Zoekalgoritmen

Bachelor IT
Sven Mariën
(sven.marien01@ap.be)

AP HOGESCHOOL ANTWERPEN

AP.BE

Zoekalgoritme: waarom ?

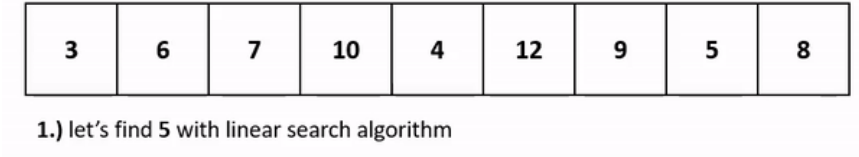
- We willen nagaan of een element aanwezig is een lijst
- We willen de plaats kennen van een bepaald element in een lijst
- We willen nagaan of een bepaald element meermaals aanwezig is in de lijst.
- …

Overzicht

- **Lineair search / Sequential search**
- **Binary search**
- **Hash table**

Lineair search

- Ook wel “sequential” search genoemd.
- Algoritme:
  - Start bij het begin van de lijst
  - Vergelijk met de te zoeken waarde
  - Ga naar het volgende element en herhaal ofwel tot
    - de waarde gevonden werd.
    - het einde van de lijst is bereikt.

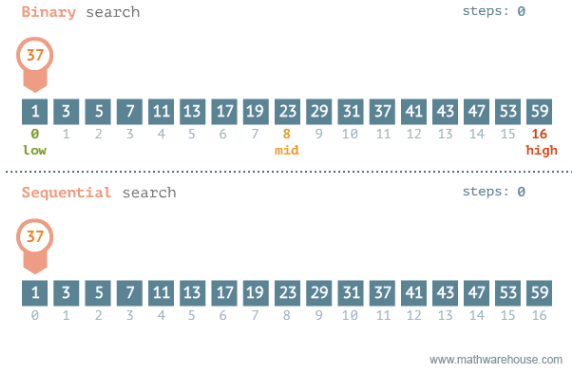


Lineair search

- Dit algoritme hadden we ook reeds gebruikt bij de linked list om een element op te zoeken
- Return waarde:
  - **True/false**: het element is aanwezig / niet-aanwezig
  - **Index** : plaats van het element in de lijst
  - **Node**: in geval van een linked list
- Voordeel:
  - Eenvoudig algoritme
  - De lijst hoeft niet gesorteerd te zijn.
- Nadeel:
  - In het slechtste geval moet de volledige lijst doorlopen worden => time complexity: O(n)

Binary Search

- Andere benamingen: “half-interval search”, “logarithmic search”
- Voorwaarde: de lijst **moet** reeds gesorteerd zijn !
- Algoritme:
  - Neem het **middelste** element in de lijst
  - **Vergelijk** met de **te zoeken waarde**
  - Indien de **gevonden waarde < te zoeken:**
    - Neem de **rechter helft** en start opnieuw
  - Indien de **gevonden waarde > te zoeken:**
    - Neem de **linker helft** en start opnieuw



Binary Search

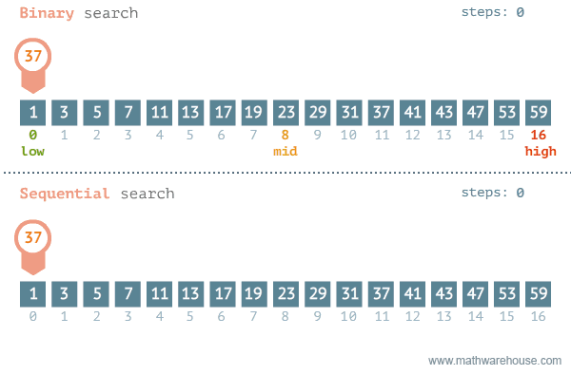
- “Divide and conquer” principe
  - Steeds de lijst in de helft delen
  - Werkt dus enkel met arrays.
  - Recursie !
- Voordeel:
  - Het is een snel algoritme: **O(log n)**
- Nadeel:
  - De lijst moet reeds gesorteerd zijn.
  - Niet bruikbaar met linked lists.

Binary Search / Binary Search Tree

- Werken op dezelfde manier
- Hebben dus ook dezelfde efficiëntie (indien het om een “balanced tree” gaat).
- Beiden: **O(log n)** time complexity

N	Max. aantal compares
7	3
15	4
512	9
1.000.000	20

www.mathwarehouse.com

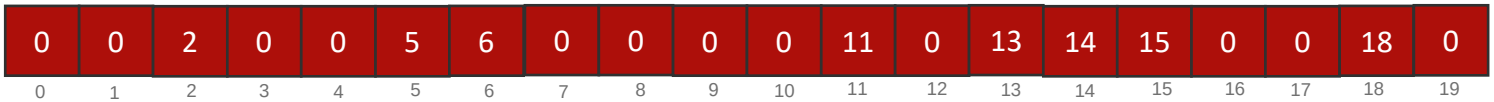


www.mathwarehouse.com

Kunnen we nog beter doen ?

