**Двоично търсене(Binary search)**

### От

### Симеон М.

### 10А

## 18.04.2023г.

## 

## Двоично търсене

## Въведение:

1. Определение
2. Двоичното търсене е **ефикасен(бърз)** алгоритъм за намиране на елемент в сортиран лист от числа. Използва подхода **„разделяй и владей“**, като продължава да разделя наполовина частта(**subarray**) на листа, в която вероятно се съдържа елемента. Това продължава, докато не се стигне само до един елемент в частта, която сме разделили.
3. Как работи алгоритъма ?
4. Заделяме необходимото място от сортирания масив.
5. Взимаме средния елемент и го сравняваме с елемента, който търсим. Ако елемента, който търсим е равен на средния, тогава връщаме индекса на този елемент и функцията на алгоритъма приключва. Ако числото, което търсим е по-малко от средния елемент, дясната част отпада и търсим в лявата, докато намерим числото. Ако търсения елемент е по-голям от средния, лявата част отпада и продължаваме да търсим в дясната, докато не го намерим.
6. Ако то липсва в масива, **return -1**.

1. Графично изобразяване на алгоритъма

Търсен елемент: **12**

80

99

65

55

50

32

20

12

10

8

7

6

5

4

3

2

1

0

1. 12 го сравняваме със средния елемент, в нашия случай 50 по формулата arr[(low + high) / 2].

10

12

20

32

50

55

65

80

99

1

3

5

7

8

6

4

2

0

* 12 е по-малко от 50, следователно търсим само в лявата част на масива.

10

12

20

32

1

3

2

0

1. Търсим средния елемент и го сравняваме с желания такъв.

20

**12**

10

32

3

2

1

0

* Елемента е намерен на позиция arr[1].

1. Имплементация(c#)

Text

Description automatically generated

* Резултат:

Text

Description automatically generated

1. Рекурсивно двоично търсене – Имплементация

Text

Description automatically generated

* Резултат:

Text

Description automatically generated

4 Времева сложност(Time complexity)

* Най-добър случай – имаме, когато търсения елемент се намира в средата т.е. имаме съвпадение при първото сравнение. Времева сложност: O(1).
* Среден случай – O(logn).
* Най-лош случай – най-лош случай имаме, когато трябва да търсим продължително в масива, докато не остане един елемент(O(logn)).

5 Памет(Space complexity)

* Тъй като алгоритъма не изисква допълнително памет, сложността е O(1).

6 Приложения

* За търсене на елемент в сортирана структура от данни.
* Може също така да бъде полезен и за няколко операции като: намиране на най-малкият елемент, или най-големия в масив.

7 Предимства

* Двоичното търсене е по-бързо от линейното, особено за голям диапазон от числа.
* Алгоритъма е сравнително лесен за имплементиране.
* Двоичното търсене може да бъде използвано като building block за по-сложни алгоритми, като онези използвани в компютърните графики и machine learning.

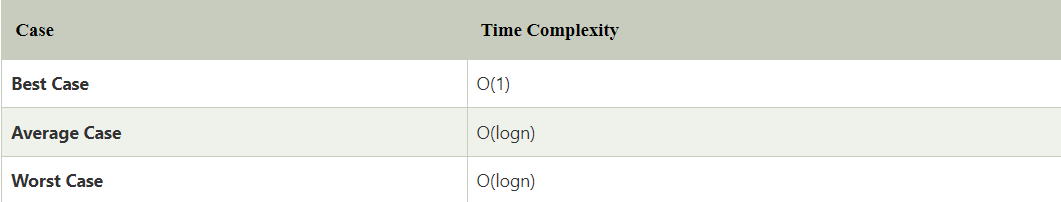
8 Недостатъци

* Трябва масива да е сортиран, ако не е сортиран трябва да се сортира.
* Двоичното търсене изисква структурата от данни, която се търси, да се съхранява в съседни места в паметта. Това може да е проблем, ако структурата на данните е твърде голяма, за да се побере в паметта, или ако се съхранява на външна памет, като например hard drive или в cloud.
* Двоичното търсене може да бъде по-малко ефективно от други алгоритми, като хеш таблици, за търсене на много големи набори от данни, които не се побират в паметта.

1. Обобщение(Conclusion)

* В заключение може да се каже, че двоичното търсене е ефективен алгоритъм за намиране на елемент в сортиран масив. Времевата сложност на двоичното търсене е O(logn). Въпреки това, един от основните недостатъци на двоичното търсене е, че масивът трябва да бъде сортиран. това е полезен алгоритъм за изграждане на по-сложни алгоритми в компютърната графика и машинното обучение или няколко други случая.

Времева сложност и памет:





Препратки

1. <http://www.btechsmartclass.com/data_structures/binary-search.html>.
2. <https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search>
3. <https://www.programiz.com/dsa/binary-search>
4. <https://www.geeksforgeeks.org/binary-search/>
5. <https://towardsdatascience.com/everything-you-need-to-know-about-the-binary-search-algorithm-6bc4f9a3127d>
6. <https://www.programmingalgorithms.com/algorithm/binary-search/>
7. <https://www.javatpoint.com/binary-search>