Lecture 11/02/16

Lecturer: Xiaodi Wu

Reading: Chapter 3.2
Recap

▶ We have defined Binary Search Tree (BST).
Binary Search Tree

Recap

- We have defined Binary Search Tree (BST).
- Insertion and Removal operation on the BST. (Required in Lab 2)
Recap

- We have defined Binary Search Tree (BST).
- Insertion and Removal operation on the BST. (Required in Lab 2)
- Complexity of each operation is $O(h)$. 
Binary Search Trees
Problems with BST

Complexity $O(h)$

$h$ could be $O(\log(n))$ or $O(n)$. The worst case complexity is $O(n)$. 

Candidate solution: AVL tree.
Problems with BST

Complexity $O(h)$
$h$ could be $O(\log(n))$ or $O(n)$. The worst case complexity is $O(n)$.

*Is there a way to force $h$ to be $O(\log(n))$ while still having efficient insertion and removal operation?*
Problems with BST

Complexity $O(h)$

$h$ could be $O(\log(n))$ or $O(n)$. The worst case complexity is $O(n)$.

Is there a way to force $h$ to be $O(\log(n))$ while still having efficient insertion and removal operation?

Candidate solution: AVL tree.
**AVL tree**

**Height-balance Property**

For every internal node $v$ of $T$, the heights of the children of $v$ can differ by at most 1.
Height-balance Property
For every internal node \( v \) of \( T \), the heights of the children of \( v \) can differ by at most 1.

AVL Tree
Any binary search tree with the **height-balance property** is called an AVL tree, named after the initials of the inventors.
AVL tree

Height-balance Property
For every internal node $v$ of $T$, the heights of the children of $v$ can differ by at most 1.

AVL Tree
Any binary search tree with the **height-balance property** is called an AVL tree, named after the initials of the inventors.
Remember the goal is to hope $h = O(\log(n))$. 
AVL trees
AVL tree: $h = O(\log(n))$

Height-balance Property

- Why not force the same height of the children?
AVL tree: $h = O(\log(n))$

Height-balance Property

- Why not force the same height of the children?
- How about allow the difference of heights to be 2?
AVL tree: $h = O(\log(n))$

Height-balance Property

- Why not force the same height of the children?
- How about allow the difference of heights to be 2? in the assignment 3.
AVL tree: \( h = O(\log(n)) \)

Height-balance Property

- Why not force the same height of the children?
- How about allow the difference of heights to be 2? in the assignment 3.

The height \( h \) of an AVL tree of \( n \) nodes

- Basically, find the relationship between \( h \) and \( n \).
AVL tree: $h = O(\log(n))$

Height-balance Property

- Why not force the same height of the children?
- How about allow the difference of heights to be 2? in the assignment 3.

The height $h$ of an AVL tree of $n$ nodes

- Basically, find the relationship between $h$ and $n$.
- Let $n(h)$ be the minimum \# nodes in a tree of height $h$. 
AVL tree: \( h = O(\log(n)) \)

Height-balance Property

- Why not force the same height of the children?
- How about allow the difference of heights to be 2? in the assignment 3.

The height \( h \) of an AVL tree of \( n \) nodes

- Basically, find the relationship between \( h \) and \( n \).
- Let \( n(h) \) be the minimum \# nodes in a tree of height \( h \).
- It suffices to show that \( n(h) = 2^{\Omega(h)} \).
AVL tree: \( h = O(\log(n)) \)

Height-balance Property

- Why not force the same height of the children?
- How about allow the difference of heights to be 2? in the assignment 3.

The height \( h \) of an AVL tree of \( n \) nodes

- Basically, find the relationship between \( h \) and \( n \).
- Let \( n(h) \) be the minimum \( \# \) nodes in a tree of height \( h \).
- It suffices to show that \( n(h) = 2^{\Omega(h)} \).
- \( n \geq 2^{ch} \Rightarrow h \leq \frac{1}{c} \log(n) \in O(\log(n)) \).
Proof:  \( h = O(\log(n)) \)

Theorem

*The height of an AVL tree storing \( n \) items is \( O(\log(n)) \).*

Proof.

- \( n(1) = 1, \ n(2) = 2. \)
Proof: $h = O(\log(n))$

Theorem

*The height of an AVL tree storing $n$ items is $O(\log(n))$.*

Proof.

- $n(1) = 1$, $n(2) = 2$.
- How about $n(3)$? What is the worst case?
Proof: $h = O(\log(n))$

Theorem

*The height of an AVL tree storing $n$ items is $O(\log(n))$.*

Proof.

- $n(1) = 1$, $n(2) = 2$.
- How about $n(3)$? What is the worst case?
- $n(3) = 1 + n(1) + n(2)!$
Theorem

The height of an AVL tree storing $n$ items is $O(\log(n))$.

Proof.

- $n(1) = 1$, $n(2) = 2$.
- How about $n(3)$? What is the worst case?
- $n(3) = 1 + n(1) + n(2)!$
- In general,

$$n(h) = 1 + n(h - 1) + n(h - 2).$$
Proof: $h = O(\log(n))$

Theorem

*The height of an AVL tree storing $n$ items is $O(\log(n))$.*

Proof.

- $n(1) = 1$, $n(2) = 2$.
- How about $n(3)$? What is the worst case?
- $n(3) = 1 + n(1) + n(2)!$
- In general,

\[
  n(h) = 1 + n(h-1) + n(h-2).
\]

- $n(h)$ is a strictly increasing function of $h$. Thus

\[
  n(h) > 2 \times n(h-2).
\]
Proof: $h = O(\log(n))$, cont’d

- In general, for any $i$ such that $h - 2i \geq 1$, we have

\[ n(h) > 2^i \times n(h - 2i). \]
Proof: $h = O(\log(n))$, cont’d

- In general, for any $i$ such that $h - 2i \geq 1$, we have
  \[ n(h) > 2^i \times n(h - 2i). \]

- One can choose $i = \lceil h/2 \rceil - 1$. Thus $n(h - 2i)$ could be $n(1)$ or $n(2)$. We have,
  \[ n(h) > 2^{\lceil h/2 \rceil - 1} n(1) \in 2^{\Omega(h)}. \]
Proof: $h = O(\log(n))$, cont’d

- In general, for any $i$ such that $h - 2i \geq 1$, we have

  $$n(h) > 2^i \times n(h - 2i).$$

- One can choose $i = \lceil h/2 \rceil - 1$. Thus $n(h - 2i)$ could be $n(1)$ or $n(2)$. We have,

  $$n(h) > 2^{\lceil h/2 \rceil - 1} n(1) \in 2^{\Omega(h)}.$$

- Precisely, we could have

  $$h < 2 \log(n) + 2.$$
AVL tree: Update

Updates like BST
Except, we might break the height-balance property. Need additional effort to re-balance it!
AVL tree: Update

Updates like BST

Except, we might break the height-balance property. Need additional effort to re-balance it!
AVL tree: Update

Updates like BST
Except, we might break the height-balance property. Need additional effort to re-balance it!