concept
REVIEW

• AVL Trees
  • BST trees with AVL property
  • $\text{Abs}(\text{height(left)} - \text{height(right)}) \leq 1$
  • Heights of subtrees can differ by at most one
  • This property must be preserved throughout the tree
• Add the following into an AVL Tree
  • {1,2,3,5,4}
• Add 2, then verify balance
• Add three, observe that the balance of ‘1’ is off.
  • What case is this?
• Add three, observe that the balance of ‘1’ is off.
  • What case is this? Right-right
• Rotate the tree to preserve balance
• Rotate the tree to preserve balance
  • What is the new root?
• Rotate the tree to preserve balance
  • What is the new root? 2
• Perform the ‘left’ rotation which brings two into the root position
• Add the 5
• Add the 5
  • Verify balance
• Add the 4
• Add the 4
  • Verify balance
• Add the 4
  • Verify balance. Which node(s) are out of balance?
• Add the 4
  • Verify balance. Which node(s) are out of balance?
• Which rotation will fix the tree?
• Which rotation will fix the tree?
  • Select the lowest out-of-balance node
• Which rotation will fix the tree?
  • Select the lowest out-of-balance node (right-left case)
What does the final tree look like?
• The grandchild (4) moves up to the unbalanced position
• The grandchild (4) moves up to the unbalanced position
  • Observe the tree is balanced
• Work among yourselves, create an AVL tree from the input sequence
On your own or in small groups, produce the AVL tree from the following sequence of inputs.

\{10,20,15,5,0,-5\}
On your own or in small groups, produce the AVL tree from the following sequence of inputs.
{10, 20, 15, 5, 0, -5}

Once you’ve finished this, think about why this balance condition is enough to give us a tree height in $O(\log n)$
REVIEW

\{10, 20, 15, 5, 0, -5\}
REVIEW

\{10, 20, 15, 5, 0, -5\}
{10, 20, 15, 5, 0, -5}
REVIEW

{10, 20, 15, 5, 0, -5}
AVL HEIGHT

- Do we get $O(\log n)$ height from this balance?
AVL HEIGHT

• Do we get $O(\log n)$ height from this balance?
  • We can get somewhat unbalanced trees
AVL HEIGHT

• Do we get $O(\log n)$ height from this balance?
  • We can get somewhat unbalanced trees
  • Are the balanced enough?
AVL HEIGHT (PROOF)

• You do not need to memorize this proof, but it is interesting to think about
AVL HEIGHT (PROOF)

• You do not need to memorize this proof, but it is interesting to think about
  • Let’s consider the most “unbalanced” AVL tree, that is: the tree for each height that has the fewest nodes
AVL HEIGHT (PROOF)

- For height 1, there is only one possible tree.
AVL HEIGHT (PROOF)

- For height 1, there is only one possible tree.

- For height 2, there are two possible trees, each with two nodes.
AVL HEIGHT (PROOF)

• For height 1, there is only one possible tree.

• For height 2, there are two possible trees, each with two nodes.
AVL HEIGHT (PROOF)

- What about for height three? What tree has the fewest number of nodes?
AVL HEIGHT (PROOF)

- What about for height three? What tree has the fewest number of nodes?
  - *Hint: balance will probably not be zero*
AVL HEIGHT (PROOF)

• What about for height three? What tree has the fewest number of nodes?
  • Hint: balance will probably not be zero
AVL HEIGHT (PROOF)

• What about for height three? What tree has the fewest number of nodes?
  • *Hint: balance will probably not be zero*

There are multiple of these trees, but what’s special about it?
AVL HEIGHT (PROOF)

• The smallest tree of size three is a node where one child is the smallest tree of size one and the other one is the smallest tree of size two.
AVL HEIGHT (PROOF)

• In general then, if $N_1 = 1$ and $N_2 = 2$ and $N_3 = 4$, what is $N_k$?
AVL HEIGHT (PROOF)

• In general then, if \( N_1 = 1 \) and \( N_2 = 2 \) and \( N_3 = 4 \), what is \( N_k \)?
  
  • Powers of two seems intuitive, but this is a good case of why 3 doesn’t always make the pattern.
AVL HEIGHT (PROOF)

• In general then, if \( N_1 = 1 \) and \( N_2 = 2 \) and \( N_3 = 4 \), what is \( N_k \)?
  • Powers of two seems intuitive, but this is a good case of why 3 doesn’t always make the pattern.
  • \( N_4 = 7 \), how do I know?
AVL HEIGHT (PROOF)

• In general then, if \( N_1 = 1 \) and \( N_2 = 2 \) and \( N_3 = 4 \), what is \( N_k \)?

