Complexiteit

Uitwerking Opgave 6

a) Merk op dat zolang *Gewisseld* True blijft (dus True wordt in regel (6)), de body van de (buitenste) while-lus gedaan wordt. Zodra *Gewisseld* False is na regel (6) stopt het algoritme. De meeste arrayvergelijkingen worden dus gedaan als *Gewisseld* zo lang mogelijk True blijft.
Per doorgang door de buitenste while-lus wordt eerst *Boven* met 1 afgelaagd, en vervolgens worden *Boven* vergelijkingen uitgevoerd, waarbij *Boven* in de eerste iteratie op $n-1$ wordt gezet, in de volgende op $n-2$, etcetera. Wanneer *Boven* in regel (3) gelijk is geworden aan 1, vindt nog 1 vergelijking plaats. Indien dan nog gewisseld wordt, zal de buitenste while loop nóg een keer doorlopen worden, maar *Gewisseld* wordt (en blijft) dan False en bij die doorgang vinden geen vergelijkingen meer plaats omdat op dat moment na regel (3) $Boven = 0$. Daarna vindt de test in regel (2) nog één keer plaats en stopt het algoritme. Indien niet meer gewisseld wordt zal de buitenste while stoppen (omdat *Gewisseld* False is) met $Boven = 1$. In beide gevallen wordt dus geen vergelijking meer gedaan.
In het slechtste geval wordt in elke ronde in de for-loop met $Boven > 1$ gewisseld, zodat de for-loop in de volgende ronde nog een keer wordt uitgevoerd. In het slechtste geval worden dus $1 + \ldots + n - 1 = \frac{1}{2}n(n-1)$ vergelijkingen gedaan. De test in regel (2) wordt dan overigens n (als $A[1] \leq A[2]$) of $n+1$ (als $A[1] > A[2]$) keer gedaan.

Om deze worst-case van het algoritme gedaan te krijgen moet de binnenste loop dus $n-1$ keer worden gestart, namelijk voor $Boven = n, n-1, \ldots, 2$ vóór regel (3). Daartoe moet na elk van de eerste $n-2$ iteraties de boolean *Gewisseld* op True zijn gezet, m.a.w. er moet in elk van de eerste $n-2$ iteraties gewisseld worden; het maakt niet uit of er in de laatste iteratie ($Boven = 2$ vóór regel (2), dus $= 1$ na regel (3)) gewisseld wordt, de vergelijking gebeurt altijd.

We bekijken nu voor wat voor invoerarrays het worst case aantal vergelijkingen gedaan wordt.
Elk element dat te veel naar rechts staat (te klein is) wordt, per iteratie, één plaats naar links verschoven. We hebben dus de eerste $n-2$ iteraties ten minste één wissel precies als er een element is dat ten minste $n-2$ plaatsen naar links verschoven moet worden (als alle elementen minder verschoven moeten worden stoppen we eerder, omdat dan in een eerdere iteratie alle elementen al op hun plaats staan). Dit kan alleen maar de waarde op plek n en/of op plek $n-1$ zijn. In de worst case moet dus ofwel $A[n-1]$ de kleinste zijn van $A[1], \ldots, A[n-1]$, of moet $A[n]$ de kleinste of één-na kleinste van het hele array zijn.

b) Aangezien *Gewisseld* op True wordt geïnitialiseerd zal de binnenste for-loop ten minste één keer worden uitgevoerd, dus de vergelijking $A[j] > A[j+1]$ gebeurt ten minste $n-1$ keer. In het beste geval worden ook precies alleen die $n-1$ vergelijkingen gedaan, en dat komt voor dan en slechts dan als meteen na de eerste ronde *Gewisseld* False is. Dat is het geval dan en slechts dan als voor alle $j = 1, 2, \ldots, n-1$ $A[j] \leq A[j+1]$. En dat betekent precies dat $A[1] \leq A[2] \leq \cdots \leq A[n]$, ofwel dat het array oplopend gesorteerd is.

c) Het aantal arrayvergelijkingen is een goede maat voor de complexiteit, en de arrayvergelijkingen in regel (5) een goede basisoperatie. Wat betreft de maatgevendheid: de operaties in regel (6) gebeuren hooguit even vaak als de vergelijking in regel (5). Die vergelijking wordt ten minste $n-1$ keer uitgevoerd (eerste ronde) en regel (3) en regel (4) (*) maximaal n keer, dus hooguit even vaak in orde van grootte. De test in regel (2) gebeurt maximaal $n+1$ keer, in orde van grootte dus hooguit even vaak als de vergelijking $A[j] > A[j+1]$. (We nemen voor het gemak aan dat $n > 1$; voor $n = 1$ is de arrayvergelijking niet maatgevend.)
(*) We tellen regel (4) hier voor het gemak als 1 operatie, maar eigenlijk moeten we de for-loop omschrijven naar een while-loop en de test $j \leq \mathit{Boven}$ vergelijken met de arrayvergelijking in regel (5).

d) Met deze aanpassing verandert de worst case niet: met een omgekeerd gesorteerd rijtje geldt dat $\mathit{Onder} = 1$ en $\mathit{Boven} = n - i$ na regel 3 voor ieder van de $n-1$ relevante iteraties. De complexiteit blijft dus hetzelfde. Hooguit wordt het aantal worst-case invoeren beperkt.

De best case verandert ook niet; in de aangepaste versie van het algoritme wordt de buitenste loop sowieso 1 keer doorlopen, en in de best case blijft dat 1 keer. Er wordt in dat geval niet gewisseld en *Onder* en *Boven* hebben in het geheel geen effect. Het aantal vergelijkingen is dus nog steeds $n-1$ en ook de best case invoer verandert niet.