Data Structures and Algorithms

István András Seres @Eötvös Loránd University

2019 April 16, Tuesday 12:30-14:00, D-0.221

1 Bináris fák levelei

Mutassuk meg, hogy egy bináris fa levél csúcsainak száma eggyel nagyobb, mint azoké, amelyeknek két gyerekük van!

2 Binary search trees

Draw a binary search tree by inserting the following numbers from left to right:

$$11, 6, 8, 19, 4, 10, 5, 17, 43, 49, 31$$

3 Keresési sorozatok

Egy bináris keresőfa kulcsai az 1 és 1000 közötti egész számok közül valók. A fában a 363 kulcsot akarjuk megkeresni. Az alábbi sorozatok közül melyek nem lehetnek keresési sorozatok?

(A) 2, 252, 401, 398, 330, 344, 397, 363.

(B) 924, 220, 911, 244, 898, 258, 362, 363.

(C) 925, 202, 911, 240, 912, 245, 363.

(D) 2, 399, 387, 219, 266, 382, 381, 278, 363.

(E) 935, 278, 347, 621, 299, 392, 358, 363.

4 Előző és következő elem bináris keresőfákban

(A) Mutassuk meg, hogy ha egy bináris keresőfa valamely csúcsának van bal oldali gyereke, akkor a megelőzőjének nincs jobb oldali gyereke!

(B) Mutassuk meg, hogy ha egy bináris keresőfa valamely csúcsának van jobb oldali gyereke, akkor a rákövetkezőjének nincs bal oldali gyereke!

(C) Mutassuk meg, hogy ha egy bináris keresőfa valamely csúcsának van bal oldali gyereke, akkor a megelőzője a csúcs bal oldali részfájának a maximális eleme!

(D) Mutassuk meg, hogy ha egy bináris keresőfa valamely csúcsának van jobb oldali gyereke, akkor a rákövetkezője a csúcs jobb oldali részfájának a minimális eleme!

(E) Mutassuk meg, hogy ha egy bináris keresőfa valamely csúcsának nincs bal oldali gyereke, akkor a megelőzője a csúcs legalacsonyabban fekvő olyan őse, amelynek a csúcs jobb oldali leszármazottja!

(F) Mutassuk meg, hogy ha egy bináris keresőfa valamely csúcsának nincs jobb oldali gyereke, akkor a rákövetkezője a csúcs legalacsonyabban fekvő olyan őse, amelynek a csúcs bal oldali leszármazottja!

5 BST operations 1.

Implement a function that returns the successor of a node in a binary search tree (the BST stores integer keys). A successor of a node n is defined as the smallest key x in the BST such that x is bigger than the value of n, or *null* if that does not exist. You may assume that the BST does not contain duplicate keys. Let assume in the TreeNode class, you are given the following functions at your disposal: getValue(), getLeft(), getRight(), and getParent(). Note that getLeft(), getRight(), and getParent() return null if the node does not have a left, a right child, or is the root, respectively.

6 BST operations 2.

We wish to augment binary search trees with the *TREE-ENUMERATE*(x, a, b) operation, which outputs all the keys k s.t. $a \leq k \leq b$ in a BST rooted at x. Describe an algorithm for implementing this operation whose worst-case running time in terms of h, the height of the tree, and m, the number of keys that are output, is $\mathcal{O}(m+h)$. Prove that the running time of your algorithm is $\mathcal{O}(m+h)$.

7 LZW lossless data compression algorithm

Encode the string *banana_bandana* with LZW!