

12.1 Binary Search Tree

A Binary Search Tree (BST) is a ~~TST~~ ~~FOR~~ ~~WHICH~~ ~~EVER~~ ~~NODE~~ ~~x~~ ~~CONTAINS~~ ~~FIELDS~~:

key[x], left[x], right[x], p[x] (+ satellite DATA)

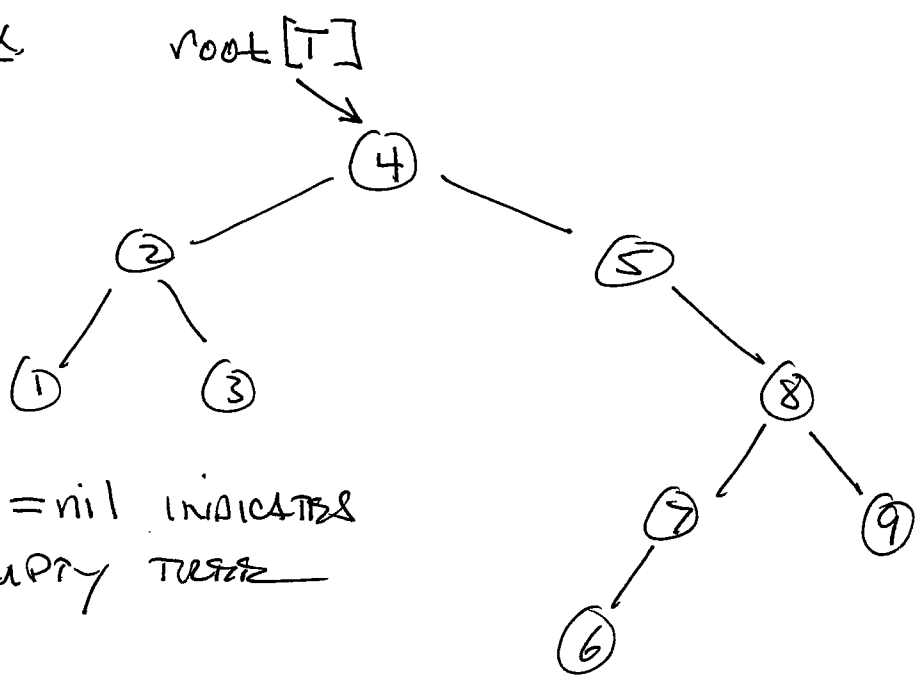
AND SATISFYING THE

Binary Search Tree Properties:

FOR ALL NODES x:

- IF y is in the LEFT SUBTREE OF x, THEN $key[y] \leq key[x]$.
- IF y is in the RIGHT SUBTREE OF x, THEN $key[x] \leq key[y]$.

EX



root[T] = nil INDICATES AN EMPTY TREE

THE BST PROPERTY MAKES IT POSSIBLE TO PRINT OUT KEYS IN ASCENDING ORDER

InOrderTreeWalk(x)

- 1.) if $x \neq \text{nil}$
- 2.) InOrderTreeWalk(left[x])
- 3.) Print key[x]
- 4.) InOrderTreeWalk(right[x])

(COULD DO OTHER PROCESSING OF x HERE)

TO PRINT FULL TREE, call:

InOrderTreeWalk(root[T])

Run Time: $\Theta(n)$ if T has n nodes. See proof on P. 255.

A PRE-ORDER TREE WALK WOULD PRINT ~~THE VALUES IN EITHER SUBTREE~~ ~~BEFORE~~ ~~IT~~ ~~PRINTS~~ ~~THE~~ ~~VALUES~~ ~~IN~~ ~~EITHER~~ ~~SUBTREE~~.
key[x] BEFORE IT PRINTS THE VALUES IN EITHER SUBTREE.

A POST-ORDER TREE WALK PRINTS ~~THE~~ key[x] AFTER IT PRINTS VALUES IN left & right subtree.

EXERCISE: WRITE PSEUDO-CODE FOR THESE OPS.

12.2 QUERIES

A BST CAN SERVE AS AN EFFICIENT STRUCTURE FOR STORING DATA. (i.e. A DATABASE).

TreeSearch(x, k)

- 1.) if $x = \text{nil}$ or $k \neq \text{key}[x]$
- 2.) return x
- 3.) if $k < \text{key}[x]$
- 4.) return $\text{TreeSearch}(\text{left}[x], k)$
- 5.) else
- 6.) return $\text{TreeSearch}(\text{right}[x], k)$

TO SEARCH FOR A NODE x WITH KEY k WE CALL

$\text{TreeSearch}(\text{root}[T], k)$.

IF SUCH A NODE EXISTS (A POINTER TO) THAT NODE IS RETURNED, OTHERWISE nil IS RETURNED.

NOTE: IN MANY APPLICATIONS (E.G. DATABASE) KEYS WILL BE DISTINCT. IF T CONTAINS MULTIPLE KEYS, A SOME NODE WITH MATCHING KEY IS RETURNED.

THE NODES ENCOUNTERED in TreeSearch
Form a DEGRADED PATH STARTING AT
THE ROOT, \therefore RUN TIME is $\Theta(h)$ (Worst case)
where $h = \text{height}(T)$.

SEE ITERATIVE VERSION ON P. 257.

TreeMinimum(x)

- 1.) while left[x] \neq nil
- 2.) $x \leftarrow \text{left}[x]$
- 3.) Return x

TreeMaximum(x)

- 1.) while right[x] \neq nil
- 2.) $x \leftarrow \text{right}[x]$
- 3.) Return x

CONCRETENESS OF BOST ALGORITHMS FOLLOWS
FROM BST PROPERTIES. ~~the worst~~
CASE RUN TIME: $\Theta(h)$
where $h = \text{height}(T)$.

THE SUCCESSOR OF A NODE x IS
THE NEXT NODE TO BE PROCESSED AFTER
x IN AN IN ORDER TREE WALK.

