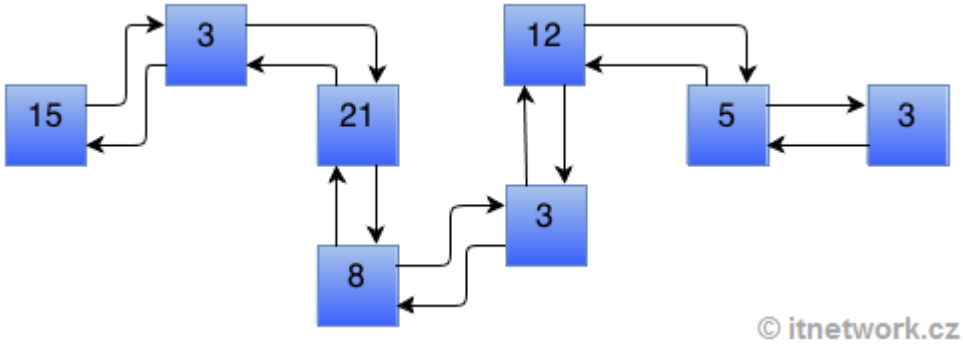
04 - Spojové struktury

Spojový seznam (linked list)

- prvky jsou v paměti rozházené a ukazují na sebe
- každý prvek nese informaci, kde se nachází následující prvek
  - většinou zároveň nese i informaci, kde se nachází předchozí prvek (obousměrný seznam)
- jde o vytvoření "řetězu"



Výhody (oproti poli)

- není omezen pevnou délkou, lze položky flexibilně přidávat / odebírat
- jednodušše se vkládají prvky doprostřed seznamu
  - netřeba prvky posouvat, pouze se přepíše ukazatel na další prvek

Nevýhody (oproti poli)

- nelze efektivně přeskočit na n-tý prvek (indexování) - vždy je nutné jít postupně

Časová složitost

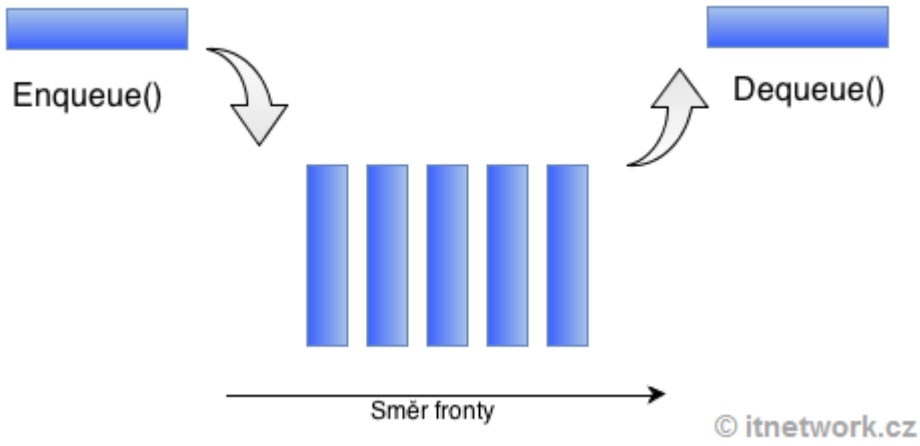
- najdi prvek: O(n)
- přidej prvek na začátek: O(1)
- přidej prvek někam: O(n)
- smaž prvek: O(n)

Shrnutí

Spojový seznam se hodí pro operace, kde nepotřebujeme indexovat prvky, ale potřebujeme je efektivně vkládat a mazat - především, pokud je vkládáme na neurčitá místa, nikoliv vždy na konec / začátek seznamu.

Fronta (queue)

- kolekce, která má 2 základní funkce - **přidat (queue)** a **vymazat (enqueue)**
- funkce pro přidání vždy přidává prvek na konec fronty
- funkce pro vymazání vždy odebírá prvek ze začátku fronty

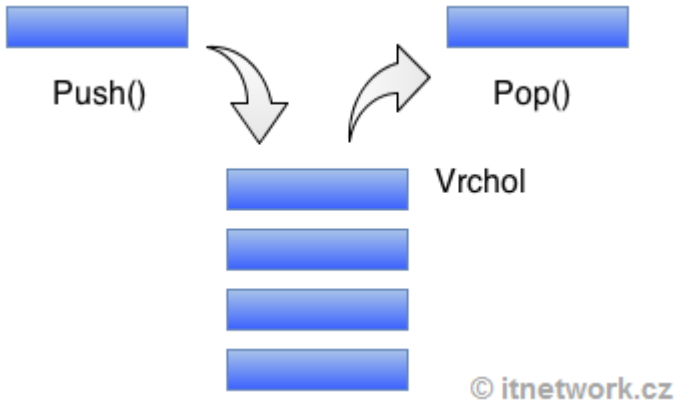


Shrnutí

Fronta se používá v případech, kdy chceme zachovat stejné pořadí prvků, jako když přišly (FIFO - first in, first out). Typickým příkladem je např. fronta u doktora. Prvky se staví za sebe a ve stejném pořadí se poté i dostávají z fronty ven.

