Complexiteit

Uitgebreide uitwerking opgave 21

a. Het kan voorkomen dat regel (11) nul keer wordt uitgevoerd, maar regel (5) $n-1$ keer (%). Dat is echt een ordeverschil, dus de test uit (11) is zeker niet maatgevend voor de totale hoeveelheid werk die het algoritme doet. Dit ((%) dus) is het geval als het startarray A reeds oplopend gesorteerd is: regel (5) wordt $n-1$ keer gedaan en er wordt geconstateerd in regel (8) dat het array al gesorteerd is, en dientengevolge worden regel (9) t/m (14) overgeslagen en stopt het algoritme.

b. Omdat `gesorteerd` in regel (1) op `False` is gezet en $n \geq 2$ zal de test in regel (5) voor $i = 1$ ten minste 1 keer gedaan worden. Als het beginarray oplopend gesorteerd is, zal `gesorteerd` voor $i = 1$ steeds `True` blijven, en zal regel (5) $n-1$ keer worden uitgevoerd. Als het beginarray niet oplopend gesorteerd is zal regel (5) voor $i = 1$ ten minste 1 keer gedaan worden, is de test in regel (8) vervolgens `True` en zal de test in regel (11) uitgevoerd worden voor $j = 2$ t/m n, dus $n-1$ keer. In beide gevallen worden dus in ronde 1 al ten minste $n-1$ arrayvergelijkingen gedaan.

Algemene observatie: zolang het array niet gesorteerd is worden in ronde i altijd $n-i$ arrayvergelijkingen in regel (11) gedaan en minstens 1 in regel (5). Als in ronde i door de verwisseling in regel (14) het array gesorteerd raakt, vindt er —als tenminste $i < n-1$[1]— nog 1 ronde plaats (ronde $i+1$ dus), waarin geconstateerd wordt dat het array inderdaad gesorteerd is. Dat levert dan $n-1-i$ vergelijkingen in regel (5) op, en vervolgens geeft de test in regel (8) False, worden regel (9) t/m (14) overgeslagen en stopt het algoritme.

c. We zagen al in **b.** dat als het beginarray niet oplopend gesorteerd is, er in ronde 1 (dus voor $i = 1$) al ten minste $1 + n - 1 = n > n - 1$ arrayvergelijkingen worden gedaan; ten minste 1 vergelijking in regel (5), en $n-1$ in regel (11). De ondergrens van $n-1$ wordt dan sowieso niet gehaald. Echter, als het beginarray gesorteerd is, is $n-1$ vergelijkingen wel haalbaar. Er wordt in dat geval maar 1 ronde gedaan; in totaal doet het algoritme dan $n-1$ vergelijkingen in regel (5) en 0 in regel (11), dus $n-1$ ijn totaal. Het moge dus duidelijk zijn dat het beste geval voorkomt dan en slechts dan als het beginarray oplopend gesorteerd is. Het aantal vergelijkingen is dan $n-1$.

d. Het slechtste geval krijg je als de buitenste while-lus zo vaak mogelijk wordt doorlopen (m.a.w. A blijft zo lang mogelijk niet gesorteerd), en als in elke ronde i regel (5) zo vaak mogelijk gebeurt, te weten $n-i$ keer. Dat zou het maximale aantal keer regel (5) opleveren. Regel (11) gebeurt in ronde i altijd[2] een vast aantal keer, namelijk $n-i$. De worst case vindt dus plaats als in elke ronde i ($i = 1, 2, \ldots, n-1$) pas in de laatste vergelijking van regel (5) wordt opgemerkt dat A niet oplopend gesorteerd is. Dat kost dan in ronde i telkens $n-i$ vergelijkingen in regel (5).

[1] Als $i = n-1$ worden verder geen vergelijkingen meer gedaan; beide binnenste while-loops worden in de volgende ronde overgeslagen en daarna stopt het algoritme.

[2] Als de test in regel (8) `True` heeft opgeleverd en de while-loop van regel (9) t/m regel (13) inderdaad plaatsvindt