• \( N_k = 1 + N_{k-1} + N_{k-2} \)
  Because the smallest AVL tree is a node (1) with a child that is the smallest AVL tree of height \( k-1 \) (\( N_{k-1} \)) and the other child is the smallest AVL tree of height \( k-2 \) (\( N_{k-2} \)).
AVL HEIGHT (PROOF)

- In general then, if $N_1 = 1$ and $N_2 = 2$ and $N_3 = 4$, what is $N_k$?
  - $N_k = 1 + N_{k-1} + N_{k-2}$
    Because the smallest AVL tree is a node (1) with a child that is the smallest AVL tree of height $k-1$ ($N_{k-1}$) and the other child is the smallest AVL tree of height $k-2$ ($N_{k-2}$).
  - This means every non-leaf has balance 1
AVL HEIGHT (PROOF)

• In general then, if \(N_1 = 1\) and \(N_2 = 2\) and \(N_3 = 4\), what is \(N_k\)?
  
  • \(N_k = 1 + N_{k-1} + N_{k-2}\)

Because the smallest AVL tree is a node (1) with a child that is the smallest AVL tree of height \(k-1\) \((N_{k-1})\) and the other child is the smallest AVL tree of height \(k-2\) \((N_{k-2})\).

• This means every non-leaf has balance 1

• Nothing in the tree is perfectly balanced.
AVL HEIGHT (PROOF)

\[ N_k = 1 + N_{k-1} + N_{k-2} \]
\[ N_{k-1} = 1 + N_{k-2} + N_{k-3} \]
AVL HEIGHT (PROOF)

\[ N_k = 1 + N_{k-1} + N_{k-2} \]
\[ N_{k-1} = 1 + N_{k-2} + N_{k-3} \]
AVL HEIGHT (PROOF)

Substitute the k-1 into the original equation

\[ N_k = 1 + N_{k-1} + N_{k-2} \]
\[ N_{k-1} = 1 + N_{k-2} + N_{k-3} \]
**AVL HEIGHT (PROOF)**

1 + \( N_{k-3} \) must be greater than zero

\[
N_k = 1 + N_{k-1} + N_{k-2}
\]

\[
N_{k-1} = 1 + N_{k-2} + N_{k-3}
\]

\[
N_k = 1 + (1 + N_{k-2} + N_{k-3}) + N_{k-2}
\]

\[
N_k = 1 + 2N_{k-2} + N_{k-3}
\]

\[
N_k > 2N_{k-2}
\]
AVL Height (Proof)

1 + N_{k-3} must be greater than zero

N_k = 1 + N_{k-1} + N_{k-2}
N_{k-1} = 1 + N_{k-2} + N_{k-3}
N_k = 1 + (1 + N_{k-2} + N_{k-3}) + N_{k-2}
N_k = 1 + 2N_{k-2} + N_{k-3}
N_k > 2N_{k-2}

This means the tree doubles in size after every two height (compared to a perfect tree which doubles with every added height).
AVL CONCLUSION

• If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
AVL CONCLUSION

• If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
  • Insert(key $k$, value $v$)
  • Find(key $k$)
AVL CONCLUSION

• If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
  • Insert(key k, value v)
  • Find(key k)
  • Delete(key k): *not covered in this class*
AVL CONCLUSION

• If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
  • Insert(key k, value v)
  • Find(key k) : $O(\text{height}) = O(\log n)$
  • Delete(key k): *not covered in this class*
AVL CONCLUSION

• If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
  • $\text{Insert}(\text{key } k, \text{ value } v) = O(\log n) + \text{balancing}$
  • $\text{Find}(\text{key } k) : O(\text{height}) = O(\log n)$
  • $\text{Delete}(\text{key } k)$: not covered in this class
AVL CONCLUSION

• If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
  • $\text{Insert(key } k, \text{ value } v) = O(\log n) + \text{ balancing}$
  • $\text{Find(key } k) : O(\text{height}) = O(\log n)$
  • $\text{Delete(key } k): \text{ not covered in this class}$

• How long does it take to perform a balance?
AVL CONCLUSION

- If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
  - Insert(key $k$, value $v$) = $O(\log n)$ + balancing
  - Find(key $k$) : $O(\text{height}) = O(\log n)$
  - Delete(key $k$): not covered in this class
- How long does it take to perform a balance?
  - There are at most three nodes and four subtrees to move around.
AVL CONCLUSION

• If AVL rotation can enforce $O(\log n)$ height, what are the asymptotic runtimes for our functions?
  • Insert(key k, value v) = $O(\log n)$ + balancing
  • Find(key k) : $O(\text{height}) = O(\log n)$
  • Delete(key k): *not covered in this class*

• How long does it take to perform a balance?
  • There are at most three nodes and four subtrees to move around. $O(1)$
AVL CONCLUSION