Zásobník (stack)

- kolekce, která má 2 základní funkce - **přidat (push)** a **vymazat (pop)**
- funkce pro přidání vždy přidává prvek na začátek fronty
- funkce pro vymazání vždy odebírá prvek ze začátku fronty

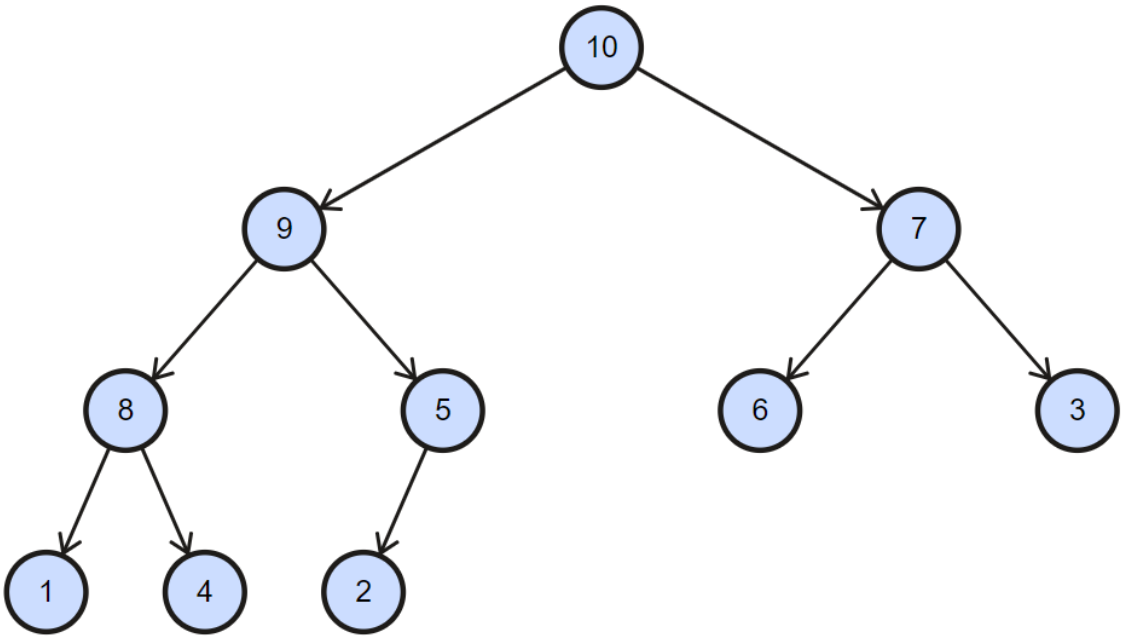


Shrnutí

Zásobník se používá v případech, kdy chceme nejdříve vyjmout prvky, které jsou v zásobníku nejkratší čas (LIFO - last in, first out). Typickým příkladem je např. sud. Prvky se skládají na sebe a pokud se chceme dostat k nejspodnějšímu, musíme nejdříve odstranit všechny prvky, co jsou nad ním.

Binární stromy

- vždy obsahují kořen (první uzel)
- z kořenu jdou další uzly - každý uzel má nejvýše 2 potomky
- v každém uzlu je uložen právě jeden prvek
- větev stromu končí, když už uzel nemá žádné další potomky



- podle logiky uložení dat se binární stromy můžou dělit na různé kategorie (halda, nevyvážený strom, vyvážený vyhledávací strom)
- stromy se časo používají např. při vyhledávání:
  - **vyvážený strom**
  - pravá větev je vždy vyšší hodnota než daný uzel, levá větev naopak menší hodnota
  - při vyhledávání můžeme jednoduše zvolit nejkratší cestu
  - **samovyvažovací strom**
  - stromy, co mají funkce navíc, které kontrolují, zda nejsou nevyvážené
    - AVL stromy (rozdíl hloubky je max 1), B-stromy (více prvků v jednom uzlu)

Shrnutí

Binární stromy se používají hlavně při vyhledávání, kde lze pole prvků upravit na formátovaný strom. Díky tomu lze v něm nalézt data rychleji než v jiných strukturách.