Complexiteit 2023 — Opgave 1 (uitwerking)

1 maart 2023

```
1 for i := 2 to n do
2 |   x := A[i];
3 |   j := i − 1;
4 |   while j > 0 and A[j] > x do
5 |   |   A[j + 1] := A[j];
6 |   |   j := j − 1;
7 |   od
8 |   A[j + 1] := x;
9 od
```

a. Merk ten eerste op dat de for-loop op regel 1 itereert van $i = 2$ tot $i = n$, dus precies $n - 1$ iteraties. Op regel 3 wordt j geïnitialiseerd op $i - 1$, dus $j \geq 1$. De eerste vergelijking $j > 0$ op regel 4 levert dus altijd *True* op, en daarmee gebeurt de vergelijking $A[j] > x$ op regel 4 minstens één keer per iteratie. De vergelijking $A[j] > x$ op regel 4 gebeurt dus minstens $n - 1$ keer.

- regels 1, 2, 3 en 8 gebeuren elk precies $n - 1$ keer (één keer per iteratie van de for-loop op regel 1), en dus hooguit even vaak als de vergelijking $A[j] > x$.
- de vergelijking $j > 0$ op regel 4 gebeurt per iteratie van de for-loop op regel 1 hooguit één keer vaker dan de vergelijking $A[j] > x$, en in totaal dus hooguit $n - 1$ keer vaker. Dat is hooguit een (constante) factor 2, dus in orde van grootte gebeurt deze vergelijking hooguit even vaak als de vergelijking $A[j] > x$.
- regels 5 en 6 gebeuren alleen als beide vergelijkingen op regel 4 *True* opleveren en dus zijn uitgevoerd, en dus hooguit even vaak als elk van deze vergelijkingen, waaronder de vergelijking $A[j] > x$.

b.

Zoals betoogd bij **a.**, wordt de vergelijking $A[j] > x$ minstens $n - 1$ keer uitgevoerd.

Dit wordt gehaald dan en slechts dan als de vergelijking precies één keer wordt uitgevoerd per iteratie van de for-loop op regel 1. Met andere woorden: de while-loop moet termineren voordat de vergelijking $A[j] > x$ een tweede keer wordt uitgevoerd. Dit gebeurt in twee gevallen: als $A[i - 1] \leq A[i]$ (dan faalt de vergelijking $A[j] > x$ in de eerste iteratie van de while-loop) of als $i - 2 \leq 0$

(dan faalt de vergelijking $j > 0$ in de tweede iteratie van de while-loop en wordt de vergelijking $A[j] > x$ niet meer gemaakt).

Hieruit volgt dat $A[i-1] \leq A[i]$ in ieder geval voor alle $3 \leq i \leq n$. Voor $i = 2$ maakt het niet uit. Dit geeft dus twee best case-invoerrijtjes: het oplopend gesorteerde rijtje, en het bijna oplopend gesorteerde rijtje waarin alleen de eerste twee elementen verkeerdom staan.

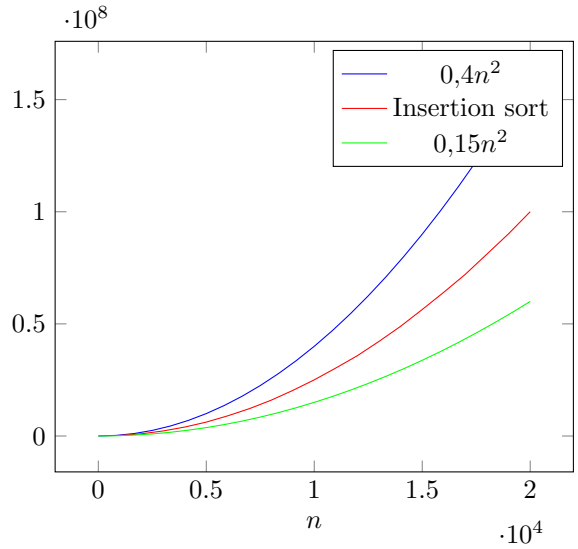
c.

Zoals opgemerkt bij **a.**, zijn er precies $n-1$ iteraties van de for-loop op regel 1. De tweede vergelijking op regel 4 kan hooguit gebeuren zolang $j > 0$. Omdat j begint op $i-1$ en telkens met één wordt verlaagd, is dit dus hooguit $i-1$ keer. In totaal wordt de vergelijking $A[j] > x$ dus hooguit $\sum_{i=2}^{n}(i-1) = \frac{1}{2}n(n-1)$ keer gedaan.

