Seminář PRG
10. hodina - 8.11.2024

Gymnázium Voděradská 2024/2025
Jan Borecký

Program dneška - Vyhledávání v poli

1. Lineární vyhledávání
2. Binární vyhledávání
3. Rekurzivní binární vyhledávání

Poznámka

Ve složce aktuální hodiny najdete příklady vypracování úloh na oblíbená jídla a volby z minulé hodiny. Můžete se inspirovat (nelze už dostat bonusovou známku)

Lineární vyhledávání

- Prochází postupně celé pole
  - for cyklus od 1. do posledního prvku
- U každého prvku zkontroluje, jestli to není ten, co chceme
  - pokud ano, vrátí ho
  - pokud ne, nedělá nic a posune se o prvek dál

```
int LinearSearch(int[] array, int target)
{
	for (int i = 0; i < array.Length; i++)
	{
		if (array[i] == target)
		{
			return i; // Vrátíme index, na kterém jsme prvek našli.
		}
	}
	return -1; // Prvek jsme nenašli.
}
```

Seřazená pole

- Jak dlouho může trvat lineární vyhledávání v poli s *n* prvky?
- Mějme seřazené integerové pole
  [1, 5, 8, 9, 13, 15, 16, 19, 21, 30, 48, 50]
- Jak dlouho může trvat lineární vyhledávání v seřazeném poli s *n* prvky?
- Můžeme nějak využít toho, že je pole seřazené k rychlejšímu nalezení prvku?

Seřazená pole

- Seřazená pole nám mohou urychlit hledání, protože můžeme odhadnout, kde se prvek asi vyskytuje, a eliminovat tím procházení všech prvků

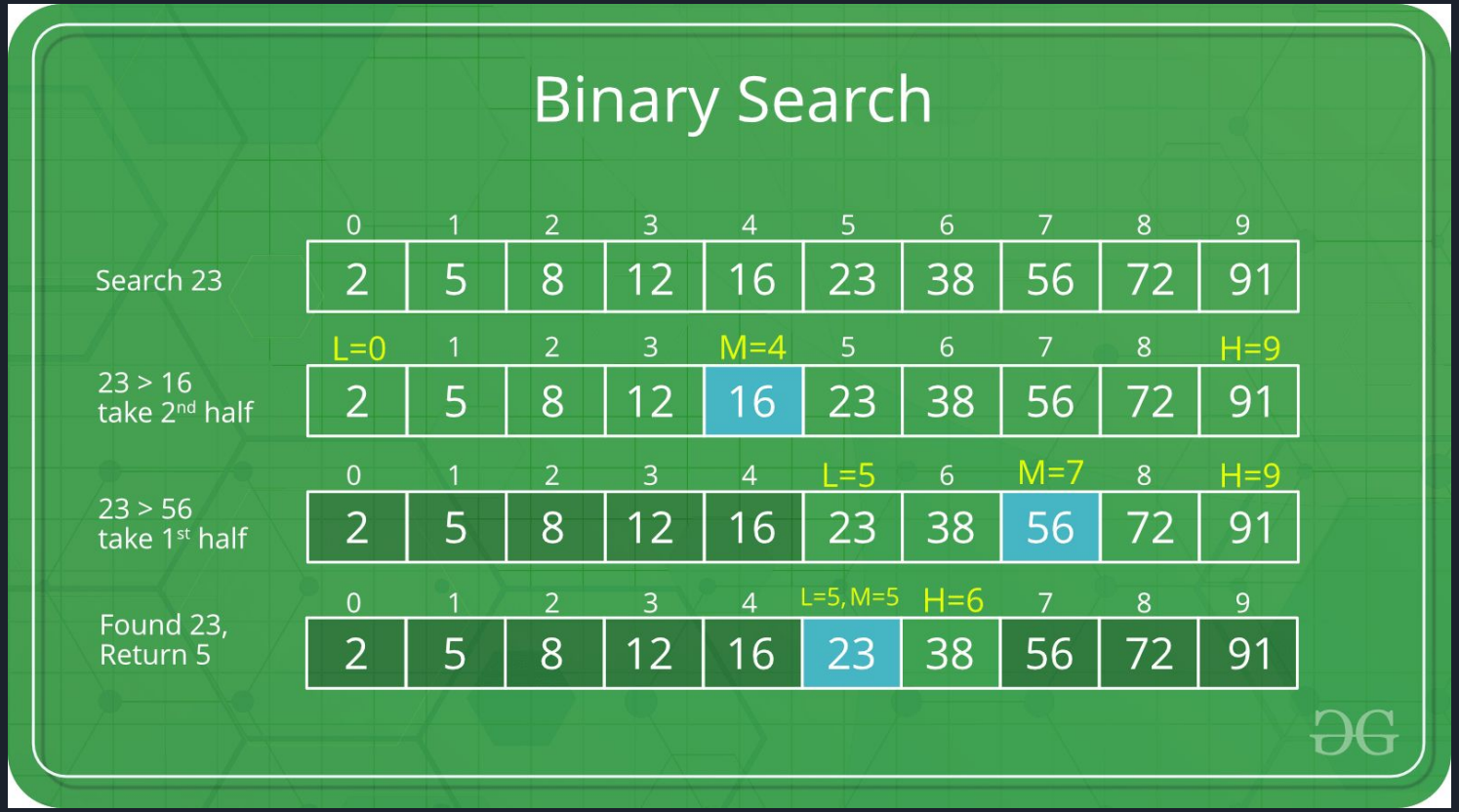
- Jak pole seřadit?
  - Řadícím algoritmům se budeme věnovat příští hodinu

