17. Vyhledávání a porovnávání

Účel vyhledávání (V čem spočívá úloha „najít hodnotu v datové struktuře“?)

- Vyhledávání v datové struktuře se používá k nalezení určité hodnoty nebo prvků v určitém rozsahu v dané struktuře
  1. Hledání konkrétního záznamu v databázi
     - Například hledání informace o konkrétním produktu nebo zákazníkovi v databázi
  2. Vyhledávání v uspořádané struktuře
     - Uspořádané struktury jako například seznamy nebo stromy se používají k ukládání dat v určitém pořadí. Vyhledávání v takové struktuře může být použito k nalezení prvního, nejmenšího, nebo největšího prvku v určitém rozsahu
  3. Hledání duplicitních záznamů
     - V některých případech může být potřeba najít duplicitní záznamy v datové struktuře
  4. Vyhledávání s podmínkami
     - Vyhledávání může být prováděno s určitými podmínkami. Například hledání všech zákazníků, kteří si koupili určitý produkt v určitém časovém období

Příklad algoritmu, který vyhledává hodnotu v posloupnosti hodnot

- Jedním z nejjednodušších algoritmů pro vyhledávání hodnoty v posloupnosti je sekvenční vyhledávání. Tento algoritmus prochází všechny prvky v posloupnosti postupně a porovnává je s hledanou hodnotou. Pokud najde hledanou hodnotu, vrátí index tohoto prvku v posloupnosti. Pokud hledanou hodnotu nenajde, vrátí zpravidla hodnotu -1

```java
public static int search(int[] arr, int x) {
    int n = arr.length;
    for (int i = 0; i < n; i++) {
        if (arr[i] == x) {
            return i; // nalezli jsme hledanou hodnotu, vrátíme index
        }
    }
    return -1; // hledanou hodnotu jsme nenašli
}
```

  - Tento algoritmus přijímá jako vstup pole **arr** obsahující prvky posloupnosti a hodnotu **x**, kterou hledáme v této posloupnosti. Algoritmus prochází všechny prvky v poli a porovnává je s hledanou hodnotou **x**. Pokud najde hodnotu, vrátí index tohoto prvku v poli. Pokud hodnotu nenajde, vrátí -1

Souvislost porovnávání a vyhledávání

- Porovnávání
  - Používá se k porovnání dvou hodnot a určení, zda jsou stejné nebo ne
  - Může být provedeno na různých typech dat, jako jsou čísla, řetězce, objekty atd. Když vyhledáváme hodnotu v datové struktuře, je často potřeba porovnávat hodnoty v této struktuře s hledanou hodnotou, aby se určilo, zda se jedná o shodné hodnoty nebo nikoliv. Porovnávání se tedy používá jako klíčová součást procesu vyhledávání
- Vyhledávání
  - Používá se k nalezení určité hodnoty v datové struktuře, například v poli, seznamu, stromu nebo databázi
  - Vyhledávací algoritmy procházejí prvky datové struktury a porovnávají je s hledanou hodnotou, aby určily, zda se jedná o hledanou hodnotu nebo nikoliv. Porovnání se tedy používá jako důležitá součást procesu vyhledávání

Jak souvisí rozdíl (odčítání) a výsledek s porovnáváním

- Při porovnávání dvou hodnot se porovnávají na základě toho, která hodnota je větší nebo menší než druhá
- Pokud jsou hodnoty stejné, jsou považovány za rovny
- Rozdíl (odčítání) se používá k výpočtu rozdílu mezi dvěma hodnotami. Například, když odečteme číslo 5 od čísla 10, výsledkem bude 5. Výsledek odčítání může být porovnán s jinými hodnotami, aby se určilo, zda je větší nebo menší než daná hodnota
- Například, když odečteme číslo 5 od čísla 10 a porovnáme výsledek s číslem 3, zjistíme, že výsledek odčítání (5) je větší než 3

Souvislost s rozhraním – přirozené řazení (Comparable), absolutní řazení (Comparator)

- Rozhraní Comparable
  - Definuje metodu **compareTo**(), která umožňuje srovnávat instance dané třídy podle určitého kritéria
  - Implementace tohoto rozhraní umožňuje tzv. přirozené řazení, tedy řazení podle výchozího kritéria pro danou třídu
  - Pokud například máme třídu Person, můžeme implementovat rozhraní **Comparable** a definovat, že instance této třídy budou řazeny podle jména, příjmení nebo věku. V případě, že chceme řadit podle jiného kritéria, musíme implementovat rozhraní Comparator
- Rozhraní Comparator
  - Umožňuje porovnávat instance tříd podle jiného kritéria, než je to výchozí pro danou třídu
  - To umožňuje tzv. absolutní řazení, tedy řazení podle určitého kritéria, které není výchozí pro danou třídu
  - Pokud například máme třídu Person, můžeme implementovat rozhraní Comparator a definovat, že instance této třídy budou řazeny podle výšky, váhy nebo národnosti

Prakticky:

- Třída dána, nastavení možnosti porovnávání, volání metody pro využití komparátoru