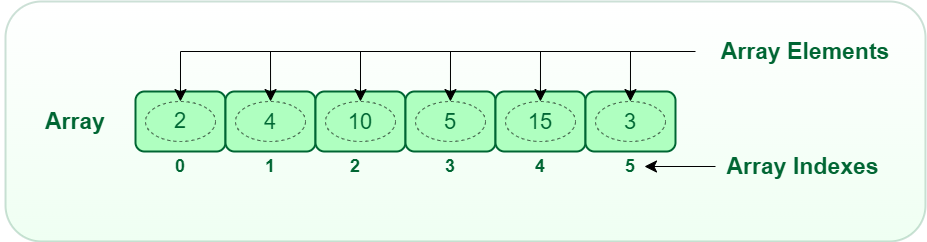
**Typy datových struktur - Pole, Spojový seznam, Strom, Fronta, Zásobník, Halda**

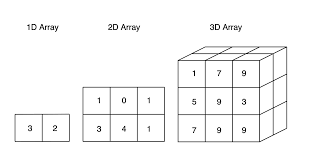
**Co je to datová struktura?**

* Způsob organizace dat v paměti
* Zajišťuje efektivitu použití dat
* Umožňuje uchovávat a zpracovávat množiny dat
  + stejného typu
  + různého typu ale logicky související

**Pole**

* Kolekce elementů
* Identifikovaný indexem
* Pevně dané pořadí prvků
* Každý prvek na konkrétním místě
* Mohou být vícerozměrná
  + Počet indexů = dimenze pole (1D, 2D)
* Velikost může být dynamická nebo statická
* Prvky jsou stejného typu
* Efektivnější při třídění
* Pomalejší při hledání (porovnání každého s každým, pokud se nehledá přes index)





**Spojový seznam**

Co je to?

LinkedList obsahuje sérii NODES, každá obsahuje hodnotu a odkaz na další NODE

* První vytvořená NODE má hodnotu a odkazuje na NULL (je zároveň poslední)
  + Je přiřazena jako HEAD LinkedListu
* Při přidání nový node je posední node atributu NEXT přiřazena nová NODE a zároveň atributu TAIL v LinkedListu je taky přiřazena nová node

Pro získání například 10. prvku, musíme procházet LinkedList od začátku popořadě dokud nedojdeme do 10. prvku

Přidávání

* O(1)

Vyhledávání

* O(N)
* Lineární
  + dynamická struktura
  + prvky navzájem lineárně provázané vzájemnými odkazy (ukazatelé nebo reference)
  + Nesmí existovat cykly v odkazech
* Jednosměrný
  + Základní varianta
  + Každý uzel obsahuje
    - DATA
    - UKAZATEL (na další uzel)
  + Poslední uzel ukazuje na NULL
  + Struktura má ukazatel na první prvek

Diagram

Description automatically generated

* Obousměrný
  + Každý uzel obsahuje
    - UKAZATEL NA PŘEDEŠLÝ PRVEK
    - DATA
    - UKAZATEL NA DALŠÍ PRVEK

Chart, box and whisker chart

Description automatically generated

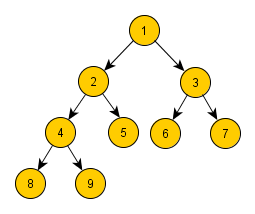
* Kruhový
  + Cyklus
  + Navázání konce a začátku
    - Poslední odkazuje vždy na začátek

Chart, box and whisker chart

Description automatically generated

**Strom**

* Hierarchická struktura dat (objektů, souborů, vyhledávání)
* Strom jde vždy ze shora dolů
* NODE stromu obsahuje
  + Pole CHILDREN
* Nevybalancovaný strom (počet NODES na obou stranách od kořene se výrazně liší)
  + Efektivita algoritmů pro strom jsou méně efektivní
* Soubor uzlů, který jsou spojeny hranami
  + Obsahuje data a odkazy na potomky a svého rodiče
* Všechny uzly (kromě kořene – začáteční prvek) mají **JEDNOHO** předka a žádný nebo **VÍCE potomků**
* PŘ. Hierarchie adresářů, struktury WEB stránek
* Cesta stromu
  + Délka cesty
    - Počet hran od kořene k uzlu (hloubka)
  + Délka vnitřní cesty
    - Součet délek cest z uzlu do uzlu
  + Hloubka stromu
    - Rovna hodnotě maximální hloubky uzlu
* Typy uspořádání stromu (metody procházení)
  + Přímé (pre-order) – 1, 2, 4, 8, 9, 5, 3, 6, 7
  + Vnitřní (in-order) – 8, 4, 9, 2, 5, 6 , 3, 7
  + Inverzní (post-order) – 8, 9, 4, 5, 2, 6, 7, 3 ,1



