Recursion

Introductie *37*

Leerkern *37*

5.1 Foundations of recursion *37*
5.2 Recursive analysis *37*
5.3 Applications of recursion *38*

Terugkoppeling *40*

– Uitwerking van de opgaven *40*

Hoofdstuk 5

Recursion

INTRODUCTIE

Veel methoden die we op een datastructuur aan kunnen roepen, zullen op een recursieve wijze geïmplementeerd worden. Recursie is een techniek waarbij een vraagstuk over een datastructuur gereduceerd wordt tot een vraagstuk over een kleinere datastructuur en dit proces te herhalen tot een vraagstuk over een basisgeval dat eenvoudig opgelost kan worden.
In deze leereenheid zullen we uitgebreid ingaan op recursie. Het hele tekstboekhoofdstuk 5 behoort tot de leerstof.

LEERDOELEN
Na het bestuderen van deze leereenheid wordt verwacht dat u
- de bestanddelen van een recursieve methode kunt noemen
- zelf een recursieve definitie kunt lezen en opstellen
- zelf een recursieve methode kunt ontwerpen
- voor- en nadelen van recursie kunt noemen
- lineaire en binaire recursie kunt herkennen
- een paar voorbeelden van recursie kunt geven.

LEERKERN

5.1 Foundations of recursion

OPGAVE 5.1
Geef een recursieve definitie van de functie machtsverheffen x^n voor $n \in \mathbb{N}$ en $x \in \mathbb{R}$.

5.2 Recursive analysis

Hoe wordt recursie door de Java virtual machine mogelijk gemaakt? Bij iedere methodeaanroep worden alle belangrijke gegevens van de aanroepende methode, zoals de plaats van de aanroep (program counter) en de waarde van de parameters en van de lokale variabelen, op een speciale plaats bewaard. Deze plaats heet de *Java-stack.* Bij terugkeer van de aangeroepen methode krijgen alle gegevens hun oorspronkelijke waarde weer terug. Daarom kan een methode ook zichzelf aanroepen zonder dat alle waarden verloren gaan. Na de aanroep kan de methode gewoon doorgaan op het punt waar zij gebleven was.
De Java-stack wordt in hoofdstuk 6 verder uitgelegd en in paragraaf 15.1 wordt nader ingegaan op het gebruik van de Java-stack bij de implementatie van recursie.

De complexiteit van een recursieve methode wordt bepaald door voor elke aanroep apart het aantal primitieve bewerkingen in de body van de methode te bepalen zonder met de recursieve aanroep rekening te houden. De complexiteit van het recursieve algoritme in zijn geheel is dan de som van de complexiteit van de individuele aanroepen.

5.3 Applications of recursion

5.3.1 LINEAR RECURSION

OPGAVE 5.2
Maak opgave R-5.4 van het tekstboek.

OPGAVE 5.3
Geef een recursief algoritme voor het vinden van het maximum in een array A van n elementen. Wat is de complexiteit van uw algoritme?

OPGAVE 5.4
Geef een recursief algoritme dat het aantal schakels telt in een enkel geschakelde lijst. Wat is de complexiteit van uw algoritme?

OPGAVE 5.5
De methode keerOm retourneert voor een gegeven string s een string waarvan de letters in omgekeerde volgorde staan. Bijvoorbeeld keerOm("bal") heeft als terugkeerwaarde "lab". Geef een recursieve implementatie in Java van de methode keerOm.
Wat is de complexiteit van uw algoritme?
Aanwijzing: beschouw een niet-lege string als bestaand uit één karakter gevolgd door een string en gebruik om die string te krijgen de methode substring(int) uit de klasse java.lang.String.

5.3.2 BINARY RECURSION

Het tekstboek gebruikt de notatie $\log_2 x$ in plaats van het in Nederland meer gebruikelijke ${}^2\log x$, om de logaritme van x met grondtal 2 aan te duiden.