We bekijken of dat kan voorkomen en zo ja voor wat voor invoer dat voorkomt en hoeveel vergelijkingen dat precies oplevert. In bovengenoemd geval moet voor elke ronde i gelden dat $A[j] < A[j+1]$ voor alle $j = i, i+1, \ldots, n-2$, maar $A[n-1] > A[n]$. Het totale aantal vergelijkingen zou in dat geval worden: $n-1+n-2+\cdots+2+1$ (regel (5)) $+n-1+n-2+\cdots+2+1$ (regel (11)). Dat is dus bij elkaar $\frac{1}{2}n(n-1)+\frac{1}{2}n(n-1) = n(n-1)$ arrayvergelijkingen.
Dit kan inderdaad voorkomen, namelijk (zoals hierboven al aangegeven) dan en slechts dan als A in elke ronde $i = 1, 2, \ldots, n-1$ oplopend gesorteerd is op het laatste tweetal na, en dat komt precies voor als het beginarray er zo uitziet, dus het beginarray is oplopend gesorteerd op de laatste twee elementen na: $A[j] < A[j+1]$ voor alle $j = 1, 2, \ldots, n-2$, maar $A[n-1] > A[n]$. Het doet er niet toe of $A[n]$ de kleinste van heel A is of niet, als $A[n]$ maar niet de grootste is. Als het beginarray A oplopend gesorteerd is op de laatste twee elementen na, dan zal A er in elke ronde ook zo uitzien (zie (*) hieronder), en doet het algoritme het maximale aantal vergelijkingen. Een voorbeeld van een worst-case invoerrijtje is $2, 4, 6, 8, 10, 5$.

(*) Stel dat in ronde i $A[1], \ldots, A[n-1]$ oplopend gesorteerd is op de laatste twee elementen na. (i) Als $A[n]$ niet de kleinste is van $A[i]$ t/m $A[n]$, dan is $A[i]$ de kleinste, en wordt in regel (14) $A[i]$ met zichzelf verwisseld en hebben we in ronde $i+1$ dus weer dezelfde situatie. (ii) Als $A[n]$ wel de kleinste is van $A[i]$ t/m $A[n]$, wordt in regel (14) $A[i]$ met $A[n]$ verwisseld, en komt de oude $A[i]$ achteraan te staan en zitten we in de ronde erna (ronde $i+1$) in het geval dat A oplopend gesorteerd is behalve de laatste twee elementen, en met de kleinste van $A[i+1]$ t/m $A[n]$ achteraan. Immers voor de oude A gold dat $A[i] < A[i+1] < \cdots < A[n-1]$.

e. Stel dat A in het begin aflopend gesorteerd is. Voor het gemak nemen we even aan dat n even is. In ronde 1 stopt de while-lus van regel (4) t/m (7) meteen na de eerste doorgang (1 vergelijking in regel (5)), en worden in de volgende while-lus (regel (10 t/m (13)), waarin de kleinste wordt gezocht, $n-1$ vergelijkingen gedaan. In regel (14) worden $A[1]$ (de grootste!) en $A[n]$ (de kleinste!) verwisseld. Nu staan dus het eerste en het laatste element goed. In ronde 2 worden $1+n-2$ vergelijkingen gedaan, en na afloop staan de eerste twee en de laatste twee elementen goed. Zo gaan we door. Na ronde $\frac{n}{2}$ staan de eerste $\frac{n}{2}$ en de laatste $\frac{n}{2}$ elementen goed, dus zijn alle array-elementen op de juiste plek gezet. Er is dan nog 1 ronde nodig (namelijk ronde $\frac{n}{2}+1$) om te constateren dat A inmiddels goed gesorteerd is.
In totaal is het aantal vergelijkingen: $\frac{n}{2}+n-1+n-2+\cdots+n-\frac{n}{2}+n-1-\frac{n}{2}$. Immers $\frac{n}{2}$ rondes, elk 1 vergelijking regel (5) + aantal vergelijkingen regel (11) rondes 1 t/m $\frac{n}{2}$ + aantal vergelijkingen regel (5) uit ronde $\frac{n}{2}+1$. Bij elkaar opgeteld is dit $\frac{3}{8}n^2+\frac{3}{4}n-1$.
Voor oneven n is dit aantal overigens $\lfloor\frac{n}{2}\rfloor+n-1+n-2+\cdots+n-\lfloor\frac{n}{2}\rfloor+n-1-\lfloor\frac{n}{2}\rfloor$ (etcetera).
Merk op dat voor dit invoerarray (aflopend gesorteerd) het aantal vergelijkingen in regel (5) gelijk is aan $\frac{n}{2}+n-1-\frac{n}{2} = n-1$ (we nemen n even), en het aantal vergelijkingen in regel (11) gelijk is aan $n-1+n-2+\cdots+\frac{n}{2} = n-1+n-2+\cdots+2+1-(\frac{n}{2}-1+\frac{n}{2}-2+\cdots+2+1) = \frac{1}{2}\cdot n(n-1)-\frac{1}{2}\cdot\frac{n}{2}(\frac{n}{2}-1) = \frac{3}{8}n^2-\frac{1}{4}n \in \Theta(n^2)$. Dat is een ordeverschil, dus ook regel (5) alleen is geen goede maat voor de complexiteit (niet maatgevend). We moeten dus altijd de arrayvergelijkingen in regel (5) en regel (11) samen tellen.