- Wat is de snelste manier om in een array te zoeken ?
- Stel: ik heb een lijstje met volgende elementen:
  - 2, 5, 6, 11, 13,14,15, 18

- Wat als ik mijn array echter op deze manier zou opvullen ?



- Wat is de performantie dan om een element op te zoeken ?
  - Time complexity: O(1) !!! ☺
  - Maar…. Space Complexity: O(max) ☹

Hashing

- Wat is hashing ?

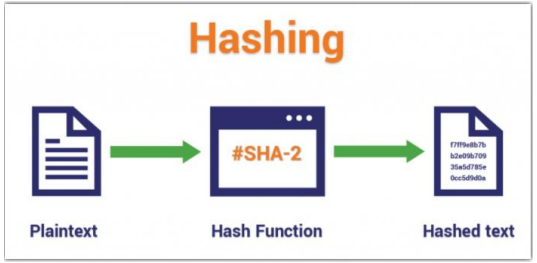
Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value. It is also used in many encryption algorithms.

Original string	“Hash value”
“hallo”	287
“dit is een tekst”	498
“hello”	189
“AP hogeschool”	344

- Hashing is “one way” !

Hashing

- Gekende algoritmes:
  - SHA (SHA-1, SHA-2)
  - MD-4, MD-5
  - …
- Toepassingen:
  - **Datastructuren:** Hashtable**,** Dictionary,..
  - **Authenticatie**: wachtwoorden worden nooit als “plain tekst” bewaard in de databank.
  - **Patronen** zoeken in teksten (bv. plagiaatdetectie)
  - …



Zoeken mbv.“hashing”

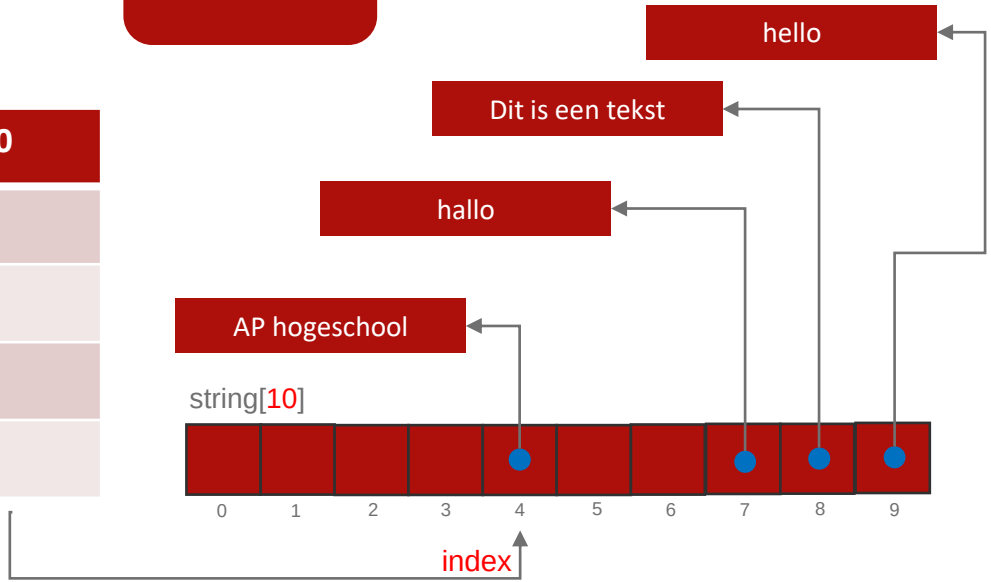
- Hoe kunnen we de hash value nu concreet gebuiken ?
  - We zouden de hash value als index kunnen nemen van onze array.
  - Dit geeft echter nog steeds veel lege ruimten in de array
- Extra toevoeging:
  - gebruik van modulo operator (%)

Original string	“Hash value”
“hallo”	287
“dit is een tekst”	498
“hello”	189
“AP hogeschool”	344

Hashing + Modulo operator = “Hash table”

- Vb. ik verwacht max. 10 elementen in mijn lijst => % 10

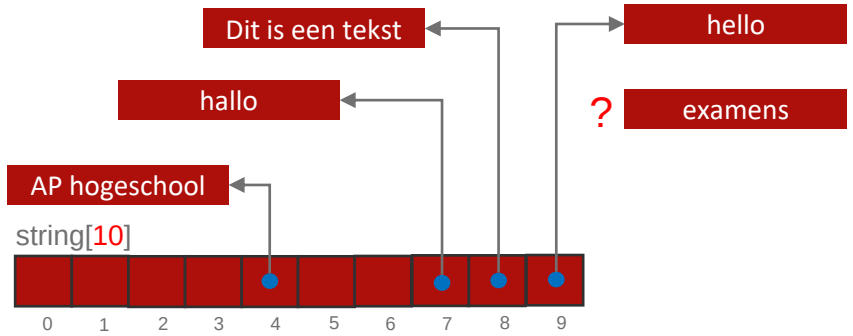
Original string	“Hash value”	%10
“hallo”	287	7
“dit is een tekst”	498	8
“hello”	189	9
“AP hogeschool”	344	4



Hashing collisions

- Het zou kunnen dan het hashing algoritme dezelfde “hash value” geeft voor verschillende “input”
- Dit noemt men “**collisions**”.