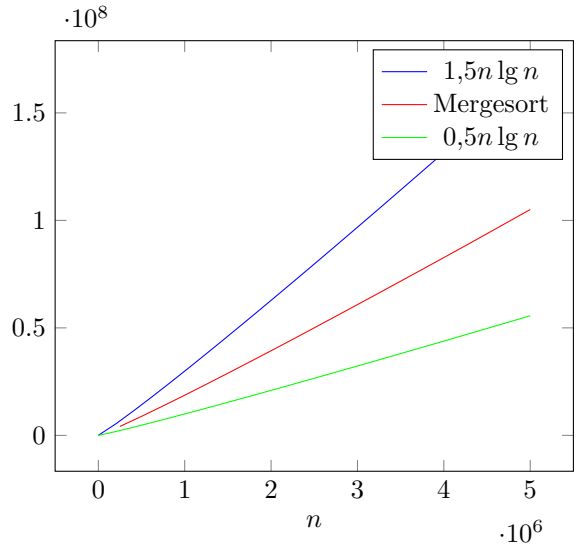
Dit wordt gehaald als voor elke bekeken i en voor elke $j > 1$ geldt dat $A[j] > A[i]$. Als dit niet zo is, termineert de while-loop immers. Wederom maakt de uitkomst van de vergelijking met $j = 1$ niet uit, daarna termineert de while-loop immers toch omdat $j \leq 0$.

Hieruit volgt dat $A[i]$ voor alle $2 \leq i \leq n$ ofwel het kleinste of het op één na kleinste element moet zijn van het deelrijtje $A[1], \ldots, A[i]$. De worst case-invoerrijtjes zijn precies alle rijtjes die hieraan voldoen. (Overtollig, maar: dit zijn er precies 2^{n-1}; voor elk van de $n-1$ posities na de eerste heb je immers twee opties.)

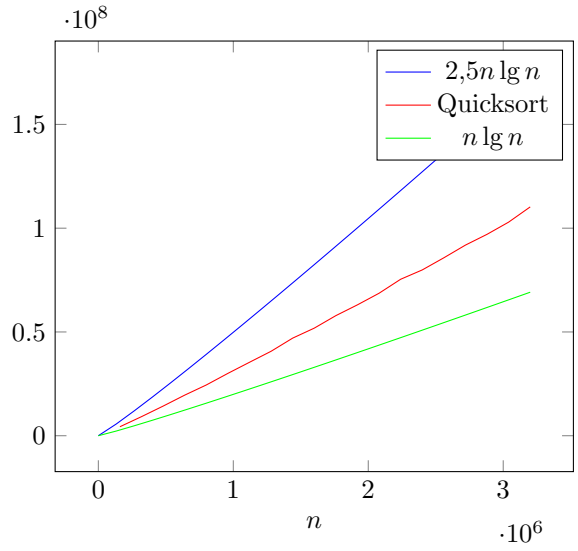
d.



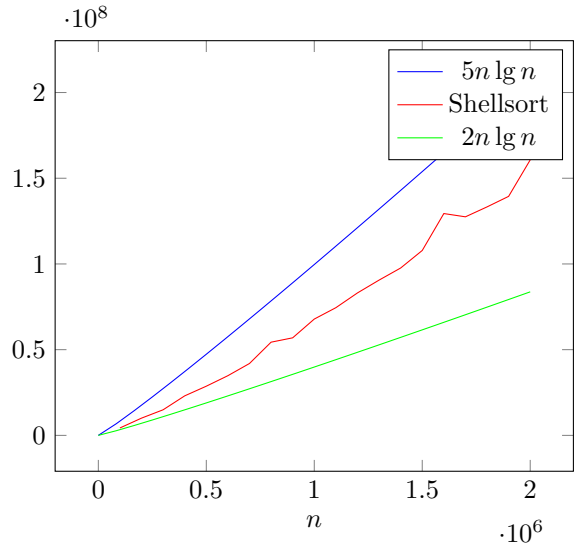
Voor Insertion sort is het op basis van de grafiek aannemelijk dat de average case-complexiteit in $O(0{,}4n^2) = O(n^2)$ en $\Omega(0{,}15n^2) = \Omega(n^2)$ en daarmee in $\Theta(n^2)$ zit.



