De grootte van de invoer

Om te bepalen hoe goed een algoritme echt is (polynomiaal of exponentieel bijvoorbeeld), moet de complexiteit van een algoritme uitgedrukt worden in de grootte van de invoer. Bij een algoritme dat N getallen moet sorteren, zeggen we gemakshalve dat de invoer N groot is, en drukken we de complexiteit uit in N: zo heeft (bijvoorbeeld) bubblesort een complexiteit van $\mathcal{O}(N^2)$ en heeft (bijvoorbeeld) mergesort een complexiteit van $\mathcal{O}(N \cdot \lg N)$. Als ieder getal in het array met een constant aantal bits (32 of 64) wordt gerepresenteerd, hoeven we dit niet mee te tellen bij de grootte van de invoer.

Het ligt anders wanneer de invoer van een algoritme alleen een getal N is. Dan is de grootte van de invoer het aantal bits dat nodig is om N te representeren. En dat is gelijk aan $k \approx \lg N$. In dat geval moeten we de complexiteit van het algoritme uitdrukken in k.

Een voorbeeld: samengesteldheid

Om na te gaan of een gegeven getal N samengesteld is of niet, kunnen we het volgende algoritme gebruiken, waarbij we gewoon alle mogelijke getallen tussen 2 en $N-1$ als deler gaan proberen:

```
Gevonden = false;
x = 2;
while (not Gevonden and (x ≤ N-1))
do   if (x is een deler van N)
     then Gevonden = true;
     else  x++;
     fi
od

if (Gevonden)
then return "ja";
else  return "nee";
fi
```

De complexiteit van dit algoritme is (in het slechtste geval) $\mathcal{O}(N)$, want in het slechtste geval doorloopt de while-lus $N-2$ iteraties. Dit moeten we uitdrukken in de grootte van de invoer, het aantal bits dus dat nodig is om N te representeren: $k \approx \lg N$. Omdat $k \approx \lg N$, is $N \approx 2^k$. Ons algoritme heeft dus complexiteit $\mathcal{O}(2^k)$, en dat is exponentieel.

Een voorbeeld: torens van Hanoi

Bij de torens van Hanoi is het nog erger. Ook daar is de invoer alleen een getal N. De recursieve functie die we kunnen gebruiken om de N schijven van het ene stokje naar het andere stokje te verplaatsen, vergt $2^N - 1$ stappen. De complexiteit is dus $\mathcal{O}(2^N)$. De grootte van de invoer is opnieuw $k \approx \lg N$, zodat $N \approx 2^k$. Wanneer we de complexiteit nu uitdrukken in k, zien we dat die $\mathcal{O}(2^{(2^k)})$ is. Dat noemen we dubbel exponentieel, en is veel erger dan 'gewoon' exponentieel.