# AVL strom

David Chudožilov

Vycházím z na cvičení už naprogramovaného binárního vyhledávacího stromu

- Funkce a metody: `avl_tree(), empty(), remove_all(node*), insert(const T&), write(), remove_node(const T&), treeprint(), find(const T&, node*& parent)`

## AVL strom

- Modifikace binárního vyhledávácího stromu, která udržuje tzv. balanční faktor každého vrcholu velký maximálně 1
- Díky tomu je možné udržet časovou náročnost vyhledávání hodnoty ve stromě O(log n) ve všech případech, tedy nemůže nastat situace, kdy strom bude silně nevyvážený a začne se přibližovat spojovému seznamu, který má složitost vyhledáváni O(n)

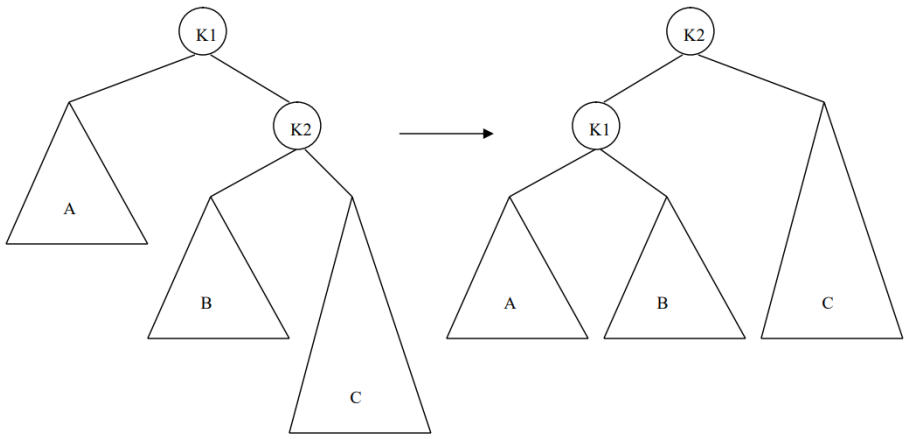
## Balanční faktor

- `int balance` - prvek třídy `avl_tree::node`
- Definován jako hloubka levého podstromu minus hloubka pravého podstromu
- Pokud jeho hodnota v absolutní hodnotě pro libovolný vrchol přesáhne 1, strom již není vybalancovaný a je třeba ho znovu vyvážit pomocí rotací

## Rotace

### LL rotace

- Odpovídající metoda `left_rotation(node* node_, node* parent)`
- Rotace se provádí na vrcholu `node_`, tedy tento vrchol je ten, který se po rotaci bude nacházet o level níž
- Argument `node* parent` vždy značí rodiče vrcholu `node_`, případně `nullptr` pokud `node_` je kořen stromu

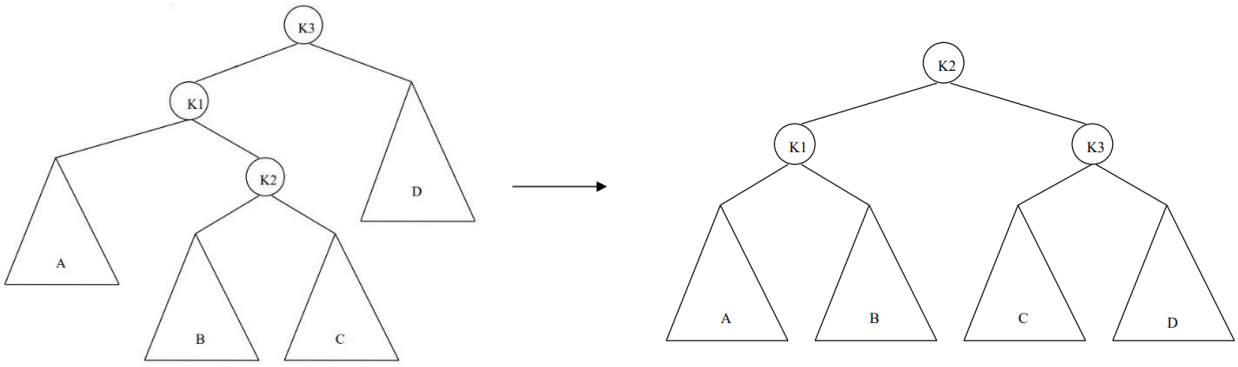



### RR rotace

- Odpovídající metoda `right_rotation(node* node_, node* parent)`
- Analogické provedení jako levá rotace

### LR rotace

- Metoda `left_right_rotation(node* node_, node* parent)`




- Na levém potomku vrcholu node_ nejprve provedena levá rotace, na vrcholu node_ následně pravá rotace

### RL rotace

- Metoda `right_left_rotation(node* node_, node* parent)`
- Na pravém potomku vrcholu node_ nejprve provedena pravá rotace, na vrcholu node_ následně levá rotace

## Algoritmus vyvažování

### Vkládání vrcholu

- Postupujeme ve stromě od kořene až k nově vloženému vrcholu a hledáme nevyvážený vrchol, který se nachází nejníže
- Jakmile tento vrchol najdeme, provedeme na něj rotaci podle toho, který podstrom je nejhlubší (tedy podle balančních faktorů daného vrcholu a jeho potomků)

a) Balanční faktor vrcholu < -1
- Pravý podstrom pravého potomka je hlubší, než levý podstrom – LL rotace
- Levý podstrom pravého potomka je hlubší, než pravý podstrom – RL rotace

b) Balanční faktor > 1
- Levý podstrom levého potomka je hlubší, než pravý podstrom – RR rotace
- Pravý podstrom levého potomka je hlubší, než levý podstrom – LR rotace

### Mazání vrcholu

- Mazaným vrcholem v případě, že má být smazán vrchol s dvěma potomky, rozumíme vrchol, s kterým se ten vrchol se dvěma potomky prohodil (ten, který má pouze jednoho nebo žádného potomka)
- Projdeme strom od kořene k rodiči mazaného vrcholu a opět nalezneme nejníže se nacházející nevyvážený vrchol – na něm provedeme rotaci podle stejného algoritmu, jako při vkládání
- Na rozdíl od vzkládání musíme po této první rotaci ještě jednou zkontrolovat a v případě nevyváženosti stejným algoritmem vyvážit i předky rodiče mazaného vrcholu

## Implementace vyvažování nevybalancovaného stromu

- Vyváženost se kontroluje a případně i provádí vyvažování po každém vložení nebo smazání vrcholu

- Funkce `balance_insert(node* end_node)` a `balance_del(node* end_node)` pro vyvažování, pokud byl do stromu nový vrchol vložen nebo byl vrchol smazán
- `node* end_node` je právě přidávaný vrchol, pokud prováděná operace je vkládání vrcholu do stromu, nebo rodič právě mazaného vrcholu, pokud se provádí mazání
- Obě funkce prochází strom od kořene po `end_node` a kontrolují balanční faktor a v případě nalezení nevyváženosti provedou rotaci na vrcholu `unbalanced_min` (nejhlubší nevyvážený vrchol), přičemž funkce `balance_del` ještě navíc provede dodatečnou kontrolu na předcích daného vrcholu nutnou při mazání vrcholu