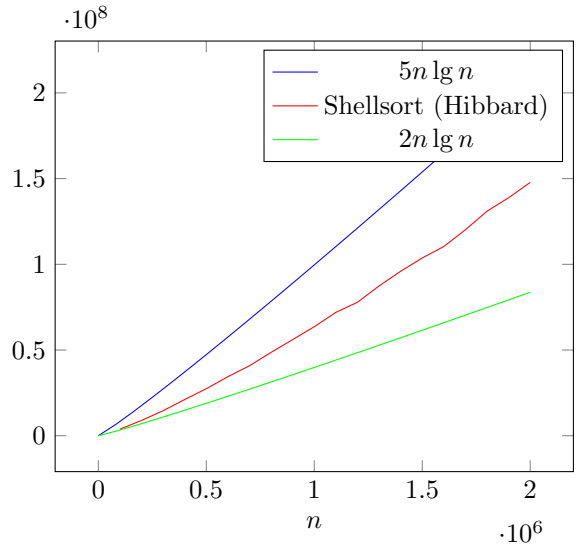
Voor Mergesort is het op basis van de grafiek aannemelijk dat de average case-complexiteit in $O(1{,}5n \lg n) = O(n \lg n)$ en $\Omega(0{,}5n \lg n) = \Omega(n \lg n)$ en daarmee in $\Theta(n \lg n)$ zit.



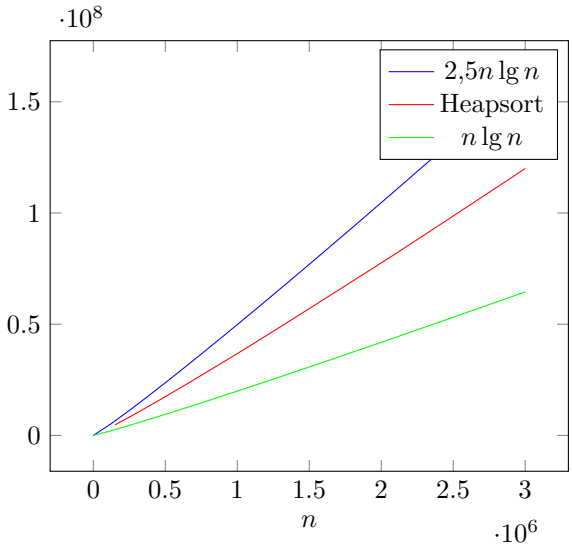
Voor Quicksort is het op basis van de grafiek aannemelijk dat de average case-complexiteit in $O(2{,}5n \lg n) = O(n \lg n)$ en $\Omega(n \lg n)$ en daarmee in $\Theta(n \lg n)$ zit.



Voor Shellsort is het op basis van de grafiek aannemelijk dat de average case-complexiteit in $O(5n \lg n) = O(n \lg n)$ en $\Omega(2n \lg n) = \Omega(n \lg n)$ en daarmee in $\Theta(n \lg n)$ zit.



Voor Shellsort (Hibbard) is het op basis van de grafiek aannemelijk dat de average case-complexiteit in $O(5n \lg n) = O(n \lg n)$ en $\Omega(2n \lg n) = \Omega(n \lg n)$ en daarmee in $\Theta(n \lg n)$ zit.



Voor Heapsort is het op basis van de grafiek aannemelijk dat de average case-complexiteit in $O(2{,}5n \lg n) = O(n \lg n)$ en $\Omega(n \lg n)$ en daarmee in $\Theta(n \lg n)$ zit.

e. Opmerking: deze vraag was slecht gesteld, het was niet nodig om dit te doen voor *alle* gevonden rijtjes bij **b** en **c**. Je krijgt de volle punten als je dit juist doet voor één gevonden rijtje.

In de volgende tabel is aangegeven hoeveel vergelijkingen alle algoritmen doen op een oplopend respectievelijk aflopend gesorteerd rijtje voor $n = 10\,000$.

Algoritme	oplopend	aflopend
Bubblesort	9 999	49 995 000
Insertion sort	9 999	49 995 000
Mergesort	69 008	64 608
Quicksort	50 014 998	50 019 998
Shellsort	120 005	172 578
Shellsort (Hibbard)	113 631	144 824
Heapsort	244 460	226 682

Het valt op dat Bubblesort en Insertion sort het significant beter doen voor een oplopend gesorteerd rijtje dan voor een gemiddeld rijtje, namelijk ongeveer n vergelijkingen. Het valt ook op dat Quicksort het beduidend slechter doet voor beide rijtjes van voor een gemiddeld rijtje, namelijk ongeveer $\frac{1}{2}n^2$ vergelijkingen. De andere getallen zijn allemaal in lijn der verwachting met de grafieken.