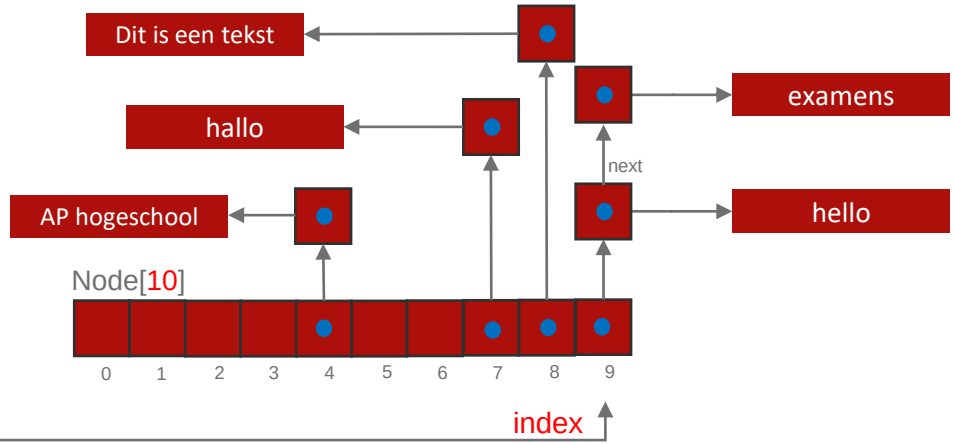
Original string	“Hash value”	%10
“hallo”	287	7
“dit is een tekst”	498	8
“hello”	189	9
“AP hogeschool”	344	4
“examens”	189	9



Hoe collisions vermijden / oplossen ?

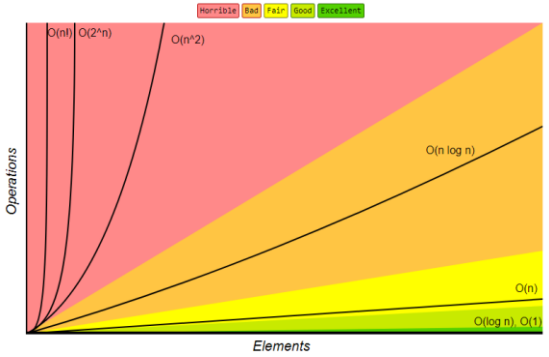
- Een goed hashing algoritme zorgt voor weinig of geen collisions
- Toch collisions ?
  - Werken met **array** + bijkomende **linked list**(s)

Original string	"Hash value"	%10
"hallo"	287	7
"dit is een tekst"	498	8
"hello"	189	9
"AP hogeschool"	344	4
"examens"	189	9



Vergelijking zoekalgoritmes

Zoekalgoritme	Time Complexity	Array	Linked List
Lineair / Sequential search	O(n)	✓	✓
Binary Search	O(log n)	✓	X
Hash table (*)	O(1)	✓	X



(*) Collisions zorgen ervoor dat de time complexity slechter wordt, doordat er extra bijkomende opzoekingen moeten gebeuren.

c# - GetHashCode()

- Op de base class ‘object’
  - bevinden zich 4 methodes
  - GetHashCode() zal een ‘default’ hashcode geven voor eender welk object of type in c#.
  - Uiteraard kan je deze overschrijven

```
int getal = 123;
Console.WriteLine($"{getal} geeft een hashcode: {getal.GetHashCode()}");

string text = "123";
Console.WriteLine($"{text} geeft een hashcode: {text.GetHashCode()}");

Auto auto = new Auto()
{
    AantalKm = 10000,
    Bouwjaar = 2005,
    Brandstof = Brandstof.Waterstof,
    Kleur = "Rood",
    Model = "Ferrari F40"
};
Console.WriteLine($"{auto} geeft een hashcode: {auto.GetHashCode()}");

Auto auto2 = new Auto()
{
    AantalKm = 20000,
    Bouwjaar = 2005,
    Brandstof = Brandstof.Electriciteit,
    Kleur = "Groen",
    Model = "Ferrari F40"
};
Console.WriteLine($"{auto2} geeft een hashcode: {auto2.GetHashCode()}");
```

```
Testclass c = new Testclass();
c.
```

Equals
GetHashCode — int object.GetHashCode() Serves as the default hash function.
GetType
ToString

```
Hello World!
123 geeft een hashcode:123
123 geeft een hashcode:1747292503
Ferrari F40 (kleur: Rood, bj:2005, km:10000) geeft een hashcode: 58225482
Ferrari F40 (kleur: Groen, bj:2005, km:20000) geeft een hashcode: 54267293
```

AP HOGESCHOOL
ANTWERPEN