Complexiteit

Uitwerkingen opgave 34 + 38

opgave 34
Het zo ontstane algoritme heet "binary insertion sort" en is zeer efficiënt qua vergelijkingen (zie ook college): het doet er hoogstens

$$\sum_{i=1}^{n}\lceil \lg i\rceil \geq \lceil \sum_{i=1}^{n} \lg i\rceil = \lceil \lg n!\rceil,$$

de theoretische ondergrens voor het het aantal arrayvergelijkingen nodig om de juiste sortering te vinden.
Het algoritme doet echter nog steeds $\Theta(n^2)$ verschuivingen van sleutels: dezelfde verschuivingen namelijk als gewoon insertion sort. Dit toont aan dat het vergelijken van sleutels eigenlijk niet een goede basisoperatie is om de complexiteit van dit algoritme mee te beschrijven. Het algoritme is weliswaar gebaseerd op deze vergelijkingen, maar het aantal vergelijkingen is niet maatgevend voor de complexiteit van het algoritme; verschuiven van sleutels is dat overigens wel. Het probleem zou kunnen worden opgeheven door een pointerlijst te gebruiken, maar dan is binair zoeken weer lastig.

opgave 38
a. Steeds wordt het grootste kind bepaald (ten koste van één vergelijking, tenzij het een knoop betreft met één kind, wat een zeldzaamheid is), waarna dit wordt vergeleken met de oorspronkelijke waarde van $A[i]$. Dus maximaal $2d$ vergelijkingen, waarbij d de hoogte van de subboom met i als wortel is. Ter herinnering: de wortel van een binaire boom zit op nivo 0, en de hoogte van de boom is het hoogste nivo dat voorkomt. Een voorbeeld: voor een boompje met alleen een wortel en twee kinderen kost het 2 vergelijkingen (hoogte = 1); als het linker kind zelf ook nog een (linker) kind heeft (hoogte = 2), wordt het maximaal 3 stuks: één minder dan de beloofde $2 * 2 = 4$.
b. We moeten de uitkomst van **a.** sommeren over alle knopen waarvoor *SiftUp* gedaan wordt. We nemen voor het gemak aan dat de boom "geheel gevuld" is (dan is $n = 2^k - 1$). In het algemene geval —het onderste niveau mogelijk niet geheel gevuld— zal het aantal vergelijkingen alleen maar minder zijn. Onderstaande uitkomst is dus in het algemeen een bovengrens op het aantal arrayvergelijkingen. We vinden zo (met $d = \lfloor \lg n \rfloor$, terwijl ℓ over de nivo's loopt):

$$\sum_{\ell=0}^{d-1} 2(d-\ell) * (\text{aantal knopen op nivo } \ell) =$$

$$2\sum_{\ell=0}^{d-1}(d-\ell)2^{\ell} = 2^{d+2} - 2d - 4 = 2^{\lfloor \lg n\rfloor+2} - 2\lfloor \lg n\rfloor - 4 \in \Theta(n),$$

waarbij de uitkomst van de sommatie met behulp van inductie bewezen kan worden (of door de formule voor meetkundige reeks en opgave 12 te gebruiken). De constructie van de heap vindt dus plaats in lineaire tijd!
c. We krijgen als bovengrens:

$$2\sum_{j=1}^{n-1}\lfloor \lg j\rfloor \leq 2(n-1)\lfloor \lg(n-1)\rfloor \leq 2n\lg n \in \Theta(n\lg n).$$

De afschatting kan preciezer, maar dit resultaat is voor ons voldoende.
d. Samen vinden we dat het algoritme als worst case complexiteit $\Theta(n \lg n)$ heeft.