Binární vyhledávání

- Celé pole postupně dělí na poloviny a postupuje k hledanému prvku

- Využijeme **POUZE** v seřazených polích

- Algoritmus:
  - Nastaví výchoze interval vyhledávání na celé pole
  - Najde prostřední prvek intervalu
  - Podívá se, jestli prostřední prvek není hledaný prvek
    - Pokud ano, vrátí ho
    - Pokud ne, podívá se, jestli je prostřední prvek **menší/větší** než hledaný prvek a upraví horní/dolní hranici intervalu podle toho
  - Vrací se do kroku 2 (najde prostřední prvek intervalu) a jede znovu


Binary Search
Search 23
0 1 2 3 4 5 6 7 8 9
2 5 8 12 16 23 38 56 72 91
23 > 16
take 2nd half
L=0 1 2 3 M=4 5 6 7 8 H=9
2 5 8 12 16 23 38 56 72 91
23 > 56
take 1st half
0 1 2 3 4 L=5 6 M=7 8 H=9
2 5 8 12 16 23 38 56 72 91
Found 23,
Return 5
0 1 2 3 4 L=5, M=5 H=6 7 8 9
2 5 8 12 16 23 38 56 72 91

Binární vyhledávání

- Jak dlouho může trvat binární vyhledávání v poli s *n* prvky?
  - O(log n) (v nejhorším případě $\log_2 n+1$ iterací)

- Porovnání časové složitosti lineárního a binárního vyhledávání u polí různých velikostí:
  - n = 10: Lineární 10, Binární 3
  - n = 100: Lineární 100, Binární 7
  - n = 1 000: Lineární 1000, Binární 10
  - n = 1 000 000: Lineární 1 000 000, Binární 20

- Pokud v poli vyhledáváme často, vyplatí se nám si pole seřadit a poté už v něm vyhledávat binárně (Oproti tomu mít pole neseřazené a vždy hledat lineárně.)

Rozděl a Panuj (Divide and Conquer)

- Rozděl a panuj je způsob, jakým přistupovat k problému

- Základem jsou 3 kroky:
  - Rozdělení problému na jednodušeji řešitelné podproblémy
  - Vyřešení podproblémů rekurzivním voláním
  - Zkombinování podproblémů k získání finálního řešení

- Využití:
  - Quicksort (Poznáme v budoucnu)
  - Merge Sort (Poznáme v budoucnu)
  - Násobení matic (Strassenův algoritmus)
  - Rychlá Fourierova Transformace (FFT)

- Jak by se dal tento přístup využít u binárního vyhledávání?

Cvičení

Ve složce aktuální hodiny si najděte projekt Search Playground a začněte na něm pracovat.

Děkuji za pozornost

Zpětná vazba:
https://forms.gle/Roe5RVSTDU9pKB7X6

Kontakt:
Mail - honza.borecky@seznam.cz
Discord - yeenya (Yeenya#6930)