Binární strom

Binární strom je nelineární datová struktura, ve které je každý prvek (uzel) spojen s nejvýše dvěma dalšími uzly, které označujeme jako **levý syn** a **pravý syn**.

Tato struktura se používá pro efektivní vyhledávání, třídění dat, reprezentaci hierarchií nebo při kompresi dat (např. Huffmanovo kódování).

Základní terminologie

Abychom pochopili práci se stromy, musíme definovat jejich části:

- **Kořen (Root):** Nejvyšší uzel stromu, od kterého vše začíná. Nemá žádného rodiče.
- **Uzel (Node):** Prvek stromu obsahující data a ukazatele na své potomky.
- **List (Leaf):** Uzel, který nemá žádné potomky (nachází se na úplném konci větve).
- **Větev (Edge):** Spojnice mezi dvěma uzly.
- **Hloubka (Výška):** Počet úrovní od kořene k nejvzdálenějšímu listu.

Typy binárních stromů

V praxi se setkáváme s různými variantami podle toho, jak jsou uzly uspořádány:

Typ	Charakteristika
Plný binární strom	Každý uzel má buď 0, nebo 2 potomky (nikdy jen jednoho).
Vyvážený strom	Rozdíl výšek levého a pravého podstromu každého uzlu je minimální. Zajišťuje rychlé operace.
Binární vyhledávací strom (BST)	Pro každý uzel platí: všechny hodnoty v levém podstromu jsou menší a v pravém větší než hodnota uzlu.

Binární vyhledávací strom (BST)

BST (Binary Search Tree) je nejpoužívanější formou binárního stromu díky své efektivitě při vyhledávání.

Jak funguje vyhledávání v BST:

Představte si, že hledáte číslo **15**:

```
1. Začnete u kořene (např. číslo **20**).
2. Protože 15 < 20, jdete do **levého** podstromu.
3. Narazíte na uzel s hodnotou **10**.
4. Protože 15 > 10, jdete do **pravého** podstromu.
5. Narazíte na uzel **15** – prvek nalezen.
```

Díky tomuto principu se při každém kroku eliminujeme polovinu zbývajících dat (podobně jako u binárního vyhledávání v poli).

Průchod stromem (Traversal)

Abychom mohli se všemi daty ve stromu pracovat (např. je vypsat), musíme stromem „projít“. Existují tři hlavní způsoby:

- **In-order (Levý-Kořen-Pravý):** U BST vypíše data seřazená od nejmenšího po největší.
- **Pre-order (Kořen-Levý-Pravý):** Vhodné pro kopírování stromu.
- **Post-order (Levý-Pravý-Kořen):** Používá se například při mazání stromu (nejdříve se mažou potomci, pak rodič).

Výhody a nevýhody

Výhody:

- **Rychlost:** Vyhledávání, vkládání a mazání je u vyvážených stromů velmi rychlé (logaritmická složitost).
- **Flexibilita:** Strom se může dynamicky zvětšovat a zmenšovat.

Nevýhody:

- **Degenerace:** Pokud vkládáme data již seřazená (1, 2, 3, 4…), strom se může změnit v „čáru“ (seznam), čímž ztrácí svou rychlost. To řeší tzv. **samovyvažující se stromy** (např. AVL nebo Red-Black stromy).

Související pojmy: B-strom, Indexování, Databáze, Algoritmus, Rekurze, Huffmanovo kódování.