OPGAVE 5.7
Maak nogmaals opgave 5.3, nu met behulp van binaire recursie. Wat is de complexiteit van uw algoritme?

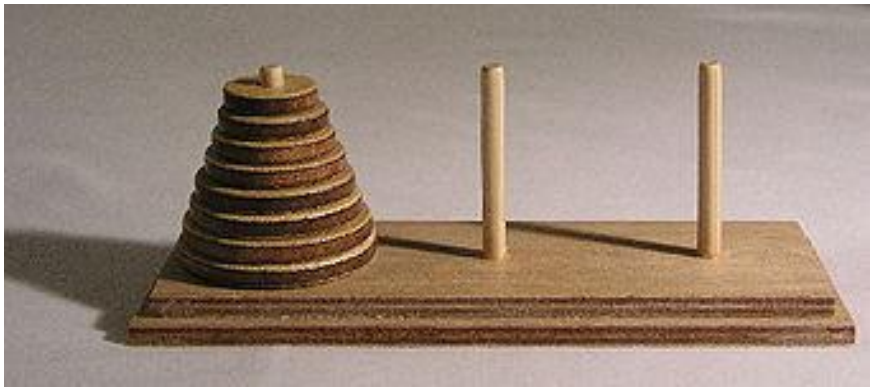
We beëindigen dit hoofdstuk met een mooie toepassing van recursie, 'De torens van Hanoi'.

'De torens van Hanoi' is een wiskundige puzzel en geldt als standaard-voorbeeld voor recursie omdat het zich op elegante wijze recursief laat oplossen.

OPGAVE 5.7

a In yOUlearn vindt u de url van 'De torens van Hanoi' op wikipedia. Lees de uitleg van het spel op deze pagina.

b Probeer de puzzel zelf op te lossen, eerst met vier schijven en daarna met een groter aantal schijven. (zie figuur 5.1).



FIGUUR 5.1 De torens van Hanoi

c Geef in woorden aan hoe u de puzzel op een recursieve manier op zou lossen.

d Denkt u dat het ook mogelijk is om 'De torens van Hanoi' op een niet-recursieve manier op te lossen en zou het dan sneller werken?

TERUGKOPPELING

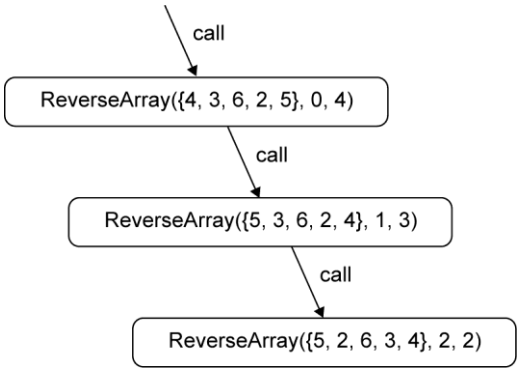
Uitwerking van de opgaven

5.1 Recursieve definitie van x^n voor $n \in \mathbf{N}$:

i $x^0 = 1$
ii $x^n = x \cdot x^{(n-1)}$ voor $n > 0$

NB: de toevoeging $n > 0$ in het ii-deel is van belang voor de juistheid van de definitie.

5.2 Een trace van deze aanroep is



FIGUUR 5.2 De trace van de methodeaanroep

5.3 Als een array maar één element bevat, is het maximum van de array-elementen gelijk aan dit element. Dit geval vormt de basis van de recursie. Het maximum van een array A met n elementen is gelijk aan het maximum van $A[n-1]$ en het maximum van het arraydeel dat bestaat uit de eerste $n-1$ elementen van de array. Om recursie te kunnen gebruiken geven we het algoritme daarom een extra parameter mee: maxArray(A, k) bepaalt het maximum van de eerste k elementen van A.

