**1.** Na obrázku je uveden BVS, který v průběhu práce vyvažujeme (AVL strom). Ten nyní upravíme tak, že z něj odstraníme operací Delete uzly s klíči 5, 25, 35 v tomto pořadí. Rozhodněte, zda a jaká rotace bude během této úpravy použita:

**2.** Nakreslete AVL strom s 8 číselnými klíči tak, aby po vložení klíče s hodnotou 19 bylo nutno provést

a) LR rotaci v koření, b) LR rotaci v uzlu, který není kořenem.

|  |  |
| --- | --- |
| 3. Uvedený kód se snaží implementovat levou rotaci v AVL stromu v uzlu *node*. Vysvětlete, jak je nutno kód doplnit nebo změnit, aby mohl být skutečně použit. | **Node leftRotation( Node node ) {**  if( node == null ) return node;  Node p1 = node.right;  if( p1 == null) return node;  node.right = p1.left;  p1.left = node;  return p1;  **}** |

**4.** Docent Omylný tvrdí, že vždy, když se AVL strom vyváží některou z rotací (jednoduchou nebo dvojitou) následující po smazání uzlu, sníží se tím také hloubka celého AVL stromu. Najděte k tomuto tvrzení protipříklad.

**5.** Z neprázdného AVL stromu nejprve odstraníme klíč x operací Delete a vzápětí jej do stromu vložíme operací Insert. Porovnáváme tvar stromu před oběma operacemi a po jejich provedení a chceme vědět, jestli je shodný nebo jestli se liší nebo zda to závisí na konkrétním tvaru stromu a/nebo hodnotě klíče x. Zdůvodněte svou odpověď.

**6.** Najděte příklad vyhledávacího stromu T a dvou jeho klíčů x a y takových, že posloupnost operací (Delete(x), Delete(y)) vytvoří jiný strom než posloupnost (Delete(y), Delete(x)). Vyhledávací strom bude

a) BST b) AVL

**7.** Houbka kořene je 0. Určete, jaký je minimální možný počet uzlů v AVL stromu s hloubkou

a) 2 b) 3 c) 4

**8.** Pokusíme se využít binární vyhledávací strom jako prioritní frontu. Jaká bude asymptotická složitost operací Insert, FindMin a DeleteMin pro a) BST, b) AVL?

**9.** Algoritmus AVL-sort specifikujeme takto: Všechny dané klíče uložíme jeden po druhém do původně prázdného AVL stromu. Poté vhodnou procedurou vypíšeme všechny klíče ve stromu v rostoucím pořadí.

Jaká bude asymptotická složitost tohoto řazení?

**10.** Do prázdného splay stromu vložte postupně klíče 2, 7, 1, 4, 3, 9, 5, 6. Nakreslete strom po každém vložení.

**11.** Splay strom byl původně prázdný, pak do něj byly vloženy klíče 1, 2, 3, ..., 10 v tomto pořadí. Poté byly odstraněny klíče 1, 2, 3 v tomto pořadí. Nakreslete výsledný strom. Předpokládejte, že při mazání je použita funkce findMaxKey aplikovaná na levý podstrom kořene a že tato funkce také přesouvá nalezený maximální prvek do kořene stromu, na nějž byla aplikována.

**12.** Splay strom obsahuje N klíčů, dva z nich jsou x a y (navzájem různé). Se stromem byla provedena posloupnost operací (Find(x), Find(y), Find(x), Find(y), ..., Find(x), Find(y)), jichž bylo celkem 2N. Jaký byl průměrný počet navštívených uzlů během jedné operace Find? Určete co nejpřesnější dolní a horní odhad tohoto průměru.

**13.** Splay tree obsahuje 2*n* −1 klíčů s hodnotou 1, 2, 3, ..., 2*n* −1 a je ideálně vyvážený, to jest má hloubku *n*−1 . Po vyhledání uzlu s klíčem 1 se tento uzel stane kořenem stromu. Jakou hloubku bude mít výsledný strom? Řešte zvlášť pro sudé a liché *n*.