Grote O-notatie en complexiteitstheorie.

Zie:
- https://nl.wikipedia.org/wiki/Grote-O-notatie
- https://en.wikipedia.org/wiki/Big_O_notation
- https://en.wikipedia.org/wiki/Computational_complexity_theory
- https://vtk.ugent.be/wiki/Complexiteit

De grote O-notatie is een wiskundige notatie.
Andere benamingen zijn:
- Bachmann-Landau notatie
- asymptotische notatie

In de informatica wordt de notatie gebruikt om algoritmen te classificeren.
De 'O' staat voor 'orde' en geeft de orde van grootte van een functie aan.

Met deze notatie kan de complexiteit van een algoritme aangegeven worden.
Hiermee wordt aangegeven in welke manier de benodigde tijd toeneemt in functie van de omvang (= aantal elementen) van het te verwerken probleem.

Een gebruikelijke onderverdeling:

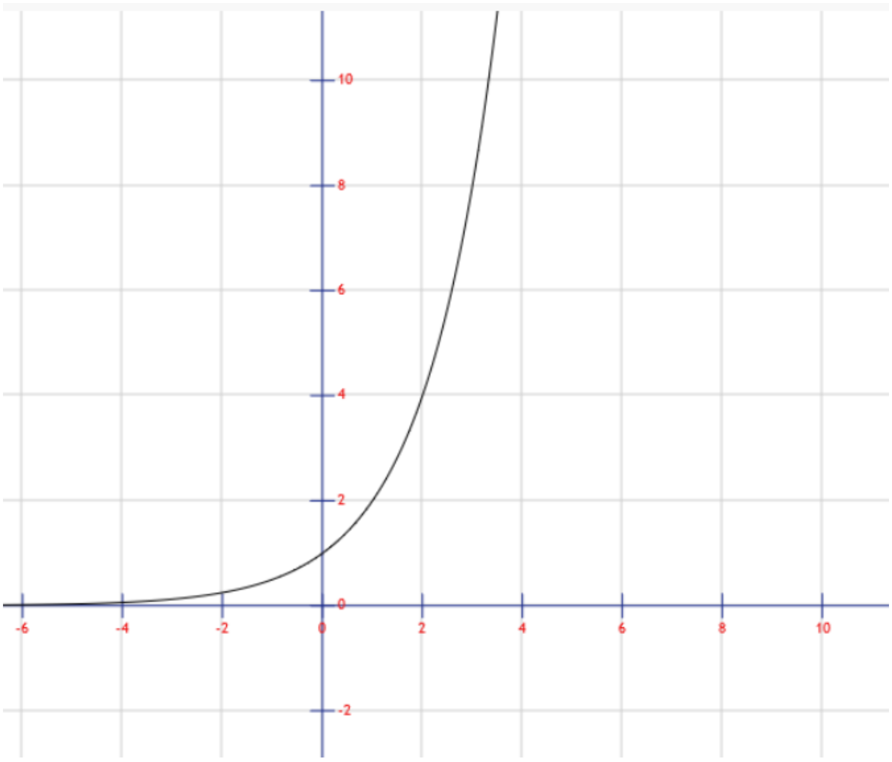
orde	**naam**	**algoritme**
O(1)	constant	is x even of oneven?
O(log(n))	logarithmic	binary search
O(n)	linear	linear search, loop
O(nlog(n))	log linear	mergesort, quicksort, heapsort
$O(n^2)$	quadratic	insertion sort, geneste dubbele loop
$O(n^3)$	cubic	
$O(n^c)$	polynomial	
$O(c^n)$	exponential	traveling salesman
O(n!)	factorial	brute force traveling salesman , recursieve fibonacci

De exponentiële functie.

Machtsverheffen: 2^n : 2*2 . . . 2 (n keer) (n is geheel en positief)
Als n positief en geheel is, is machtsverheffen hetzelfde als herhaald vermenigvuldigen.
De definitie van 2^n kan uitgebreid worden tot reële getallen.

$f(x) = 2^x$

2^x



Een belangrijke eigenschap bij de exponentiële functie is:

