**Интерполационно търсене(Interpolation search)**

### От

### Симеон М.

### 10А

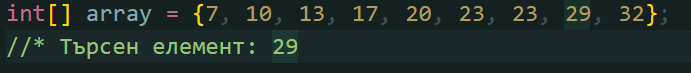
## 22.04.2023

## Интерполационно търсене

## Въведение:

1. Определение
2. Интерполационното търсене е алгоритъм за търсене на ключ в масива, който е **подреден** по числови стойности, присвоени на ключовете. Интерполационното търсене е вид, чрез който хората търсят телефонен номер с име. Интерполационното търсене е подобрение в сравнение с двоичното търсене на инстанции, където стойностите в сортиран масив са равномерно разпределени. Двоичното търсене винаги отива до средния елемент за проверка. От друга страна, търсенето с интерполация може да отиде на различни места според стойността на ключа, който се търси. Интерполационното търсене намира конкретен елемент чрез изчисляване на проучената позиция. Първоначално проучената позиция, е позицията на **най-средния елемент** от колекцията. Ако възникне съвпадение, тогава се връща **индексът на елемента**.
3. Как работи алгоритъма ?
4. Започва да търси **желания елемент** от средата на листа.
5. Ако има съвпадение, връща индексът на елемента.
6. Ако няма, изчислява проучената позиция.
7. Разделя листът, използвайки формулата за изчисление на позицията и намира **нов среден елемент**.
8. Ако търсения елемент е по-голям от средата, търси в дясната част.
9. Ако е по-малък, търси в лявата част.
10. Повтаряме, докато имаме съвпадение.

1. Постъпково разглеждане на алгоритъма



1. Ще изчислим позицията на елемента по формулата:

mid = low + ((x – A[low]) \* (high – low) / (A[high] – A[low]))

Получаваме:

low = 0

high = 8

mid = 0 + ((29 – 7) \* (8 – 0) / (32 – 7))

mid = 6

1. В тази ситуация можем да видим, че изчислената позиция е отдясно, тъй като необходимата стойност е от дясната страна на масива, по подобен начин, ако стойността е била от лявата страна на масива, стойността на изчислената позиция щеше да остане отляво.

Diagram

Description automatically generated

low = 7

high = 8

mid = 7 + ((29 – 29) \* (8 – 7) / (32 – 29))

mid = 7

1. Така открихме елемента, който търсим се на позиция arr[7].
2. Имплементация(C#)

Text

Description automatically generated

* Резултат:

Text

Description automatically generated

1. Времева сложност(Time complexity)

* **Най-добър случай –** **O(1)**
* Най-добрият случай възниква, когато целта е намерена точно като първата очаквана позиция, изчислена с помощта на формулата. Тъй като извършваме само едно сравнение, времевата сложност е O(1).
* **Най-лош случай – O(n)**
* Най-лошият случай възниква, когато даден набор от данни е експоненциално разпределен.
* **Среден случай -** **O(log(log(n)))**
* Ако наборът от данни е сортиран и равномерно разпределен, тогава отнема O(log(log(n))) време като средно (log(log(n))) се правят сравнения.

1. Памет(Space complexity)

* O(1)

1. Приложения

* Потенциален алгоритъм за бързо търсене. За търсене в рамките на равномерно разпределено и сортирано пространство за търсене (като телефонен указател)

1. Предимства

* Когато всички елементи в списъка са сортирани и равномерно разпределени, тогава времето за изпълнение на алгоритъма за търсене чрез интерполация е **log(log n).**

1. Недостатъци

* Когато елементите в списъка се увеличават експоненциално, тогава времето за изпълнение на алгоритъма за търсене с интерполация е 0(n), т.е. най-лошият случай.

1. Оптимизация

* Средна времева сложност от **O(log(logn))** може да бъде постигната като използваме динамична структура от данни.

1. Обобщение(Summary)

* Интерполационно търсене е ефективен алгоритъм за търсене, използван за търсене на конкретна целева стойност в сортиран масив. Той използва интерполация, за да оцени позицията на целевата стойност в масива и след това извършва линейно търсене, за да намери целевата стойност.
* Алгоритъмът има времева сложност средно O(log log n), което го прави по-бърз от двоичното търсене за големи, равномерно разпределени масиви. Въпреки това, той може да работи лошо за масиви с неравномерно разпределени данни или данни с много дубликати.
* Интерполационното търсене може да бъде полезен инструмент за търсене в големи, сортирани масиви, но е важно да се вземе предвид разпределението на данните, преди да се използва. Като цяло търсенето с интерполация е добър избор за търсене в големи, равномерно разпределени масиви, но други алгоритми може да са по-подходящи за различни типове данни.

Препратки:

* <https://www.mygreatlearning.com/data-structure/tutorials/interpolation-search>
* <https://www.studytonight.com/post/interpolation-search-algorithm?utm_content=cmp-true>
* <https://www.codingninjas.com/codestudio/library/time-space-complexity-of-searching-algorithms>
* <https://iq.opengenus.org/interpolation-search-algorithm/>
* <https://www.2braces.com/data-structures/interpolation-search>
* <https://github.com/BG-IT-Edu/School-Programming/blob/main/Courses/Applied-Programmer/Algo-and-Data-Structures-Basics/05-%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8-%D0%B7%D0%B0-%D1%82%D1%8A%D1%80%D1%81%D0%B5%D0%BD%D0%B5/05.1.%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8-%D0%B7%D0%B0-%D1%82%D1%8A%D1%80%D1%81%D0%B5%D0%BD%D0%B5.pdf>