Binäre Suche

Informatik 12 Februar 2024

Gegeben: Ein sortiertes Array **a** mit den **n** Komponenten **a[0], ... , a[n-1]**

Aufgabe: Entscheide, ob ein für die Variable **value** eingegebener Suchwert mit dem Wert einer Komponente des Arrays **a** übereinstimmt.

Wir durchlaufen den Algorithmus schrittweise anhand des folgenden Beispiels.

Gegeben: Array **a** mit den Komponenten **a[0], ... , a[9]**; n = len(a) = 10
value = **13**

Die rekursiv formulierte Boolesche Funktion **binarysearch** liefert den Wert **`True`**, falls **value** mit dem Wert irgendeiner Komponente von **a** übereinstimmt, andernfalls liefert sie den Wert **`False`**.

Wir übergeben **value** und die Liste **a[0], . . . , a[9]**

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
3	4	5	5	7	8	11	13	19	21

der Funktion **binarysearch**,
welche **a[0], . . . , a[9]** als lokale Liste **array[0], . . . , array[9]** fortführt:

array[0]	array [1]	array [2]	array [3]	array [4]	array [5]	array [6]	array [7]	array [8]	array [9]
3	4	5	5	7	8	11	13	19	21

1. Schritt:
Bestimme den mittleren Index **middle** des Arrays **array**: **middle** = len(array)//2 = 10//2 = 5

2. Schritt:
midvalue = array[middle] = array[5] = **8**
Vergleiche **value** mit **midvalue**:
Falls **`value == midvalue`**: **binarysearch** gibt den Wert **`True`** zurück; gefunden!
Falls **`value < midvalue`**: suche in der Liste array[0], . . . , array[4] links von array[5]
Falls **`value > midvalue`**: suche in der Liste array[6], . . . , array[9] rechts von array[5]

hier: wegen **13 > 8** suche in der Liste array[6], . . . , array[9] .

binarysearch übergibt **value** und die Liste **array[6], . . . , array[9]**

array[6]	array [7]	array [8]	array [9]
11	13	19	21

der Funktion **binarysearch**,
welche **array [6], . . . , array [9]** als lokale Liste **array[0], . . . , array[3]** fortführt:

array[0]	array[1]	array[2]	array[3]
11	13	19	21

1. Schritt:
Bestimme den mittleren Index **middle** des Arrays **array**: **middle** = len(array)//2 = 4//2 = 2

2. Schritt:
midvalue = array[middle] = array[2] = 19
Vergleiche **value** mit **midvalue**:
Falls **`value == midvalue`**: **binarysearch** gibt den Wert **`True`** zurück; gefunden!
Falls **`value < midvalue`**: suche in der Liste array[0], array[1] links von array[2]
Falls **`value > midvalue`**: suche in der Liste array[3] rechts von array[2]

hier: wegen **13 < 19** suche in der Liste array[0], . . . , array[1] .

binarysearch übergibt **value** und die Liste **array[0], . . . , array[1]**

array[0]	array[1]
11	13

der Funktion **binarysearch**,
welche **array [0], . . . , array [1]** als lokale Liste **array[0], . . . , array[1]** fortführt:

array[0]	array[1]
11	13

1. Schritt:
Bestimme den mittleren Index **middle** des Arrays **array**: **middle** = len(array)//2 = 2//2 = 1

2. Schritt:
midvalue = array[middle] = array[1] = 13
Vergleiche **value** mit **midvalue**:
Falls **`value == midvalue`**: **binarysearch** gibt den Wert **`True`** zurück; gefunden!
Falls **`value < midvalue`**: suche in der Liste array[0] links von array[1]
Falls **`value > midvalue`**: die Liste [] rechts von array[1] ist leer;
dann gibt **binarysearch** den Wert **`False`** zurück: nicht gefunden!

Hier: da der Suchwert **value** und **array[middle]** übereinstimmen, hat der Boolesche Term **`value == midvalue`** den Wert **`True`**; folglich liefert **binarysearch** den Wert **`True`**: gefunden!

Wäre 12 der Suchwert, erhielte man wegen 12<13 im 2. Schritt: suche in der Liste array[0] links von array[1]

binarysearch übergibt **value** und die Liste **array[0]**

array[0]
11

der Funktion **binarysearch**, welche den Wert **`False`** liefert, da array[0]≠value und array die Länge 1 hat.
Zusammengefaßt: **binarysearch** liefert den Wert **`False`**, falls

```
array == [] or (array[0] != value  and  len(array) == 1)
```

den Wert **`True`** annimmt.