Algorithm maxArray(A, k)
Input: een array A gevuld met gehele getallen en een natuurlijk getal k > 0.
Output: het maximum van de eerste k getallen uit A

als k = 1
return A[k – 1]
anders
return max(A[k – 1], maxArray(A, k – 1])

Aanroep van deze methode met als tweede parameter A.length geeft het maximum van het gehele array.
De complexiteit van de body van maxArray(A,k) zonder rekening te houden met recursieve aanroepen, is O(1).
De complexiteit van maxArray(A, k) is de som van de complexiteit van alle (k) aanroepen, dus O(k). De complexiteit voor het berekenen van het maximum van het gehele array is O(n), zelfs Θ(n).

5.4 **Algorithm** lengte(v)
Input: Een schakel van de lijst.
Output: De lengte van de lijst vanaf de schakel v

als v = null
return 0
anders
return lengte(v.getNext()) + 1

Aanroep van deze methode op de eerste schakel geeft de lengte van de totale lijst.
De complexiteit van de body van de methode zonder rekening te houden met de recursieve aanroepen, is O(1). Voor een lijst met lengte n zijn er n aanroepen nodig, de complexiteit is dan ook O(n), zelfs Θ(n).

5.5 In deze oplossing kan de string s ook de lege string zijn.

```
/**
 * Geeft de omkering van een gegeven string.
 * @param s de string
 * @return de omkering van s.
 */
public static String keerOm(String s) {
  if (s.length()== 0) {
    return "";
  }
  else {
    return keerOm(s.substring(1)) + s.charAt(0);
  }
}
```

Als de complexiteit van substring en charAt O(1) zijn, is de complexiteit van de body van de methode zonder rekening te houden met de recursieve aanroepen O(1); er zijn n recursieve aanroepen als n de lengte van de string is, de complexiteit is dus O(n), zelfs Θ(n).

5.6 Nu splitsen we, net als in codefragment 5.10, de array bij elke recursiestap in twee delen:

Algorithm maxArray(A, i, n)
Input: een array A gevuld met gehele getallen en natuurlijk getallen i en n
Output: het maximum van de n getallen uit A vanaf index i

als n = 1
return A[i]
anders
return max(maxArray (A, i, $\lceil n/2 \rceil$), maxArray(A, i + $\lceil n/2 \rceil$, $\lfloor n/2 \rfloor$))

Aanroep van de methode maxArray(A, 0, A.length) berekent het maximum van alle elementen uit A.
De complexiteit van de methode zonder rekening te houden met de recursieve aanroepen is O(1). Er zijn 2n-1 aanroepen, zoals uitgelegd in paragraaf 5.3.2, dus de complexiteit is O(n), zelfs Θ(n).

5.7 a Het spel 'De torens van Hanoi' bestaat uit een stapel van N schijven en drie stokken, de bronstok , de hulpstok en de doelstok. De stapel schijven moet verplaatst worden van de bronstok naar de doelstok. Daarbij kan men gebruikmaken van de hulpstok. Men mag maar één schijf tegelijk verplaatsen en er mag nooit een grotere schijf op een kleinere schijf geplaatst worden.
b In yOUlearn vindt u een link naar een animatie.
c 1 schijf kan direct van bronstok naar doelstok verplaatst worden.
Het verplaatsen van N schijven kan als volgt worden opgelost.Verplaats de bovenste N – 1 schijven van bronstok naar hulpstok met doelstok als hulppin
Verplaats de grootste schijf van bronstok naar doelstok
Verplaats de N – 1 schijven van hulpstok naar doelstok met bronstok als hulppin

d Alles wat recursief opgelost kan worden, is ook iteratief op te lossen. In het geval van 'De torens van Hanoi' zullen iteratieve oplossingen complexer worden. Een recursieve methode maakt impliciet gebruik van de Java-stack om tot de oplossing te komen. Een iteratieve methode voor 'De torens van Hanoi' zou bijvoorbeeld zelf alle tussenresultaten kunnen opslaan, net zoals het in de Java-stack gebeurt.
Dit zou zeker niet sneller zijn dan een recursieve methode.