* Metody struktury stromu pro vyhledávání
  + Dokonale vyvážený strom
    - Pro každý vrchol platí = počet vrcholů v levém podstromu se liší NEJVÍCE o 1
  + Vyhledávací
    - *všechny klíče v levém podstromu jsou menší  
      než a všechny klíče v pravém podstromu jsou větší než klíč v  
      daném vrcholu.*

Binární strom

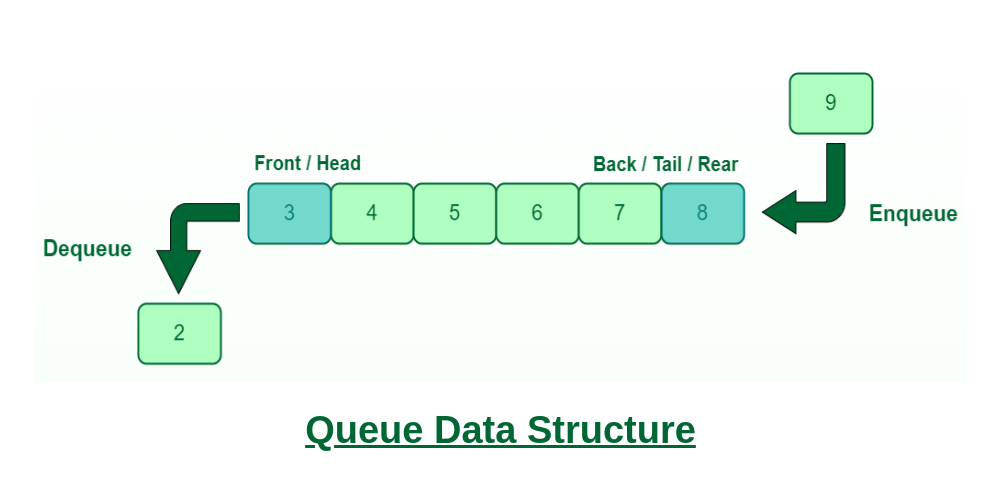
rodič může mít vždy jen 2 děti (ne méně, ne více)

Jeho NODE obsahuje

* LEFT
* RIGHT

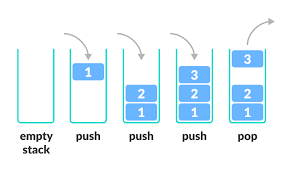
**Fronta**

* Ukládá prvky v tzv. FRONTĚ
* Charakteristika: FIFO (First in, First out)
* Funguje na principu reálné fronty
  + Prvky jsou uloženy v pořadí ve kterém byly přidány
  + Prvky jsou odebrány v opačném pořadí

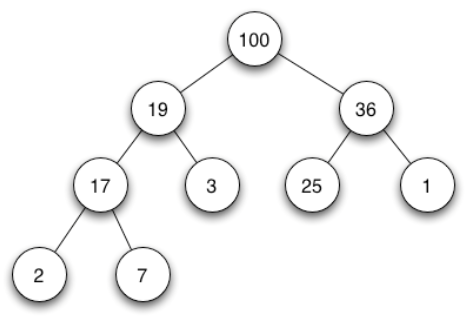


**Zásobník**

* Ukládá prvky v tzv. ZÁSOBNÍKU
* Charakteristika: LIFO (Last in, First out)
* Funguje na principu zásobníku zbraně
  + Prvky jsou uloženy v pořadí, ve kterém jsou přidány
  + Prvky jsou odebrány z konce – poslední přidaný prvek půjde ven jako první při zavolání metody POP



**Halda**

* Stromová datová struktura
* Uzel má žádného nebo více potomků
* Každý potomek má hodnotu menší nebo rovno jeho rodičovi
* Využívá většinou pro účely řazení a vyhledávání
* Využívá pro hledání nejkratších cest, třídění grafů, řízení paměti
* **MAX-HEAP**
  + v kořenu stromu je prvek s nejmenším klíčem
  + každý uzel má hodnotu větší nebo rovnu hodnotě svých potomků
* **MIN-HEAP**
  + v kořenu stromu je prvek s největším klíčem
  + každý uzel má hodnotu menší nebo rovnu hodnotě svých potomků