- By using AVL rotations, we can keep the tree balanced
AVL CONCLUSION

• By using AVL rotations, we can keep the tree balanced
• An AVL tree has $O(\log n)$ height
AVL CONCLUSION

• By using AVL rotations, we can keep the tree balanced
• An AVL tree has $O(\log n)$ height
• This does not come at an increased asymptotic runtime for insert.
AVL CONCLUSION

• By using AVL rotations, we can keep the tree balanced
• An AVL tree has $O(\log n)$ height
• This does not come at an increased asymptotic runtime for insert.
• Rotations take a constant time.
MEMORY ANALYSIS

• Similar to runtime analysis
MEMORY ANALYSIS

• Similar to runtime analysis
  • Consider the worst case
MEMORY ANALYSIS

• Similar to runtime analysis
  • Rather than counting the number of operations, we count the amount of memory needed
MEMORY ANALYSIS

• Similar to runtime analysis
  • Rather than counting the number of operations, we count the amount of memory needed
  • During the operation, when does the algorithm need to “keep track” of the most number of things?
MEMORY ANALYSIS

- Breadth first search
MEMORY ANALYSIS

- Breadth first search
  - The Queue keeps track of the elements that need to be analyzed next.
MEMORY ANALYSIS

• Breadth first search
  • The Queue keeps track of the elements that need to be analyzed next.
  • This is the memory we need to consider
MEMORY ANALYSIS

• Breadth first search
  • The Queue keeps track of the elements that need to be analyzed next.
  • This is the memory we need to consider.
  • At what point does the Queue have the most amount stored in it?
MEMORY ANALYSIS

• Breadth first search
  • The Queue keeps track of the elements that need to be analyzed next.
  • This is the memory we need to consider
  • At what point does the Queue have the *most* amount stored in it?
  • When the tree is at its widest – how many nodes is that?
MEMORY ANALYSIS

• Breadth first search
  • The Queue keeps track of the elements that need to be analyzed next.
  • This is the memory we need to consider
  • At what point does the Queue have the *most* amount stored in it?
  • When the tree is at its widest – how many nodes is that?
  • **N/2**: half the nodes of a tree are leaves
MEMORY ANALYSIS

• Consider finding an element in a sorted linked list
MEMORY ANALYSIS

• Consider finding an element in a sorted linked list
  • How much memory does this take?
MEMORY ANALYSIS

• Consider finding an element in a sorted linked list
  • How much memory does this take?
  • Don’t count the data structure, only count the amount of memory that the actual algorithm uses.
MEMORY ANALYSIS

• Consider finding an element in a sorted linked list
  • How much memory does this take?
  • Don’t count the data structure, only count the amount of memory that the actual algorithm uses.
  • What does it need to “keep track” of?
MEMORY ANALYSIS

• Consider finding an element in a sorted linked list
  • How much memory does this take?
  • Don’t count the data structure, only count the amount of memory that the actual algorithm uses.
  • What does it need to “keep track” of?
  • *Just the think we’re looking for!*
MEMORY ANALYSIS

• Consider finding an element in a sorted linked list
  • How much memory does this take?
  • Don’t count the data structure, only count the amount of memory that the actual algorithm uses.
  • What does it need to “keep track” of?
  • *Just the think we’re looking for!* $O(1)$
MEMORY ANALYSIS

• We care about the asymptotic memory usage
MEMORY ANALYSIS

• We care about the asymptotic memory usage

• That is, as the input size of the data structures increases, does the amount of extra memory increase?
MEMORY ANALYSIS

• We care about the asymptotic memory usage
• That is, as the input size of the data structures increases, does the amount of extra memory increase?
  • AVL Insert?
MEMORY ANALYSIS

• We care about the asymptotic memory usage

• That is, as the input size of the data structures increases, does the amount of extra memory increase?
  • AVL Insert? No, we only need to keep track of the parent and grandparent.
MEMORY ANALYSIS

• We care about the asymptotic memory usage

• That is, as the input size of the data structures increases, does the amount of extra memory increase?
  • AVL Insert? No, we only need to keep track of the parent and grandparent.
  • DFS?
MEMORY ANALYSIS

• We care about the asymptotic memory usage

• That is, as the input size of the data structures increases, does the amount of extra memory increase?
  • AVL Insert? No, we only need to keep track of the parent and grandparent.
  • DFS? Yes, we need to keep track of all the elements leading back up to the root
NEXT WEEK

• Hashtables
  • The O(1) holy grail!
NEXT WEEK

• Hashtables
  • The O(1) holy grail!
• Exam review on Wednesday
NEXT WEEK

• Hashtables
  • The O(1) holy grail!
• Exam review on Wednesday
• Exam on Friday!