Introduction to Computer Programming

LECTURE 8
Randomly built binary search trees
• Expected node depth
• AVL Tree
  • Insert
  • Delete
Binary Search Tree

A binary tree
Searching
Insertion
Deletion
Insertion

TREE-INSERT(T, z)

1. y ← NIL
2. x ← root[T]
3. while x != NIL
   4. do y ← x
      5. if key[z] < key[x]
         6. then x ← left[x]
         7. else x ← right[x]
   8. p[z] ← y
9. if y = NIL
   10. then root[T] ← z               → Tree T was empty
   11. else if key[z] < key[y]
       12. then left[y] ← z
   13. else right[y] ← z
Deletion

TREE-DELETION(T,z)
1. if left[z] = NIL or right[z] = NIL
2. then y ← z
3. else y ← TREE-SUCCESSOR(z)
4. if left[y] != NIL
5. then x ← left[y]
6. else y ← right[y]
7. if x != NIL
8. then p[x] ← p[y]
9. if p[y] = NULL
10. then root[T] ← x
11. else if y = left[p[y]]
12. then left[p[y]] ← x
13. else right[p[y]] ← x
14. if y != z
15. then key[z] ← key[y]
16. return y
Three Cases
-Delete 13
Three Cases
-Delete 13
Three Cases

- Delete 13
Three Cases
-Delete 16
Three Cases
-Delete 16
Three Cases
-Delete 16
Three Cases
-Delete 6
Three Cases

-Delete 6
Three Cases
-Delete 6
Binary-search-tree sort

\( T \leftarrow \emptyset \quad \triangleright \text{Create an empty BST} \\
\text{for } i = 1 \text{ to } n \\
\quad \text{do TREE-INSERT}(T, A[i]) \\
\text{Perform an inorder tree walk of } T. \\

\text{Example:} \\
A = [3 1 8 2 6 7 5] \\

Tree-walk time = \( O(n) \), but how long does it take to build the BST?
Analysis of BST sort

BST sort performs the same comparisons as quicksort, but in a different order!

The expected time to build the tree is asymptotically the same as the running time of quicksort.
Node depth

The depth of a node = the number of comparisons made during TREE-INSERT. Assuming all input permutations are equally likely, we have

Average node depth

\[
\frac{1}{n} E \left[ \sum_{i=1}^{n} (\text{# comparisons to insert node } i) \right]
\]

\[
= \frac{1}{n} O(n \lg n)
\]

= \( \mathcal{O}(\lg n) \) (quicksort analysis)
Expected tree height

But, average node depth of a randomly built BST = $O(lg \, n)$ does not necessarily mean that its expected height is also $O(lg \, n)$ (although it is).

Example.

\[
\text{Ave. depth} \leq \frac{1}{n} \left( n \cdot lg \, n + \frac{\sqrt{n} \cdot \sqrt{n}}{2} \right) = O(lg \, n)
\]
AVL Tree

• AVL(Adelson-Velsky and Landis) tree is a height balanced tree.

• It has good properties to make searching completed in $O(\log n)$ time.
Insertion into a AVL Tree
Case 1: Inserting into Short Subtree

Before:

```
    C
   /+
  B  E
 /   |
D    F
```

After:

```
    C
   /-
  B  E
 /   |
A    F
```

Insert A
Case 2-a

Insert G
Case 2-a

Perform CCW rotation !!!

Before:

- C
  - B
    - D
      - F
  - E

After:

- C
  - B
    - D
      - F
  - E
    - G
Case 2-a

Perform CCW rotation !!!
Case 2-a

1. **Insert**: Inserting a new node at the right position of node B. The tree height increases by 1.

2. **CCW Rotation**: Performing a counterclockwise rotation at node B. The tree height remains the same.

The final tree height is h+1.
Case 2-b (Vice versa case of 2-a)

Perform CW rotation !!!

Insert A
Case 2-b (Vice versa case of 2-a)

Perform CW rotation !!!
Case 3-a

Perform double CCW rotation !!!
Case 3-a

Perform double CCW rotation !!!
Case 3-a

Perform double CCW rotation !!!
Case 3-b (Symmetric case)