Hledání bez zarážky

(sekvenční hledání)

Úvod:

- používá se při vyhledávání v poli (seznamu), které je neseřazené

- princip: sekvenčně (v cyklu) procházím prvky pole, dokud prvek nenaleznu nebo neprojdu celé pole

Ukázka v kódu:

```
public static int SekvencniVyhledavani(int[] pole, int x)
{
    for (int i = 0; i < pole.Length; i++)
    {
        if (pole[i] == x)
            return i; // vrací index, pokud jsme hledaný prvek nalezli
    }
    return -1; // při neúspěchu vrací -1
}
```

Výhody:

- **Jednoduchost**: Velmi snadná implementace i pochopení.
- **Flexibilita**: Funguje na neseřazených datech, což je často užitečné.
- **Nezávislost na velikosti pole**: Není třeba předem znát velikost datové struktury nebo ji měnit.

Nevýhody:

•**Pomalejší pro velká data**: kvůli Lineární složitosti bývá doba vyhledání pro velká pole dlouhá

•**Nemožnost optimalizace bez seřazení**: Bez seřazení dat nelze použít efektivnější metody (např. binární vyhledávání).

Časová složitost

- **Nejlepší případ**: O(1) – Pokud se hledaný prvek nachází hned na začátku pole.

- **Průměrný případ**: O(n/2) – Průměrně najdeme prvek někde uprostřed pole, ale kvůli lineární složitosti to označujeme jako **O(n)**.

- **Nejhorší případ**: O(n) – Pokud se hledaný prvek nachází na konci pole nebo tam vůbec není, musíme projít všech **n** prvků.

Příklad vyhledávání:

Máme pole: [3, 8, 1, **7**, 5]
Hledaný prvek: **7**
Krok za krokem

1.Vyhledávání začne na indexu 0 a postupně prochází každý prvek, dokud nenajde shodu.

2.Na indexu 3 je nalezena hodnota 7, což odpovídá hledanému prvku.

3.Algoritmus končí s výsledkem index: **3**

Index	Hodnota v poli	Hledaný prvek = 7	Výsledek
0	3	Ne	Pokračuj
1	8	Ne	Pokračuj
2	1	Ne	Pokračuj
3	**7**	Ano	Konec