$a^{x+y} = a^x * a^y$

De logaritmische functie

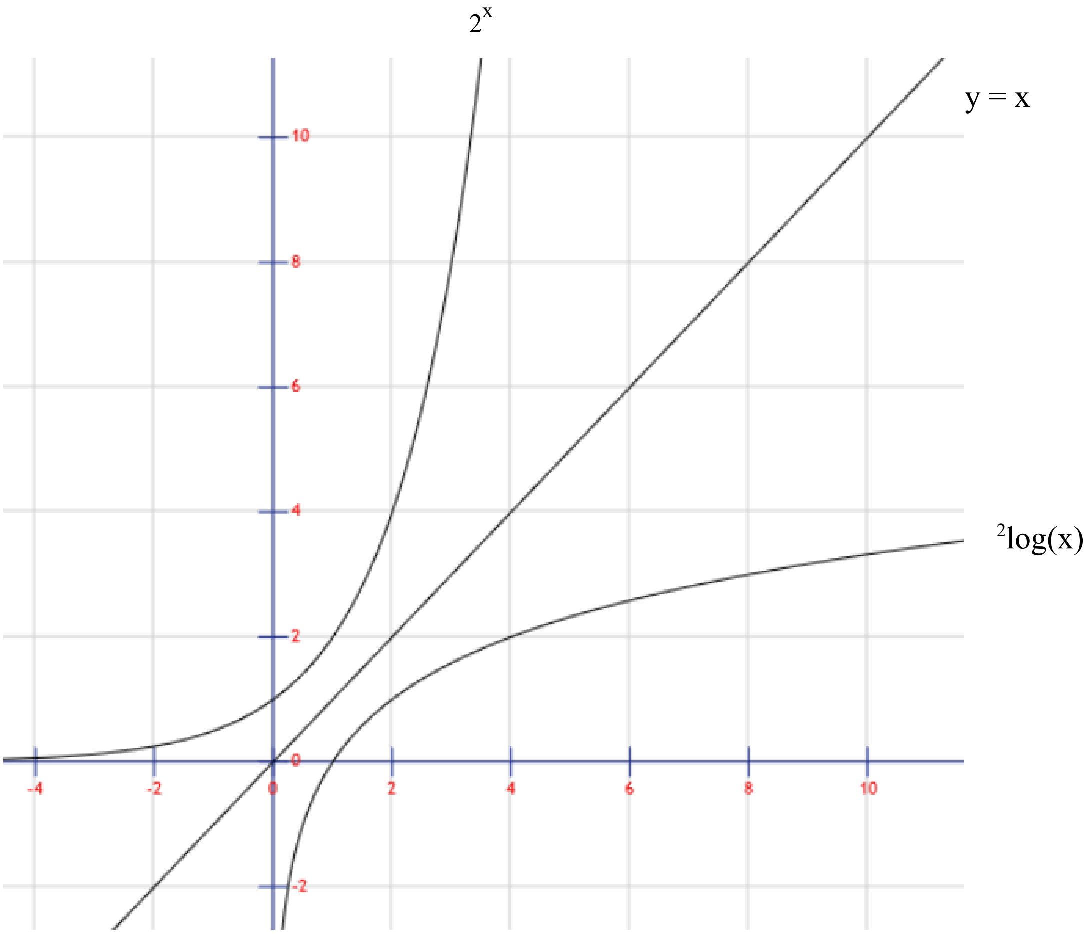
De logaritmische functie is de inverse functie van de exponentiële functie.

Als $y = 2^x$ dan is $x = {}^2\log(y)$

Een voorbeeld: ${}^2\log(64) = 6$ want $2^6 = 64$

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

n	${}^2\log(n)$
1	0
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1024	10



De grafiek laat het volgende zien:

- de functie 2^x en $^2\log(x)$ zijn elkaars gespriegelde t.o.v. de lijn y=x
- $^2\log(x)$ stijgt maar wordt op den duur steeds vlakker

Algemeen: de inverse functie van a^x is: $^a\log(x)$.

Dit houdt in dat $a^{^a\log(x)} = x$ en $^a\log(a^x) = x$

Bij rekenmachines zijn vaak twee logarithmitische functies aanwezig:

- log(x)
- ln(x) , de natuurlijke logaritme

log(x) is de inverse functie van 10^x en is gelijk aaan $^{10}\log(x)$
ln(x) is de inverse functie van exp(x) = e^x en is gelijk aan $^e\log(x)$

(e is de Euler-constante en is gelijk aan 2.718281828459045...)

De volgende eigenschap geldt:

$$^a\log(b) * {}^b\log(c) = {}^a\log(c)$$

Daarom geldt ook: $^b\log c = \dfrac{^a\log(c)}{^a\log(b)}$

en dus ook: $^a\log(b) = \dfrac{^e\log(a)}{^e\log(b)} = \dfrac{\ln(a)}{\ln(b)}$

$$^2\log(x) = \frac{\ln(x)}{\ln(2)}$$

$$^{10}\log(x) = \frac{\ln(x)}{\ln(10)}$$

Python kent de volgende logfuncties:

- math.log(x) : ln(x)
- math.log(x,a) : $^a\log(x)$ (= ln(x)/ ln(a))
- math.log2(x) : $^2\log(x)$ (nauwkeuriger dan math.log(x,2))
- math.log10(x) : $^{10}\log(x)$ (nauwkeuriger dan math.log(x,10))

Oefenopgaven.

Ga uit van een lijst 'a' met len(a) = n

Kies uit de volgende antwoorden: O(1) , O(log(n)) , O(n) , O (n^2)

1. Van welke orde is de opdracht

```
for e in a:
    print(e)
```

2. Van welke orde is de opdracht

```
print(a[0])
```

3. Van welke orde is de opdracht

```
k = len(a)-1
while k > 0:
      print(a[k])
      k //= 2
```

4: Van welke orde is de opdracht:

```
for e1 in a:
   for e2 in a:
       print(e1-e2)
```

Sites:

- https://nl.wikipedia.org/wiki/Grote-O-notatie
- https://en.wikipedia.org/wiki/Big_O_notation
-
- http://www.nldit.com/programmering/computer-programming-languages/201309/86643.html#.WC8HIFx4RQM
- https://justin.abrah.ms/computer-science/big-o-notation-explained.html
- http://bigocheatsheet.com/
- https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/
- http://pages.cs.wisc.edu/~vernon/cs367/notes/3.COMPLEXITY.html#introduction