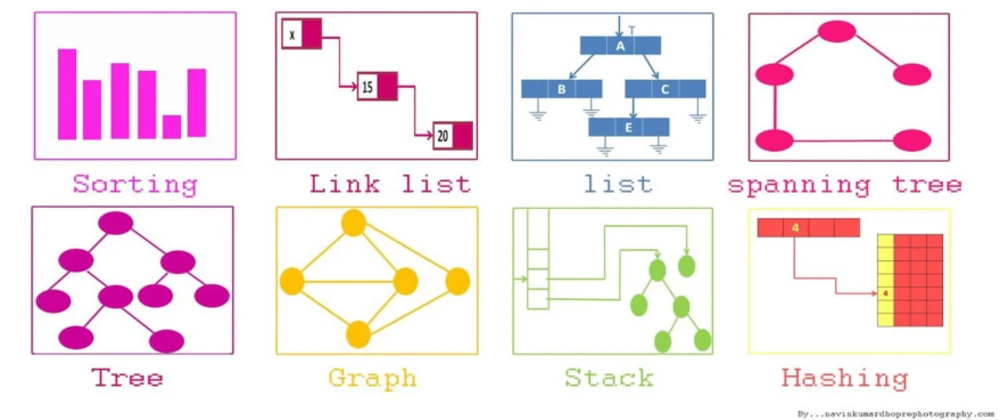
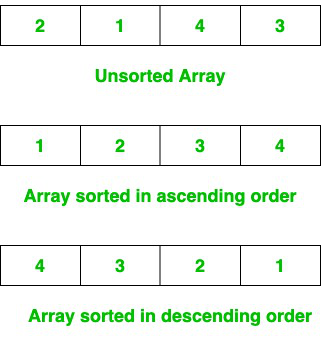
**Vlastnosti datových struktur – Seřazenost a opakování prvků, Indexace, hashování a klíče prvků**

**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

**Seřazenost a opakování**

* Jak jsou prvky v dat. struk. uspořádány
* **POLE (Array) nebo ŘETĚZCE (String)**
  + Pevně dané pořadí prvků
  + Každý prvek na konkrétním místě
  + Prvky stejného typu
  + Efektivnější při třídění
  + Pomalejší při hledání (porovnání každého s každým, pokud se nehledá přes index)
* **MNOŽINY nebo HASHOVACÍ TABULKY (HashSet, Dictionary)**
  + Nemají definované pořadí prvků
  + Unikátní prvky (bez duplicit)
  + Každý prvek uložen pouze jednou
  + Rychlejší přístup a lepší výkon při operacích s prvky
    - Vyhledávání pomocí klíče
* Výběr vhodné dat. struktury závisí na konkrétním použití
* Různé dat. struktury mají různé časové a prostorové nároky pro různé operace

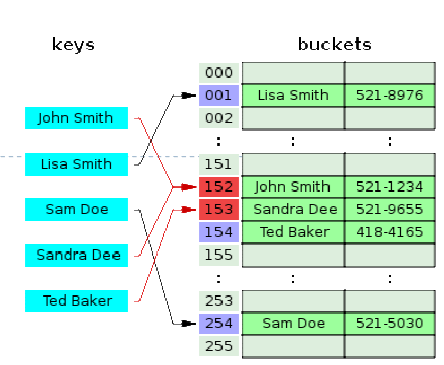


**Indexace**

* Proces přidělování identifikátorů prvkům
* Tzv. pozice v struktuře
* Indexy se používají k rychlému přístupu
* Klíč v množině se nepočítá jako index
* U pole typicky celé číslo (klíč u množin může být cokoliv)
* Je potřeba dávat pozor, aby index nevyhledával mimo rozsah
  + Chyba nebo pád programu
* Indexování ve většině programovacích jazycích začíná 0
  + PASCAL začíná 1
* Indexace funguje pomocí pointerů
  + První prvek typu int má 4 bytes
  + Ostatní prvky se naléznou pomocí vyhledání paměťvě pomocí bytes

**Hashování**

* Technika pro efektivní
  + Vyhledávání
  + Vkládání
  + Odstraňování
* Využívá unikátní kódování každého prvku, který slouží jako KLÍČ
  + Převádí (nezasahovaný) klíč libovolného typu na unikátní číselnou hodnotu (v ideálním případě)
  + V praxi nemusí být vždy možné mít unikátní hash pro každý prvek
    - Kolize – hashovací funkce není dostatečně komplexní nebo malý rozsah hodnot
    - Omezení paměti – hashovací tabulka omezena velikostí = při velkém počtu prvků se přeplní a budou se opakovat
    - Efektivita – složitější hashovací funkce jsou pomalejší a náročnější na výpočet
* Dobrá hashovací funkce s řešením kolizí bude mít časovou konstantu pro vykonávání operací O(1)
* PŘ.
  + Vyhledání jména na seznamu
    - Spočítáme hash pro každé jméno a použijeme jako index
    - Pro vyhledání se spočítá hash pro vyhledávané jméno a zpřístupní se v tabulce podle hashe



**Klíče prvků**

* Hodnota používaná k označení pomocí unikátního identifikátoru prvku
* V DICTIONARY je většínout String
* V STROMĚ většinou hodnota uzlu
* Zaručují efektivní vyhledávání